Jyrki Joutsensalo, Timo Hämäläinen, and Alexander Sayenko
"QoS Supported Networks, Scheduling, and Pricing; Theory and Applications"

November 18, 2008

UNIVERSITY OF JYVÄSKYLÄ 2008

PREFACE AND ACKNOWLEDGMENTS

The basic premise of our subject, QoS management in communication networks, is that, as more users start to use modern networks, and as their usage patterns evolve to include more QoS -intensive networking applications such as data browsing on the world wide web, Java applications, video conferencing, etc., there emerges an acute need for very high-bandwidth transport network facilities. Research and development with QoS in modern networks have matured considerably over the past few years, with a number of commercial QoS tools already available. A number of different experimental prototypes have been and are being deployed and tested in the U.S., Europe, and Japan, with significant support from their respective government agencies and telecommunication providers. There now exists a large and expanding interest on this topic to better understand the issues and challenges in designing such networks. It is anticipated that the next generation of the Internet will employ fully QoS supported edge nodes and backbones.

Many electrical engineering, computer engineering, and computer science programs around the world have started to offer a graduate course on this topic. That is, research and development QoS issues in communication networks have matured significantly to the extent that some of these principles are being moved from the research laboratories to the formal (graduate) classroom setting. However, there are no textbooks to fill this emerging void, except for those that deal some very small details. These observations led us to the decision to write a textbook on this topic. This book is designed to serve as a text for university students, who are interested in telecommunications, especially, Quality of Service (QoS), pricing issues, scheduling etc. As a backround we presume that the reader has a thorough understanding of basic calculus and telecommunications networking theory.

Much of the book's material is based on research that we have conducted over the years with our graduate students and research scientists visiting at our laboratory, and we would like to acknowledge them all. We would also like to acknowledge personally following individuals: Dr. Jian Zhang, Dr. Kari Luostarinen, Dr. Jarmo Siltanen, Dr. Kimmo Kaario, Dr. Isto Kannisto, and Dr. Ari Viinikainen.

ACRONYMS

AF	Assured Forwarding
AQM	Active Queue Management
ATM	Asynchronous Transfer Mode
BB	Bandwidth Broker
BE	Best Effort
CBQ	Class-Based Queuing
CBS	Committed Burst Size
CIR	Committed Information Rate
CL	Controlled Load
COPS	Common Open Policy Service protocol
CSFQ	Core-Stateless Fair Queuing
DB	Delay Bound
DiffServ	Differentiated Services
DRR	Deficit Round Robin
DSCP	DiffServ Codepoint
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
FCFS	First Come First Served
FTP	File Transfer Protocol
GIST	General Internet Signalling Transport
GPS	General Processor Sharing
GS	Guaranteed Service
HTTP	Hypertext Transfer Protocol
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IntServ	Integrated Services
IP	Internet Protocol
LE	Lower/Limited Effort
LLQ	Low Latency Queue
LP	Linear Programming
MPLS	Multi-protocol Label Switching
MRED	Multilevel RED

MTU	Maximum Transmission Unit
NMS	Network Management Station
NS-2	Network Simulator v2
PBS	Peak Burst Size
PDB	Per-Domain Behaviour
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHB	Per-Hop Behaviour
PIB	Policy Information Base
PIR	Peak Information Rate
PQ	Priority Queuing
QoS	Quality of Service
RED	Random Early Detection
RIO	RED with In and Out
RSVP	Resource Reservation Protocol
RTP	Real-time Protocol
SCFQ	Self-Clocked Fair Queuing
SCTP	Stream Control Transmission Protocol
SFQ	Stochastic Fair Queuing
SIP	Session Initiation Protocol
SLA	Service Level Agreement
SLS	Service Level Specification
SNMP	Simple Network Management Protocol
SPFQ	Start-Potential Fair Queuing
srTCM	Single-rate Three-Color Marker
STFQ	Start-Time Fair Queuing
TCP	Transmission Control Protocol
TSWTCM	Time Sliding Window Three-Color Marker
trTCM	Two-rate Three-Color Marker
UDP	User Datagram Protocol
VoD	Video on Demand
VoIP	Voice over IP
W^2FQ	Worst-case Weighted Fair Queuing
WFQ	Weighted Fair Queuing

WLAN	Wireless LAN
WRED	Weighted RED
WRR	Weighted Round Robin

LIST OF SYMBOLS

B	interface output bandwidth
B_i	per-class bandwidth
B_i^{max}	maximum per-class bandwidth
B_i^f	per-flow bandwidth
C_i	service price
D_i	queuing delay
D^{HW}	hardware delay
L_i	mean packet size
L_i^{max}	maximum packet size
L_i^{min}	minimum packet size
m	number of service classes
N_i	number of active flows
P_i	peak rate
Q_i	quantum value
r	instantaneous revenue
R	total revenue
U	set of classes with the bandwidth limitations
V_i	amount of transmitted data
w_i	weight
γ_i	allocation of additional bandwidth
ρ_i	long-term bounding rate
σ_i	burst size
$f(.)$	pricing function
λ	Lagrangian penalty term
$v_i(t)$	temporary value of the weight
t	time
μ	forgetting factor in the adaptive algorithm

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Resource allocation in the multiservice communication networks presents a very important problem in the design of the future multiclass IP networks. The main motivation for the research in this field lies in the necessity for structural changes in the way the Internet is designed. The current Internet offers a single class of best-effort service, although some traffic prioritization will be active in the new network router implementations. However, the Internet is changing and it is becoming an important channel for critical information and the fundamental technology for information systems of most advanced companies and organizations. Users are becoming increasingly reliant on the Internet for up-to-date personal, professional and business information. The substantial changes transforming the Internet from a communication and browsing infrastructure to a medium for conducting personal business and e-commerce are making the Quality of Service (QoS) an increasingly critical issue. Meanwhile, to realize the above changes, the future IP networks must be able to support a wide range of different traffic types with different QoS requirements. For example, new sophisticated real-time applications such as Voice over IP (VoIP), Video-on-Demand (VoD) and Video- Conferencing require firm performance guarantees from the network where certain resources should be reserved for them. Hence, efficient resource allocation mechanisms are needed to distribute network resources among all competing service classes for achieving their QoS requirements.

The current architecture of the Internet assumes the best-effort model which means "as much as possible as soon as possible". According to this definition, each packet has the same expectation of treatment as it transits a network. This has sufficed for traditional Internet applications, such as Web, E-mail, news groups etc. However, new multimedia applications and services create a strong impetus to bring different levels of service to packet traffic. The reason is that depending on a service, the traffic characteristics and, as a result, requirements for network transport functionality may vary. Some services may be long-lived telnet logins with a little traffic sent but needing interactive response. Some may be the File Transfer Protocol (FTP) sessions with bursts of kilobytes or even megabytes. Some may be the Hypertext Transfer Protocol (HTTP) transactions that open a

transport connection to transmit a handful of packets. Some may be audio or video streams with the fixed rates and no way to slow down. If the Internet handles all data in the same way, then it can result in unacceptable, if not completely unusable, service. For instance, if telephony and file transfer traffic are mixed together in the same part of the network, then the file transfer may experience congestions and packet losses due to the telephony applications that send data at the constant rate. In turn, the telephony traffic will not slow down, but will experience losses and poor voice quality due to the presence of the bursty file transfer operations. Thus, the enhanced Internet architecture should provide different service classes to indicate the treatment of individual packets and flows and to allocate resources among them.

These tasks are solved within the Quality-of-Service (QoS) research area. The objective of network QoS is to quantify the treatment a particular packet can expect. A workable QoS architecture must provide a means for specifying performance objectives for different classes of packets as well as means of delivering on those performance objectives. It should be noted that QoS cannot create additional bandwidth. When some packets get better treatment, other packets will get worse treatment. Thus, the task of QoS is to distribute resources in such a way that all the performance objectives are met.

The premise of QoS is that some traffic is more important and should be treated appropriately. Furthermore, there are also economical reasons because the Internet has become mission-critical to many companies. In such a framework a provider can offer services with specific performance with cost linked to quality. Further, as the QoS becomes available in the general Internet, more companies and end-users can migrate to the Internet increasing a provider's revenue while decreasing network purchasing costs.

From the viewpoint of a customer (either a user, or an organization, or another provider), the first step towards the QoS is the Service Level Agreement (SLA), which is negotiated with a provider. The SLA defines the QoS requirements, the anticipated load, actions to take if the load increases the negotiated value, pricing etc. Since the SLA includes rules and actions in the human readable form, it has to be translated into the machine readable representation. For these purposes, the SLA is partitioned into several documents. The Service Level Objectives (SLO) specifies metrics and operation information to enforce and monitor the SLA. The Service Level Specification (SLS) specifies the handling of a customer's traffic by a service provider.

To characterize the QoS requirements and actions in the SLS, a provider and a customer must specify them with a set of well-known parameters, or performance metrics, so that a provider can translate them into the router configuration. The fundamental parameters are throughput, delay, jitter, and packet loss.

1.1 QoS definitions

Quality of Service can be defined in various ways, most of which are equivalent or complimentary. It is the ability of a network element (e.g. a router/switch or Web server) to have some degree of assurance that its traffic and service requirements can be satisfied. It describes the assurance of sufficiently low delay and packet loss for certain types of applications or traffic. The provision of QoS in a network, especially one as large as the global Internet, is not a trivial matter. The cooperation of all network layers from top-to-bottom (i.e. layer one to layer seven of the ISO-OSI model) in addition to every network element from end-to-end (i.e. from sender to receiver) is required. The need for QoS is becoming progressively more evident daily. For example, many companies rely on the Internet for the day-to-day management of their global enterprises. Many more also utilize the Internet to conduct business (e.g. to place order with their suppliers, interact with customers, etc.). These companies are willing to invest a substantial amount of money for the best possible QoS from the Internet. After all, building and maintaining private high-speed global communication networks would represent a major capital investment for any multi-national corporation. On a potentially much larger scale, there are myriads of individual users and organizations that are willing to pay a higher Internet service fee for better QoS in order to take advantage of demanding applications like IP-telephony, video-conferencing and online games. Of course, there will always be a large constituency of users who want to pay nothing at all (excluding Internet service provider fees) for the more traditional services such as email and Web surfing. As is expected, applications differ in their QoS requirements. A loss-sensitive application is one that cannot accept the loss of data. For example, an ftp file transfer is loss-sensitive since every bit is important while a telephony application can handle the loss of an occasional packet (without retransmission). A delay-sensitive application is dependant on the timely arrival of its packets. File transfers are not delay sensitive although human patience places lower bounds on throughput. Multimedia applications like streaming video and audio can only handle small delays with predictable variation (jitter) and still maintain their utility. QoS has two components: performance assurance and service differentiation. The former relates to bandwidth, delay and jitter and packet loss. Bandwidth is the fundamental resource since it affects the other three. Service differentiation addresses providing different QoS to different applications with differing requirements. In both cases, this is also a way for Internet Service Providers (ISPs) to increase revenues. Absolute QoS vs Relative QoS

Performance assurance typically relies on absolute (quantitative) QoS specifications that in some way deal with loss or delay bounds. As an example consider the statements no packet will experience a delay greater than 200 ms or packet loss will not exceed 1 percent. In the absolute model, if the network can guarantee requested performance level of the user, he or she will be admitted access to the network. Typically the user will be rejected if the network can not provide

the requested assurances. It is an all or nothing proposition. This is typically between specific endpoints. Relative QoS is precisely what service differentiation addresses. The only assurance from the network is that a higher class will receive better or at least no worse service than any lower class. This model can not offer hard guarantees on delay or packet loss because the amount of service received by a class and the resulting QoS perceived by an application depend on the current network load in each class. This model also requires integration with a pricing or policy-based scheme to make higher classes more costly than lower classes. Disincentives (e.g. high cost) must exist so that everyone does not prefer to use the higher priority classes since this would effectively shrink the relative QoS differences between the classes. The existence of QoS classes supports the dynamic change of classes by a user or application that is able to actively adapt based on the observed performance. For example, a telephony application can dynamically switch classes to find an acceptable QoS at the most economical cost (i.e. no need to pay for a high priority class if a lower priority, and hence cheaper, class meets its requirements). Typically, this model supports arbitrary endpoints.

End-to-End QoS

The QoS protocols of IntServ and DiffServ [16] are capable of furnishing a QoS to flows or aggregates of flows in an end-to-end manner but it is unlikely that either one will ever be ubiquitous enough to do that. Instead, they will likely be used together in the real world to provide end-to-end QoS. By mixing and matching their capabilities in a variety of possible architectures, the goal of end-to-end QoS is nearing reality. Now at least MPLS (Multi-Protocol Label Switching [127]) and the combination of RSVP and DiffServ have been proposed to provide end-toend QoS. However, end-to-end QoS does not take place only between the network nodes of a network. The end hosts play an integral part in this. Most of the research on QoS is oriented towards network issues such as scheduling and routing. However, network QoS by itself is not sufficient to support end-to-end QoS because bandwidth management and congestion avoidance cannot resolve scheduling or bottleneck problems at the end hosts (Web servers). The FIFO scheduling performed by most servers can undermine any QoS improvements made by the network since a busyWeb server can indiscriminately drop high priority network packets. Thus, a true end-to-end Internet QoS solution has as an essential component aWeb server with QoS mechanisms to provide overload protection and to enable differentiated services. In addition, satisfying the end-to-end QoS requirement (delay in particular) of an application traversing a number of network elements is usually addressed by partitioning the end-to-end QoS requirement into the local QoS requirements in each individual element involved and then achieving them respectively. Hence, the problem of allocating resources among multiple service classes based on the local QoS requirements in a single network element is of practical importance.

1.1.1 QoS parameters

The fundamental QoS parameters can be defined as a set of parameters where other sets can always be mapped to. Following this definition, a sufficient set of QoS parameters include delay, throughput, jitter, packet loss ratio and availability.

The packet delay parameter is critical for the real-time applications. One can think of the packet delay as consisting of several subcomponents: the propagation delay, the serialization delay, and the queuing delay. The propagation delay is defined as the time it takes one bit of data to reach another end of a link. It depends only on the link characteristics and does not depend on the packet size. The serialization delay is the time it takes a router to output completely a packet. This delay depends on the link characteristics and the packet size. The queuing delay is the time a packet spends in a router's internal queue. The sum of all the propagation, serialization, and queuing delays along the datapath constitutes the end-to-end delay.

Throughput specifies the amount of bytes (or bits) that an application can send during a time unit without losses. It is one of the most important parameters because most applications include it in the set of their QoS requirements. It should be noted that the throughput stands for the long-term rate of an application. Due to the packet-based nature of most networks, the short-term rate may differ from the long-term value. Thus, it is usually the case that the throughput refers to the average rate of an application. Consequently, one can use other parameters such as the maximum or the rate and the minimum rate.

Jitter specifies the delay variation between the packets. It is an important parameter for the interactive applications, such as on-line audio and video conversations. Since data exchange between two applications involves sending a significant number of packets, it is often the case that jitter specifies the *maximum* delay variation observed between the two consecutive packets. However, one can also use a smoothing equation to obtain some mean value over the sequence of packets. Regardless of the interpretation, ideally the jitter should equal zero because the bigger its value is, the bigger buffer a receiving application must have to compensate delay variations between the packets.

Packet loss, as its name indicates, characterizes the number of packets dropped during transmission. This parameter is critical for those applications that perform guaranteed data delivery because every time a router drops a packet, a sending application has to retransmit it, which results in ineffective bandwidth usage. It is also important for some real-time applications since packet drops reduce the quality of transmitted video and/or audio data. Since the number of dropped packets depends on the duration of a session, the packet loss is expressed usually as a ratio of the number of the dropped packets to the overall number of packets.

Availability is usually directly related to the services of the network. It has a close relation to reliability, and there are lots of mechanisms to enhance the availability of the Internet, such as router redundancy protocols VRRP and HSRP.

Service Level Agreement

A Service-Level-Agreement (SLA), which is negotiated and committed between service providers and service consumers (either another service providers or common users or both), defines the QoS metrics for each class of service, the anticipated per-class workload intensity and the pricing strategy by which the service payment will be determined. The problem with SLAs is that the rules that are readable and understandable for human beings have to be translated into a form that is readable for machines. Differentiated Services framework suggests that the negotiated and committed SLA may be represented by Service Level Objectives (SLOs) for machine readability. As a summary of terminology related to SLAs, we refer to the definitions of RFC 3198 as follows [159]:

- Service Level Agreement (SLA): The documented result of a negotiation between a customer/consumer and a provider of a service, which specifies the levels of availability, serviceability, performance, operation or other attributes of the service.

- Service Level Objective (SLO): Partitions an SLA into individual metrics and operational information to enforce and/or monitor the SLA. SLO may be defined as part of an SLA, an SLS, or in a separate document. It is a set of parameters and their values. The actions of enforcing and reporting monitored compliance can be implemented as one or more policies.

- Service Level Specification (SLS): Specifies the handling of the traffic by a service provider. It is negotiated between a customer and a service provider. For instance, in a DiffServ network environment, it defines parameters such as specific Code Points and the Per-Hop-Behavior, profile characteristics and the treatment of the traffic for those Code Points. An SLS is a specific SLA and its SLOs (the individual metrics and operational data to enforce) guarantee the QoS required by traffic.

1.2 QoS mechanisms

The task of providing QoS spans many technologies and network architectural layers. A packet may leave a station via the Ethernet interface, travel through several network domains that use the Asynchronous Transfer Mode (ATM) technology, and finally reach the destination station that is connected to the Wireless LAN (WLAN) network. Such a diversity of the link layer technologies makes it complicated to deploy QoS. It is quite natural that each link layer technology may provide some QoS features that cannot be mapped easily to another link layer medium. Thus, it is easier to operate with QoS on the transport layer and then map its requirements onto the underlying link layer. Since at the moment the Internet Protocol (IP) protocol is de-facto the standard protocol to exchange data, most QoS mechanisms target the IP networks. At the same time, there are

standardization documents and research works on how to map IP level QoS to the concrete link layer medium.

The subsequent subsections present an overview of the most common QoS mechanisms used in the IP networks. Also, a good overview of the basic QoS mechanisms and interconnections between them is given in [164].

1.2.1 Packet classification and marking

To ensure the QoS requirements, the network has to identify somehow a flow, packets of which require a special treatment. The classical way to identify a flow is to use the following five parameters: source address, source port, destination address, destination port, protocol number. Such an approach requires that each router, which provides the QoS guarantees, uses the multi-field classifiers that can identify packets belonging to a particular flow. Though it is not a complicated task, the number of data structures a router has to keep grows proportionally to the number of flows. Furthermore, every time a packet arrives, a router has to find the matching set of values in the classifier which, of course, takes some time.

To diminish this overhead, the incoming data can be classified based on some *label* stored in a packet header. Such an approach reduces significantly the time it takes a router to find the matching entry in the classifier. Besides, the classifier data structures become considerably simpler. However, either the source applications or some router in the network must write the label into the packet header so that the packets are provided appropriate treatment. Depending on the QoS architecture, this task is solved differently.

1.2.2 Traffic regulation

Having made a contract with a customer, a provider should not anticipate that a customer application will send data at a rate that is smaller than or equals the value mentioned in the agreement. In fact, a user application is likely to send more data as a matter of normal operation. This is due to adaptive applications, packet level acknowledgements, and incapability of informing an application that it has to transmit data at some rate. As a result, a provider has to use traffic regulation models so that ill-behaving applications do not impact other flows.

In the traditional queuing theory, most models are based on stochastic processes. Among them the most popular are the Poisson model for data [89], on-off model [23] for voice sources and more complicated Markovian models [100] for video sources. In general, these models are either too simple to characterize the important properties of the source or too complex for tractable analysis. Furthermore, as reported in [120], the distribution of packets arrivals follows a self-similar law where the underlying distributions are heavy-tailed rather than the Poisson distribution.

In practice, other models are used that bound the traffic rather than characterize the process exactly. It is sufficient for the resource management algorithms

to know just the bounds to allocate resources in most cases. The simplest model that bound the traffic is the (r, T) model [55]. A traffic stream is said to satisfy the (r, T) model if no more than rT bits are transmitted on any interval of length T. Unfortunately, it is suitable only for the fluid traffic model which is not the case for the packet networks. Alternatively, a traffic stream satisfies the (σ, ρ) model [39] if during any interval of length T the number of bits in that interval is less than $\rho T + \sigma$.[1] Similarly, rather than using one bounding rate, it is possible to use a set of rate-interval pairs. The D-BIND [90] model captures an intuitive property that over longer interval lengths a source may be bounded by a lower rate. There are also other models that take parameters other than the bounding rate into account. In [48], the $(X_{min}, X_{ave}, I, S_{max})$ model was proposed. A traffic stream satisfies this model if the inter-arrival time between any two packets in the stream is more than X_{min}, the average packet inter-arrival time during any interval of length I is more than X_{ave}, and the maximum packet size is S_{max}.

Among presented models, the (σ, ρ) model, which is also referred to as Token Bucket, is widely used and supported by many manufacturers of the software and hardware routers. It is simple, tractable, and, at the same time, parameter σ allows to control flexibly the burstiness of a stream. What is more important, by knowing the value of σ it is possible to express the worst-case queuing delay for a given scheduling mechanism.

On the other hand, in the future multiclass Internet, each class of customers may have to pay network service providers for their received level of QoS based on the pricing strategy agreed upon in the Service-Level-Agreements between them. A Service-Level-Agreement (SLA) defines the QoS metrics for each class of service, the anticipated per-class workload intensity and the pricing strategy by which the service payment will be determined. Obviously, the pricing strategy will specify the relationship between the QoS level offered to each class of customers and the relevant price which should be paid by them, for example, the service provider will receive a certain amount of revenue from a class of customers if the offered QoS level is more than the minimal requirement of that class and suffer another certain amount of penalty for failing to meet that. Thus, from service providers' point of view, the optimal resource allocation scheme, which can achieve the maximization of SLA revenues under a given amount of network resources (e.g., bandwidth) and a given pricing strategy, is very desirable. The use pricing as a means for allocating resources in communication networks has received much attention in recent years. A smart charging method for network usage is presented in [96]. This paper studies individual packet bid for transporting while the network only serves packets which bid above a certain (congestion-dependent) cutoff amount. Charges that increase with either realized flow rate or with the share of the network resources consumed by a traffic flow is studied in [87, 88]. Packet-based pricing schemes (e.g. [54]) has also been proposed as an incentive for more efficient flow control. The fundamental problem of achieving the system optimum that maximizes the aggregate user utility using only the

[1] Some articles and standardization documents use Latin letters r and b instead of ρ and σ, especially in those cases when it is impossible to typeset the Greek letters.

information available at the end hosts is studied in [91]. They assume that the users are of elastic traffic and can adjust their rates based on their estimates of network congestion level. Pricing and link allocation for real-time traffic that requires strict QoS guarantees is studied e.g. in [117, 119]. Such QoS guarantees can often be translated into a preset resource amount that has to be allocated to a traffic flow at all links in its route through the network. If the resource is bandwidth, this resource amount can be some sort of effective bandwidth (see, e.g., [85] for a survey of effective bandwidth characterizations and [70] for similar notions in the multiclass case). In this setting, [86, 36] propose the pricing scheme of real-time traffic with QoS requirements in terms of its effective bandwidth. Their pricing scheme can also be called as a static one and it has clear implementation advantages: charges are predictable by end users, evolve in a slower time-scale than congestion phenomena, and no real-time mechanism is needed to communicate tariffs with the users.

1.2.3 Resource sharing

Packet classification and regulation do not necessarily mean that a provider allocates enough bandwidth resources for each flow. Though it is possible to share the output bandwidth by dropping packets that are out of a traffic profile, it is not efficient from the viewpoint of utilization. Thus, a provider has to use appropriate scheduling disciplines to share the output bandwidth. The choice of an appropriate service discipline is the key in providing QoS. Since this work aims to consider the adaptive scheduling, an exhaustive overview of schedulers is given in section 2.1. Also, a good overview of the existent scheduling disciplines is presented in [171] and [145]. We will mention briefly that there is no universal scheduler and the choice for the particular discipline is governed by the tradeoff between the accuracy and the computational complexity.

1.2.4 Congestion management

A provider should not anticipate that data transmission between user applications will occur at some constant rate. Instead, there can be silent periods and periods when the significant number of packets is sent. As a result, it is quite normal that the resulting rate of all user applications can be larger than the available bandwidth. Small rate fluctuations can be compensated by router buffers. However, if applications send data at larger rates continuously, then sooner or later buffer will be full. In such a case, the congestion occurs.

The straightforward behaviour of a router during congestion is to drop the packets. Though it is the simplest approach, it has several disadvantages. The first one is that several source applications receive at the same time a notification that a packet has been dropped which causes them to slow down data transmission. The second problem, as a consequence of the first one, is that source applications synchronize their transmission phases, which results in bandwidth oscillations.

To overcome these problems, the Random Early Dropping (RED) [51] algorithm was proposed. This technique starts to drop packets randomly as the mean queue size crosses the minimum threshold. The closer the mean queue size to the maximum threshold is, the more aggressively RED drops packets. Such an approach ensures that different applications receive notifications at different moments of time. Furthermore, instead of dropping a packet, RED can set a certain bit in the IP header to signal the sending application about the congestion. This technology is known as Explicit Congestion Notification (ECN) [123].

ATM networks use the modification of the dropping mechanism which is referred to as the Early Packet Discard (EPD) [126]. The effective throughput of protocols, such as TCP, over ATM can be quite low when cells are dropped at the congested ATM switch. This is due to wasted bandwidth as the congested link transmits cells packets in which at least one cell is dropped by the switch. EPD prevents fragmentation by dropping whole packets prior to buffer overflow.

There is another dropping strategy called Selective Packet Discard (SPD), which can be used with DropTail or RED. The purpose of SPD is to avoid dropping important packets that may belong to the network management or routing protocols. Practically it is accomplished by putting the important packets into the special priority queue from which a router drops packets only after it drops all packets from the user queue.

1.2.5 Signalling

While the network can be configured statically to ensure some level of QoS, a provider can achieve much better functioning if the network is informed about active flows and their requirements, and resources are allotted on demand. For these purposes, a provider uses the signalling protocols that belong either to the horizontal or vertical management. The horizontal signalling protocols carry the information between the routers and customer applications, and also between the routers. An example of the horizontal protocol is Resource Reservation Protocol (RSVP), Session Initiation Protocol (SIP), and General Internet Signalling Transport (GIST). The vertical signalling protocols are used by the centralized network management entities to configure routers. For instance, Common Open Policy Protocol (COPS) belongs to the vertical management. The Simple Network Management Protocol (SNMP) [28] can also be used in the vertical management, though its functionality allows merely to monitor the state of routers, not to configure them.

1.2.6 Routing and traffic management

Routing is one of the most important functions of the network because it allows a packet to reach a destination node based on its address. The network may function properly without many QoS mechanisms described earlier, however, the absence of the routing information will make the network unusable.

The traditional routing is based on the address of the destination node and assumes that the shortest path is selected. However, the shortest path is not al-

ways the best one. There may be a longer path that provides, for instance, smaller delay or has less congestions. In other words, the choice for the best path should not be done based only on the distance, but also on some preferences. For these purposes, each path may have associated QoS metrics that are analysed by the routing agents while they make a decision concerning the best path. In such a case, the best path does not mean the shortest, but the one that has the best value from the viewpoint of the used metrics.

Along with the QoS-routing, a provider may use the MPLS technology to manage traffic in complicated network environments where multiple paths exist. In particular, MPLS allows to create a trunk, i.e. a path through providers' networks, passing through the required set of links. By sending data packets to different trunks, the QoS guarantees can be provided.

1.3 Adaptive approaches for the resource allocation

The problem of efficient allocation of resources has gained recently significant attention, including the creation of new scheduling disciplines and a combination of the existent ones. Another direction is the dynamic adaptation of parameters to the varying network environments. Here, we present a brief overview of research works devoted to the adaptive allocation of resources in different queuing disciplines.

The problem of adjusting weights of the Weighted Round Robin (WRR) policy to support the premium service has been considered in [156]. It is proposed to allocate resources according to the dynamics of the average queue size that is calculated by using a low-pass filter. If the average queue size of the Expedited Forwarding (EF) aggregate increases, then the processing resources are taken from the Assured Forwarding (AF) and Best Effort (BE) aggregates, otherwise they are returned back to the AF class. However, it was not considered how the resources should be allocated for the AF aggregate to ensure all QoS guarantees.

A similar kind of algorithm, in which the state of the queues are used to adapt weights, has been proposed in [74]. It relies upon the WFQ policy and makes the dynamic assignment of weights based on the usage of queues. Unfortunately, this algorithm was considered only for the IntServ framework. It might be necessary to refine it for the DiffServ architecture, in which, for instance, short physical queues are built and, at the same time a significant amount of resources are allocated for the EF aggregate.

In [168], a modified WRR scheme, which is called Fair WRR, has been proposed to protect BE traffic from AF out-profile packets in the core routers. This policy adjusts dynamically the service weights and buffer allocations by using congestion hints in order to avoid unfair bandwidth sharing. However, the EF traffic aggregate was not considered and no recommendations were provided on how to reserve resources for this aggregate.

In [76], the Variable WRR policing model has been introduced that uses the average packet length to adapt weights in the WRR policy. A weight value, which

is based on bandwidth requirements, is corrected using the average length of packets. However, the simulation presented in that paper only includes the general case. Neither DiffServ architecture nor the settings to implement PHBs with the proposed policy were considered. Moreover, it is not clear how to choose a base value of a weight.

In [92], the effective bandwidth is used to adjust bandwidth allocation in the DiffServ framework. The proposed measurement-based adaptive scheme, which is referred to as Dynamic WRR, either increases or decreases the bandwidth based on the values of the estimated bandwidth, multiplexing gain factor, and the measured loss ratios. The proposed scheme might work well for the AF traffic aggregates, but it fails to take into account the delay requirements of the EF class. Besides, the article does not clarify how bandwidth allocations are translated into the weight values of the WRR scheduler.

In [80], the general approach for using revenue as a means to adapt weights of the WFQ scheduler has been presented. It was proposed to use the pricing function and the delay requirements to share adaptively resources between several service classes.

In [99], a configuration scheme has been considered, which guarantees maximum revenue for the service provider while keeping utilization high. The proposed scheme, which is based on the WFQ scheduler, selects those traffic flows that maximize the benefit. However, the study has focused only on the best effort service class. Neither EF nor the other behaviour aggregates have been considered.

The objective of the TEQUILA project is to study, specify, implement and validate a set of service definition and traffic engineering tools to obtain quantitative end-to-end QoS guarantees through careful planning, dimensioning and dynamic control of scalable and simple qualitative traffic management techniques within the Internet. The project has five key objectives: a) study the issues behind, develop architectures for, and propose algorithms and protocols to enable: negotiation, monitoring and enforcement of SLS between service providers and their customers and between peer providers in the Internet, b) develop a functional model of co-operating components, related algorithms and mechanisms to offer a complete solution for intra-domain traffic engineering to meet contracted SLSs in a cost effective manner, c) develop a scalable approach, architecture and set of protocols for interdomain SLS negotiation and QoS-based routing to enforce end-to-end quality across the Internet, d) validate the above through both simulation and/or testbed experimentation, and e) use, enhance and contribute to drafts, specifications and standards of the wider international community.

The COST 290 project (Wi-QoST: traffic engineering and QoS management in wireless multimedia networks) aims to increase the knowledge on future advanced multiservice wireless networks and specifically on traffic nature and behaviour and its impact on network architecture, performance and planning. The project gives a special attention to the QoS and related aspects in both access networks and core networks in the presence of mixed multimedia traffic. To accomplish this, new analytical tools, software implementations and prototypes

are developed and validated. As the TEQUILA project, this project aims to contribute to standardization bodies on the basis of the obtained achievements by creating standards inputs leading to well supported decisions. Furthermore, it tries to coordinate the research relations, and foster exchange and networking of researchers, between European participating organizations and research groups being active in the field.

1.4 Outline of the book

The rest of the book is organized as follows. Chapter 2 presents the analytical background for the proposed adaptive models. The major scheduling disciplines used in the telecommunication networks are considered, and their advantages and disadvantages are analysed. For each chosen scheduling discipline, an analytical adaptive model is presented. This chapter also presents the preliminary results concerning the efficiency of each adaptive model based on the analytical expressions.

Chapter 3 continues with analytical solutions in of upper delay bound of GPS- based packetized fair queuing algorithms. Approach of deriving the suboptimal resource allocation scheme in a GPS-based network node is effective and the derived resource allocation scheme can achieve the highest SLA revenue under a given amount of network resources and flat pricing strategy.

Chapter 4 presents the major QoS frameworks proposed by the IETF and provides explanations on how the adaptive models can be integrated into these frameworks. This chapter considers issues, such as the interaction with the signalling protocols, the availability of the QoS information, and the integration with the forwarding mechanisms.

Chapter 5 provides practical results. Each adaptive model is tested in various environments. First, adaptive models are compared to disciplines with a static configuration within a simple environment without any particular QoS framework and signalling mechanisms. Then, a simulation case for the Integrated Services frameworks is considered. Finally, this chapter presents simulation results for a hybrid framework, in which signalling mechanisms from the Integrated Services are combined with the forwarding mechanisms of the Differentiated Services.

Chapter 6 focuses on the delay performance of different service class and *mean packet delay* and *packet delay* are chosen as the QoS metric in a SLA. When the *mean packet delay* is used as the QoS parameter in a SLA, the measurement period of packet delays should also be specified in the SLA so that the SLA revenues can be collected periodically based on the deployed pricing strategies and the periodical QoS performance measurements (in this case, the *mean packet delay*). Whereas, with *packet delay* as the QoS metric, the SLA revenues are collected based on the used pricing strategies and the delay of each inbound packet.

Chapter 7 presents the problem of maximizing the revenue attained in the

hosting of an e-commerce site with a SLA contract under a given amount of server resources by optimally partitioning the server resources among all the supported classes. A Web server farm is typically deployed to host several Web sites simultaneously on the same platform.

Chapter 8 summarizes contents of the book and draws final conclusions concerning presented models based on analytical and simulation results. This chapter also shows some future research directions.

2 ADAPTIVE SCHEDULING

2.1 Scheduling disciplines

The choice of an appropriate scheduling discipline is the key to providing QoS in the packet networks. In the simplest case, a provider has to share the output bandwidth equally between all data flows so that each flow obtains a fair portion of bandwidth resources. Such an approach assumes that all the flows have the same requirements. However, due to the diversity of the existent applications, a data flow may require a certain minimum amount of the output bandwidth. It leads to the unequal bandwidth allocation between the flows. Thus, a provider has to use appropriate disciplines to ensure requirements of all the data flows. Though it is possible to share resources using the buffer management mechanisms [65], it is not as efficient as the usage of schedulers or the combination of both approaches.

A service discipline can be classified as either work-conserving or non-work-conserving [171]. With a *work-conserving* discipline, a scheduler is never idle when there is a packet to send. With a *non-work-conserving* discipline, each packet is assigned an eligible time. If no packet is eligible, none will be transmitted even when a scheduler is idle. Both these classes have drawbacks and advantages. The work-conserving disciplines are suitable for those environments, in which applications can adjust their transmission rates and can react to the packet losses. Indeed, if there are available bandwidth resources, the application can start to send more data achieving the higher throughput and the better network utilization. If later a congestion occurs, some number of packets will be dropped which will signal the sending application to slow down data transmission. Though the work-conserving disciplines allow to utilize completely output bandwidth, they use to change the traffic profile as packets move from one node to another. As a result, the provision of the end-to-end guarantees, and especially the delay guarantees, becomes an art that involves many network management solutions. As opposed to this, the non-work-conserving disciplines do not change the traffic profile, which simplifies the provision of the end-to-end guarantees. The non-

work conserving disciplines eliminate traffic distortions by delaying packets and, thus, preserving the traffic profile. However, as mentioned above, some bandwidth resources remain unused.

It is often the case that a provider has to support several service types ranging from best-effort data to applications with the tight bandwidth and delay requirements. Thus, it is not so easy to choose which class of disciplines must be used. Unfortunately, it is quite difficult to use both two classes of scheduling disciplines along the same forwarding path because they will diminish effect of each other. Thus, a provider has to decide which class is of bigger importance. At the moment, most manufactures of the telecommunication equipment rely upon the work-conserving disciplines. The main reason is that they allow to utilize bandwidth resources efficiently. Furthermore, two QoS frameworks proposed by the IETF rely upon the work-conserving schedulers. Hence, we will also focus on this class of scheduling disciplines.

It must be noted that along with the domination of work-conserving disciplines there are many solutions taken from the non-work-conserving schedulers. For instance, it is possible to set rate limits [173]. Another solution is various traffic shapers that delay packets to cope with traffic distortions. For instance, if an upstream provider wants to ensure that the traffic sent to the downstream network is within the negotiated profile, then packets can be delayed at the egress routers.

2.1.1 Work-conserving disciplines

Basic work-conserving scheduling disciplines can be classified into several major groups: first-come-first-served, the priority queuing, the round-rodin queuing, and the fair queuing. There are also complex schedulers that combine several simpler ones. They are considered in section 2.1.3.

First-come-first-served (FCFS) determines the service order of packets strictly based on their arrival order. In other words, packets are output exactly in the same order as they arrive. It is the simplest queuing discipline that is used in routers that do not have to perform a special treatment of packets. Furthermore, if there is a router that performs only retransmission functions, or the output bandwidth is higher than the bandwidth of the input links, then FCFS is an ideal solution. However, if a router has the bottleneck link, then this discipline cannot perform the necessary bandwidth allocation and provide the required QoS because all packets are treated equally. However, it must be noted that FCFS can provide implicitly certain QoS guarantees. For instance, knowing the output bandwidth and the maximum queue size, it is possible to calculate the worst-case queuing delay in a router. Thus, by setting the maximum queue size to the certain value at all intermediate routers, one can achieve the worst-case end-to-end delay. It is quite a simple and efficient solution for those networks that transmit only one class of packets, which requires some delay guarantees. Of course, this solution does not work well in the multi-service networks, in which data from various applications are sent over the same links.

Priority Queuing (PQ) [89] was aimed to overcome some problems of FCFS that treats all the incoming packets equally. PQ works according to a simple scheme - packets from the lower-priority queue can be transmitted only if the higher-priority queue is empty. In other words, PQ absolutely prefers queues with higher priority and, therefore, packets of a higher priority queue are always served first. Such a behaviour leads to very small configuration and management efforts. To achieve better treatment of some packets, it is enough to associate them with the higher priority. Unfortunately, it works only for simple cases when there are a few queues. The problem is that by putting the TCP-based applications to the higher priority we take the risk of not providing bandwidth for the low priority queues. If the higher priority queue is always full, then the lower priority queues are never served. Thus, the most typical configuration is to put all the critical UDP-based traffic into the higher priority queue, and the remaining traffic, including the non-critical UDP applications, into the lower priority queue. To solve the problem with the greedy TCP applications, a provider can assign rate limits [172].[1] Such a scheme will ensure that the higher priority traffic will not occupy all the available bandwidth. However, it makes the overall network management more complicated because it will be necessary to change the rate limits depending on the bandwidth requirements within each service queue. There is another problem reported in [49]. PQ leads to the increased burstiness which may cause heavy packet loss. Thus, PQ is the superior discipline for a simple case, but its behaviour becomes unpredictable when there are multiple flows with diverse requirements.

The simplest solution for the controlled bandwidth sharing is to use the Round Robin (RR) scheduler. It serves the input queues in a round-robin manner and outputs one packet from each queue. Of course, such a scheme cannot provide any proper QoS requirements. However, RR has several interesting features. First, it can provide the fairness between data flows. By associating each flow with a queue, it is possible to share the bandwidth fairly between the data flows.[2] Secondly, the RR scheduler can interleave packets to decrease the resulting burstiness.

The Weighted Round Robin (WRR) discipline [84] is an enhanced version of the RR scheduler. Like in RR, queues are served in the round-robin manner, but unlike in RR, each queue is assigned a weight that determines the number of packets to be transmitted. This scheme is very simple in implementation and, as a result, is used in many hardware routers. It allows a high-speed router to share the output bandwidth not wasting time on making a decision about which packet should be transmitted next. However, the WRR scheduler does not take the packet size into account. Thus, if some queue has a bigger average packet size than the other one, the former receives more bandwidth implicitly. It is not a problem for those environments that have the constant packet size, such as the ATM networks. But if the packet size varies significantly, then it is quite difficult

---

1 Authors of this scheduler referred to it as Rate-Controlled Static Priority (RCSP).

2 Practically, it is achieved by the Stochastic Fair Queuing (SFQ) discipline that uses a hash function to map an incoming packet to one of the RR queues.

to predict the resulting bandwidth allocation. It is exactly for these reasons that it is quite difficult to use WRR to provide the delay guarantees. If we do not know how much time the WRR scheduler will spend serving each queue, we cannot allocate enough resources for the delay-critical one.

To overcome the problem of the WRR discipline, Deficit Round Robin (DRR) [143] has been proposed. Like WRR, it works in the round-robin way serving consequently the input queues. Thus, it has the same computational complexity. The difference between WRR and DRR is that the DRR scheduler associates a so-called deficit counter with each queue. The deficit counter represents the number of bytes a scheduler is allowed to output from a queue before it starts to serve the next one. Every time the scheduler sends a packet, it decrements the packet size from the deficit counter. If the next packet to be transmitted has the bigger size than the current value of the deficit counter, the scheduler starts to serve the next queue.[3] It enables to share accurately the output bandwidth between several queues. At the end of a round, the deficit counter of each queue is updated with the associated quantum value. Thus, by setting different quantum values for queues, different bandwidth allocation is achieved. Though DRR takes the packet size into account, it cannot achieve high accuracy in sharing the output bandwidth. Another problem is that due to the round-robin nature it cannot provide tight delay guarantees.

These disadvantages were overcome with fair queuing (FQ). For sure, the best known and studied technique among the FQ schedulers is Weighted Fair Queuing (WFQ) [44]. It schedules packets according to their arrival time, size, and the associated weight. Every time a packet arrives at a router, WFQ calculates its virtual finishing time and adds it to the sorted list of packets. The packet with the least virtual finishing time is output first. It enables to share the output bandwidth with a high accuracy. Furthermore, it is possible to express the worst-case queuing delay of a packet. The price for these advantages is the increased load on a router that has to put a newly arrived packet to the sorted list based on its virtual finishing time. Also, a router has to keep track of the active sessions. Thus, the WFQ discipline is used in routers that have relatively slow output links. In this case, accurate bandwidth sharing is of big importance and a router has time for the scheduling decision. High-speed routers rely upon the simpler scheduling disciplines implemented at the hardware level [19]. It bear mention that WFQ cannot decouple the bandwidth and delay guarantees. In other words, WFQ cannot allocate resources efficiently for those services, that have small bandwidth but very tight delay requirements. To ensure the delay guarantees, WFQ has to allocate bigger bandwidth. If a flow is allocated a small portion of the output bandwidth, it is likely that its packets will experience considerable queuing delays. It must be noted that the schedulers based on the round-robin scheme provide better delay characteristics for each service class.

There are many variants of WFQ. Most of them target the problem of the computational burden that WFQ puts on a router while inserting a packet into the sorted list and updating a list of active sessions. Self-Clocked Fair Queuing (SCFQ) [56] uses the evolution of the virtual start time of the packet currently be-

[3] This is a scheme proposed by authors of the DRR scheduler. It is a little bit inefficient as it requires the scheduler to peek the next packet's size to decide whether to transmit it or not. To reduce this small overhead, the practical implementation behaves a little bit differently. The scheduler just outputs packets while the deficit counter has the non-negative value.

ing in service. It reduces greatly the amount of computational need. The price for it is a larger worst-case delay estimation that depends on the number of sessions. The fact the the queuing delay, and, as a result, the end-to-end delay, depends on the number of sessions is the main obstacle for using this discipline. Start-time Fair Queuing (STFQ) [60] is very similar to SCFQ with the main difference being that the STFQ scheduler outputs a packet with the smallest virtual start time, as opposed to the smallest virtual finish time. Frame-based Fair Queuing (FFQ) [147] is another scheduling discipline that provides the same delay guarantees as WFQ, but has the less computational complexity. Another interesting variant called Worst-case Weighted Fair Queueing (W^2FQ) [8] aims to achieve better emulation of the fair queuing.

2.1.2 Per-flow versus per-class queuing

While providing the per-flow QoS guarantees, a provider has two options for the configuration of the scheduler. The first one is to associate each flow's traffic with a separate queue. Since each flow has a separate queue, it is isolated from other flows that may start to send data at higher data rates. If so, their queues will overflow and a certain number of packets will be dropped while the well-behaved flow will not be punished. The task of the scheduler in this case is to provide a certain minimum amount of the output bandwidth for each queue, i.e. for each flow. Though it is the most accurate solution, it works well only if the number of flows is not great. Otherwise, a router will have to use a huge amount of resources to support a classifier entry and a queue for each flow. Furthermore, when a packet arrives at the router, the latter will spend some time finding an appropriate queue. From the viewpoint of the scheduler, the bandwidth sharing between a huge number of queues is also a challenging task. Only FQ schedulers are capable of accurate bandwidth sharing, while the RR schedulers, which rely upon the integer weight values, may fail to share the bandwidth at all.

As a result, the prevailing method for the bandwidth sharing between a huge number of flows is to group flows with identical or similar requirements into a class and associate a class with a queue. It is often the case that there are only a few classes. Thus, classifiers are very simple, lookup procedure works fast, and the scheduler is not overburdened with a huge number of sessions. However, if there is an ill-behaved flow, then it can influence easily other flows as they share the same bandwidth allocated by the scheduler. Thus, the task of isolation of data flows becomes quite important. It can be solved by using several techniques or by combining some of them. One of the most powerful and well-proven solutions is to use Stochastic Fair Queuing (SFQ) [104] that approximates equal bandwidth sharing between the data flows. The SFQ scheduler organizes several internal queues, which are served in the round-robin order, and a hash function that maps each incoming packet to one of the queues. However, SFQ cannot be combined with the FQ schedulers. Another solution is to use the per-flow buffer management by organizing the virtual queues [65]. Though all flows belonging to the same class share the same physical queue, each flow may have

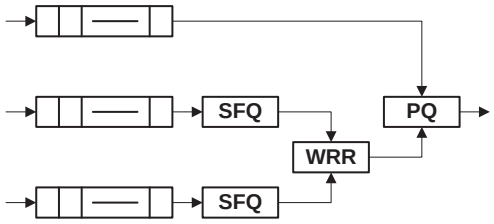


FIGURE 1 Combination of schedulers.

a packet counter. Thus, if the counter of a flow reaches the threshold value, packets belonging to this flow will be dropped. Unfortunately, this solution requires a router to keep the per-flow information which is not efficient in the backbone networks. In [150], authors have proposed an architecture for the core routers that allows to share bandwidth between the data flows without maintaining the per-flow states.

It must be noted that a provider does not have to deploy the same resource allocation scheme in all routers. While routers in the backbone network may rely upon the per-class queuing, routers in the access networks may use the per-flow classification and scheduling [67]. Such an approach, which is used in many QoS frameworks, enables to find the tradeoff between the high accuracy and the fast data transmission.

2.1.3 Complex schedulers

Combination of schedulers

One can construct a complex scheduler simply by combining several simple disciplines. In fact, most software-based routers allow to do so, which is not the case for most hardware-based routers. The reason is that the possibility to combine the schedulers requires the more complex kernel data structures that, in turn, can slow down data transmission. However, the hardware routers use to implement some well-proven combinations of schedulers [154].

Instead of considering all the possible combinations of schedulers, a simple example will be given. Fig. 1 illustrates the combination of the PQ, WRR, and SFQ schedulers. There is the PQ scheduler at the top level that decides from which queue a packet must be taken. If the first queue is empty, then the control is passed to the WRR scheduler that selects, based on the saved state, a queue to serve. Note that the second and the third queues have the SFQ scheduler attached, which provides the fairness between data flows within a particular queue. This is a powerful combination that allows to associate the delay-critical data with the first queue and to share bandwidth between the non-critical queues by using WRR. Note that other schedulers, such as DRR and WFQ, may be used in place of WRR.

It is worth mentioning that it is quite difficult to provide analytical estimations for all the QoS metrics for any combination of schedulers. As a result, if a

provider cannot predict the behaviour of the resulting scheduler, he cannot set it up properly to meet all the QoS requirements. Referring back to Fig. 1, if the first queue is always busy, then the WRR scheduler will not get any chance to transmit data from the second and third queue.

Hierarchical link sharing

The basic scheduling disciplines allow to share bandwidth to support different service classes. Alternatively, a link may be shared by several different organizations or departments. Each of these may require further partitioning into some classes. At the same time, unutilized link resources may be allocated for those organizations and departments that are willing to transfer data. Thus, the hierarchical structure of organizations and services suggests a hierarchical allocation of bandwidth. Schedulers, which follow this scheme, are often referred to as the hierarchical scheduling. They aim to perform two major tasks:

- allocation of the necessary bandwidth resources to support the QoS guarantees
- distribution of free resources in the controlled and predictable way

It must be noted that the basic scheduling disciplines fail to perform the second task because the available bandwidth is distributed between the existent classes according to their weight values. In other words, it is impossible to allocate all free resources to one particular class. From this point of view, the hierarchical link sharing schedulers are quite attractive for the network providers since they allow to control the allocation of free bandwidth.

The simplest implementation of the hierarchical link sharing is Hierarchical Round Robin (HRR). The scheduler cycles through slots at the top level. If a slot corresponds to a connection, one packet is transmitted. If the scheduler cycles through a slot that has further hierarchy, it will service a certain slot at the lower level. The original idea of HRR was to make it non-conserving in the sense that if HRR cycles through a slot with no packets waiting, the scheduler will leave the server idle. It is understandable that it can be turned into the work-conserving scheduler easily. HRR cannot provide the required QoS guarantees as it aims to achieve fairness between slots at each level. However, by associating weights with slots at each level, more complicated resource allocation can be achieved.

One of the best known hierarchical link sharing algorithms is Class-Based Queuing (CBQ) [52]. The goals of CBQ are to guarantee roughly the bandwidth and to control which classes may borrow bandwidth from other ones. There is no requirement concerning the scheduling disciplines but the practical implementation relies upon the combination of PQ and WRR. The fair queuing can be used in a hierarchical manner as well. At the top level, the weight reflects the link sharing requirements, while the lower weights provide QoS guarantees for individual classes. This implementation is known as Hierarchical Packet Fair Queuing (HPFQ) [9].

The biggest disadvantage of the hierarchical link sharing algorithms is the computational burden. To detect that a certain class does not consume allocated resources, it is necessary to measure the mean bandwidth all the time. Based on this information and the borrowing properties, the scheduler decides to which class these resources must be allocated. Thus, it is often the case that only software-based routers provide the implementation of these algorithms, while the high-speed hardware routers rely upon the simpler mechanisms. The complexity of the hierarchical link sharing algorithms raises another problem. Unlike the simple scheduling disciplines that have one configuration parameter per queue, these disciplines have many parameters. It makes them more complicated in management. Failing to provide all the parameters with correct values may lead to an incorrect bandwidth sharing.

2.2 Revenue-based adaptation

Regardless of the chosen scheduling discipline, a provider has to configure routers so that all the QoS guarantees are ensured. One solution is to use a static configuration. Though such an approach does not require significant management efforts, it is understandable that a provider has to *overprovide* its network with resources to meet all the QoS requirements, regardless of the current number of active flows. It results in an inefficient allocation of resources. Furthermore, if a provider has allocated bandwidth statically for ten flows, the eleventh flow will impact the provision of the QoS guarantees for all the flows within a class. Thus, a better solution is to track the number of active data flows and to allocate resources on demand. Though it requires the presence of additional management solutions and signalling protocols, the result is the more efficient allocation of resources and the ensured QoS guarantees.

If a provider tracks the number of active flows, he can always determine the minimum amount of bandwidth to be allotted. However, if there are free bandwidth resources, then there are several ways on how to allocate them. The most straightforward solution is to distribute free bandwidth equally between the active flows with regard to their bandwidth requirements. In fact, most work-conserving disciplines provide implicitly this behaviour. However, a provider may allocate free resources to the delay critical flows to minimize possible delays and jitter. On another hand, it is also possible to allocate free resources to the best-effort flows to reduce the packet loss. Thus, depending on the used criterion, different bandwidth allocation can be used.

Among the existent criteria, it is worth noting the following ones: the mean packet size [76], queue size [156], and packet loss [168]. We propose to use the prices of network services as the main criterion to allocate free resources. Along with the BE service available for all users, a provider may implement additional services with better and/or strict QoS guarantees. As considered in [35], if there are several service classes, then some of form of service class sensitive pricing is

required to attain the desired level of performance. It is reasonable to anticipate that customized services attract an increment on the service tariff because the provision of a distinguished service is undertaken with some level of additional network resources to support the service, and the tariff reflects this altered resource allocation. The difference in the charge between different service classes will depend on the difference in performance between the classes. In such a framework, it is intuitively understandable that instead of allocating free resources for the BE users, who pay only a fixed fee or even do not pay anything at all, it is worthy of providing more resources to those users, who are willing to pay for the better service.

There are several charging models that are used in pricing the Internet services [124], among which the *access-dependent* and the *volume-dependent* charging are the most popular. The access-dependent charging is usually one of two types: allowing unlimited use, or allowing limited duration of connections, and charging per connection time. The first type is known as the *flat* charging. It implies that a customer pays only the joining fee and has an access to the network resources which is limited only by the available bandwidth resources. It is the commonly adopted model in both the public Internet and within the corporate networks where the client service tariff is based on the characteristics of access to the service, rather than that of the actual use of the service. Advantages of the flat charging [105] are a) predictable monthly charges, b) no overhead costs for counting packets and preparing the usage-based reports, and c) absence of problems when some network services, such as the mail system and network file servers, generate traffic on behalf of the actual user. The time-dependent charging is widely used in the telephone and the mobile networks, but it is not generally used in the Internet.

The introduction of QoS services creates a strong impetus to move to the usage-based tariffs [75]. Experiments have shown that the usage pricing is a fair way to charge customers and to allocate network resources [3]. Internet services usually use the volume-based rather than the time-based charging because the former reflects duration of a connection and access speeds. Furthermore, the time-based charging works only when resource demands per time unit are roughly uniform, which is not the case for the Internet applications. It is also worth mentioning that even telecommunication providers use to apply the usage-based charging when ordinary data is transmitted. Thus, this research will use the usage pricing, which is based on the amount of resources used or reserved by a customer.

The price for the volume unit can be fixed or it can depend on parameters, such as the time of the day, congestion level, and provided bandwidth. Prices can also be affected by the regulatory environment and the cost structure of the relevant technologies. In this research, we assume that prices are almost constant, i.e. they change vary rarely comparing to the transmission time of one packet or do not change at all.

2.3 Adaptive models

2.3.1 Choice for the scheduler

As discussed previously, the scheduler is one of the most important components and is the basis for allocating resources. The choice of an appropriate service discipline is the key in providing QoS. Thus, it is crucial to choose the scheduler that will enable a provider to perform the adaptive resource allocation based on the QoS requirements and prices for the network services.

Among the scheduling disciplines presented in section 2.1, we are not going to consider the hierarchical schedulers. The main reason is the absence of analytical expressions for the major QoS parameters and the computational burden they put on a router. Furthermore, most of them are just a sophisticated combination of simpler disciplines. As a result, it is better to rely upon the simpler and faster schedulers omitting an additional "layer" introduced by the hierarchical structures.

However, every basic scheduler has its advantages and drawbacks. For instance, the FQ disciplines provide high accuracy at the expense of the increased computational load. At the same time, the RR schedulers are simple in implementation but are not capable of providing tight delay guarantees. Since there is no universal solution, we will consider the WFQ, WRR, and DRR schedulers as representatives of the FQ and RR resource allocation approaches. The reason we do not consider PQ is that it does not scale well in the multiservice networks.

2.3.2 Resource allocation

As considered earlier, by allocating resources on the per-class basis, it is possible to build scalable and simple in management solutions that depend on the number of classes, rather than on the number of flows. Indeed, when a flow appears and disappears, it may cause changes in the configuration of a service class, but it does not cause routers to add or delete a service class and associated control structures. The anticipated number of service classes will not be large. It is likely that there will be a service class for the delay critical applications, such as Voice-over-IP (VoIP), a service class for the mission-critical applications that require significant bandwidth requirements, a service class for those users who want better than BE treatment, and a class for the BE data. Another reason to introduce a small number of well-defined services is to achieve end-to-end service coherency when spanning multiple network domains [27]. A small set of distinguished services can be supported across a large set of service providers by equipment vendors and application designers.

Since resources are going to be allocated on the per-class basis, it is important to estimate the resulting QoS requirements of a class. The point is that a provider allocates resources for the traffic classes, which exist only within a provider’s domain and are visible neither to the customers nor to other providers.

At the same time, user requirements are applied to an individual flow, not to the class. Thus, it is necessary to translate individual QoS requirements into the class parameters so that a provider can allocate enough resources.

Among all the QoS parameters, we are going to consider bandwidth and delay. The reason is that it is a challenging task to provide jitter by using the work-conserving schedulers because they use to change the traffic profile. Furthermore, it is even a more difficult task when jitter is decoupled from delay, i.e. when we have to ensure the relatively big delay and a small jitter. Since jitter is a difference between two consecutive delays, it is simpler to limit the maximum delay for packets. In other words, jitter can be ensured implicitly by the delay guarantees. The packet loss can be avoided completely for the UDP flows simply by providing enough bandwidth and buffer resources. Buffers will compensate small peaks of the input traffic, while the scheduler will output packets at some constant rate. However, it is almost impossible to avoid packet loss for the TCP flows since packet drops use to control TCP throughput. They act as a signal for the TCP sender that it has to slow down data transmission. However, there are technologies, such as ECN, that can assist in reducing the amount of packet drops. Thus, the zero-packet loss for the well-behaving flows can be achieved by providing enough buffer and bandwidth resources.

In the case of bandwidth, the resulting class bandwidth can be approximated by a sum of the bandwidth requirements of all data flows. It should be the responsibility of a router to allocate this bandwidth fairly between flows. To provide the delay guarantees, it is necessary to estimate the resulting burst size when multiple data flows are aggregated. The worst-case estimation is a sum of all the burst sizes. However, in practice, it is true if and only if all the flows start to send data simultaneously so that packets from different data flows arrive at the router at the same moments of time. Furthermore, such a situation is possible only when each flow has a dedicated link to the router. Since in practice flows start to transmit data at different moments of time and several data flows share the same link, the resulting burst size is smaller. There are several research works which aimed to give more resources conserving estimation of the burst size, but most of them rely upon the strong assumptions, such as the synchronization unit [158]. Another reason for using the worst-case estimation is that any other estimation cannot guarantee the strict delay bound, but rather leads to the statistical delay bounds.

2.3.3 Pricing criterion

To charge customers, a provider uses the *pricing function* that establishes the cost for the resource usage. In this research we are going to consider a simple usage-based pricing function that depends on the fixed price and the amount of the transferred data:

$$R(t) = \sum_i V_i(t) C_i \,. \tag{1}$$

Here, $V_i(t)$ specifies the amount of data sent within the ith class by the time t, and C_i is the price measured in monetary units per one unit of data. It should be noted that (1) does not preclude a provider from using more complicated functions. A provider can use any usage-based function that may include additional components, such as constant monthly fee. Furthermore, the price can be constant or it can depend on parameters, such as time, congestion level etc.

As presented in (1), the total revenue is the amount of the transferred data times the price for one data unit. Unfortunately, such an interpretation does not allow us to maximize it. However, one can think of the total revenue R as a sum of instantaneous revenues obtained during short periods of time. Based on this, we can rewrite (1) in the following form:

$$R(t) = \sum_i \int_0^t r_i\big(C_i, w_i(t)\big)\,dt = \int_0^t \left(\sum_i r_i\big(C_i, w_i(t)\big)\right) dt\,. \tag{2}$$

Here, r_i is the instantaneous revenue of the ith class, which is the function of price C_i and the parameter of the scheduler $w_i(t)$. Since the parameters of the scheduler control the amount of transferred data, they have a direct relationship to the instantaneous revenue. Thus, by manipulating the parameters w_i, different instantaneous revenues can be obtained. Since the task of a provider is to maximize the total revenue, it can be achieved by maximizing r_i:

$$\max\left\{\sum_i r_i\big(C_i, w_i\big)\right\}. \tag{3}$$

Since expression (3) presents the instantaneous revenue at the moment of time t, then it is possible to assume that $w_i(t) = w_i$. It must be noted that (3) is a subject to the QoS constraints. Indeed, a provider cannot allocate infinitely resources for some service class as it violates the QoS requirements of another class. It also interesting to note that (3) does not depend on the number of active flows. If a provider has a switching equipment with a certain capacity, then it does not matter how many flows there are. The more data streams compete for the available resources, the less bandwidth each stream has. However, the total amount of data, capable of being transferred over a period of time, remains the same (which is true if all flows send data continuously and use all resources).

Though (1)-(3) assume that prices are associated with service classes, the actual charging is done on the per-customer (or on the per-flow) basis. It does not change the general considerations because $V_i(t)$ is just a sum over data sent by each customer within the ith service class. However, the per-customer charging requires the presence of a node that can identify a particular flow and track the amount of transferred data.

The subsequent sections present the basic scheduling disciplines, such as WFQ, WRR, and DRR, and the QoS constraints that these disciplines impose on the revenue-based adaptive resource allocation.

2.3.4 Weighted Fair Queuing

WFQ scheduler

WFQ schedules packets according to their arrival time, size, and the associated weight. Upon arrival of a new packet, the virtual finishing time is calculated and the packet is scheduled for departure in the right order with respect to the other packets. Then, WFQ outputs packets in the ascending order of the virtual finishing time. Such an approach enables the sharing of resources between service classes in a fair and predictable way. Furthermore, it is possible to estimate the bandwidth allocation and the worst-case delay performance, which makes the use of the WFQ discipline very attractive for the provision of QoS, and especially for the provision for the end-to-end guarantees.

Suppose that B is the total throughput of an output link on which a router implements WFQ. If all sessions of the WFQ scheduler are active, then each class receives a portion of the total bandwidth, which is determined by its weight w_i and is equal to $w_i B$ [114]. Hence, to simplify the expressions, we assume that it holds for all weights w_i that

$$\sum_i w_i = 1, \; w_i \in (0;1) \, . \tag{4}$$

By knowing the QoS requirements of all data flows, we can find values for w_i, such that all the QoS guarantees are ensured.

QoS requirements

Each service class can have an associated weight that specifies the allocated bandwidth. If there are N_i active flows within the ith class, then each flow has bandwidth that can be approximated by[4]

$$B_i^f = \frac{w_i B}{N_i} \, . \tag{5}$$

B_i^f can be treated as one of the QoS parameters that specifies the required bandwidth of a flow belonging to the ith service class. Thus, the minimum value of the weight, which provides the necessary amount of bandwidth for every flow, can be given by

$$w_i \geq N_i \frac{B_i^f}{B} \, . \tag{6}$$

The inequality states that a provider can allocate more resources than necessary. Indeed, if the network has free bandwidth resources, then a provider can allocate, either explicitly or implicitly, more bandwidth to a service class.

[4] In the case of WFQ, the fairness between data flows within a service class can be achieved by the per-flow buffer management or by the per-flow policing performed at some routers.

It is often the case that instead of specifying the requirements for a single traffic stream, there is a need to allocate resources for the whole class not taking the number of active flows into account. Such a behaviour may be necessary, when a provider wants to allocate bandwidth for the best-effort class. For these purposes, a modified version of (6) is proposed:

$$w_i \geq \frac{B_i}{B}\,. \tag{7}$$

Here B_i specifies the minimum amount of bandwidth resources for the whole ith class. Depending on the resource allocation strategy, a provider uses either (6) or (7), or both. In the latter case, it is possible to reserve a certain minimum amount of bandwidth regardless of the current number of active flows.

Due to the buffering, scheduling, and transmission of packets the size of a router queue varies all the time. In turn, the length of a queue in a routing node has an impact on the queuing delay and on the overall end-to-end delay of a packet. It can be shown that under WFQ the worst-case queuing delay is given by the following expression, where L^{max} denotes the maximum packet size:

$$D = \frac{\sigma}{\rho} + \frac{L^{max}}{B}\,. \tag{8}$$

In (8), it has been assumed that each incoming flow is regulated by the Token Bucket [39] with the bucket depth σ and token rate ρ. Parameters σ and ρ can be viewed as the maximum burst size and the long term bounding rate respectively.

Some authors use another form of (8) that includes L/ρ term referred to as the packetization delay. It is caused by the fact that before making a scheduling decision, a router has to receive a whole packet to do some amount of header processing before a packet can be given to the scheduler. Furthermore, a router might also have an internal transmission queue [49] where a packet sits after a being selected by the output scheduler. Thus, the expression below presents the worst-case delay estimation that takes the packetization and the hardware delay into account:

$$D = \frac{\sigma}{\rho} + \frac{L}{\rho} + \frac{L^{max}}{B} + D^{HW}\,. \tag{9}$$

Here, D^{HW} stands for the largest delay caused by the router's internal components. For the sake of brevity, we will be using (8), but will also present the final formulas that include the packetization and hardware delay. When multiple flows are aggregated, it is possible to assume that ρ is equal to the bandwidth allocated for a whole aggregate. As shown above, it is equal to $w_i B$. In turn, the worst-case estimation for the resulting burst size is $N_i\sigma$. Thus, (8) can be given in the following way, where D_i is the worst-case delay of a packet in the ith class:

$$D_i = \frac{N_i\sigma}{w_i B} + \frac{L^{max}}{B}\,. \tag{10}$$

Since there is a need to know the value of w_i, under which the required queuing delay can be guaranteed, it is possible to use (10) to obtain it:

$$w_i \geq \frac{N_i \sigma}{BD_i - L^{max}}. \tag{11}$$

Here, w_i specifies the minimum value that is necessary to provide the delay guarantees. It is clear that the more active flows there are and the least the required delay is, the bigger portion of resources must be allocated. It is interesting to note that expression $BD_i - L^{max}$ must have a positive value because the weight value cannot be negative. Thus, inequality $L^{max} < BD_i$ must always hold. In other words, if the WFQ scheduler cannot provide the necessary delay guarantees at least for one packet of the maximum size, then it will be impossible to provide the delay guarantees for the whole service class regardless of its burst size. If a provider wants to take the packetization and the hardware delay into account, then (11) can be modified as follows:

$$w_i \geq \frac{N_i \sigma + L_i}{B(D_i - D^{HW}) - L^{max}}. \tag{12}$$

By combining (6), (7), and (11) it is possible to present the final expression that reserves the necessary resources based on the bandwidth and delay requirements:

$$w_i \geq \max\left\{\frac{N_i B_i^f}{B}, \frac{B_i}{B}, \frac{N_i \sigma}{BD_i - L^{max}}\right\}, \forall i = \overline{1, m}, \tag{13}$$

where m stands for the number of service classes.

Upper constraints

Along with constraints that reserve the minimum amount of resources, a provider may need to introduce constraints that limit the allocation of resources. For instance, if we know that packets of a certain class will be routed along a certain path, which has a certain maximum capacity, then it makes sense to limit the amount of the allocated bandwidth. Otherwise, unnecessary bandwidth resources will be allotted to this class and sooner or later packets will be dropped in the bottleneck link. Another reason to construct the upper constraint is the availability of information concerning the flow peak rate. If we know the peak rates of all the flows within a service class, then we can determine the maximum bandwidth this class occupies by summing all the peak rate values.

If B_i^{max} is the maximum bandwidth of the ith service class, then it is possible to introduce easily the following upper constraint:

$$w_i \leq \frac{B_i^{max}}{B}, \forall i \in U, \tag{14}$$

where U represents a set of service classes that have information about the maximum bandwidth available along a data path.

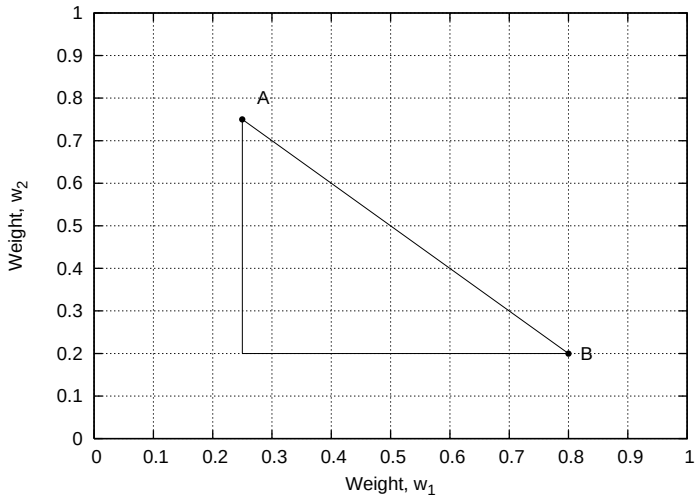


FIGURE 2 Graphical interpretation of constraints for WFQ.

Graphical interpretation

It is possible to build the graphical interpretation of constraints that reserve the necessary amount of resource. For the sake of simplicity, we will consider two service classes. The per-flow bandwidth requirements within each service class are 10 Kbps and 5 Kbps respectively. The output bandwidth is 1 Mbps. Suppose the current number of active flows within each service class is 25 and 40. Then Fig. 2 presents the constraints for the weight values of the WFQ scheduler.

Each point within the constrained area represents a set of the weight values that satisfy the given QoS requirements. However, there are only two extreme solutions – (0.25, 0.75) and (0.8, 0.2) – denoted by points A and B respectively. The first one corresponds to the case when the first class is allocated only the minimum amount of resources, while the remaining bandwidth is provided for the second class. In the second extreme solution, all free resources are allocated for the first class. Note that the number of the extreme solutions is always equal to the number of service classes.

As follows from the graphical interpretation, it is always possible to obtain a feasible solution if a router has enough output bandwidth. If there is no feasible solution, then the router does not have enough resources to ensure all the QoS requirements for the given number of active flows.

Pricing criterion

As follows from Fig. 2, depending on the number of service classes, there is a certain number of extreme weight combinations that satisfy the given QoS requirements. However, there is only one solution that optimizes the resource allocation from the viewpoint of the used criterion. As considered in section 2.3.3, a provider may use prices for the service classes to increase the total revenue. For these purposes, expression (3) should be reformulated for the WFQ scheduler.

If the fluid model is taken into consideration, then the expression $w_i B$ ap-

proximates the amount of the ith service class data a scheduler outputs during a time unit. Of course, it is true if all sessions of the WFQ scheduler are active. However, since a session corresponds to a service class that aggregates multiple data flows, it is likely that all sessions of the WFQ scheduler are active most of the time. Thus, the instantaneous revenue for the ith class can be given by:

$$r_i(C_i, w_i) = C_i w_i B \ \ [monet.\ units/second]\,. \tag{15}$$

Since resources are shared between several classes, the overall instantaneous revenue can be written in the form:

$$\sum_{i}^{m} r_i(C_i, w_i) = \sum_{i=1}^{m} C_i w_i B \ \ [monet.\ units/second]\,. \tag{16}$$

Thus, by manipulating the weight values w_i, different amount of data is transferred. As a result, the instantaneous revenue is different which affects the resulting total revenue.

General model

The adaptive model for the WFQ scheduler consists of the pricing function (16), which hence will be referred to as the *target function*, and a set of constraints (4), (13), and (14).

It is proposed to add a new parameter γ_i to the target function. Its purpose is to disable or enable the allocation of excess resources for the ith service class. Suppose, there is a service class consisting of applications that generate data at the constant rate. Although it may be the most expensive class, all the excess resources, allotted to it, will be shared among the other classes, because these applications will not increase their transmission rates. Therefore, the allocation of the excess resources can be disabled by setting $\gamma_i = 0$. If more bandwidth is allocated for a service class that consists predominantly of TCP flows, then the applications will increase their window sizes and, as a result, their transmission rates. Thus, it makes sense to set $\gamma_i = 1$. It should also be noted that the target function can be simplified by removing the constant component B:

$$\max\left\{\sum_{i=1}^{m} \gamma_i C_i w_i\right\} \tag{17}$$

subject to:

$$\sum_{i=1}^{m} w_i = 1,\ w_i \in (0;1)\,,$$

$$w_i \geq \max\left\{\frac{N_i B_i^f}{B}, \frac{B_i}{B}, \frac{N_i \sigma}{BD_i - L^{max}}\right\},\ \forall i = \overline{1,m}\,,$$

$$w_i \leq \frac{B_i^{max}}{B},\ \forall i \in U\,.$$

The model presented in (17) is a *linear optimization* problem. The purposes of this optimization task is to find the optimal weight values w_i, such that all the QoS guarantees are ensured and the total revenue is maximized. One of the methods that can be used to obtain the optimal values of the w_i is the *simplex* algorithm [151]. If a provider has to ensure only the bandwidth guarantees, then the delay term can be removed from the constraint, but the general model remains the same.

Other fair queuing disciplines

The adaptive model presented above can be used with other FQ schedulers, such as W^{2}FQ, SCFQ, STFQ, FFQ, and SPFQ [149]. Only minor differences in constraints are necessary. As an example, we will consider briefly the SCFQ scheduler since it is implemented in many hardware routers.

SCFQ is very similar to WFQ except that the former is simpler in implementation. Unlike WFQ, it updates the system virtual time only when a packet transmission starts or when a packet arrives to an empty system. SCFQ provides exactly the same bandwidth guarantees as WFQ, but the worst case delay estimation depends on the number of sessions [57]:

$$D = \frac{\sigma}{\rho} + (m-1)\frac{L^{max}}{B}. \tag{18}$$

It makes SCFQ quite inefficient when the number of sessions is huge. However, since we consider the case when resources are shared between several service classes, each session corresponds to a class, not to a distinctive flow. Thus, there are only a few sessions and the worst case delay estimation does not differ significantly from the one considered for WFQ.

The general model for SCFQ is exactly the same as (17), but constraint (13) is slightly different. The adaptive model for the SCFQ scheduler is

$$\max\left\{\sum_{i=1}^{m}\gamma_i C_i w_i\right\} \tag{19}$$

subject to:

$$\sum_{i=1}^{m} w_i = 1,\ w_i \in (0;1),$$

$$w_i \geq \max\left\{\frac{N_i B_i^f}{B}, \frac{B_i}{B}, \frac{N_i\sigma}{BD_i - (m-1)L^{max}}\right\}, \forall i = \overline{1,m}. \tag{20}$$

2.3.5 Weighted Round Robin

WRR scheduler

The WRR scheduler works in a cyclic manner serving consequently the input queues. During a cycle, a certain number of packets, determined by the associated weight, are sent from each queue. If a queue has fewer packets than the

value of the weight, the WRR scheduler outputs the existent number of packets and begins to serve the next queue. The WRR scheduler does not take the size of the transmitted packets into account. As a result, it is difficult to predict the actual bandwidth that each queue obtains. In other words, it is difficult to use only the weight values as the means to specify the amount of the output bandwidth.

Suppose that w_i is the value of the weight associated with the ith queue. If L_i is the mean packet size of the ith input queue, then $w_i L_i$ bytes of data are sent during each cycle on average. If there are m input queues, then it is easy to show that the average amount of data transmitted from all queues during one cycle can be approximated by:

$$\sum_{i=1}^{m} w_i L_i \,. \tag{21}$$

Expression (21) is referred to as the *frame size*. Taking the mean packet size and weights of all queues into account, it is possible to approximate the output bandwidth for the given kth queue:

$$\frac{w_k L_k}{\sum\limits_i w_i L_i} B, \;\; k \in [1; m] \,, \tag{22}$$

where B specifies the output bandwidth of an interface, on which a router implements WRR. By approximating the average bandwidth of each queue, it is possible to provide the QoS guarantees.

QoS requirements

Let us assume that each service class is associated with a queue of the WRR scheduler. Then, (22) approximates the bandwidth allocated for the whole class. However, if class k contains N_k active data flows, then each data stream obtains the bandwidth that can be expressed as follows:[5]

$$B_k^f = \frac{w_k L_k}{N_k \sum\limits_i w_i L_i} B \,. \tag{23}$$

Parameter B_k^f can be understood as one of the QoS parameters that specifies the bandwidth that should be provided for each data flow in the kth traffic class. Thus, if B_k^f is given, then a router must allocate a certain minimum amount of resources to satisfy the QoS requirements of all data streams. Based on this, it is possible to rewrite (23) in the following form:

$$\frac{w_k L_k}{N_k \sum\limits_i w_i L_i} B \geq B_k^f \,. \tag{24}$$

[5] The fairness between data flows within a service class can be achieved by the Stochastic Fair Queuing (SFQ)[104] combined with the WRR scheduler and / or by the per-flow buffer management.

While the right side of this inequality specifies the minimum amount of resources to be allocated, the left side specifies the amount of provided resources. Since the weights of the WRR scheduler control the allocation of resources, the task is to find such values of w_k that ensure the bandwidth requirements. It is possible to use (24) to determine w_k:

$$w_k \geq \frac{B_k^f N_k}{L_k(B - B_k^f N_k)} \sum_{\substack{i=1 \\ i \neq k}}^{m} w_i L_i \,. \tag{25}$$

As in the case of the adaptive model for the WFQ scheduler, sometimes there is a need to allocate resources regardless of the number of flows and their requirements. For instance, the best-effort class has no requirements at all, but resources should be provided for this class as well. For these purposes, it is possible to modify (25) as follows:

$$w_k \geq \frac{B_k}{L_k(B - B_k)} \sum_{\substack{i=1 \\ i \neq k}}^{m} w_i L_i \,. \tag{26}$$

Here B_k stands for the bandwidth requirements of the whole service class. Depending on the resource allocation strategy, a service provider chooses either (25) or (26), or both. It is possible to combine these expressions to obtain one constraint:

$$w_k \geq \max\left\{ \frac{B_k}{L_k(B - B_k)}, \frac{B_k^f N_k}{L_k(B - B_k^f N_k)} \right\} \sum_{\substack{i=1 \\ i \neq k}}^{m} w_i L_i, \forall k = \overline{1, m} \,. \tag{27}$$

Along with bandwidth requirements, certain service classes must be provided with the delay guarantees. Since the WRR scheduler serves the input queues in a cyclic manner, the processing of a packet in the kth queue can be delayed by

$$\left(\sum_{i, i \neq k} w_i L_i^{max} \right) / B$$

seconds. In the worst case, a packet belonging to the kth queue will enter it when the WRR scheduler starts to serve the k+1th queue. As a result, the packet will have to wait for a round. Note that we use $w_i L_i^{max}$ to estimate the largest amount of data the WRR scheduler can output during a round. To decrease this delay, it is possible to introduce the Low Latency Queue (LLQ) that can work in two modes [154]: *strict priority mode* and *alternate priority mode*. In the strict priority mode, the WRR scheduler always outputs packets from LLQ first. Such a scheme is identical to the one presented in Fig. 1. However, it is difficult to predict the allocation of bandwidth for other queues in this case. Thus, the alternate priority mode will be considered, in which LLQ is served in between queues of the other service classes. For instance, if there are 3 input queues, numbered from 1 to 3, and queue 1 is LLQ, then the queues are served in the following order: 1–2–1–3–.... In such a

scheme, the processing of a packet in LLQ can be delayed by

$$\max_i \{w_i L_i^{max}\} / B$$

seconds. As in the normal WRR scheduler, each queue is allowed to transmit no more than w_i packets during a round.

If a router implements LLQ, it is necessary to reformulate the considered above equations (21)-(26). Suppose, LLQ is identified by index l, where $l \in [1; m]$. Then, the frame size is approximated by the following expression:

$$(m-1)w_l L_l + \sum_{\substack{i=1 \\ i \neq l}}^{m} w_i L_i \,. \tag{28}$$

Taking account of the presented above considerations (22) – (25), it is possible to derive expressions for the minimum weight values that satisfy all the bandwidth requirements of each service class:

$$w_k \geq \frac{B_k^f N_k}{L_k(B - B_k^f N_k)} \left((m-1)w_l L_l + \sum_{\substack{i=1 \\ i \neq k, i \neq l}}^{m} w_i L_i \right), \; k \neq l \,, \tag{29}$$

$$w_l \geq \frac{B_l^f N_l}{(m-1)L_l(B - B_l^f N_l)} \sum_{\substack{i=1 \\ i \neq l}}^{m} w_i L_i \,. \tag{30}$$

As in the case of (27), it is possible to extend these constraints so that the bandwidth requirements for a class are taken into account. It should be noted that (29) and (30) only reserve bandwidth for normal queues and LLQ when the WRR scheduler works in the LLQ mode. However, they do not provide any delay guarantees. Thus, additional constraints are necessary, which will limit the weight values based on the delay requirements.

Suppose, each data flow, belonging to the class that has the delay requirements, is regulated by the Token Bucket policer with the mean rate ρ and the burst size σ. Since a service class aggregates multiple data flows, the resulting burst size of the whole class is $N_l\sigma$. Thus, it takes the WRR scheduler $N_l\sigma / B$ seconds to transmit the received burst under ideal conditions. However, if $N_l\sigma$ is bigger than $w_l L_l^{min}$, then more time is needed to output the burst because the WRR scheduler will start to serve another queue. While the scheduler serves that queue, the packets in LLQ can be delayed by

$$\max_{i, i \neq l} \{w_i L_i^{max}\} / B$$

seconds at most. Thus, the queuing delay of packets in LLQ can be estimated by:

$$D_l = \frac{N_l\sigma}{B} + \max \left\{ \left\lceil \frac{N_l\sigma}{w_l L_l^{min}} \right\rceil - 1, 0 \right\} \frac{\max_{i, i \neq l} \{w_i L_i^{max}\}}{B} \,. \tag{31}$$

Here D_l stands for the worst-case delay, experienced by packets in LLQ. The queuing delay consists of the burst queuing delay and the delay caused by the fact the WRR scheduler can start to serve another queue. The term

$$\max\left\{\left\lceil\frac{N_l\sigma}{w_l L_l^{min}}\right\rceil - 1, 0\right\}$$

estimates the number of times the LLQ is interrupted by other queues. If $N_l\sigma$ is less than $w_i L_i^{min}$, then the burst is transmitted completely during one round.

Expression (31) does not consider the fact that the initial processing of LLQ can be delayed by the other queue being served when a LLQ packet arrives to an empty queue. This happens due to the non-preempted nature of the WRR scheduler. Thus, it is possible to introduce the corrected estimation:

$$\begin{aligned} D_l &= \frac{N_l\sigma}{B} + \max\left\{\left\lceil\frac{N_l\sigma}{w_l L_l^{min}}\right\rceil - 1, 0\right\}\frac{\max\limits_{i,i\neq l}\{w_i L_i^{max}\}}{B} + \frac{\max\limits_{i,i\neq l}\{w_i L_i^{max}\}}{B} = \\ &= \frac{N_l\sigma}{B} + \max\left\{\left\lceil\frac{N_l\sigma}{w_l L_l^{min}}\right\rceil, 1\right\}\frac{\max\limits_{i,i\neq l}\{w_i L_i^{max}\}}{B}. \end{aligned} \tag{32}$$

Based on the values of $N_l\sigma$ and $w_l L_l^{min}$, we can consider two distinctive cases:

$$D_l = \begin{cases} \dfrac{1}{B}\left(N_l\sigma + \max\limits_{i,i\neq l}\{w_i L_i^{max}\}\right), & \text{if } N_l\sigma \leq w_l L_l^{min}, \quad \text{(33a)} \\ \dfrac{1}{B}\left(N_l\sigma + \left\lceil\dfrac{N_l\sigma}{w_l L_l^{min}}\right\rceil \max\limits_{i,i\neq l}\{w_i L_i^{max}\}\right), & \text{if } N_l\sigma > w_l L_l^{min}. \quad \text{(33b)} \end{cases}$$

The first expression corresponds to the case when a burst is output completely in one round. Since the resulting burst size of the whole service class is usually larger than $w_l L_l^{min}$, we will consider (33b).

It is obvious that it is not possible to use (33b) in the linear optimization task because it has the non-linear operator. To simplify it, we can observe the fact that if $N_l\sigma \gg w_l L_l$, then

$$\left\lceil\frac{N_l\sigma}{w_l L_l^{min}}\right\rceil \approx \frac{N_l\sigma}{w_l L_l^{min}}. \tag{34}$$

Indeed, if a router aggregates a huge amount of data flows within a class, then the error will be insignificant compared to the value of $N_l\sigma$. However, if there are only a few flows, then the error is considerably bigger. It will be presented later in chapter 4 that (34) fails when there are a few flows. Based on assumption (34), we can rewrite (33b) in the following form:

$$D_l = \frac{N_l \sigma}{B} \left(1 + \frac{\max\limits_{i,i \neq l} \{ w_i L_i^{max} \}}{w_l L_l^{min}} \right) . \tag{35}$$

Expression (35) presents a simple and tractable estimation for the worst-case delay. If $\max_{i,i \neq l} \{ w_i L_i^{max} \}$ equals zero, i.e. all the WRR sessions except LLQ are inactive, then it takes $N_l \sigma / B$ seconds to output the burst. To decrease the queuing delay when all the sessions are active, we have to assign bigger values to w_l and less values to $w_i, i \neq l$, which corresponds to the intuitive expectations.

It is also possible to modify (35) to the from that takes the packetization and the hardware delays into account:

$$D_l = \frac{N_l \sigma + L_l}{B} \left(1 + \frac{\max\limits_{i,i \neq l} \{ w_i L_i^{max} \}}{w_l L_l^{min}} \right) + D^{HW} . \tag{36}$$

As in the case of WFQ, for the sake of clarity expression (35) will be considered. It will be presented in the form of the inequality meaning that packets in LLQ can experience various queuing delays; however the worst-case delay must not exceed the certain value given by D_l:

$$D_l \geq \frac{N_l \sigma}{B} \left(1 + \frac{\max\limits_{i,i \neq l} \{ w_i L_i^{max} \}}{w_l L_l^{min}} \right) . \tag{37}$$

We can also rewrite it in the following form that is suitable for the optimization problem:

$$\left(\frac{B D_l}{N_l \sigma} - 1 \right) w_l L_l^{min} - \max_{i,i \neq l} \{ w_i L_i^{max} \} \geq 0 . \tag{38}$$

In turn, (38) can be represented as a set of constraints:

$$\left(\frac{B D_l}{N_l \sigma} - 1 \right) w_l L_l^{min} - w_i L_i^{max} \geq 0, \forall k = \overline{1, m}, k \neq l . \tag{39}$$

Upper constraints

As in the case of the adaptive model for the WFQ scheduler, upper constraints may be necessary to achieve better resource allocation between service classes. It is easy to derive it for the non-LLQ mode based on (26). If B_k^{max} is the maximum bandwidth available for the kth class along the forwarding path, then the upper constraint will be as follows:

$$w_k \leq \frac{B_k^{max}}{L_k (B - B_k^{max})} \sum_{\substack{i=1 \\ i \neq k}}^{m} w_i L_i, \forall k \in U . \tag{40}$$

In the similar way one can derive the upper constraints for the LLQ mode based on (29) and (30).

Graphical interpretation

To illustrate constraints for the WRR scheduler, the same parameters as in section 2.3.4 will be used. Fig. 3 presents weight values w_k for that simple case.[6] It can be noticed that there are two lines that bound the area of possible combinations of the weight values. For instance, if w_1 equals 1, then the value of w_2 could be 1, 2, or 3. On the other hand, if w_2 equals 1, then the value of w_1 can range from 1 to 4. At the same time, there are a lot of other possible combinations within the constrained area. All these combinations result in different allocation of bandwidth resources, but all of them satisfy the given QoS requirements.

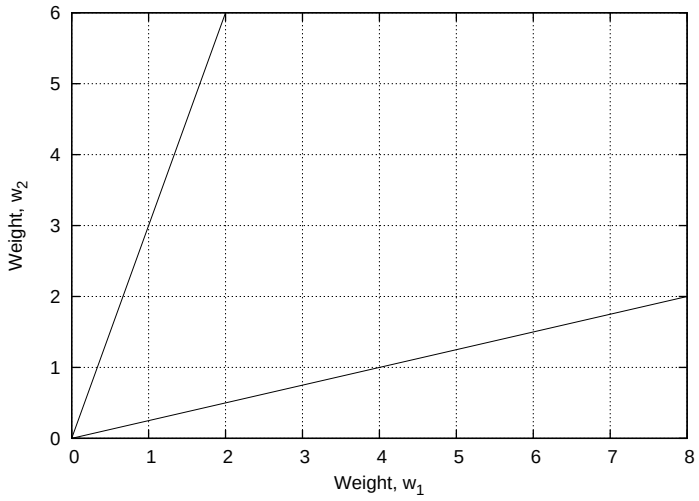


FIGURE 3 Graphical interpretation of constraints for WRR.

As follows from Fig. 3, the area between two constraints is not bounded from above. It means that there are an infinite number of possible solutions, from which it is impossible to choose the best one. For instance, combinations of weights such as (1,3) and (2,6) are possible. As can be noticed, the ratio of weights remains the same. Thus, these two combinations are similar from the viewpoint of the measured bandwidth. However, if we consider how the WRR scheduler outputs packets, then the difference becomes obvious. Combination (2,6) results in a bigger frame size, which causes larger queuing delays.

The requirement to keep the frame size as small as possible leads to the following considerations. The minimum value of the weight for each traffic class must be 1. It cannot be less than 1 because in such a case a class will not get a chance to transmit any packet. Thus, it is possible to find the maximum weight values that other traffic aggregates can have. Referring to Fig. 3, if w_1 is equal to 1, the maximum value of w_2 can be 3. In turn, if w_2 is equal to 1, then the maximum value of w_1 can be 4. If there are more than two service classes, then the same considerations are applied to each w_k. Thus, it is possible to add the following constraints:

[6] For the sake of simplicity we imply that w_i is not an integer value but a real one.

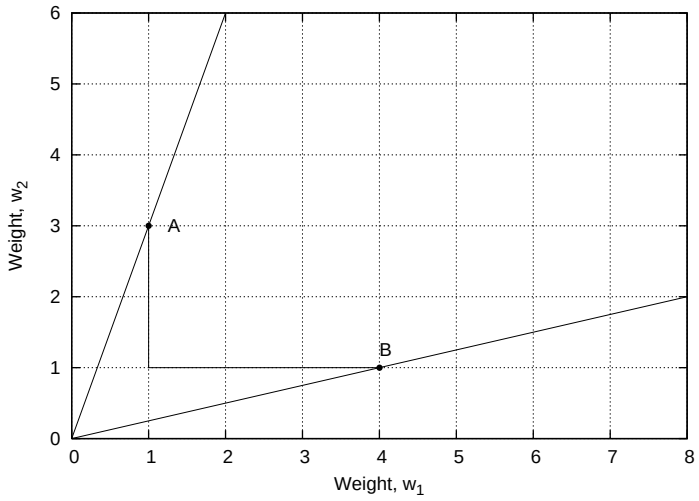


FIGURE 4 Graphical interpretation of constraints for WRR.

$$w_k \geq 1, w_k \in N, \forall k = \overline{1,m}\,. \tag{41}$$

The graphical interpretation of these additional constraints is presented in Fig. 4. As follows from this figure, there are two extreme points A and B. Point A corresponds to the case when the first class is provided only with the minimum amount of resources, and the remaining bandwidth is allocated for the second class. Point B corresponds to the opposite resource allocation scheme - the first class consumes the remaining bandwidth. As the number of service classes grows, the number of extreme points increases proportionally. For instance, if there are three classes, then there are three points that correspond to the extreme resource allocation cases.

Pricing criterion

The pricing function for the WRR scheduler differs slightly from the one considered earlier for WFQ. Though it is possible to approximate the instantaneous revenue per time unit, it makes the function too complicated and non-linear. Instead, it is possible to approximate the revenue obtained during a round. As presented earlier, (21) approximates the frame size, i.e. the amount of data the WRR scheduler outputs during one cycle. Thus, the instantaneous revenue (3) can be given by:

$$\sum_{i}^{m} r_i(C_i, w_i) = \sum_{i=1}^{m} C_i w_i L_i \;\; [monet.units/round]\,. \tag{42}$$

This function is valid only for the non-LLQ mode. Since (28) approximates the amount of data the WRR scheduler outputs in the LLQ mode, the pricing function can be modified to the form

$$(m-1)C_l w_l L_l + \sum_{\substack{i=1 \\ i\neq l}}^{m} C_i w_i L_i \quad [monet.units/round]\,. \tag{43}$$

General model

The general adaptive model for the WRR scheduler comprises the pricing function (43) and constraints (29), (30), (39), and (41). Parameter γ_i has the same purpose as in the case of the adaptive model based upon WFQ.

$$\max\left\{(m-1)\gamma_l C_l w_l L_l + \sum_{\substack{i=1 \\ i\neq l}}^{m} \gamma_i C_i w_i L_i\right\} \tag{44}$$

subject to:

$$(m-1)B_k^f N_k w_l L_l + B_k^f N_k \sum_{\substack{i=1 \\ i\neq k, i\neq l}}^{m} w_i L_i + (B_k^f N_k - B) w_k L_k \leq 0,\ \forall k = \overline{1,m}, k \neq l\,,$$

$$(m-1)(B_l^f N_l - B) w_l L_l + B_l^f N_l \sum_{\substack{i=1 \\ i\neq l}}^{m} w_i L_i \leq 0\,,$$

$$\left(\frac{BD_l}{N_l \sigma_l} - 1\right) w_l L_l^{min} - w_k L_k^{max} \geq 0,\ \forall k = \overline{1,m}, k \neq l\,,$$

$$w_k \geq 1,\ w_k \in N, \forall k = \overline{1,m}\,.$$

Alternatively, if a provider has to provide only the bandwidth guarantees, then the model given above can be simplified significantly, i.e. the pricing function (42) and the constraints (25) and (41) are used:

$$\max\left\{\sum_{i=1}^{m} \gamma_i C_i w_i L_i\right\} \tag{45}$$

subject to:

$$B_k^f N_k \sum_{\substack{i=1 \\ i\neq k}}^{m} w_i L_i + (B_k^f N_k - B) w_k L_k \leq 0, \forall k = \overline{1,m}\,,$$

$$w_k \geq 1,\ w_k \in N, \forall k = \overline{1,m}\,.$$

The presented adaptive model is the *integer linear* optimization problem. Since weights of the WRR scheduler are integer values, the solution for the optimization problem must be a set of integer weight values. One of the methods that can be used to calculate the optimal values of w_i is the *branch & bound* algorithm [50].

It should be noted that if the packet size is constant, like in the ATM networks, then we can simplify the model, i.e. there is no need to include terms L_i, L_i^{min}, L_i^{max} into the constraints and the target function.

2.3.6 Deficit Round Robin

Though DRR belongs to the class of the RR schedulers and shares common properties with WRR, it has several features that make it attractive for the provision of QoS. The most noticeable one is that the DRR scheduler is capable of taking the packet size into account while being very simple in implementation. The general considerations for DRR are the same as for WRR. So, we will present briefly the final formulas.

DRR scheduler

The DRR scheduler works in a cyclic manner serving consequently the input queues. During a round, a certain number of packets, determined by the value of the deficit counter, are sent from each queue. As all queues are served, the DRR scheduler updates the deficit counter using the *quantum value* and begins the next cycle. Unlike weights of the WRR scheduler, the quantum values specify the amount of bytes, not the number of packets. As a result, the deficit counter determines the amount of data the scheduler outputs during a round while the quantum values determine the long-term bandwidth allocation. It enables to share bandwidth accurately when the packets are of different size.

Like in WRR, it is possible to approximate the bandwidth allotted for a given class k based on the frame size and the quantum values Q_i:

$$\frac{Q_k}{\sum\limits_i Q_i} B, k \in [1; m]\,. \tag{46}$$

where B is the bandwidth of the output link.

QoS requirements

Since quantum values control the allocation of the output bandwidth between service classes, the task is to find such values of Q_k that all the QoS requirements are satisfied:

$$Q_k \geq \frac{B_k^f N_k}{B - B_k^f N_k} \sum_{\substack{i=1 \\ i \neq k}}^{m} Q_i, \forall k = \overline{1, m}\,. \tag{47}$$

In the LLQ mode, the following constraints reserve the minimum amount of bandwidth resources:

$$Q_k \geq \frac{B_k^f N_k}{B - B_k^f N_k} \left((m-1) Q_l + \sum_{\substack{i=1 \\ i \neq k, i \neq l}}^{m} Q_i \right),\ k \neq l\,, \tag{48}$$

$$Q_l \geq \frac{B_l^f N_l}{(m-1)(B - B_l^f N_l)} \sum_{\substack{i=1 \\ i \neq l}}^{m} Q_i\,. \tag{49}$$

The worst-case delay of a packet in the LLQ is determined as follows:

$$D_l = \begin{cases} \frac{1}{B}\left(N_l\sigma + \max\limits_{i,i\neq l}\{Q_i\}\right), & \text{if } N_l\sigma \leq Q_l\,, \quad (50a) \\ \frac{1}{B}\left(N_l\sigma + \left\lceil \frac{N_l\sigma}{Q_l} \right\rceil \max\limits_{i,i\neq l}\{Q_i\}\right), & \text{if } N_l\sigma > Q_l\,. \quad (50b) \end{cases}$$

If we assume that

$$\left\lceil \frac{N_l\sigma}{Q_l} \right\rceil \approx \frac{N_l\sigma}{Q_l}$$

then (50b) yields the set of final constraints that ensure the delay requirements:

$$\left(\frac{BD_l}{N_l\sigma} - 1\right) Q_l - Q_k \geq 0, \forall k = \overline{1,m}, k \neq l\,. \quad (51)$$

Having analysed expressions (46)-(51), it is possible to notice that they do not include the mean packet size, neither do they include maximum and minimum packet sizes. Thus the adaptive model for the DRR scheduler can be used in those networks where packets are not of the same size within a service class.

Upper constraints

The upper constraints for the DRR scheduler are of the same form as for the WRR scheduler, but the quantum values are used:

$$Q_k \leq \frac{B_k^{max}}{B - B_k^{max}} \sum_{\substack{i=1 \\ i\neq k}}^{m} Q_i, \forall k \in U\,. \quad (52)$$

Graphical interpretation

Since the quantum values specify the amount of bytes, not the number of packets, it is does not make any sense to set the minimum value of the quantum values to 1. Instead, the minimum quantum values must be set so that at least one packet is output during a round from each queue. Thus, the quantum value for the kth queue must be larger than or equal the maximum packet size observed within that queue. Such a requirement yields the following set of constraints:

$$Q_k \geq L_k^{max}, Q_k \in N, \forall k = \overline{1,m}\,. \quad (53)$$

Since it might be a challenging task to track the maximum packet size within each queue, a provider may set the quantum values to be larger than the Maximum Transmission Unit (MTU) of the output link. For instance, such an approach is taken in the Cisco routers [154]. However, according to our preliminary simulations, it results in a less efficient resource allocation.

Fig. 5 presents the graphical interpretation of the DRR constraints. There are two service classes, which parameters are the same as considered for the WFQ

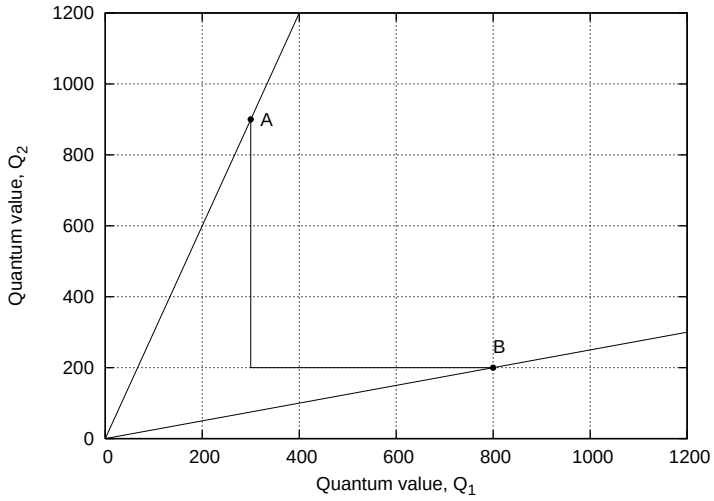


FIGURE 5 Graphical interpretation of constraints for DRR.

and WRR schedulers. The first class has the maximum packet size of 300 bytes, while the maximum packet size of the second class is 200 bytes.

Pricing criterion

The pricing function for the DRR scheduler is very similar to WRR, except that quantum values Q_i are used:

$$\sum_{i=1}^{m} C_i Q_i \quad [monet.units/round]\,. \tag{54}$$

In the LLQ mode it has the following form:

$$(m-1)C_l Q_l + \sum_{\substack{i=1 \\ i\neq l}}^{m} C_i Q_i \quad [monet.units/round]\,. \tag{55}$$

General model

The general adaptive model for the DRR scheduler consists of the pricing function (55) and the QoS constraints (48), (49), (51), and (53):

$$\max \left\{ (m-1)\gamma_l C_l Q_l + \sum_{\substack{i=1 \\ i\neq l}}^{m} \gamma_i C_i Q_i \right\} \tag{56}$$

subject to:

$$(m-1)B_k^f N_k Q_l + B_k^f N_k \sum_{\substack{i=1 \\ i\neq k, i\neq l}}^{m} Q_i + (B_k^f N_k - B)Q_k \leq 0,\ \forall k = \overline{1,m},\ k \neq l\,,$$

$$(m-1)(B_l^f N_l - B)Q_l + B_l^f N_l \sum_{\substack{i=1 \\ i\neq l}}^{m} Q_i \leq 0\,,$$

$$\left(\frac{BD_l}{N_l\sigma_l} - 1\right) Q_l - Q_k \geq 0,\ \forall k = \overline{1,m}, k \neq l\,,$$

$$Q_k \geq L_k^{max},\ Q_k \in N, \forall k = \overline{1,m}\,.$$

As in the case of the WRR scheduling discipline, the previous optimization task can be simplified in terms of constraints and the target function if only bandwidth guarantees are necessary:

$$\max\left\{\sum_{i=1}^{m} \gamma_i C_i Q_i\right\} \tag{57}$$

subject to:

$$B_k^f N_k \sum_{\substack{i=1 \\ i\neq k}}^{m} Q_i + (B_k^f N_k - B)Q_k \leq 0,\ \forall k = \overline{1,m}\,,$$

$$Q_k \geq L_k^{max},\ Q_k \in N, \forall k = \overline{1,m}\,.$$

The adaptive model for the DRR scheduler is more efficient than the one considered for WRR. The main reason for this is that it does not depend on the mean packets, neither does it require the minimum, maximum, and mean packet size. At the same time, DRR is very fast and simple in implementation. It makes the DRR scheduler very attractive for the adaptive resource sharing in the packet networks when both bandwidth and delay guarantees are necessary.

2.3.7 Other round-robin disciplines

The adaptive models for WRR and DRR can be modified easily to support any RR discipline. As an example, we will present Modified Weighted Round Robin (MWRR) [154] that is implemented in Cisco Catalyst and series 8500 routers. The MWRR scheduler combines properties of WRR and DRR. On the one hand, each queue has the associated weight that determines the number of *slots* the scheduler outputs during a round. On the other hand, there is a deficit counter that tracks the number of transmitted slots so that a router can share bandwidth more os less accurately even if the packet size varies.

The general form of the adaptive model will be similar to WRR. However, since all the slots are of the same size and the scheduler maintains the deficit counter, there is no need to include the mean packet size into the target function and the QoS constraints:

$$\max\left\{\sum_{i=1}^{m}\gamma_i C_i w_i\right\} \tag{58}$$

subject to:

$$B_k^f N_k \sum_{\substack{i=1\\ i\neq k}}^{m} w_i + (B_k^f N_k - B) w_k \leq 0,\ \forall k = \overline{1,m}\,,$$

$$w_k \geq 1,\ w_k \in N, \forall k = \overline{1,m}\,.$$

2.3.8 Weighted Delay Minimization and Revenue Optimization

In this subsection we propose a scheduling model that guarantees latency and optimizes the network service provider's revenue, not just in the worst case as in [99], but in a general case. The proposed algorithm ensures less delay for the users paying more for the connection (i.e. higher service class) than those paying less. In fact, our approach minimizes the weighted mean delay for connections, and therefore satisfaction of the customers can be achieved. We extend our previous study [**?**, 81] from the single node to the multinode case in the non-trivial way. We see that closed form, optimal, unique, distributed, and simple solution for revenue maximization can be achieved.

We show that this kind of solution can be obtained also in the multinode system, yielding a temptating scenario for customers and the service provider. The closed form formula for updating weights is independent on any statistical behavior of the connections. Therefore, it is also robust against erroneous estimates of customers' behavior.

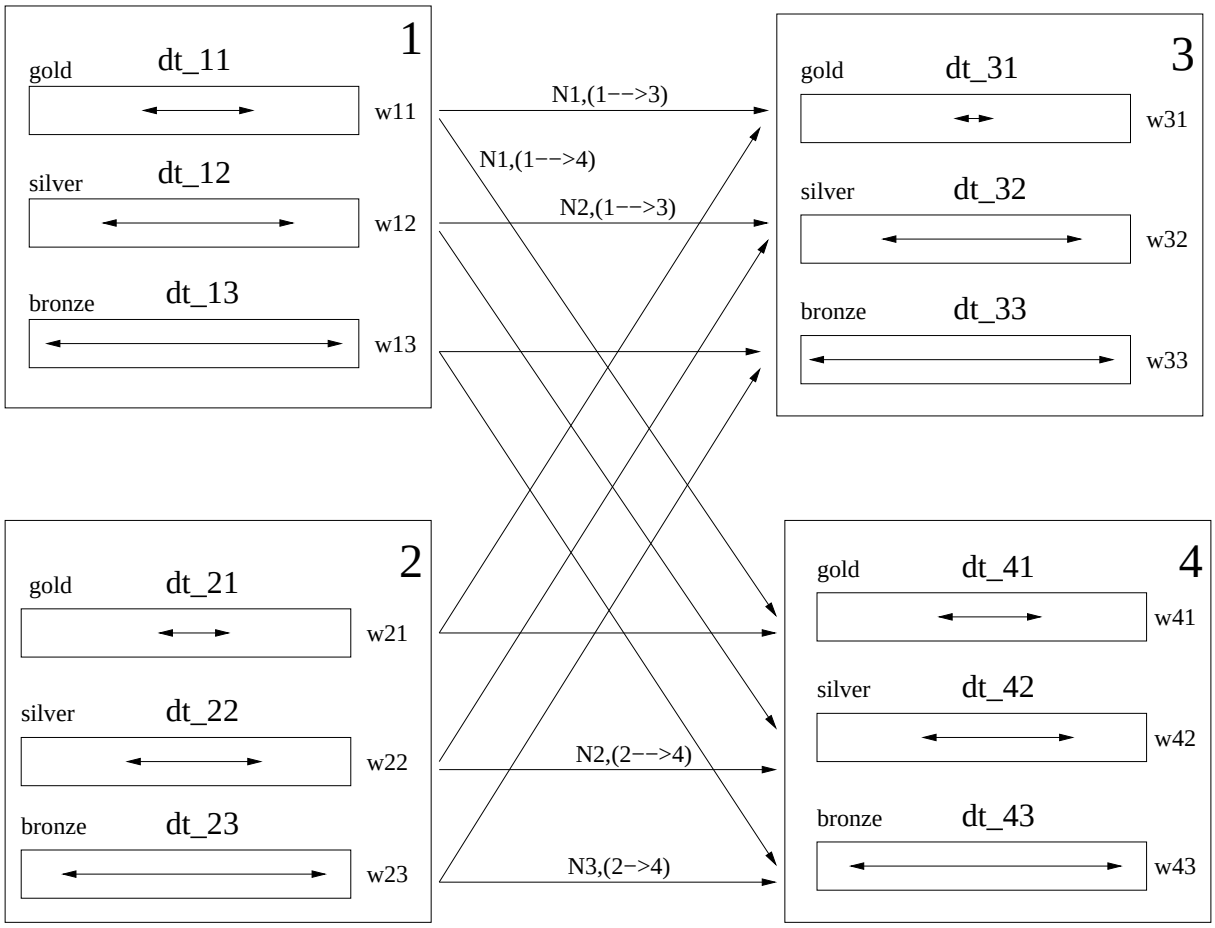


FIGURE 6 Four node system. In the medium of the system there is a core network, which is not shown in the figure. Parameter $N_j^{i\to p}$ denotes the number of such connections in the jth service class, who transfer data packets through both switches i and p.

2.3.9 Multinode network, delays, and revenue optimization

Network model and delays

Although the algorithm operates in the general multinode system with arbitrary number of switches (nodes) and service classes, we present it by using a four-node case to avoid complicated notation. Consider the four-node system with schedulers as illustrated in Fig. 6. An example of one scheduler is illustrated in more detail in Fig. 7. There are three service classes, namely gold, silver, and bronze. Gold class customers pay most of money while getting the best service, and silver class customers pay least of money. Data are transmitted from nodes 1 and 2 to either nodes 3 and 4. Parameter Δt_{ij} denotes the time which passes when data is transferred through the queue j to the output in the switch i, when $w_{ij} = 1$. If the queue is almost empty, the delay is small, and when the buffer is full, it is large. Variable w_{ij} is the weight allocated for switch i and class j. Constraint for

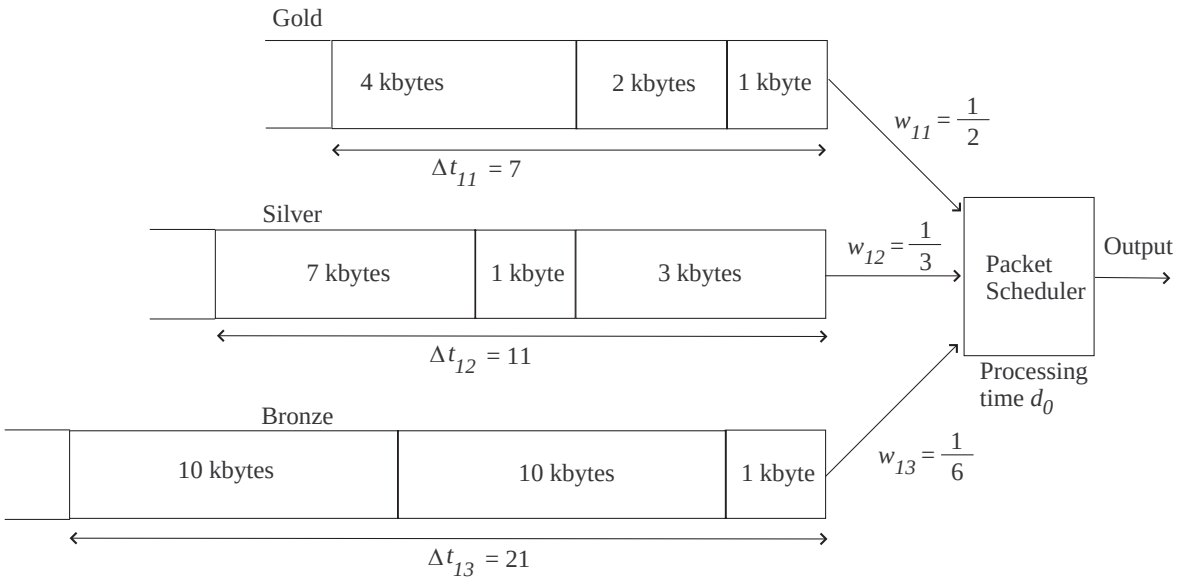


FIGURE 7 Scheduler and queues. Parameter Δt_{1j}, $j = 1, \ldots, 3$ denotes time which passes when data is transferred through queue. It depends on the processing time d_0 of the scheduler. Variables w_{1j} control the overall delay.

weights w_{ij} are

$$\sum_{j=1}^{m} w_{ij} = 1, \quad w_{ij} > 0. \tag{59}$$

Variables w_{ij} give weights, how long time queues ij are served per total time. Therefore the delay d_{ij} in the queue (i, j) is actually

$$d_{ij} = \frac{\Delta t_{ij}}{w_{ij}}. \tag{60}$$

Without loss of generality, only non-empty queues are considered, and therefore

$$w_{ij} \neq 0, \quad i = 1, \ldots, m, \tag{61}$$

where m is the number of service classes. When one queue becomes empty, $m \to m - 1$. Parameter $N_j^{i \to p}$ has the following meaning: it is denoting the number of such connections in the jth service class, which transfer data packets through both switches i and p. Because there are four schedulers and three classes, there are maximally 12 arrows from the elements 1 and 2 to 3 and 4 (for simplicity, but without loss of generality, other connections such as $1 \to 2$ are not considered here). The total number of connections in the node i and queue j is denoted by N_{ij}, and it obeys the general condition

$$N_{ij} = \sum_{p=1}^{n} N_j^{i \to p}, \tag{62}$$

where n denotes the number of the nodes. In our sample case, $n = 4$, and

$$N_{11} = N_1^{1\to3} + N_1^{1\to4}, \tag{63}$$

$$N_{12} = N_2^{1\to3} + N_2^{1\to4}, \tag{64}$$

$$N_{13} = N_3^{1\to3} + N_3^{1\to4}, \tag{65}$$

$$N_{21} = N_1^{2\to3} + N_1^{2\to4}, \tag{66}$$

$$N_{22} = N_2^{2\to3} + N_2^{2\to4}, \tag{67}$$

$$N_{23} = N_3^{2\to3} + N_3^{2\to4}, \tag{68}$$

$$N_{31} = N_1^{1\to3} + N_1^{2\to3}, \tag{69}$$

$$N_{32} = N_2^{1\to3} + N_2^{2\to3}, \tag{70}$$

$$N_{33} = N_3^{1\to3} + N_3^{2\to3}, \tag{71}$$

$$N_{41} = N_1^{1\to4} + N_1^{2\to4}, \tag{72}$$

$$N_{42} = N_2^{1\to4} + N_2^{2\to4}, \tag{73}$$

$$N_{43} = N_3^{1\to4} + N_3^{2\to4}, \tag{74}$$

Pricing and revenue

Consider the price paid by customers in the class j to the service provider. It is depending on the end-to-end delay of the data. The price $r_j(d)$ is decreasing with respect to the delay d. We consider *linear* pricing function.

Definition. *The function*

$$r_j(d) = -r_j d + k_j, \quad i = 1, \ldots, m, \tag{75}$$

$$r_j > 0, \tag{76}$$

$$k_j > 0, \tag{77}$$

is called linear pricing function.

Parameters r_j penalize if the delay is increasing. Parameter k_j is included in the function to guarantee positive revenue. Three typical pricing functions are shown in Fig. 8, where k_js are positive. Delay d includes delays in the switching element (depends on the weights), and insertion delays, transmission delays etc. (not depending on the weights). Without the loss of generality, those delays not depending on the weights are not considered here, because they have no effect on the derivation of the algorithm. They can simply be considered as constant shifts. This can be seen as follows. If the total delay is denoted by $d = d_1 + d_2$, where d_1 is the delay in the stwitch, and d_2 includes the other delays, one can write

$$r_j(d) = -r_j d_1 + (k_j - r_j d_2). \tag{78}$$

Then one can merge the right hand side term so that one obtains

$$r_j(d) = -r_j d_1 + \tilde{k}_j, \tag{79}$$

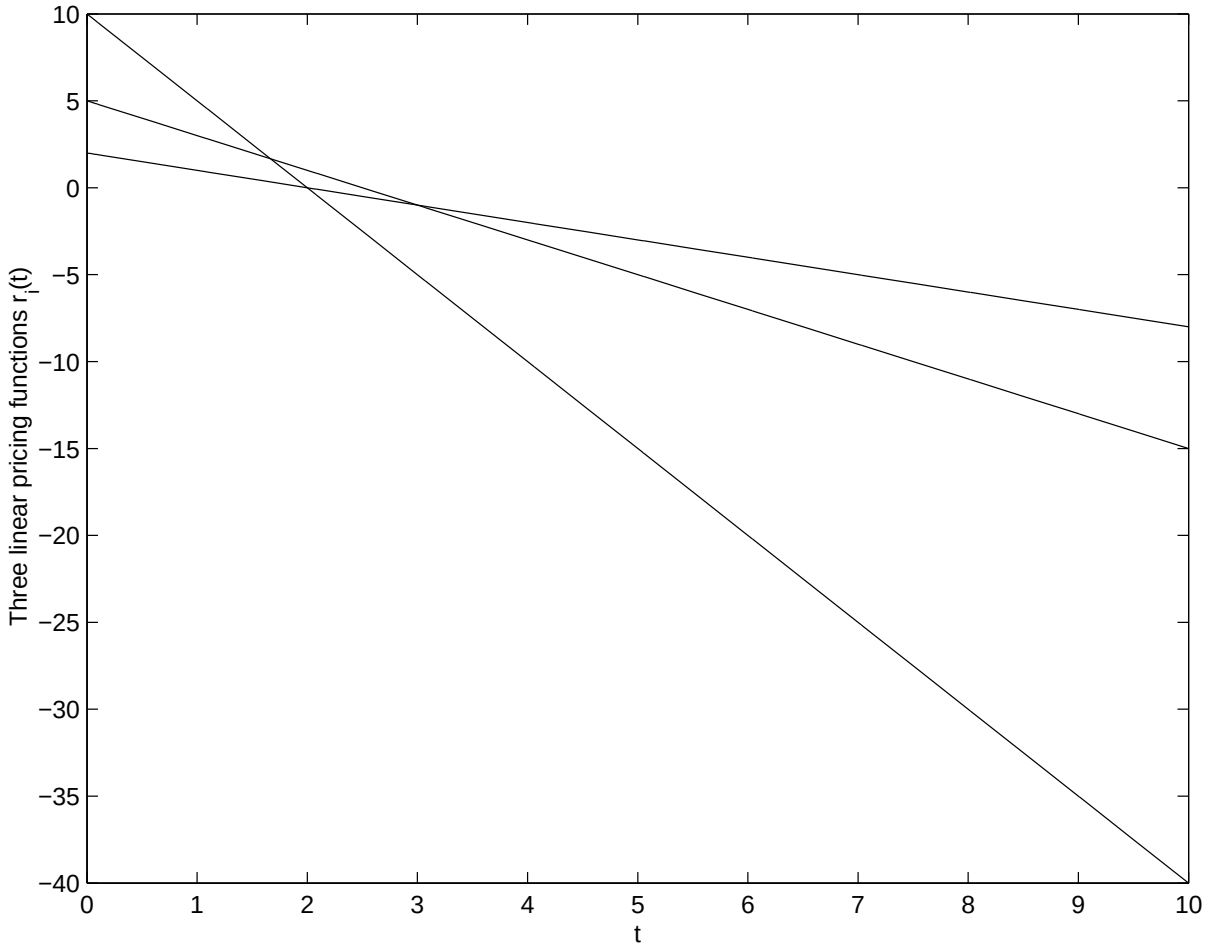


FIGURE 8 Three linear pricing functions. For gold class, $r_1(d) = -5d + 10$; for bronze class, $r_3(d) = -d + 2$. For highest priority class, both penalty factor r_j as well as constant shift k_j in the pricing model $r_j(d) = -r_j d + k_j$ are highest.

where

$$\tilde{k}_j = k_j - r_j d_2. \tag{80}$$

The revenue obtained from those customers who transfer data through switches $i_1 \to i_2$ is denoted by $F_{i_1 \to i_2}$. Then in our four-node system, the total revenue is

$$F = F_{1\to 3} + F_{1\to 4} + F_{2\to 3} + F_{2\to 4}. \tag{81}$$

Consider for example $F_{1\to 3}$. Because there are $N_1^{1\to 3}$ connections in this route with the total delay $\Delta t_{11}/w_{11} + \Delta t_{31}/w_{31}$, then the price paid by those customers is

$$F_{1\to 3} = -N_1^{1\to 3} r_1 \left(\frac{\Delta t_{11}}{w_{11}} + \frac{\Delta t_{31}}{w_{31}} \right) + N_1^{1\to 3} k_1. \tag{82}$$

In the same manner, the total revenue is

$$\begin{aligned}
F &= -N_1^{1\to 3} r_1 \left(\frac{\Delta t_{11}}{w_{11}} + \frac{\Delta t_{31}}{w_{31}}\right) + N_1^{1\to 3} k_1 \\
&- N_2^{1\to 3} r_2 \left(\frac{\Delta t_{12}}{w_{12}} + \frac{\Delta t_{32}}{w_{32}}\right) + N_2^{1\to 3} k_2 \\
&- N_3^{1\to 3} r_3 \left(\frac{\Delta t_{13}}{w_{13}} + \frac{\Delta t_{33}}{w_{33}}\right) + N_3^{1\to 3} k_3 \\
&- N_1^{1\to 4} r_1 \left(\frac{\Delta t_{11}}{w_{11}} + \frac{\Delta t_{41}}{w_{41}}\right) + N_1^{1\to 4} k_1 \\
&- N_2^{1\to 4} r_2 \left(\frac{\Delta t_{12}}{w_{12}} + \frac{\Delta t_{42}}{w_{42}}\right) + N_2^{1\to 4} k_2 \\
&- N_3^{1\to 4} r_3 \left(\frac{\Delta t_{13}}{w_{13}} + \frac{\Delta t_{43}}{w_{43}}\right) + N_3^{1\to 4} k_3 \\
&- N_1^{2\to 3} r_1 \left(\frac{\Delta t_{21}}{w_{21}} + \frac{\Delta t_{31}}{w_{31}}\right) + N_1^{2\to 3} k_1 \\
&- N_2^{2\to 3} r_2 \left(\frac{\Delta t_{22}}{w_{22}} + \frac{\Delta t_{32}}{w_{32}}\right) + N_2^{2\to 3} k_2 \\
&- N_3^{2\to 3} r_3 \left(\frac{\Delta t_{23}}{w_{23}} + \frac{\Delta t_{33}}{w_{33}}\right) + N_3^{2\to 3} k_3 \\
&- N_1^{2\to 4} r_1 \left(\frac{\Delta t_{21}}{w_{21}} + \frac{\Delta t_{41}}{w_{41}}\right) + N_1^{2\to 4} k_1 \\
&- N_2^{2\to 4} r_2 \left(\frac{\Delta t_{22}}{w_{22}} + \frac{\Delta t_{42}}{w_{42}}\right) + N_2^{2\to 4} k_2 \\
&- N_3^{2\to 4} r_3 \left(\frac{\Delta t_{23}}{w_{23}} + \frac{\Delta t_{43}}{w_{43}}\right) + N_3^{2\to 4} k_3
\end{aligned} \tag{83}$$

Optimal weights

Here we present the theorem for optimal weights:

Theorem 1: *For linear pricing functions, the maximum revenue F is achieved by using the weights*

$$w_{ij} = \frac{\sqrt{N_{ij} r_j \Delta t_{ij}}}{\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}, \quad i = 1, \ldots, n, \quad j = 1 \ldots, m, \tag{84}$$

where n is the number of nodes, and m is the number of service classes. In addition, F is unique.

Proof: Regroup the terms of Eq. (83), remember equalities (63)-(74), and add the Lagrangian penalty term as follows:

$$
\begin{aligned}
F &= -(N_1^{1\to 3} + N_1^{1\to 4}) r_1 \frac{\Delta t_{11}}{w_{11}} \\
&- (N_1^{1\to 3} + N_1^{2\to 3}) r_1 \frac{\Delta t_{31}}{w_{31}} \\
&- \cdots \\
&= -N_{11} r_1 \frac{\Delta t_{11}}{w_{11}} \\
&- N_{12} r_2 \frac{\Delta t_{12}}{w_{12}} \\
&- N_{13} r_3 \frac{\Delta t_{13}}{w_{13}} \\
&+ \lambda_1 (1 - \sum_{j=1}^{3} w_{1j}) \\
&- N_{21} r_1 \frac{\Delta t_{21}}{w_{21}} \\
&- N_{22} r_2 \frac{\Delta t_{22}}{w_{22}} \\
&- N_{23} r_3 \frac{\Delta t_{23}}{w_{23}} \\
&+ \lambda_2 (1 - \sum_{j=1}^{3} w_{2j}) \\
&- N_{31} r_1 \frac{\Delta t_{31}}{w_{31}} \\
&- N_{32} r_2 \frac{\Delta t_{32}}{w_{32}} \\
&- N_{33} r_3 \frac{\Delta t_{33}}{w_{33}} \\
&+ \lambda_3 (1 - \sum_{j=1}^{3} w_{3j}) \\
&- N_{41} r_1 \frac{\Delta t_{41}}{w_{41}} \\
&- N_{42} r_2 \frac{\Delta t_{42}}{w_{42}} \\
&- N_{43} r_3 \frac{\Delta t_{43}}{w_{43}} \\
&+ \lambda_4 (1 - \sum_{j=1}^{3} w_{4j}) \\
&+ \sum_{j=1}^{n} k_j \sum_{i=1}^{n} N_{ij}. \qquad (85)
\end{aligned}
$$

Thus the total revenue has the *general* form - not only the case of four nodes

$$\begin{aligned} F &= -\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{N_{ij}r_j\Delta t_{ij}}{w_{ij}} \\ &+ \sum_{j=1}^{m}k_j\sum_{i=1}^{n}N_{ij} \\ &+ \sum_{i=1}^{n}\lambda_i(1-\sum_{j=1}^{m}w_{ij}), \end{aligned} \tag{86}$$

where n is the number of nodes, and m is the number of service classes, Algorithm for updating weights w_{ij} optimally is obtained by taking derivative and remembering that $\sum_{j=1}^{m} w_{ij} = 1$:

$$\frac{\partial F}{\partial w_{ij}} = \frac{N_{ij}r_j\Delta t_{ij}}{w_{ij}^2} - \lambda_i = 0, \tag{87}$$

$$\begin{aligned} w_{ij} &= \frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sqrt{\lambda_i}} \\ &= \frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sqrt{\lambda_i}\sum_{l=1}^{m}w_{il}} \\ &= \frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sqrt{\lambda_i}\sum_{l=1}^{m}\frac{\sqrt{N_{il}r_l\Delta t_{il}}}{\sqrt{\lambda_i}}} \\ &= \frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sum_{l=1}^{m}\sqrt{N_{il}r_l\Delta t_{il}}}. \end{aligned} \tag{88}$$

So

$$\frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sqrt{\lambda_i}} = \frac{\sqrt{N_{ij}r_j\Delta t_{ij}}}{\sum_{l=1}^{m}\sqrt{N_{il}r_l\Delta t_{il}}}, \tag{89}$$

and

$$\lambda_i = \left(\sum_{l=1}^{m}\sqrt{N_{il}r_l\Delta t_{il}}\right)^2, \tag{90}$$

Derivative (87) is then

$$\frac{\partial F}{\partial w_{ij}} = \frac{N_{ij}r_j\Delta t_{ij}}{w_{ij}^2} - \left(\sum_{l=1}^{m}\sqrt{N_{il}r_l\Delta t_{il}}\right)^2. \tag{91}$$

The uniqueness of the solution is verified as follows:

$$\frac{\partial^2 F}{\partial w_{ij}^2} = -\frac{2N_{ij}r_j\Delta t_{ij}}{w_{ij}^3} < 0. \tag{92}$$

Therefore, F is concave with respect to the weights, and the global optimal solution is

$$w_{ij} = \frac{\sqrt{N_{ij} r_j \Delta t_{ij}}}{\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}, \quad i = 1, \ldots, n, \quad j = 1 \ldots, m, \tag{93}$$

as stated in the theorem. **Q.E.D.**

Solving weights out from the revenue expression (86), one obtains formula

$$\begin{aligned} F &= -\sum_{i=1}^{n} \sum_{j=1}^{m} \sqrt{N_{ij} r_j \Delta t_{ij}} \sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}} \\ &+ \sum_{j=1}^{m} k_j \sum_{i=1}^{n} N_{ij} \\ &= -\sum_{i=1}^{n} (\sum_{j=1}^{m} \sqrt{N_{ij} r_j \Delta t_{ij}})^2 + \sum_{j=1}^{m} k_j \sum_{i=1}^{n} N_{ij} \end{aligned} \tag{94}$$

to the revenue. The delay updating rule (143) has an important advantage that it allows *local updating in the nodes*, because in the calculation of w_{ij} there are used only the parameters of the node i:

- $r_1, \ldots, r_m$ are the same for all nodes,
- $N_{i1}, \ldots, N_{im}$ are the number of connections in the node i at different service classes,
- Parameters $\Delta t_{i1}, \ldots, \Delta t_{im}$ are the delays in the node i.

In the more general case - where the closed form optimal solution is possible to achieve - the pricing function may be

$$r_j(d) = -r_j d^{-p} + k_j, \tag{95}$$

where $p > 0$ to guarantee the decreasing price when delay is increasing. Then

$$F = -\sum_{i=1}^{n} \sum_{j=1}^{n} N_{ij} r_j \frac{\Delta t_{ij}}{w_{ij}}^{-p} + \sum_{j=1}^{m} k_j \sum_{i=1}^{m} N_{ij} + \sum_{i=1}^{n} \lambda_i (1 - \sum_{j=1}^{m} w_{ij}), \tag{96}$$

$$w_{ij} = \frac{(N_{ij} r_j \Delta t_{ij})^{-\frac{1}{p+1}}}{\sum_{l=1}^{m} (N_{il} r_l \Delta t_{il})^{-\frac{1}{p+1}}}. \tag{97}$$

The second derivative is

$$\frac{\partial^2 F}{\partial w_{ij}^2} = -(p+1) p N_{ij} r_j \Delta t_{ij} w_{ij}^{-(p+2)} < 0, \tag{98}$$

when $p > 0$.

To see that the number of connections cannot become infinity, consider the revenue formula F in Eq. (94). For simplicity, assume here that the number of

connections is continuous variable. The second order derivative with respect to the number of connections N_{ij} is

$$\frac{\partial^2 F}{\partial N_{ij}^2} = \frac{1}{2} r_j \Delta t_{ij}^2 \frac{-\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}} \sqrt{N_{ij} r_j \Delta t_{ij}} + 1}{\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}} (N_{ij} r_j \Delta t_{ij})^{\frac{3}{2}}}. \tag{99}$$

Then

$$\frac{\partial^2 F}{\partial N_{ij}^2} < 0, \tag{100}$$

when

$$-\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}} \sqrt{N_{ij} r_j \Delta t_{ij}} + 1 < 0. \tag{101}$$

But this is the case, when N_{ij} (and hence Δt_{ij}, too) becomes sufficiently large. In that domain, F is concave, having only one maximum. On the other hand, the first order derivative is

$$\begin{aligned} \frac{\partial F}{\partial N_{ij}} &= -\frac{\sqrt{r_j \Delta t_{ij}} \sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}{\sqrt{N_{ij}}} + k_j \\ &= -r_j \Delta t_{ij} - \frac{\sqrt{r_j \Delta t_{ij}} \sum_{l \neq j}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}{\sqrt{N_{ij}}}. \end{aligned} \tag{102}$$

When N_{ij} increases, Δt_{ij} increases. Then the first order derivative becomes negative for sufficiently large values of N_{ij} and Δt_{ij}. As a consequence, F begins the decrease. Therefore, finite positive pricing factors k_j in the pricing function (75) or the revenue formula (94) can never be so large that they should dominate the revenue formula, when the sufficiently large number of connections exists; on that condition, negative "penalty" term becomes dominating. Then, the optimal value of N_{ij} is finite.

Upper bound for the revenue is obtained as follows:

Theorem 2: *Upper bound for revenue (94) for optimal weights for given number of connections and delays is*

$$F \leq \sum_{i=1}^{n} \sum_{j=1}^{m} N_{ij} (k_j - r_j \Delta t_{ij}). \tag{103}$$

Proof: Notice that in Eq. (94),

$$(\sum_{j=1}^{m} \sqrt{N_{ij} r_j \Delta t_{ij}})^2 \geq \sum_{j=1}^{m} N_{ij} r_j \Delta t_{ij}. \tag{104}$$

Then

$$F \leq -\sum_{i=1}^{n} \sum_{j=1}^{m} N_{ij} r_j \Delta t_{ij} + \sum_{j=1}^{m} k_j \sum_{i=1}^{n} N_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{ij} (k_j - r_j \Delta t_{ij}). \tag{105}$$

Q.E.D.

The natural interpretation of the result is that

- positive pricing factors k_j increase upper bound,
- positive penalty factors r_j decrease upper bound, and
- positive delays Δt_{ij} - which also penalize revenue - decrease upper bound.

It is difficult rigorously to see, if the revenue is positive for some number N_{ij} of the connections, and what is the maximum revenue for the optimal choice of N_{ij}. However, we perform semi-analytical consideration of the case. Again, we must assume that N_{ij} is continuous. First, notice that $F = 0$, when no connections exist. Then, if the first order derivative with respect to N_{ij} is positive in the vicinity of $N_{ij} = 0$, F is increasing in the vicinity of $F = 0$, and it becomes positive, when N_{ij} increases. Let us take the first order derivative of optimal F in Eq. (94) as follows:

$$\frac{\partial F}{\partial N_{ij}} = -\frac{\sqrt{r_j \Delta t_{ij}} \sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}{\sqrt{N_{ij}}} + k_j. \tag{106}$$

Consider the case, where factor $r_j \Delta t_{ij}$ is so small that

$$r_j \Delta t_{ij} \leq 1. \tag{107}$$

This inequality is always possible to get by rescaling of pricing factors r_j and units of money to be paid (e.g. U.S. dollars are replaced by English pounds). In addition, it is quite natural that Δt_{ij} should be small for small N_{ij}. Then

$$\sqrt{r_j \Delta t_{ij}} \leq 1. \tag{108}$$

From the first order derivative, we then obtain the inequalities

$$\begin{aligned} \frac{\partial F}{\partial N_{ij}} &\geq -\frac{\sum_{l=1}^{m} \sqrt{N_{il} r_l \Delta t_{il}}}{\sqrt{N_{ij}}} + k_j \\ &\geq -\frac{\sum_{l=1}^{m} \sqrt{N_{il}}}{\sqrt{N_{ij}}} + k_j. \end{aligned} \tag{109}$$

Then, if

$$k_j \geq \frac{\sum_{l=1}^{m} \sqrt{N_{il}}}{\sqrt{N_{ij}}}, \tag{110}$$

F is increasing, and it is positive for the suitably chosen number N_{ij} of connections. It is quite straigthforward to construct such combinations of the variables and parameters that obey this condition. Notice from the second order derivative (99), that for very small products $N_{ij} r_j \Delta t_{ij}$, revenue function F is convex. Then, if buffers are almost empty, F is positive, increasing, and convex.

Minimization of the weighted mean delay

Next we show that the revenue maximization formula also minimizes the overall weighted mean delay.

Theorem 3: *Weighted mean delay* $E(r_j d) = E\left(r_j \frac{\Delta t_{ij}}{w_{ij}}\right)$ *is minimized by Eq. (143), and it is unique.*

Proof: First use again the four-node case, and notice that when data is transmitted through switches i_1 and i_2 in the class j, the delay is

$$d = \frac{\Delta t_{i_1,j}}{w_{i_1,j}} + \frac{\Delta t_{i_2,j}}{w_{i_2,j}}. \tag{111}$$

Because the number of these connections is $N_j^{i_1 \to i_2}$, then the weight for the corresponding delay is

$$N_j^{i_1 \to i_2} \left(\frac{\Delta t_{i_1,j}}{w_{i_1,j}} + \frac{\Delta t_{i_2,j}}{w_{i_2,j}} \right). \tag{112}$$

By using similar regrouping strategy as used for deriving the revenue formula, and remembering that the sum of the weighting connection terms (63)-(74) becomes scaling factors in the denominator, we get the general expression for unweighted overall mean delay as follows:

$$E(d) = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij}} \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{N_{ij}\Delta t_{ij}}{w_{ij}}. \tag{113}$$

Then, weighted mean delay optimization formula is

$$\begin{aligned} E(r_j d) &= \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij}} \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{N_{ij} r_j \Delta t_{ij}}{w_{ij}} \\ &+ \sum_{i=1}^{n} \lambda_i (1 - \sum_{j=1}^{m} w_{ij}). \end{aligned} \tag{114}$$

The first order derivative is

$$\frac{\partial E(r_j d)}{\partial w_{ij}} = -\frac{N_{ij} r_j \Delta t_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl} w_{ij}^2} - \lambda_i. \tag{115}$$

The weights are solved as follows:

$$\begin{aligned} w_{ij} &= \sqrt{-\lambda_i \frac{N_{ij} r_j \Delta t_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}} \\ &= \frac{\sqrt{-\lambda_i \frac{N_{ij} r_j \Delta t_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}}}{\sum_{h=1}^{m} w_{ih}} \\ &= \frac{\sqrt{-\lambda_i \frac{N_{ij} r_j \Delta t_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}}}{\sum_{h=1}^{m} \sqrt{-\lambda_i \frac{N_{ih} r_h \Delta t_{ih}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}}} \\ &= \frac{\sqrt{N_{ij} r_j \Delta t_{ij}}}{\sum_{h=1}^{m} \sqrt{N_{ih} r_h \Delta t_{ih}}}. \end{aligned} \tag{116}$$

But this is the same as Eq. (143). Uniqueness is derived as follows. First, λ_i is solved from Eq. (116):

$$\sqrt{\frac{-\lambda_i}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}} = \frac{1}{\sum_{h=1}^{m}\sqrt{N_{ih} r_h \Delta t_{ih}}}, \tag{117}$$

$$\lambda_i = -\frac{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}{(\sum_{h=1}^{m}\sqrt{N_{ih} r_h \Delta t_{ih}})^2} \tag{118}$$

Then the first order derivative is

$$\begin{aligned} \frac{\partial E(r_j d)}{\partial w_{ij}} &= -\frac{1}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}} \frac{N_{ij} r_j \Delta t_{ij}}{w_{ij}^2} \\ &+ \frac{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl}}{(\sum_{h=1}^{m}\sqrt{N_{ih} r_h \Delta t_{ih}})^2}. \end{aligned} \tag{119}$$

The second order derivative is

$$\frac{\partial^2 E(r_j d)}{\partial w_{ij}^2} = \frac{2 N_{ij} r_j \Delta t_{ij}}{\sum_{k=1}^{n}\sum_{l=1}^{m} N_{kl} w_{ij}^3} > 0, \tag{120}$$

which guarantees unique minimum in the interval $0 < w_{ij} \leq 1$. That completes proof. **Q.E.D.**

The conclusion is that because the optimal revenue is obtained by the same updating rule as the optimal minimum weighted mean delays, both the service provider and the customers can be considered satisfied using our approach.

Weights can be solved out from the expression of the weighted mean delay formula (114), and it becomes

$$E(r_j d) = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij}} \sum_{i=1}^{n} (\sum_{j=1}^{m} \sqrt{N_{ij} r_j \Delta t_{ij}})^2. \tag{121}$$

Call Admission Control

In the Call Admission Control, the revenue is checked and weights are updated in the *connection level*, not in the packet level, to avoid too costly computation. Weights are then updated only when the connection appears (or disappears). Let the state - i.e. number of connections - at the discrete time moment t be $N_{ij}(t)$, $t = 1, \ldots, m$. Let the new hypothetical state at the moment $t+1$ be $\tilde{N}_{ij}(t+1)$, $t = 1, \ldots, m$, when one or several connections appear in some class / classes. In hypothesis testing, revenue formula (94) is applied as follows:

$$\begin{aligned} F(t) &= -\sum_{i=1}^{n} (\sum_{j=1}^{m} \sqrt{N_{ij}(t) r_j \Delta t_{ij}})^2 \\ &+ \sum_{j=1}^{m} k_j \sum_{i=1}^{n} N_{ij}(t) \end{aligned} \tag{122}$$

$$\begin{aligned}\tilde{F}(t+1) &= -\sum_{i=1}^{n}(\sum_{j=1}^{m}\sqrt{\tilde{N}_{ij}(t+1)r_j\Delta t_{ij}})^2 \\ &+ \sum_{j=1}^{m}k_j\sum_{i=1}^{n}\tilde{N}_{ij}(t+1) \qquad (123)\end{aligned}$$

If $F(t) > \tilde{F}(t+1)$, then the call is rejected, otherwise it is accepted.

Experiments

In this section we demonstrate by simulation the operation of the linear pricing algorithm (75) with $m = 3$ service classes and four nodes. We compare the optimal weights (143) with a version of the algorithm that obtains the weights by "brute-force", for which the possible values are $w_{ij} = 0.1, \ldots, 0.9$ (i.e. the step size is 0.1). We present five different scenarios, which illustrate the behavior of the algorithm with and without CAC mechanism. We also present ways to guarantee different maximum delays for the users, but at the expense of reduced revenue for the operator.

The connections in the different service classes have different average data packet sizes $E(b_1) = 50$, $E(b_2) = 25$ and $E(b_3) = 10$, with a standard deviation of 1 (i.e. in a specific service class the connections have similar demand for bandwidth). The processing time of each packet scheduler is chosen as $T = 1/10000$ s/kbyte. The arrival rates of the connections to the nodes 1 and 2 (Fig. 6) are Poisson distributed and they are $\alpha_1 = 0.30$, $\alpha_2 = 0.40$ and $\alpha_3 = 0.50$ per unit time for the gold, silver, and bronze classes, respectively. Each connection has equal probability of being routed to node 3 or 4. The life time of a connection (i.e. the time the connection is served and has packets in the queue) is exponentially distributed. The duration parameters (i.e. "decay rates") for the connections are $\beta_1 = 0.01$, $\beta_2 = 0.007$ and $\beta_3 = 0.003$, where the probability density functions for the durations are

$$p_i(t) = \beta_i e^{-\beta_i t}, \quad , i = 1,2,3 \quad t \geq 0. \qquad (124)$$

The penalty factors were chosen as $r_1 = 2500$, $r_2 = 500$ and $r_3 = 100$ and the shifting factors k_j as $k_1 = 1200$, $k_2 = 800$ and $k_3 = 400$, for the gold, silver and bronze classes, respectively. The number of unit times (i.e. instances in which a new connection may or appear or disappear) in the experiments was $\tau = 2000$.

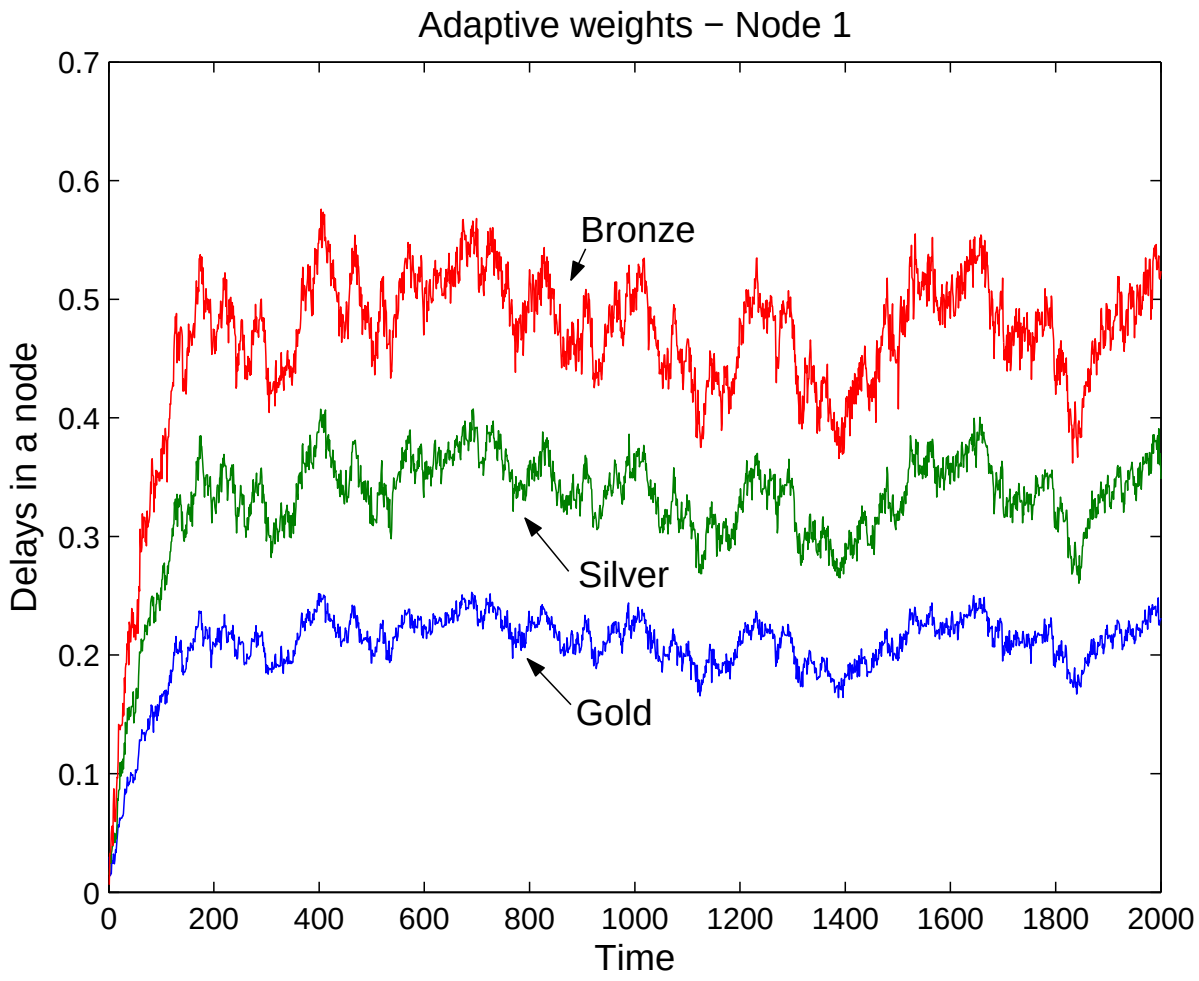


FIGURE 9 Adaptive weights - Evolution of the delays in node 1 as a function of time in the first experiment.

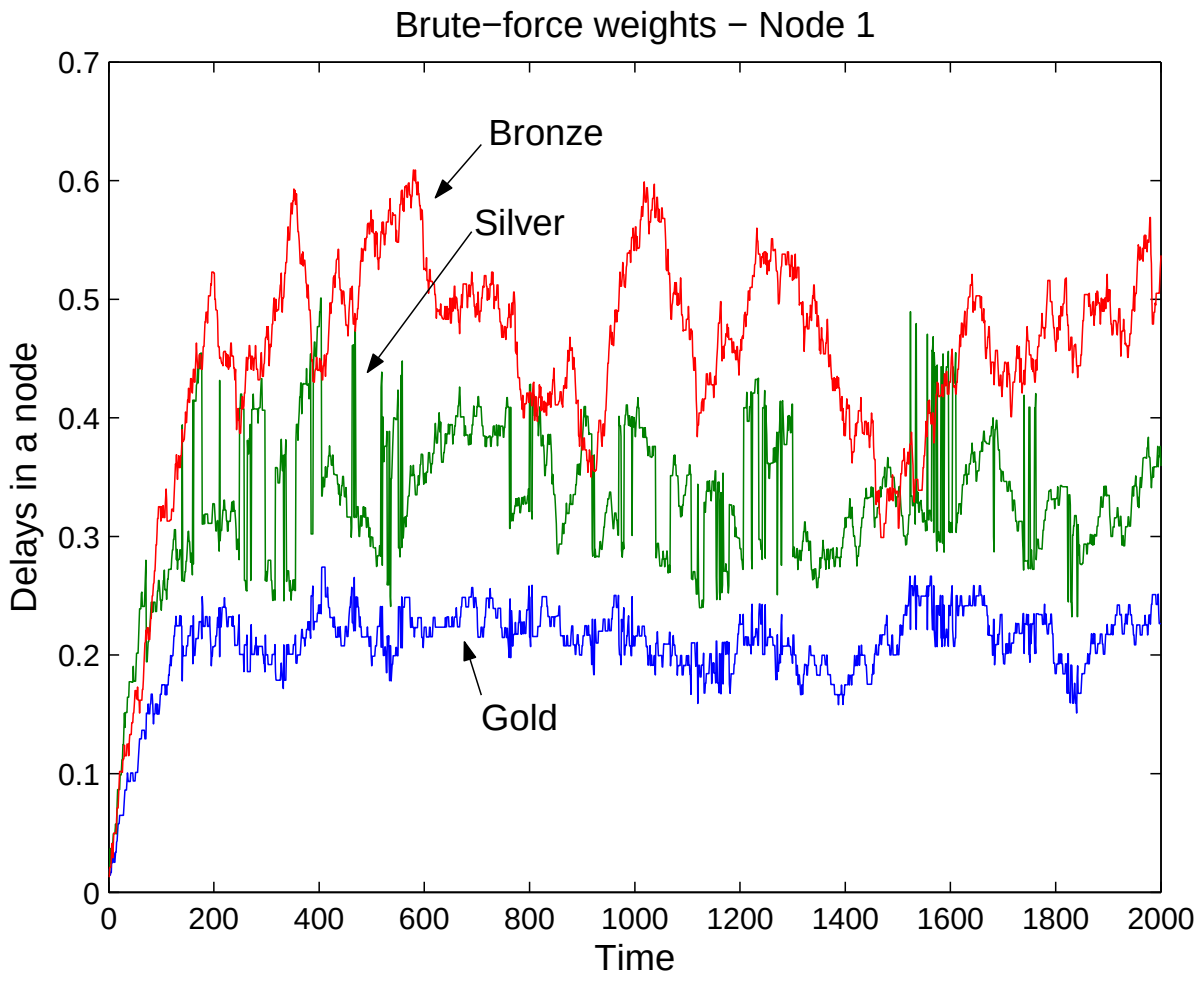


FIGURE 10 “Brute force” weights - Evolution of the delays in node 1 as a function of time in the first experiment.

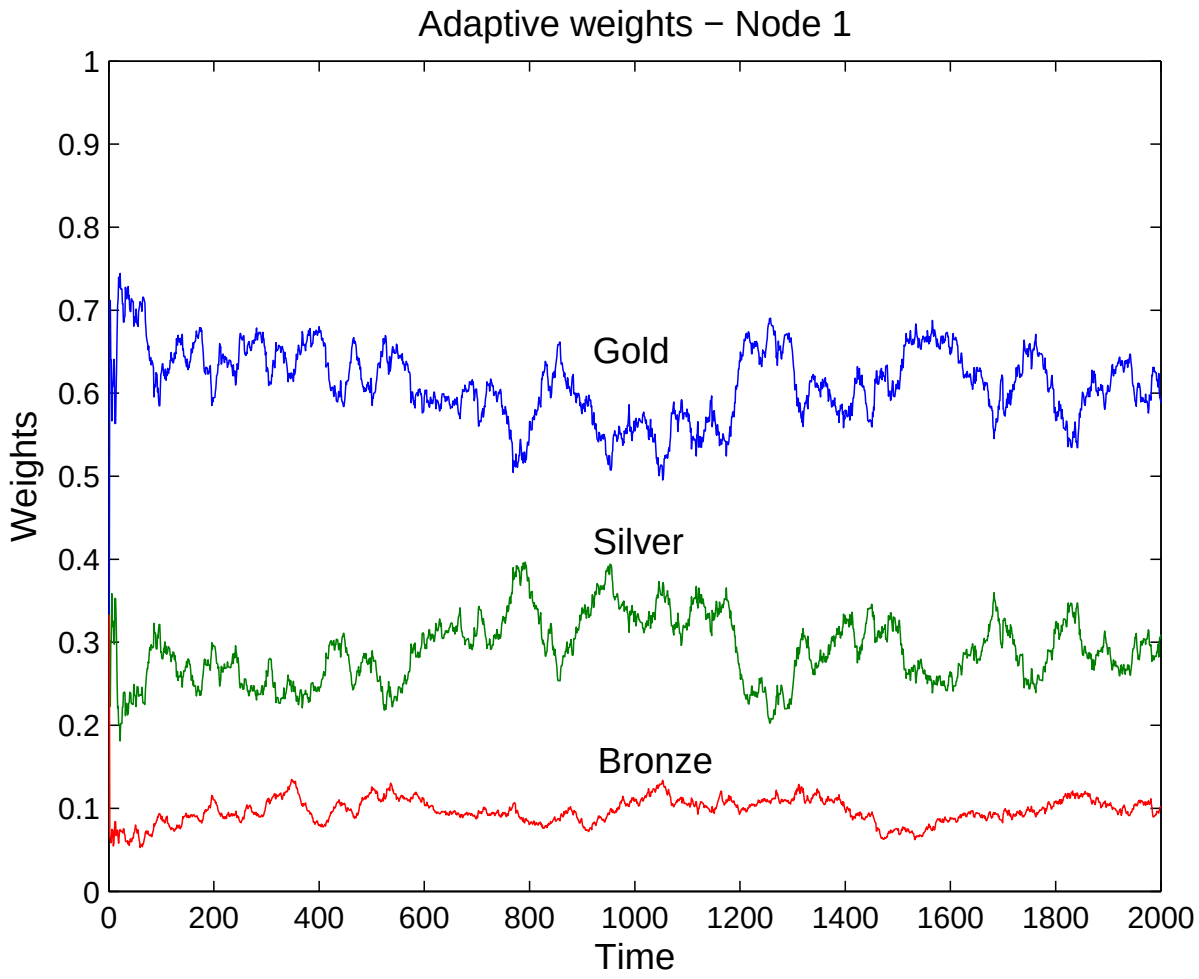


FIGURE 11 Adaptive weights - Evolution of the weights of node 1 as a function of time in the first experiment.

Experiment 1. In this experiment, the CAC mechanism described by (122) and (123) is used to limit the serviced connections. Figs. 9 and 10 show the delays in a single node for adaptive and “brute force” weights, respectively. As can bee seen, the adaptive weights offer smoother delay deriving to a better delay jitter. The delay variation of the “brute force” weights is due to the abrupt large changes in the weight values, which could be compensated by decreasing the step size (e.g. from 0.1 to 0.01) that is used by the “brute force” mechanism, but this would lead to even larger computation times. As an example, Figs. 11 and 12 show the weights in a single node for adaptive and “brute force” weights, respectively. The number of connections of node 1 can be seen in Figs. 13 and 14 for the adaptive and “brute force” weights, respectively.

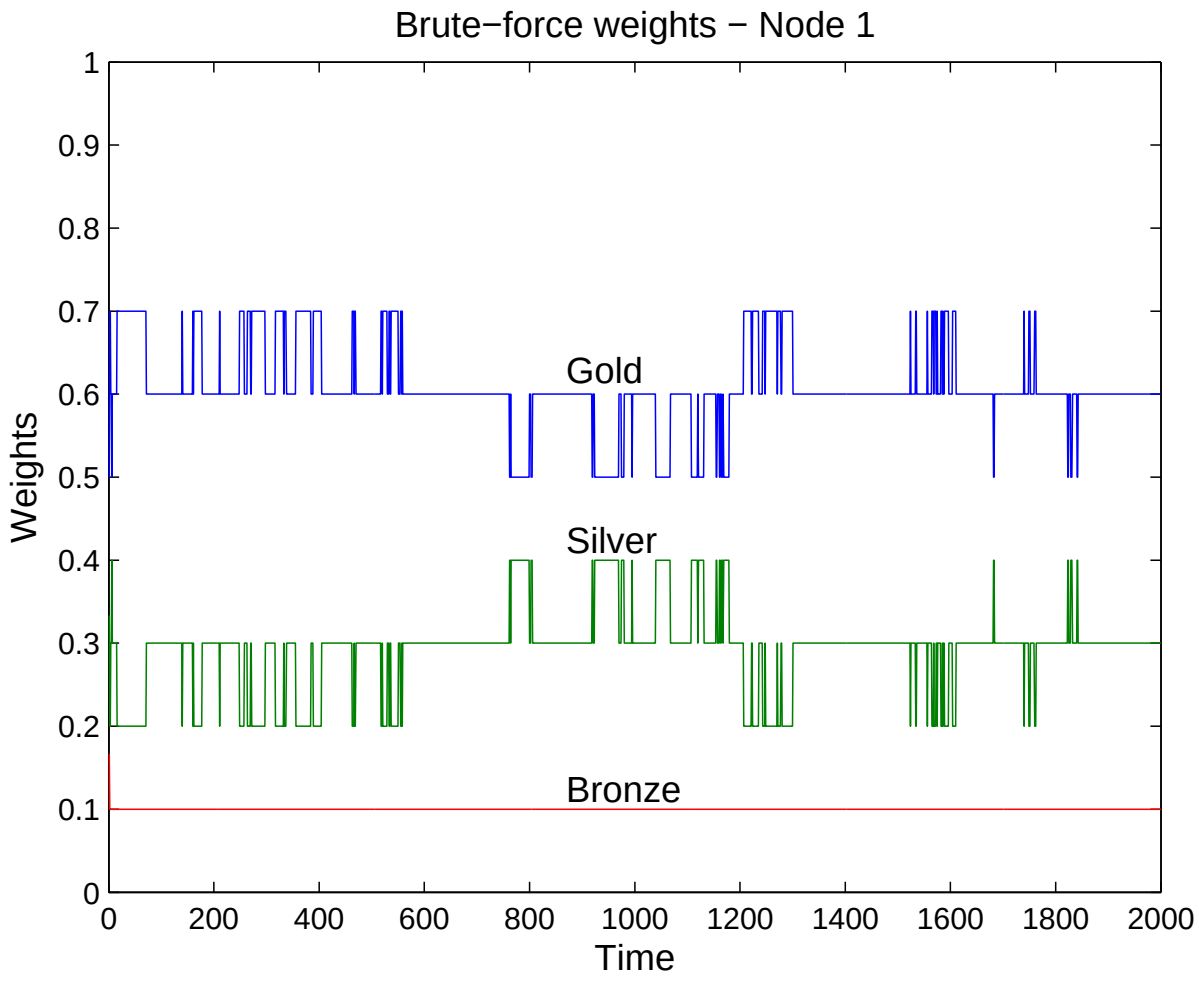


FIGURE 12 “Brute force” weights - Evolution of the weights of node 1 as a function of time in the first experiment.

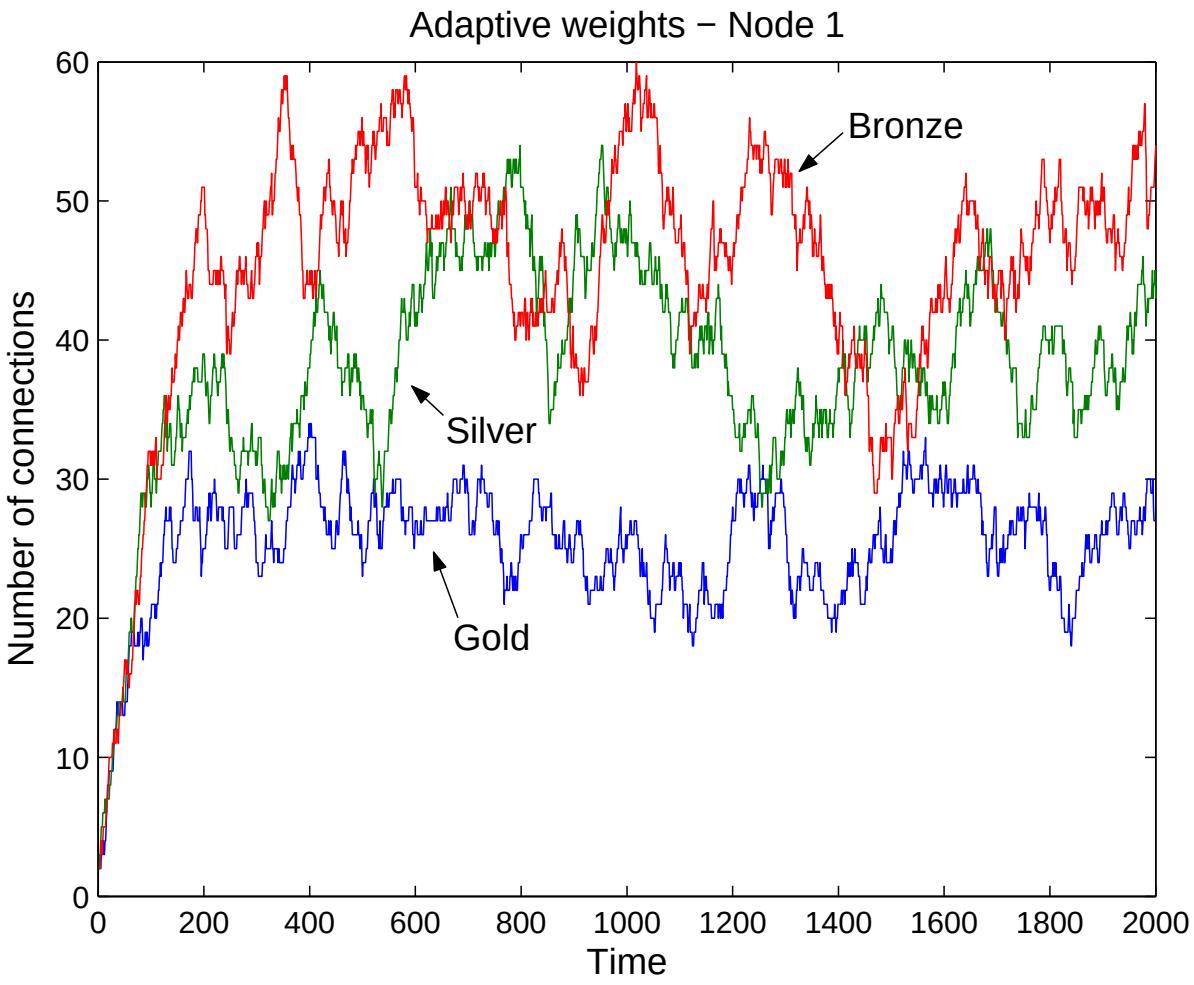


FIGURE 13 Adaptive weights - Evolution of the number of connections as a function of time in the first experiment.

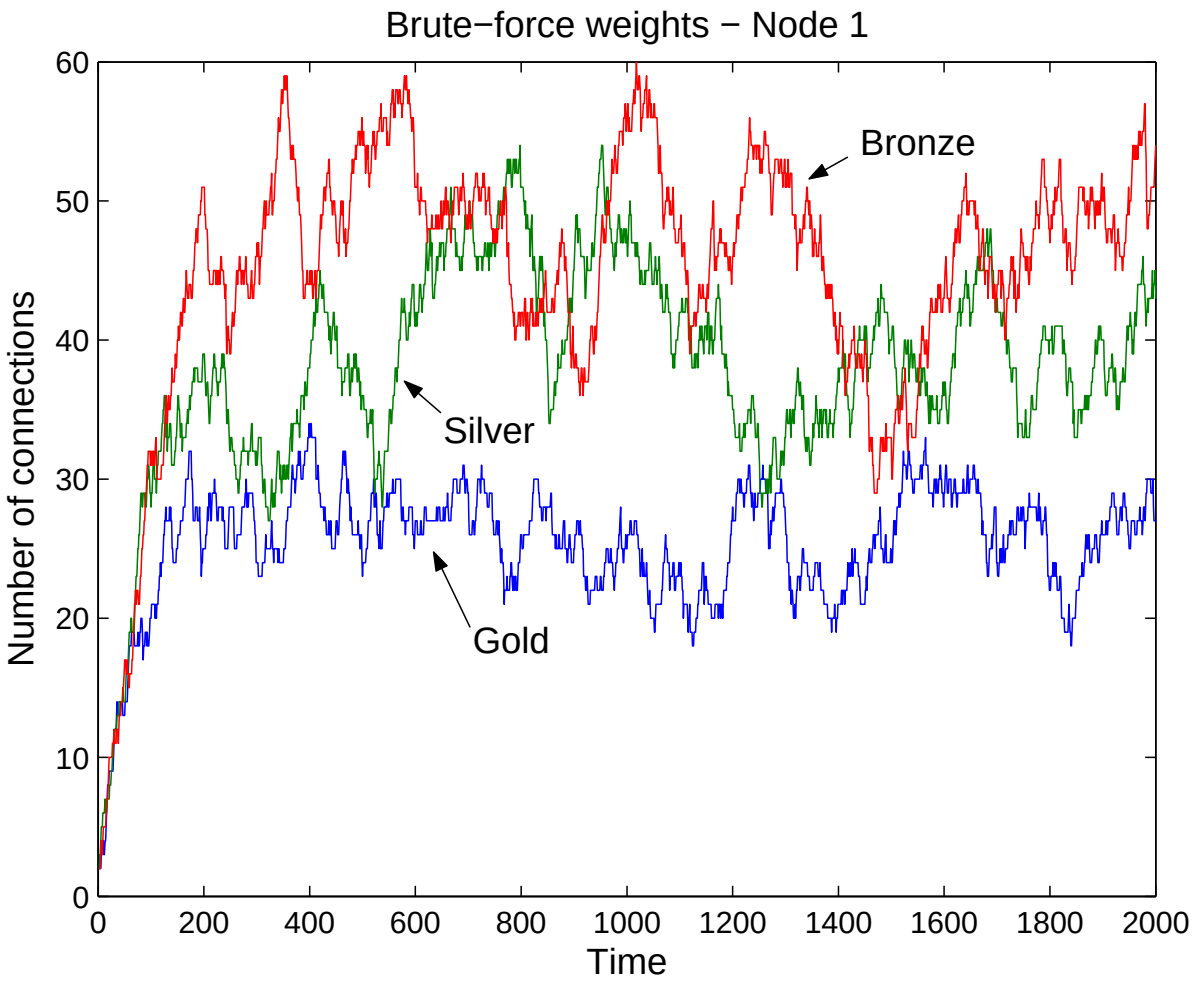


FIGURE 14 “Brute force” weights - Evolution of the number of connections as a function of time in the first experiment.

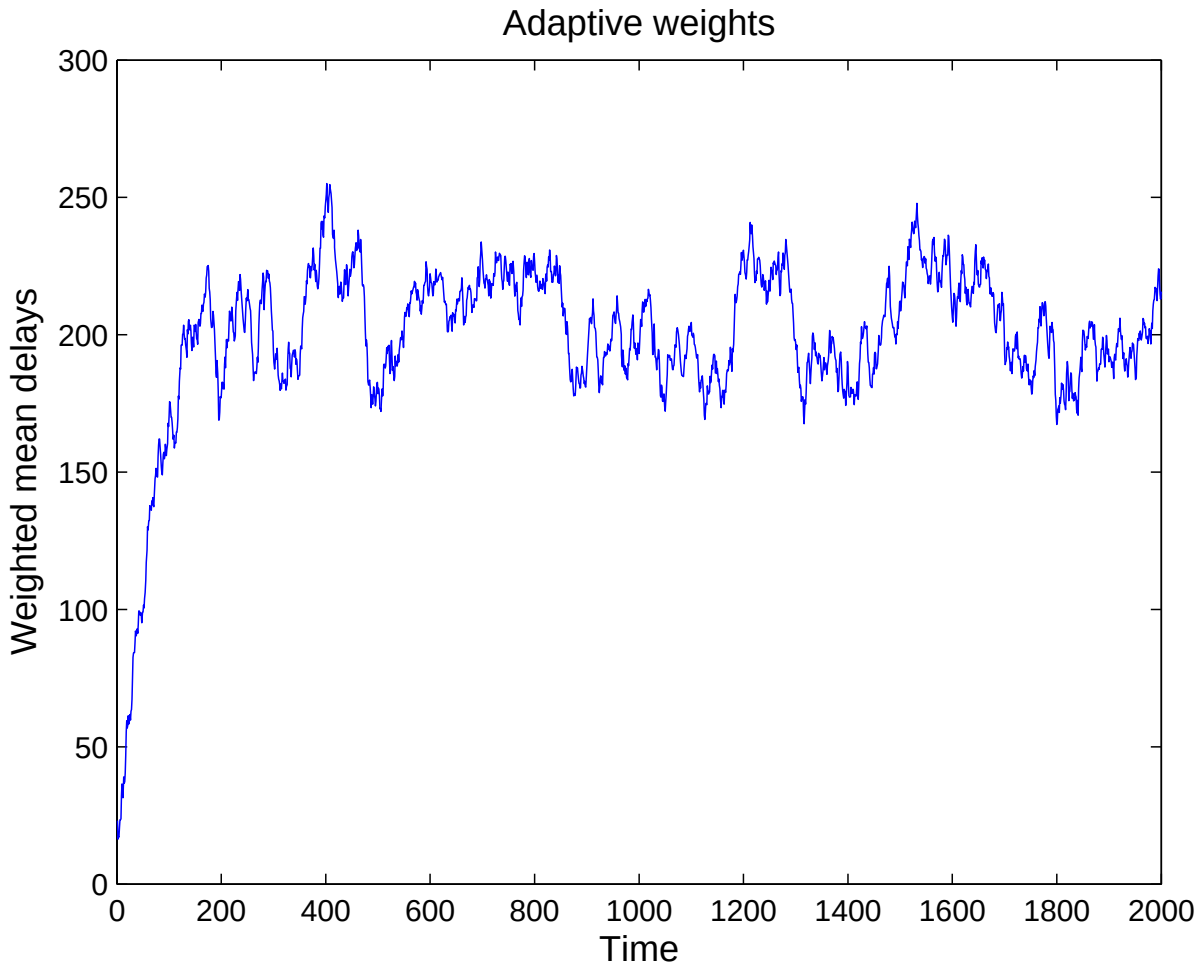


FIGURE 15 Adaptive weights - Evolution of the weighted mean delays as a function of time in the first experiment.

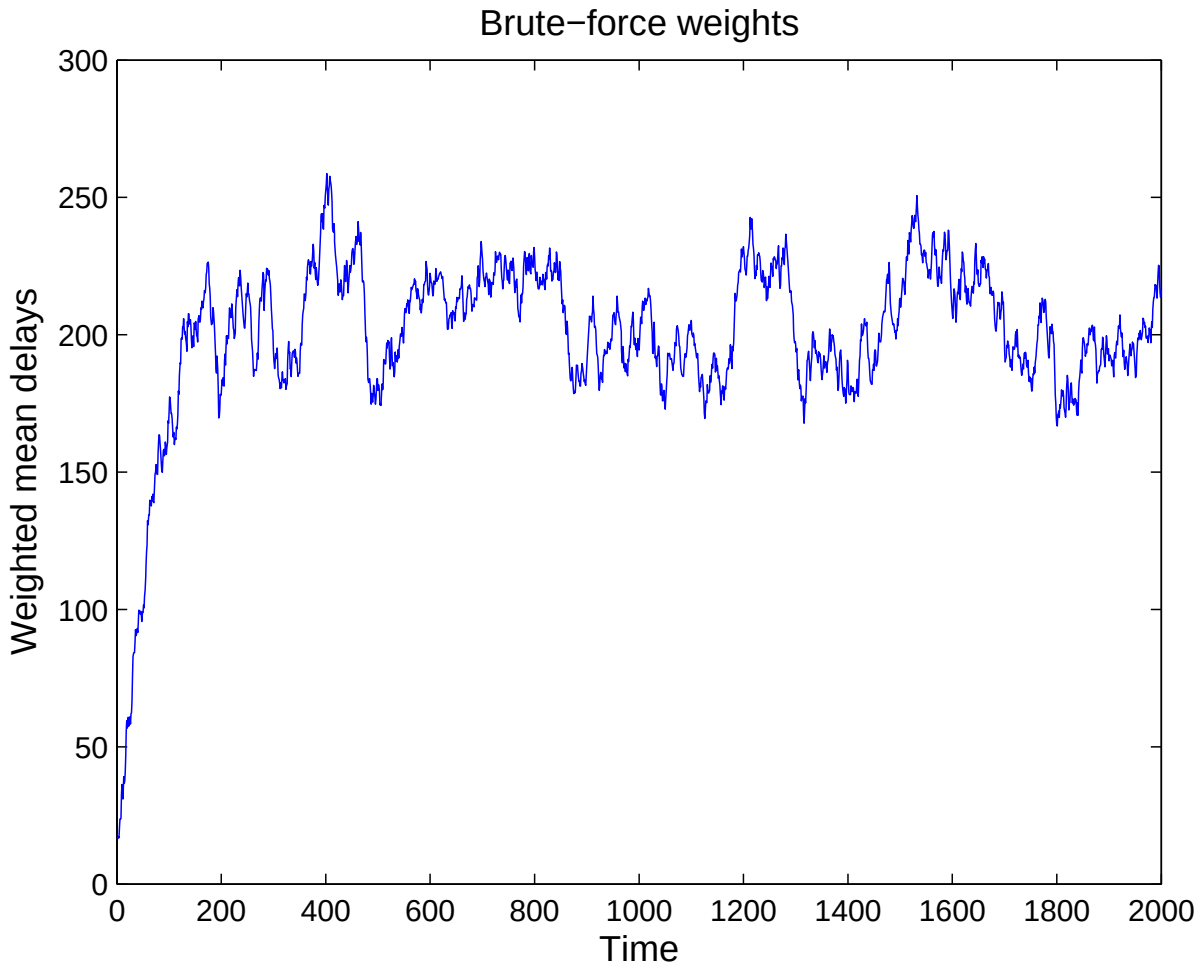


FIGURE 16 “Brute force” weights - Evolution of the weighted mean delays as a function of time in the first experiment.

Figs. 15 and 16 present the weighted mean delays for all classes through the network, for the adaptive and “brute force” weights, respectively. The average values of the weighted mean delays for “brute force” weights is 199.9 which is slightly larger than 199.1 for adaptive weights. From Figs. 17 and 18 it is seen that the revenues are close, with the adaptive weight revenue (mean 231394.7) being slightly greater than the “brute force’ weight revenue (mean 230806.6). The adaptive weights give optimum results with little computational complexity. By using a finer scale on the possible “brute force” weights, would result values closer to the optimum, but at the expense of increased computation time.

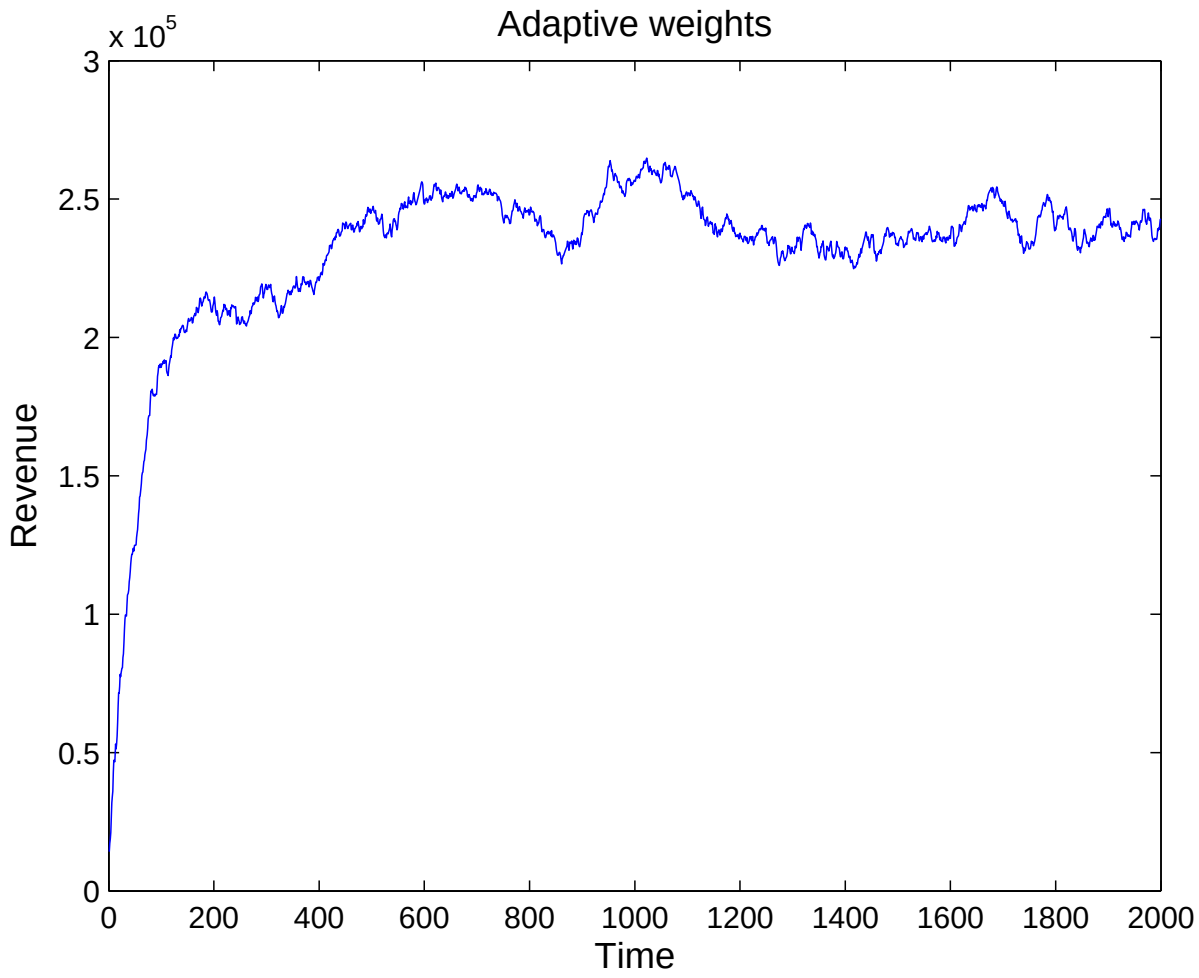


FIGURE 17 Adaptive weights - Evolution of the revenue as a function of time in the first experiment.

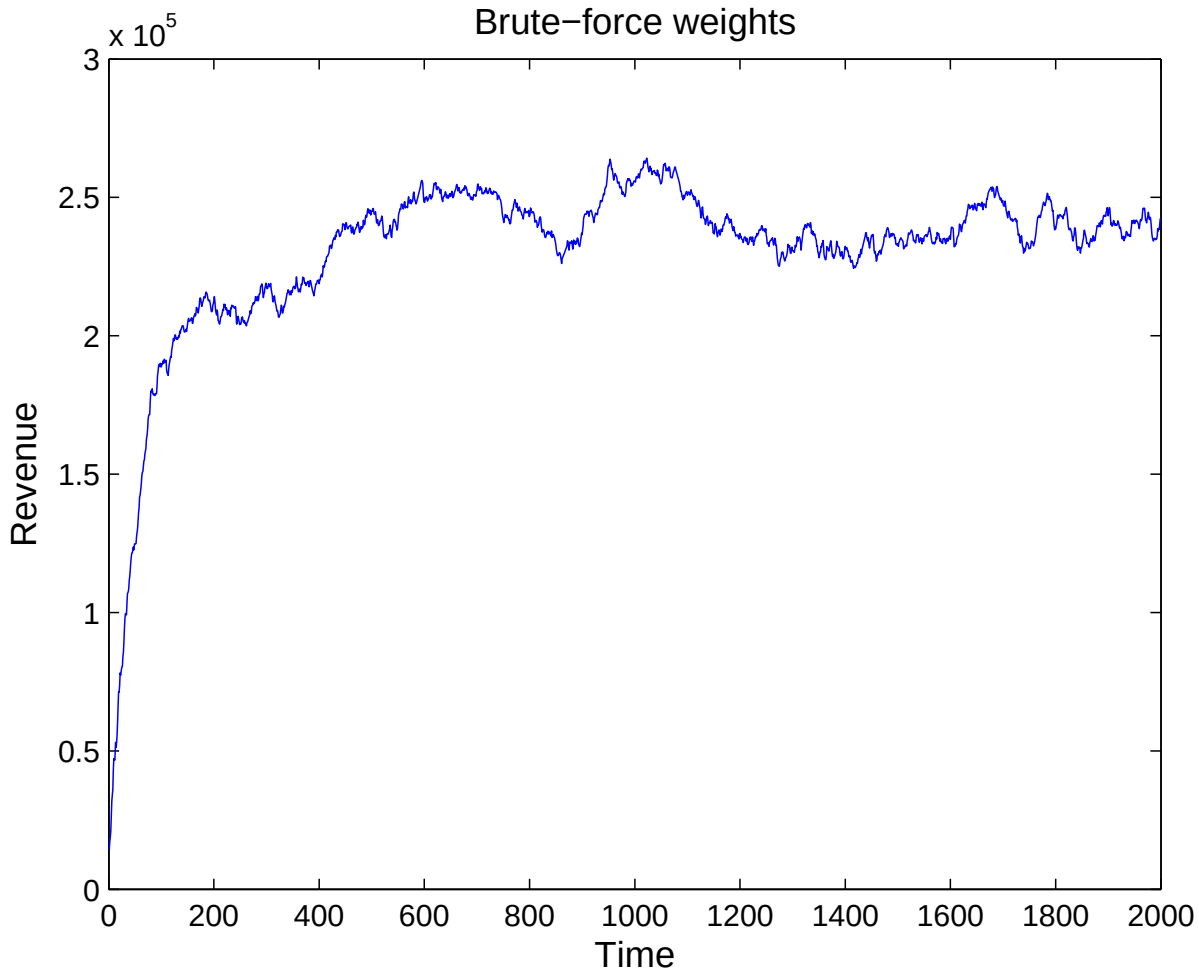


FIGURE 18 “Brute force” weights - Evolution of the revenue as a function of time in the first experiment.

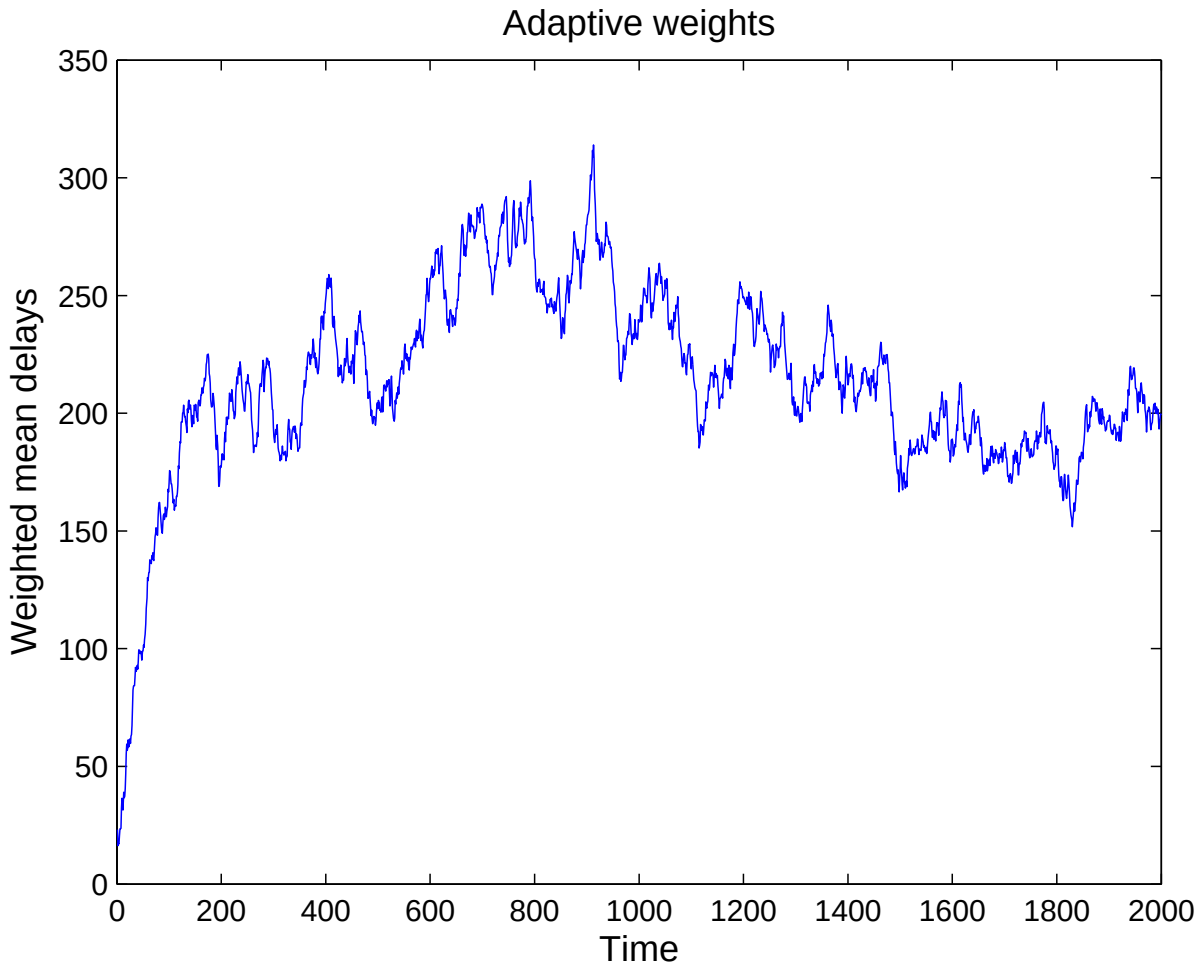


FIGURE 19 Adaptive weights - Evolution of the weighted mean delays as a function of time in the second experiment.

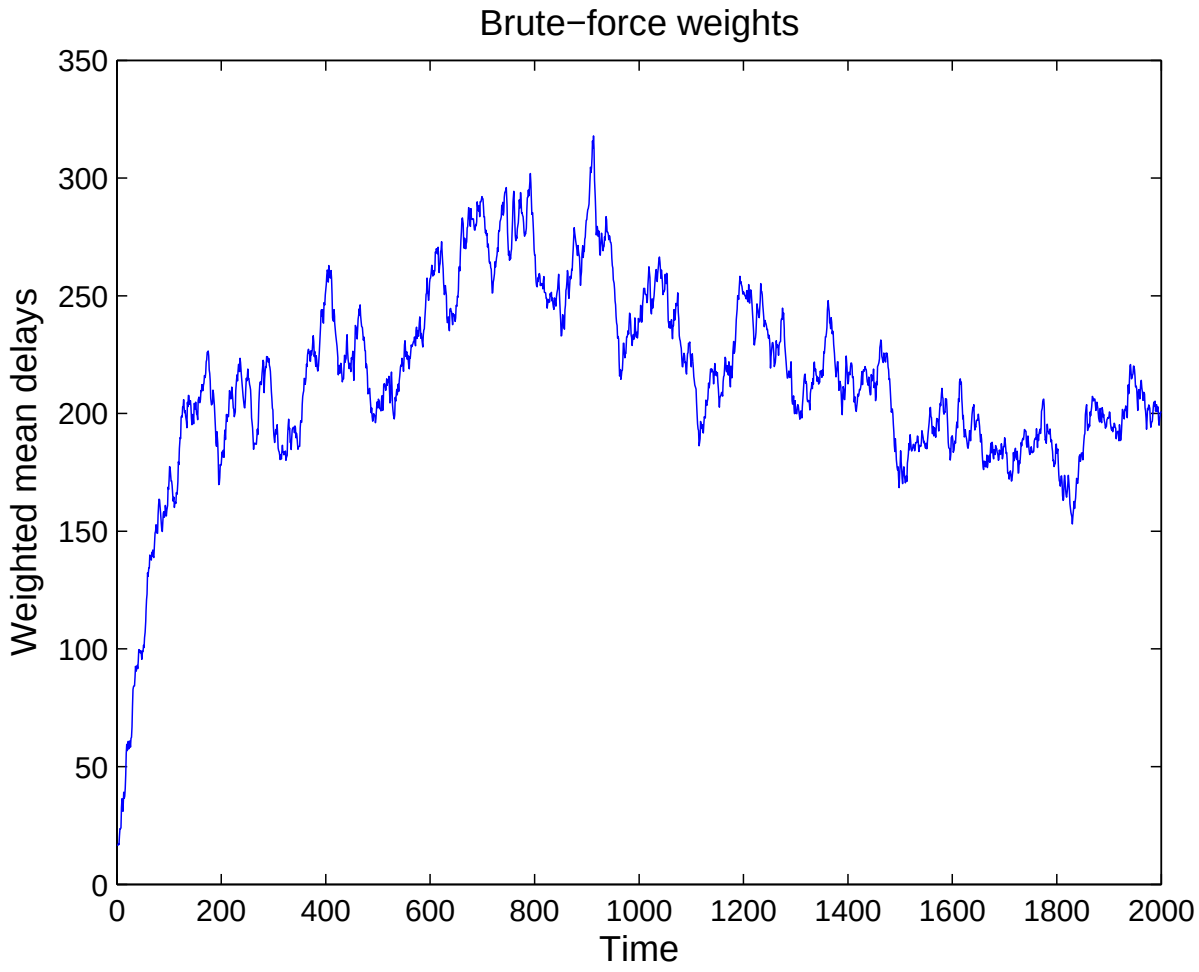


FIGURE 20 "Brute force" weights - Evolution of the weighted mean delays as a function of time in the second experiment.

Experiment 2. In this experiment the CAC mechanism is not used (i.e. all the appearing connections are accepted). By comparing the weighted mean delays of the no CAC case presented in Figs. 19 and 20 to Figs. 15 and 16 of Experiment 1, it noticed that without the CAC mechanism the weighted mean delays are larger. Still, the weighted mean delay for "brute force" weights is larger (mean 214.6) than for adaptive weights (mean 213.2). Also, the revenue without the CAC mechanism is smaller than with the CAC mechanism as can be seen by comparing Figs. 21 and 22 to Figs. 17 and 18. Again, the adaptive weight revenue (mean 229824.6) is greater than the "brute force' weight revenue (mean 229039.5).

Experiment 3. In this experiment the CAC mechanism is accompanied by a guarantee on the minimum delay on the route from node i_1 to i_2. The delay through ingress node i_1 and egress node i_2 for class j is

$$d^j_{i_1,i_2} = \frac{\Delta t_{i_1,j}}{w_{i_1,j}} + \frac{\Delta t_{i_2,j}}{w_{i_2,j}}, \tag{125}$$

where $\Delta t_{i_1,j} = \sum_{k=1}^{N_{i_1,j}} b_{i_1,j,k} \times T$ and $\Delta t_{i_2,j} = \sum_{k=1}^{N_{i_2,j}} b_{i_2,j,k} \times T$ are the delays that depend on the packet sizes of the queue of class j in nodes i_1 and i_2, respectively.

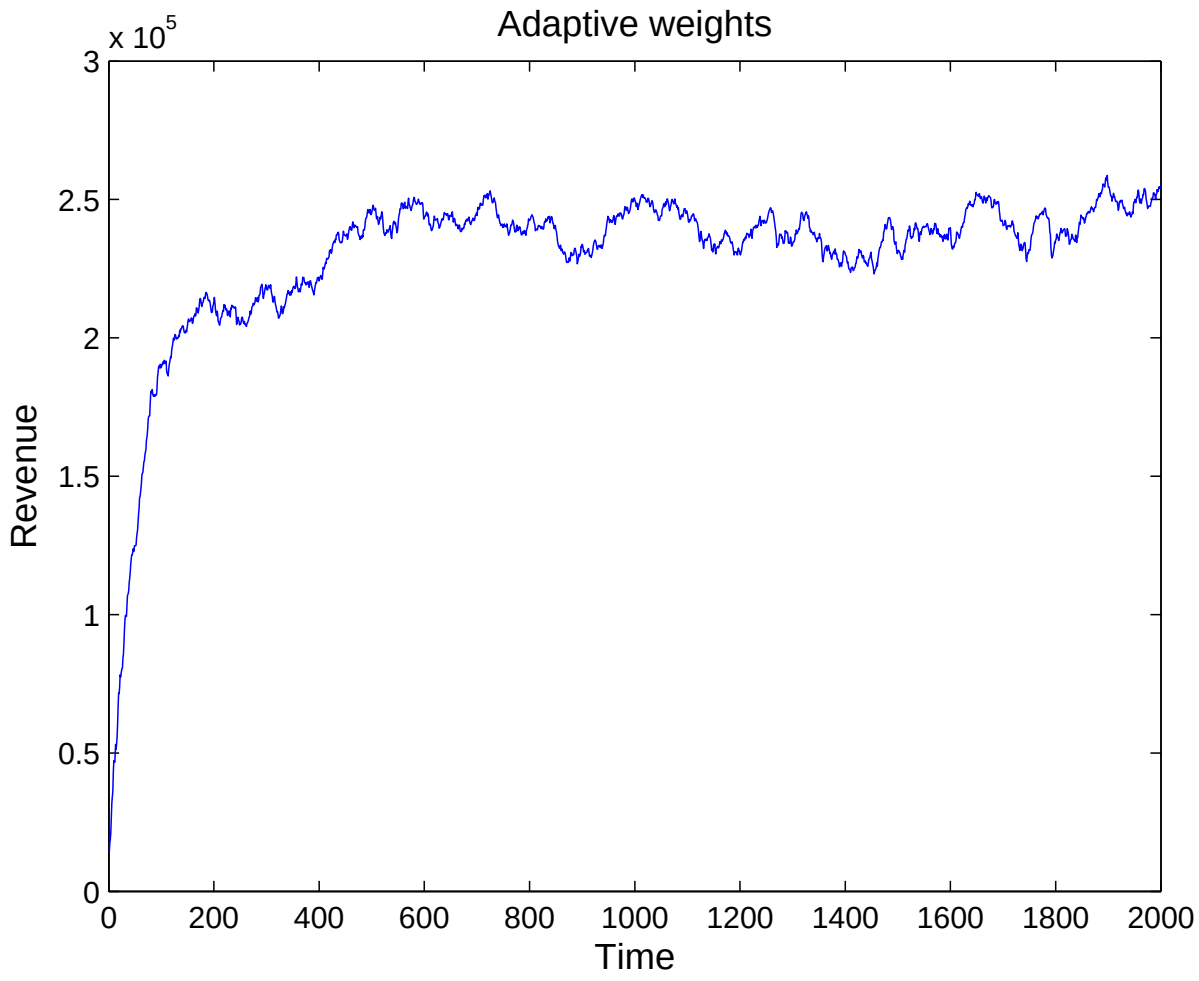


FIGURE 21 Adaptive weights - Evolution of the revenue as a function of time in the second experiment.

Now, we can express (125) at discrete time moment t as

$$d^{j}_{i_1,i_2}(t) = T \times \left(\frac{\sum_{k=1}^{N_{i_1,j}(t)} b_{i_1,j,k}}{w_{i_1,j}} + \frac{\sum_{k=1}^{N_{i_2,j}(t)} b_{i_2,j,k}}{w_{i_2,j}} \right), \tag{126}$$

and when a new connection appears (from node i_1 to i_2), the new hypothetical delay at time $t+1$ is

$$\tilde{d}^{j}_{i_1,i_2}(t+1) = T \times \left(\frac{\sum_{k=1}^{N_{i_1,j}(t+1)} b_{i_1,j,k}}{w_{i_1,j}} + \frac{\sum_{k=1}^{N_{i_2,j}(t+1)} b_{i_2,j,k}}{w_{i_2,j}} \right). \tag{127}$$

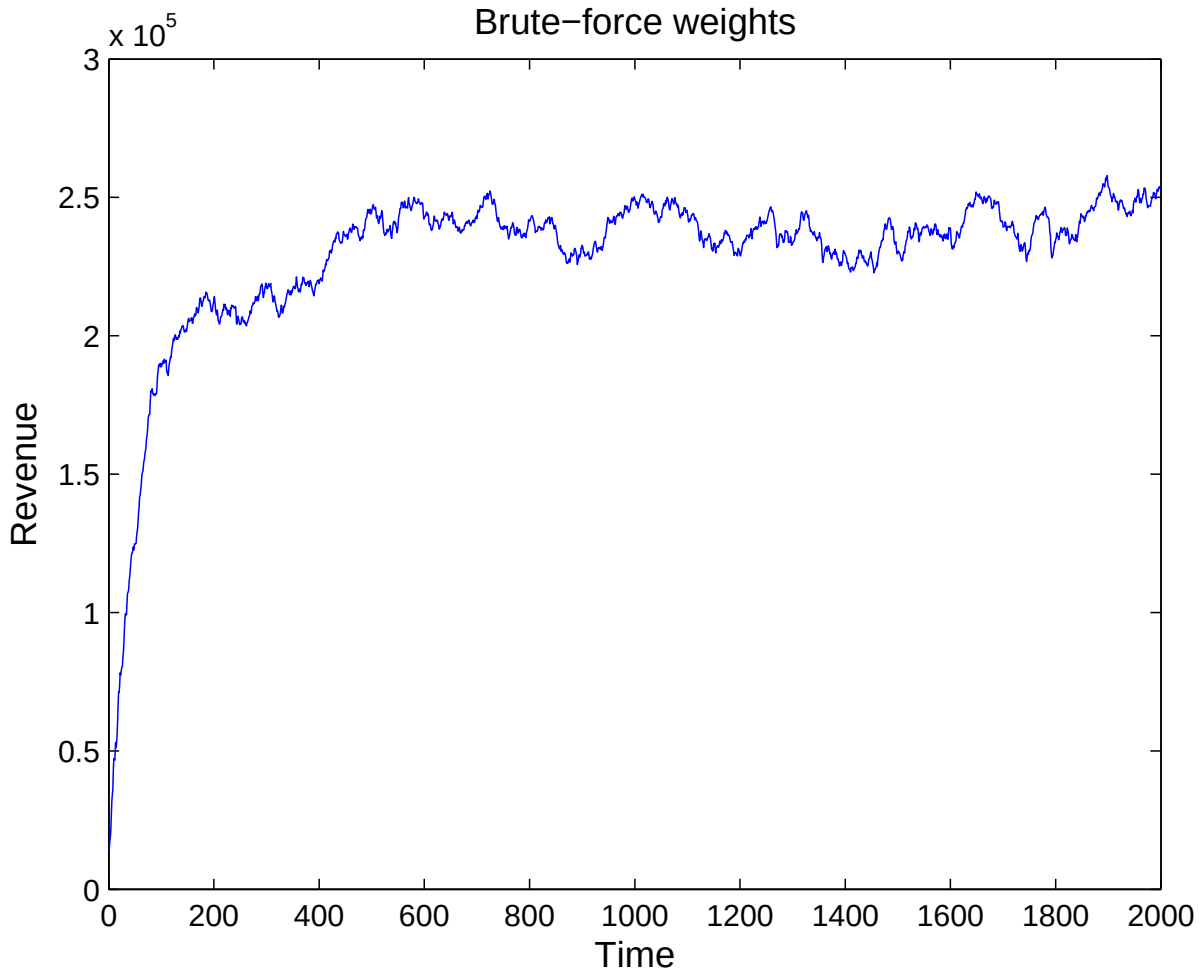


FIGURE 22 “Brute force” weights - Evolution of the revenue as a function of time in the second experiment.

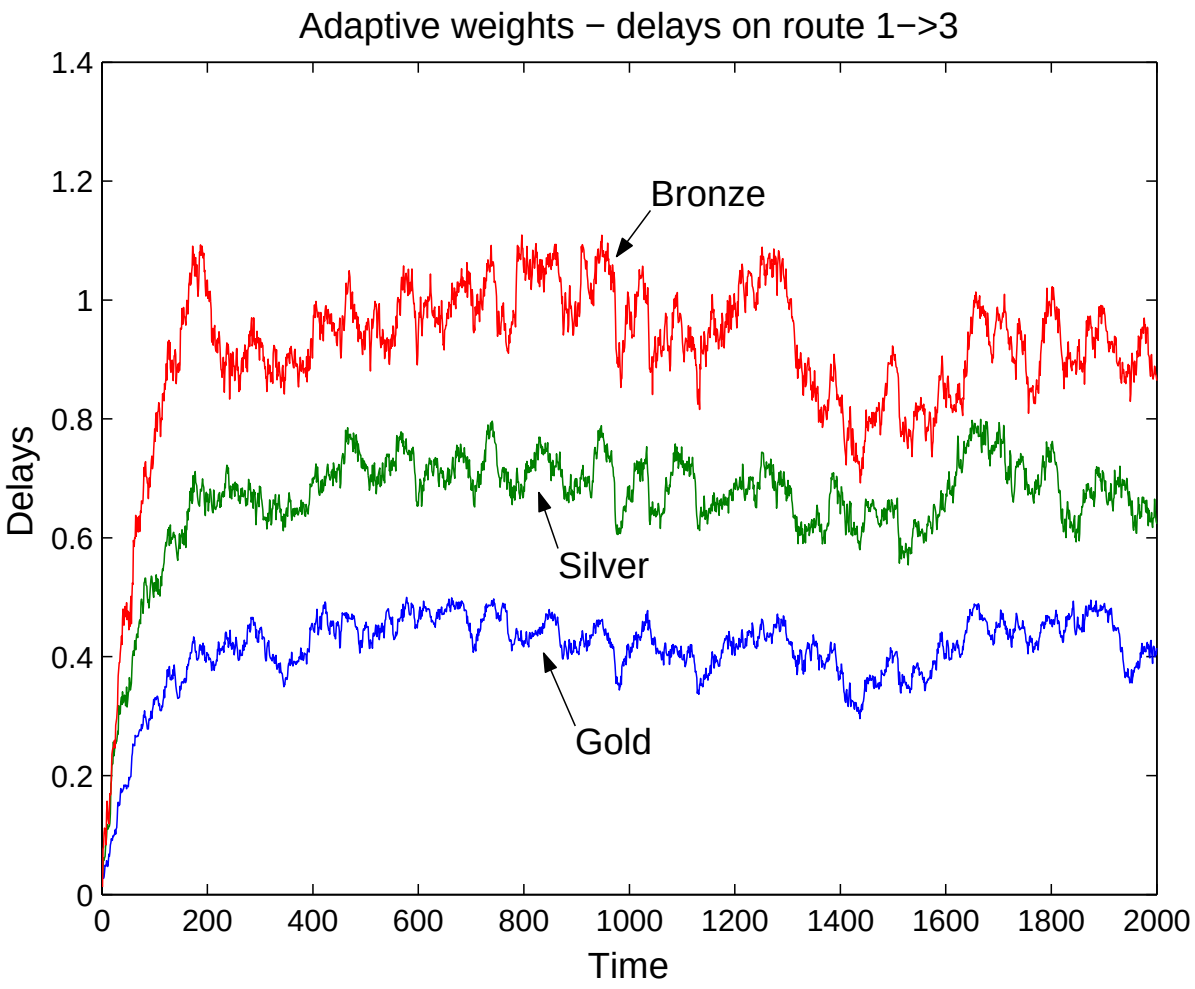


FIGURE 23 Adaptive weights - Evolution of the delays on route 1 → 3 as a function of time in the third experiment.

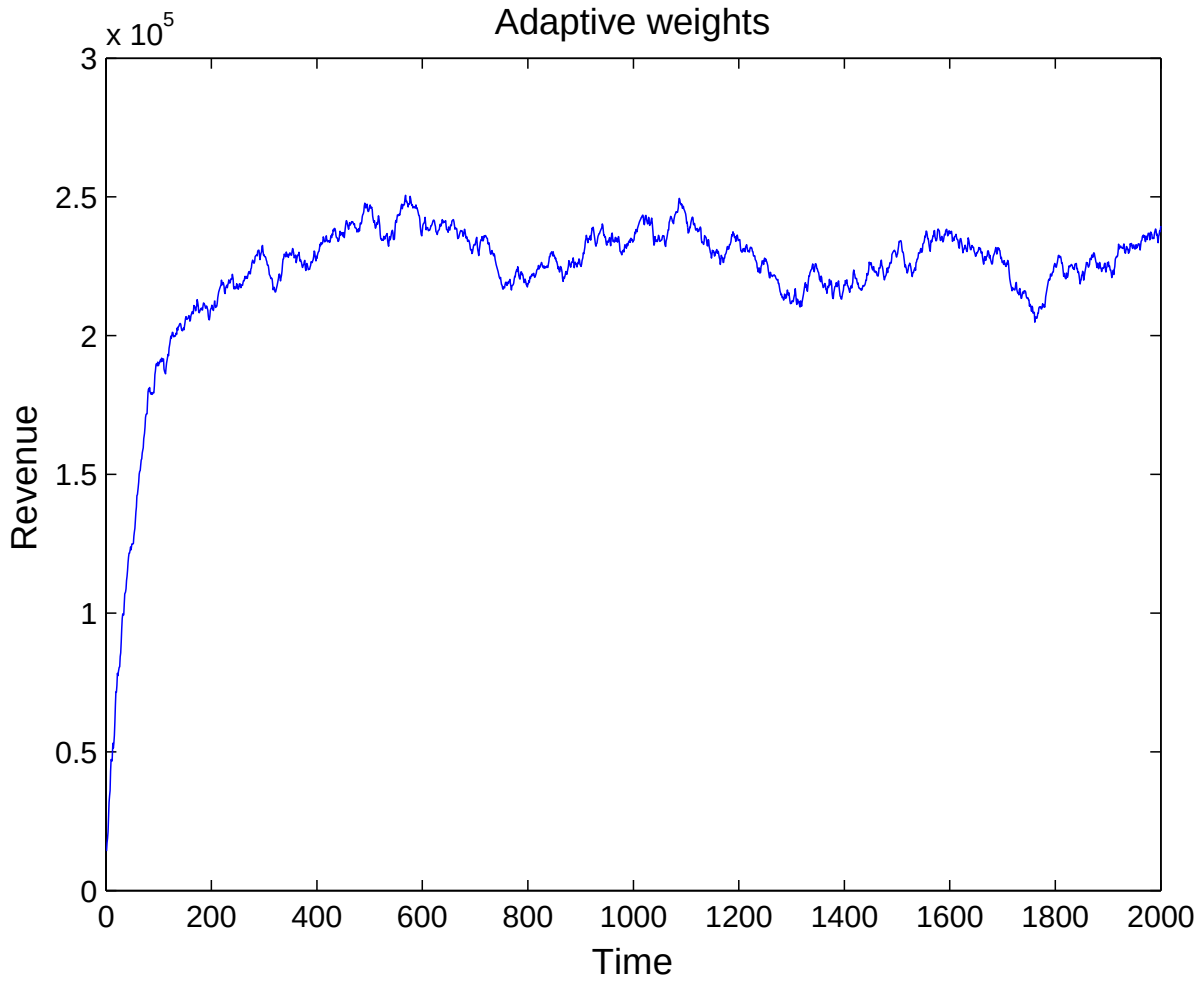


FIGURE 24 Adaptive weights - Evolution of the revenue as a function of time in the third experiment.

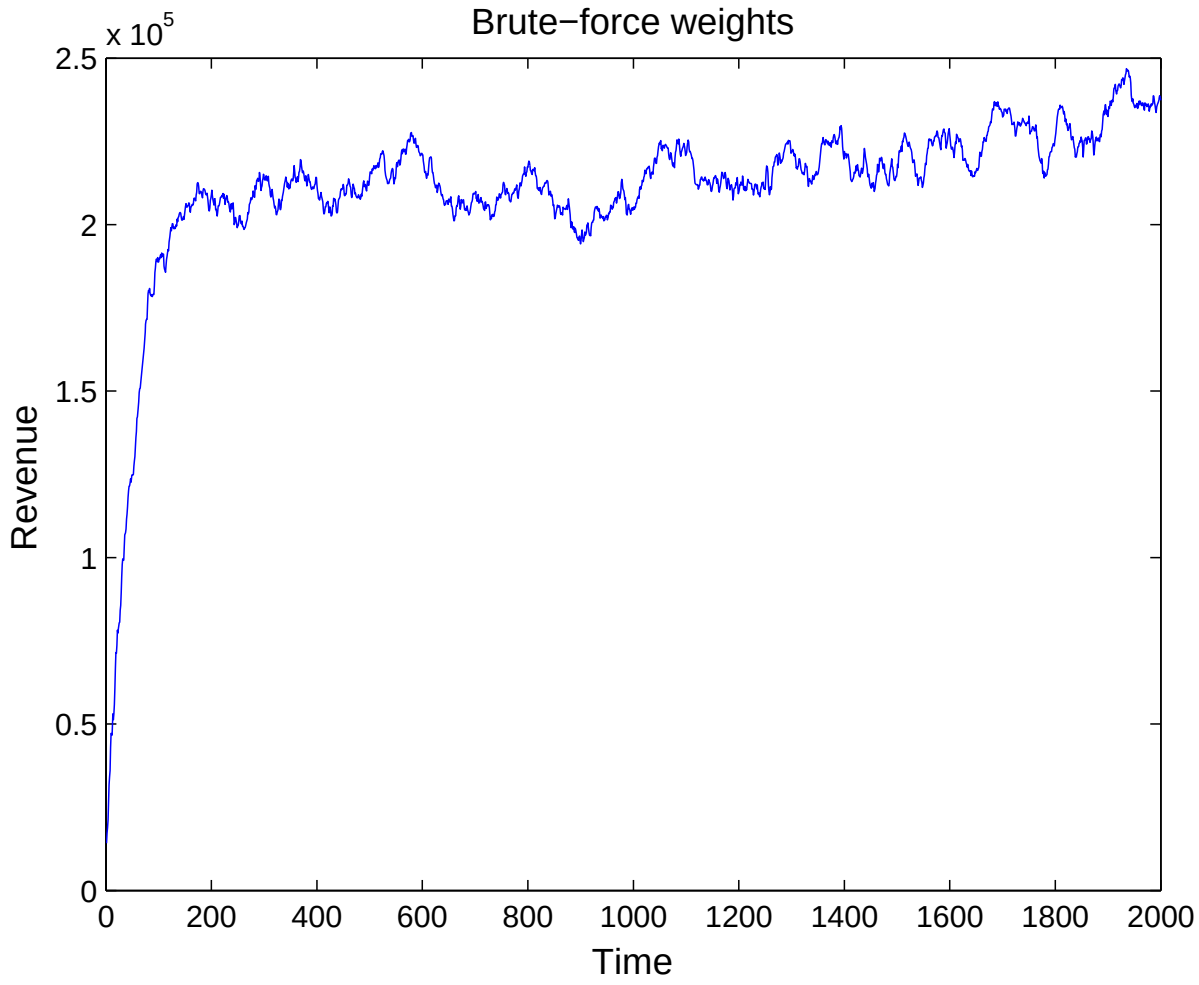


FIGURE 25 "Brute force" weights - Evolution of the revenue as a function of time in the third experiment.

Now, the connection is rejected if $\tilde{d}^j_{i_1,i_2}(t+1) > d^j_{lim}$ for any pair of ingress and egress nodes and for any class $j = 1,\ldots,m$, otherwise the connection is accepted. In the simulations the limits were chosen so that this additional CAC mechanism rejects some connections which were accepted in Experiment 1. The limits were 0.5, 0.8 and 1.1 for the gold, silver, and bronze classes, respectively. In Figs. 23 and **??** is seen (for the route between nodes 1 and 3) how delays stay under the limits in both cases, for adaptive and "brute force" weights, respectively. The revenues are seen in Figs. 24 and 25 and the mean values are 222281.3 and 211109.7 for adaptive and "brute force" weights, respectively. Notice that the revenue of the adaptive weights is about 5% larger, because it yields smaller delays in single nodes. However, when compared to the previous experiments the revenues are smaller, because more connections are rejected than would be necessary for the optimized revenue and weighted mean delay.

Experiment 4. In this experiment the CAC mechanism is accompanied by a guarantee on the minimum (unweighted) overall mean delay. From (113) the

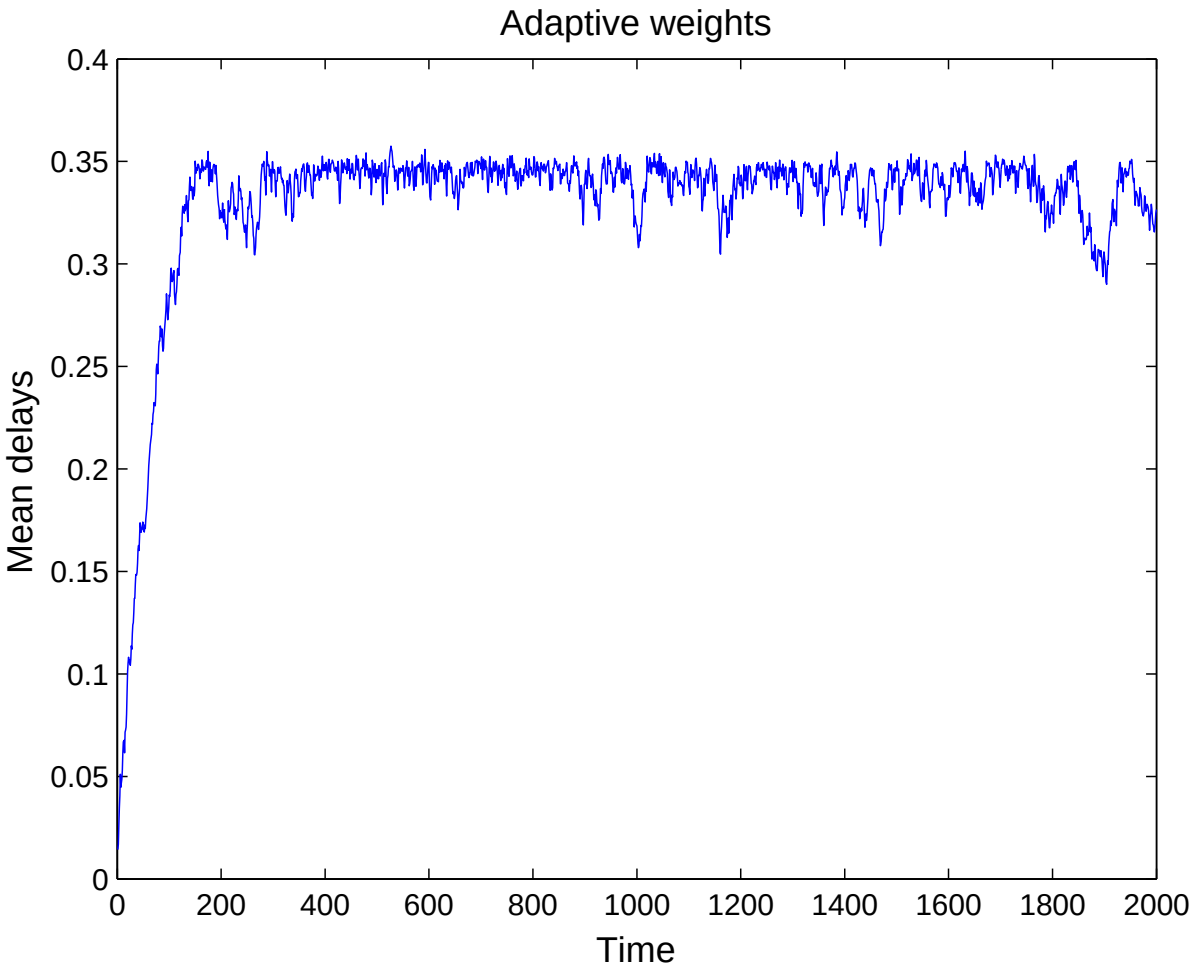


FIGURE 26 Adaptive weights - Evolution of the mean delay as a function of time in the fourth experiment.

new hypothetical state at time $t+1$ is

$$\tilde{E}(d(t+1)) = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij}(t+1)} \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{N_{ij}(t+1)\Delta t_{ij}(t+1)}{w_{ij}} \qquad (128)$$

and the connection is rejected if $\tilde{E}(d(t+1)) > E(d)_{lim}$ for the overall mean delay, otherwise the connection is accepted. The limit was chosen as $E(d)_{lim} = 0.35$, to drop some of those connections which were accepted in Experiment 1. In Figs. 26 and 27 the mean delay is shown to always stay under the limit, for adaptive and "brute force" weights, respectively. The revenues are seen in Figs. 28 and 29 and the mean values are 207216.3 and 214724.3 for adaptive and "brute force" weights, respectively. Now, the the "brute force" weights outperform the adaptive weights.

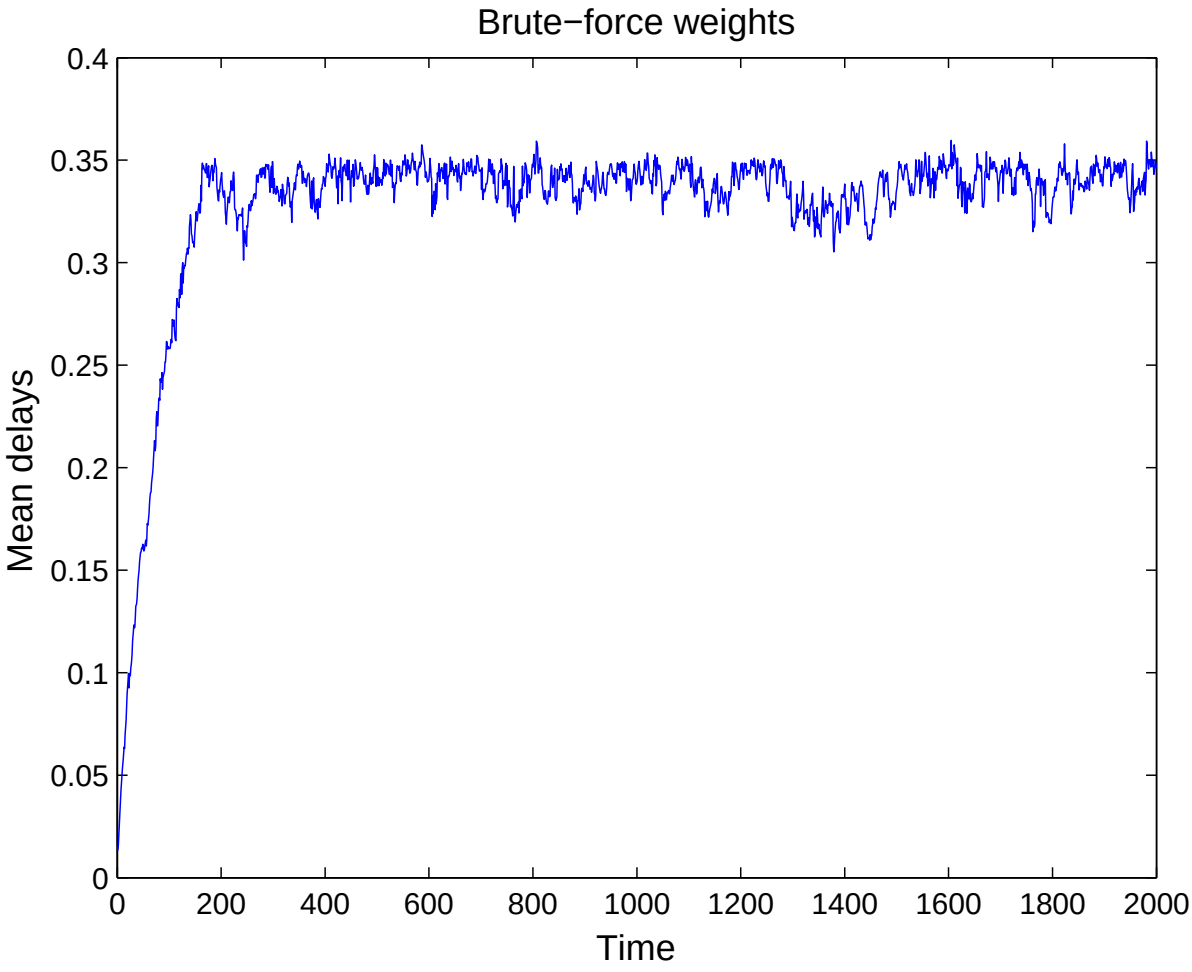


FIGURE 27 “Brute force” weights - Evolution of the mean delay as a function of time in the fourth experiment.

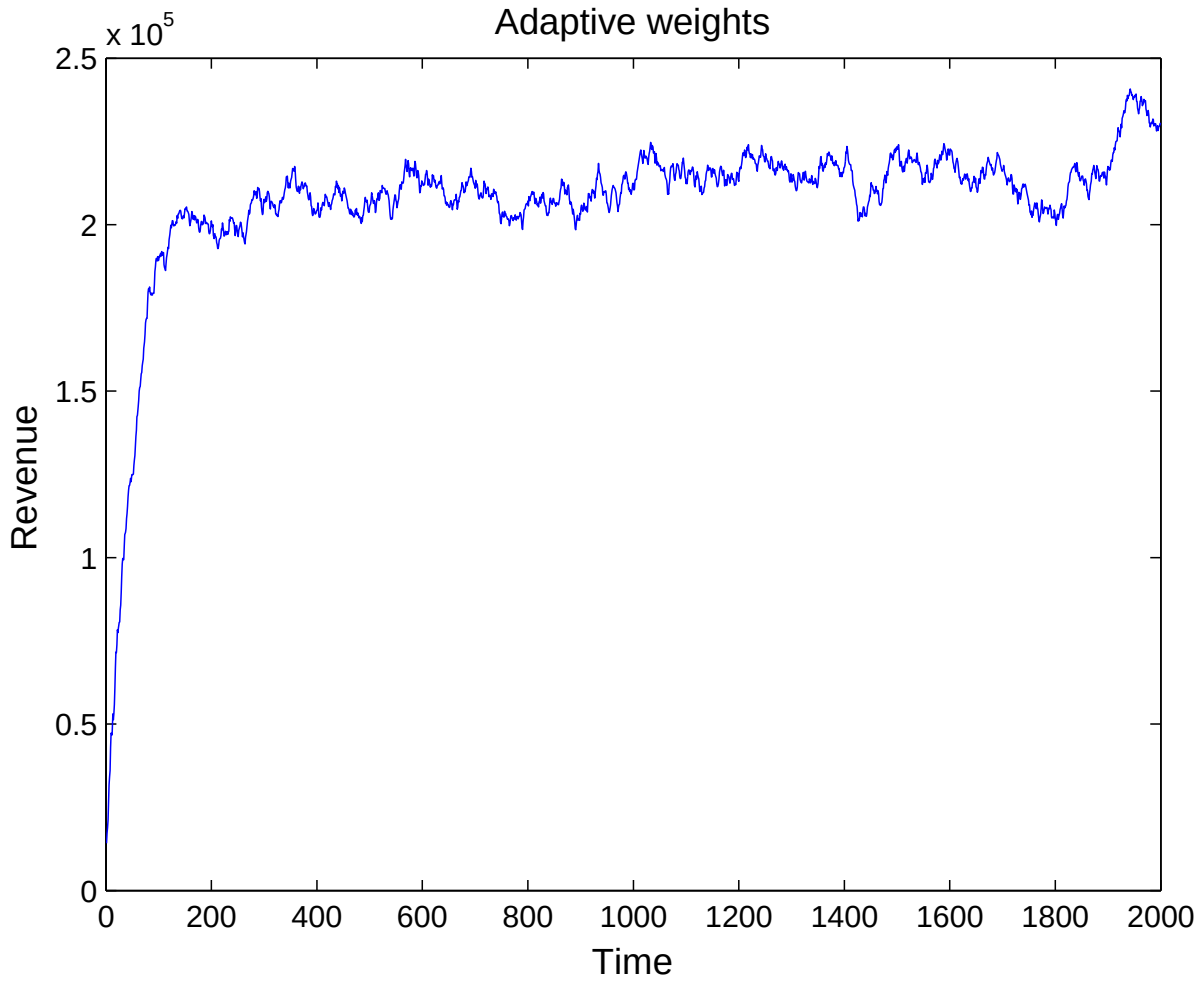


FIGURE 28 Adaptive weights - Evolution of the revenue as a function of time in the fourth experiment.

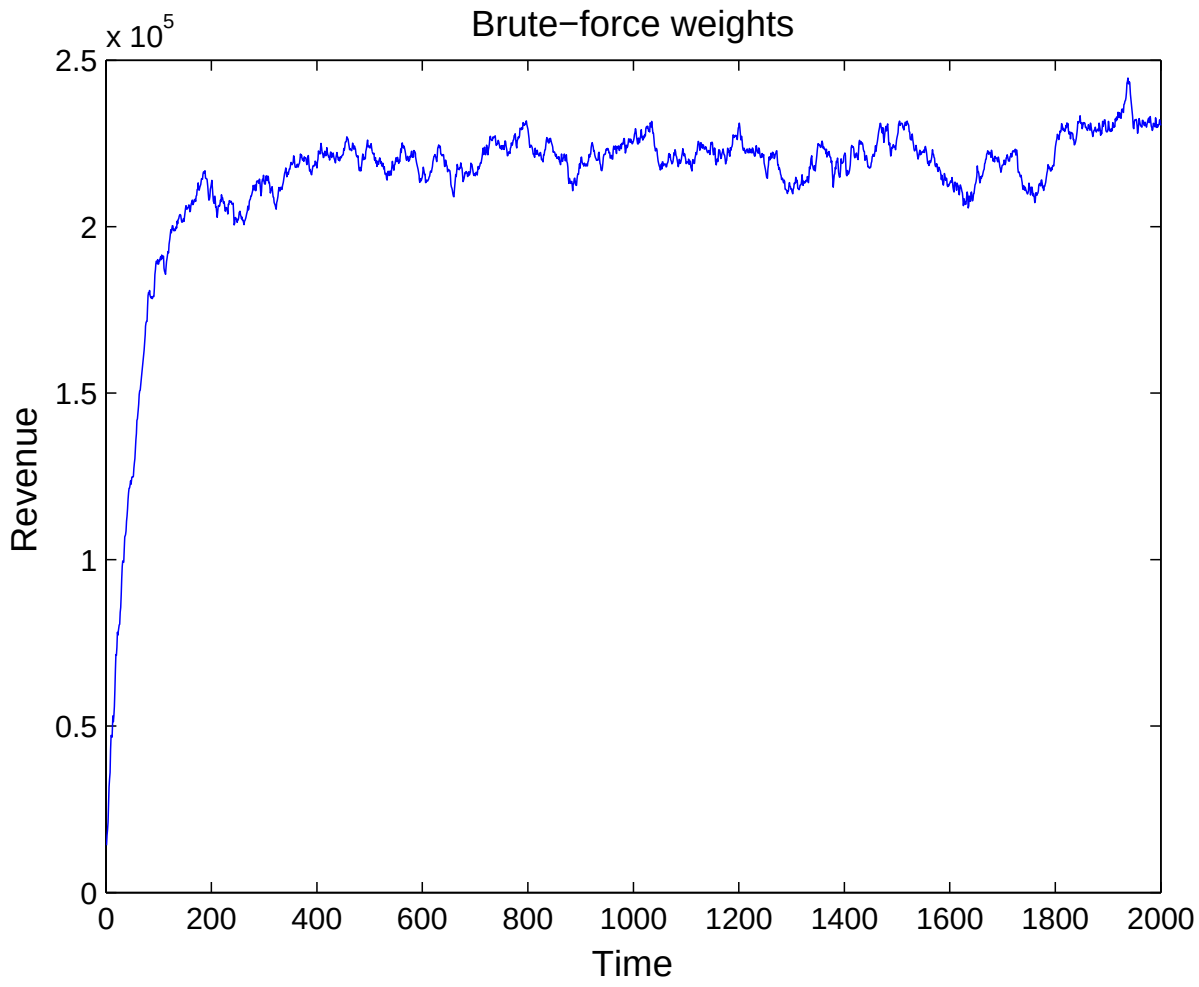


FIGURE 29 “Brute force” weights - Evolution of the revenue as a function of time in the fourth experiment.

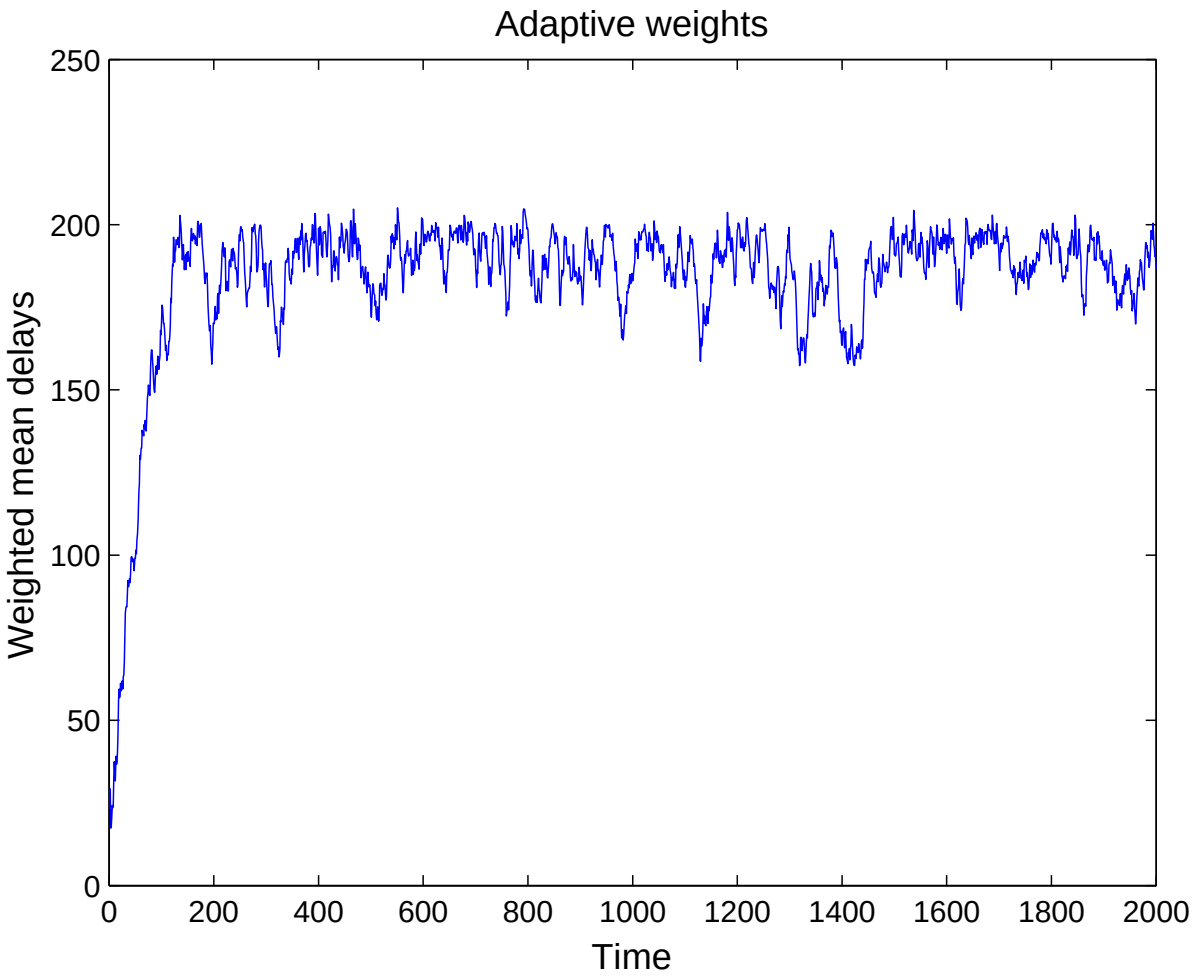


FIGURE 30 Adaptive weights - Evolution of the weighted mean delay as a function of time in the fifth experiment.

Experiment 5. In this experiment the CAC mechanism is accompanied by a guarantee on the minimum weighted mean delay . From (121) the new hypothetical state at time $t+1$ is

$$\tilde{E}(r_j d(t+1)) = \frac{1}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{ij}(t+1)} \sum_{i=1}^{n}(\sum_{j=1}^{m} \sqrt{N_{ij}(t+1) r_j \Delta t_{ij}(t+1)})^2. \quad (129)$$

and the connection is rejected if $\tilde{E}(r_j d(t+1)) > E(r_j d)_{lim}$ for the weighted mean delay, otherwise the connection is accepted. The limit was chosen as $E(r_j d)_{lim} = 200$, to drop some of those connections which were accepted in Experiment 1. I. In Figs. 30 and 31 the weighted mean delay is shown to stay under the limit, for adaptive and “brute force” weights, respectively. The revenues are seen in Figs. 32 and 33 and the mean values are 223801.8 and 223741.0 for adaptive and “brute force” weights, respectively. As the proposed algorithm minimizes the weighted mean delay, the limit on the weighted mean delay also guarantees maximized revenue, in contrast to the previous experiment.

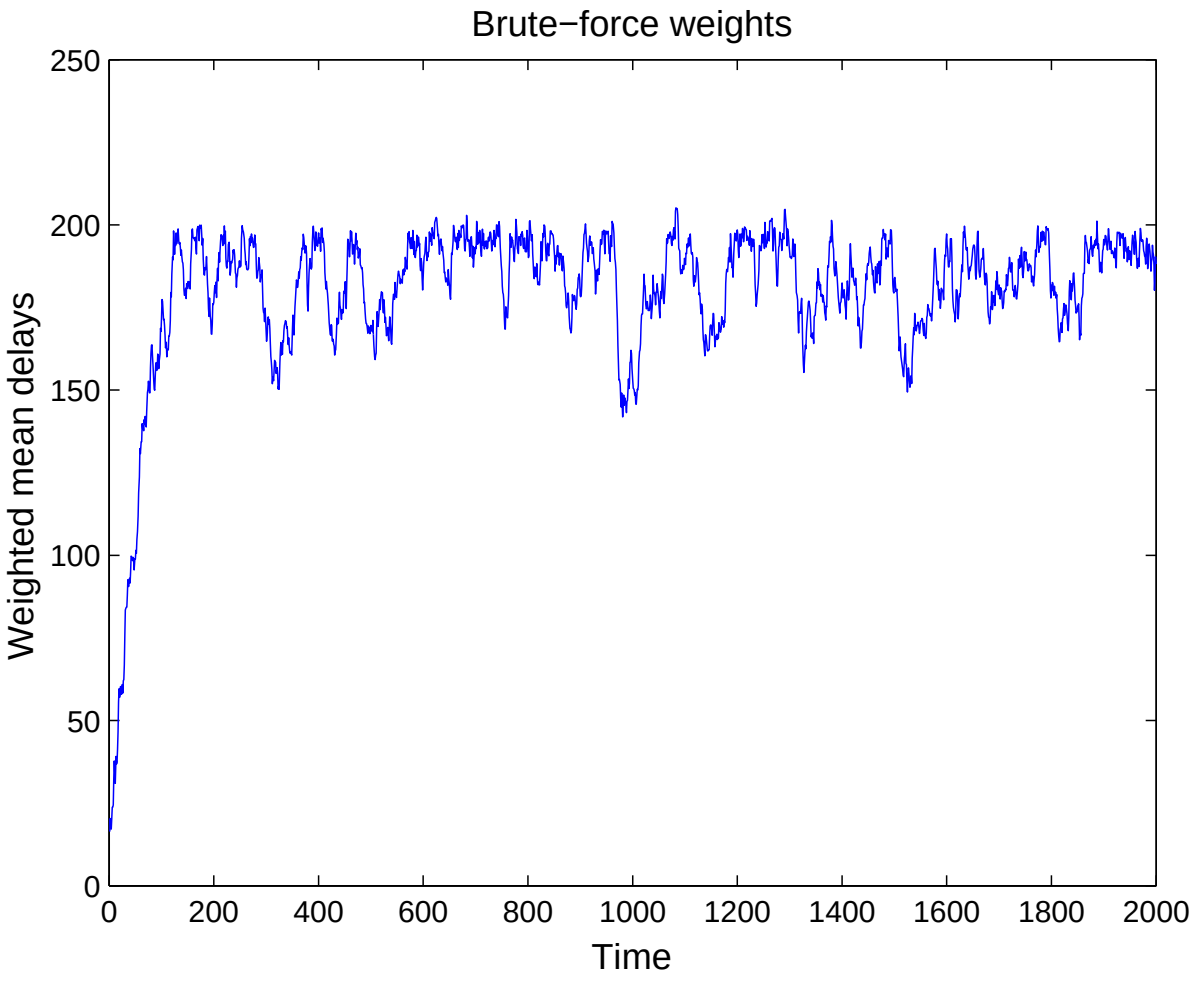


FIGURE 31 “Brute force” weights - Evolution of the weighted mean delay as a function of time in the fifth experiment.

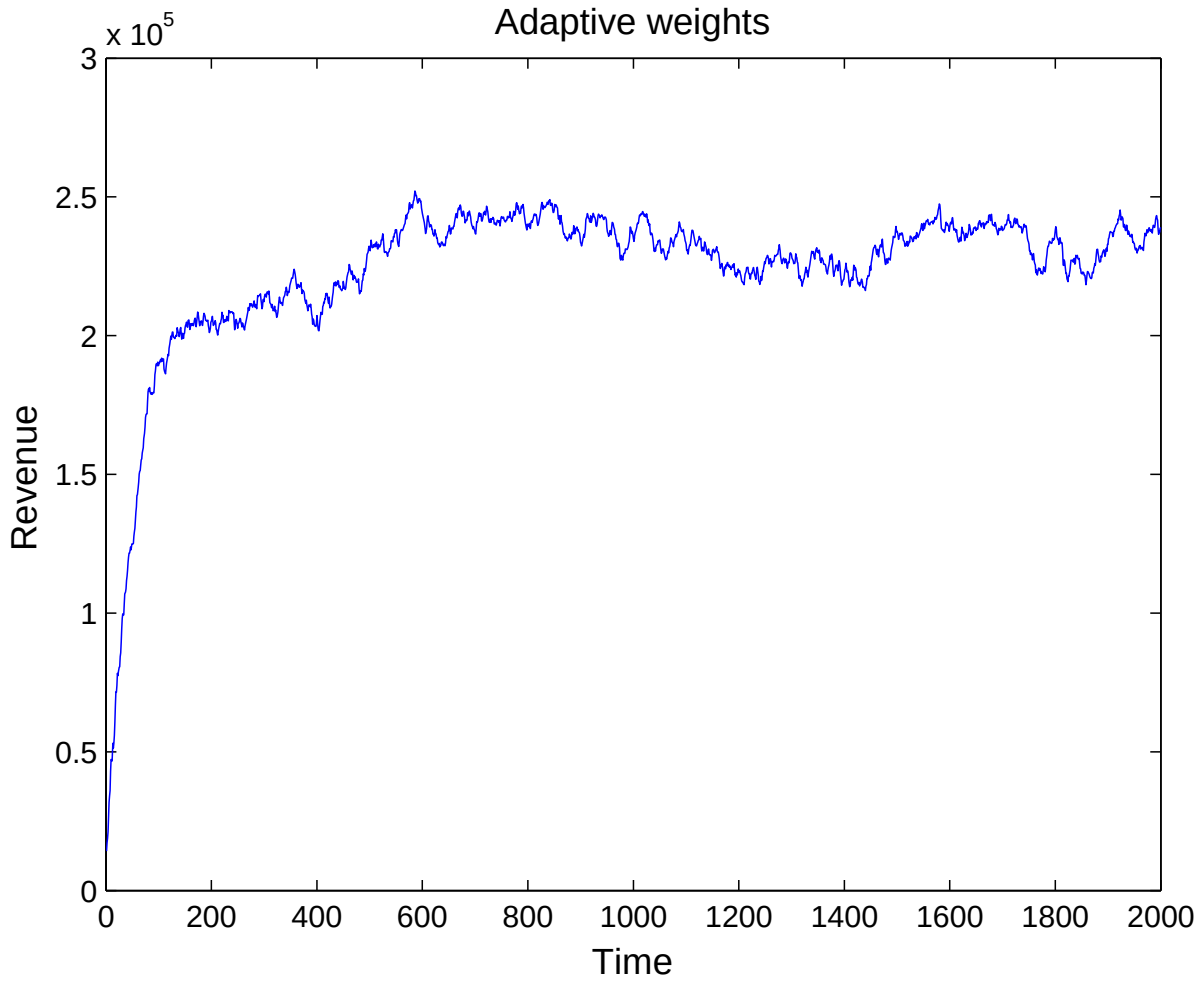


FIGURE 32 Adaptive weights - Evolution of the revenue as a function of time in the fifth experiment.

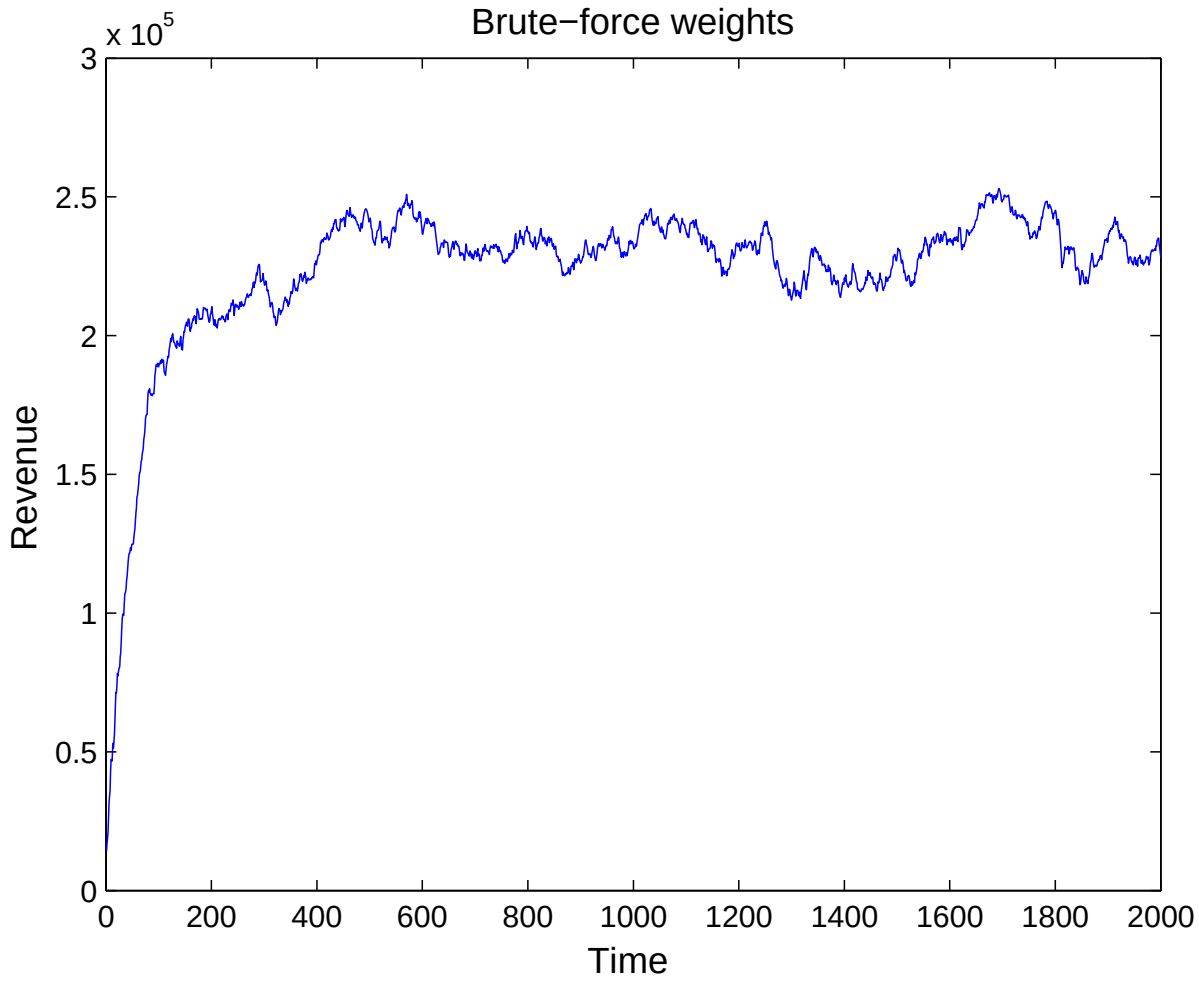


FIGURE 33 “Brute force” weights - Evolution of the revenue as a function of time in the fifth experiment.

Discussion

Here we discuss the properties of the algorithms as well as conclusions made by the experiments. We also converse implementation and computational complexity.

Algorithms and experiments

We make the following conclusions shown by algorithms and experiments:

- In the pricing scenario, we have derived an analytic form to the revenue algorithm for updating the weights w_i, which allocate data traffic to the connections of different service classes.
- The updating procedure is deterministic and nonparametric i.e. it does not make any assumptions of the statistical behavior of the traffic and connections. Thus it is robust against the errors that may occur from the wrong models.
- The algorithm is unique and optimal, which has been proved by Lagrangian optimization method.
- Our solution is quite simple, especially in the case where the pricing function is linear. Simplicity is obtained, too, by treating the number of connections N_{ij} and delays Δt_{ij} as separate parameters. Although they are in practice related to each other, their relationship in usually nonlinear e.g. due to the effect of the variable packet sizes.
- The maximum delay can be guaranteed in call admission mechanism by adding a specific constraint to the updating rule. In the experiments, delays always stay below the maximum limits.
- When the penalty pricing factors r_i are high, the corresponding connections obtain less delay.
- Because all penalty and gain factors are positive, all classes obtain service in a fair way.
- The optimal algorithm gives larger revenue than the brute-force algorithm. When all requests are accepted, the difference in the revenues is even greater. "All accept" mechanism should be favored, because it is preferred in telephones, Internet etc., i.e. it is tempting from the point of view of customers.
- Our algorithm yields minimum weighted mean delays. Because optimal revenue is obtained by the same updating rule, we can conclude that both the service provider and the customers are satisfied by using our algorithm.
- The delay updating rule (143) has an important advantage that it allows local updating in the nodes.

- Due to the nature of the pricing model, the optimum number of connections N_{ij} is finite, although finite-valued pricing factors k_j should be selected arbitrarily large. This prevents delays to become unbearably large.

- The semi-analysis shows, that the optimal selection of the weights yields positive revenue for sufficiently small N_{ij}, r_j, and Δt_{ij}, and for sufficiently large k_j. Roughly speaking, for $N_{ij} = 0$, $F = 0$. When N_{ij} begins to increase, F increases, being convex and positive with respect to the N_{ij}. At some domain, F becomes concave, while still increasing until reaching the maximum optimal point. After that, F begins to decrease, being concave for large values of N_{ij}.

2.3.10 Pricing Based Adaptive Scheduling Method for Bandwidth Allocation

In this section, we formulate an expression for the bandwidth (bit rate) of the data. As an example, consider the traffic classification at the output buffers in Fig. 34. The customers of the gold class pay more than the customers of the silver class - in our case for the available bandwidth, thus obtaining more bandwidth. Naturally, the price is often concave with respect to the bandwidth; for example, when images are transferred, the price may be twice compared to the price when voice is transferred, while the bandwidth ratio - image bandwidth / voice bandwidth - is much more than two.

Let the processing time of 1 kbyte of data be T [seconds / kbyte] in the packet scheduler of Fig. 34. Looking at the gold queue, we see three connections, denoted by N_{11} (packet size $b_{11} = 1$ kbyte), N_{12} (packet size $b_{12} = 2$ kbytes), and N_{13} (packet size $b_{13} = 4$ kbytes). Let us consider the time moment t_0 in Fig. 34, when the gold class connection N_{11}'s data packet is moved to the packet scheduler. During the time $delay = t_1 - t_0$, which passes before N_{11}'s next packet is moved to the scheduler, the other connections of the class are handled by the scheduler, thus the *delay* [seconds] induced by the other connections in the same queue (i.e. service class) is

$$\begin{aligned} delay[\mathrm{s}] &= (1+2+4)T = 3 \times \frac{1}{3}(1+2+4)T \\ &= N_1 \frac{1}{N_1} \sum_{j=1}^{3} b_{1j} T = N_1 E(b_1) T = 7T, \end{aligned} \tag{130}$$

where N_i is the number of connections in the queue of class i ($N_1 = 3$ in this example), b_{ij} is the size of the packet (relative to 1 kbyte) of the jth connection in the ith class and $E(b_i)$ is the average packet size of the connections in the ith class. The delay of the gold class is also affected by the connections in the other service classes. In Fig. 34, a weight $w_1 = 2/3$ is allotted for the gold class so, for every 2 kbytes of the gold class, 1 kbyte of the silver class is moved to the output - on the average. Thus, the overall delay [seconds] in the ith class is

$$d_i = \frac{N_i E(b_i) T}{w_i} (= 7 \times 3/2T = 10.5T). \tag{131}$$

Because sizes of the different packets vary, the delay d_{ij} scaled by the specific packet of size b_{ij} is

$$d_{ij} \quad [\mathrm{seconds/kB}] = \frac{N_i E(b_i) T}{b_{ij} w_i} \tag{132}$$

The bit rate B_{ij} (kbytes / second) of the class i is inversely proportional to the de-

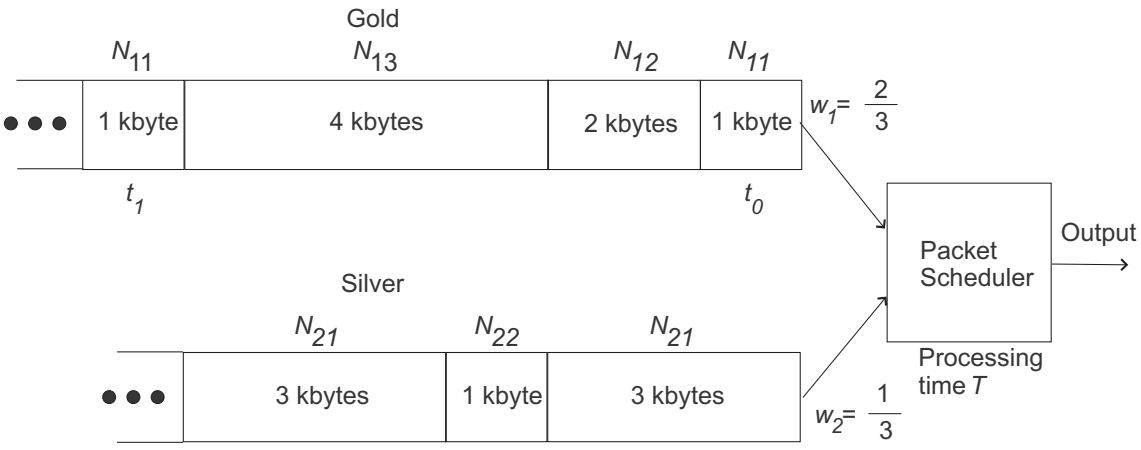


FIGURE 34 An example of a packet scheduler with two classes

lay / packet of the class i

$$\begin{aligned} B_{ij}[kB/s] &= \frac{1}{d_{ij}} = \frac{b_{ij}w_i}{N_i E(b_i)T} \\ &= \frac{b_{ij}w_i}{N_i E(b_i)} = \frac{b_{ij}w_i}{\sum_{l=1}^{N_i} b_{il}} \end{aligned} \tag{133}$$

where $\sum_{l=1}^{N_i} b_{il}$ is the sum of all packet sizes in class i and the processing time T can be scaled $T = 1$ [s / kB], without loss of generality. Here

$$E(b_i) = \frac{1}{N_i} \sum_{j=1}^{N_i} b_{ij} \tag{134}$$

is mean packet length in the class i, and N_i is the number of connections in the class i. In our example, the bit rate of the connection N_{11} is

$$B_{11} = \frac{b_{11}w_1}{N_1 E(b_1)} = \frac{1 \times 2/3}{7} = \frac{2}{21}, \tag{135}$$

while the bit rate of the connection N_{12} is

$$B_{12} = \frac{b_{12}w_1}{N_1 E(b_1)} = \frac{2 \times 2/3}{7} = \frac{4}{21}. \tag{136}$$

When T was scaled to $T = 1$,

$$B_{ij,max} = 1. \tag{137}$$

This occurs when there is only one connection in only one class, e.i. all the capacity is allocated to one connection. It is noticed naturally that

$$\begin{aligned} \sum_{i=1}^{m} \sum_{j=1}^{N_i} B_{ij} &= \sum_{i=1}^{m} \sum_{j=1}^{N_i} \frac{b_{ij}w_i}{\sum_{l=1}^{N_i} b_{il}} \\ &= \sum_{i=1}^{m} w_i \sum_{j=1}^{N_i} \frac{b_{ij}}{\sum_{l=1}^{N_i} b_{il}} = 1, \end{aligned} \tag{138}$$

where m is the number of service classes. From (133) we get plausible results:

- Larger the packet size b_{ij} is, larger is the bandwidth for that connection.
- Larger the weight w_i is, larger is the bandwidth in that class.
- Larger the number of users N_i is, smaller is the bandwidth in that class.
- Larger the mean packet size $E(b_i)$ is, smaller is the bandwidth in that class, because $\sum_{l=1}^{N_i} b_{il}$ becomes large in the denominator.

Pricing models and revenue maximization

In this section we formulate the pricing scenario and functional form of the revenue. Let the revenue of class i be $f_i(B_{ij})$ which is nonnegative, increasing, and concave. As examples of this kind of functions are $f_i(B_{ij}) = \log(B_{ij}+1)$ or $f_i(B_{ij}) = \sqrt{B_{ij}}$. The revenue obtained from class i is

$$F_i(B_{ij}) = \sum_{j=1}^{N_i} f_i(B_{ij}). \tag{139}$$

The total revenue is the sum of the revenues from all the service classes

$$F = \sum_{i=1}^{m} F_i(B_{ij}) = \sum_{i=1}^{m}\sum_{j=1}^{N_i} f_i(B_{ij}) \tag{140}$$

under the constraint

$$\sum_{i=1}^{m} w_i = 1. \tag{141}$$

We consider especially *polynomial* pricing, because it allows to obtain analytical optimal solution for the weights w_i.

Definition: *The pricing function*

$$P_i(B_{ij}) = r_i B_{ij}^p, \quad r_i > 0, \quad p > 0 \tag{142}$$

is called polynomial pricing function. In the function, $r_i > 0$ is a factor depending on the money paid for the class (bandwidth of the connection). "Better" the class i is, larger is r_i. Next we present a theorem for optimal weights.

Theorem: *Optimal weights for revenue F under the polynomial pricing model is*

$$w_i = \frac{\left(\sum_{j=1}^{N_i} b_{ij}\right)^{\frac{p}{p-1}}}{r_i^{\frac{1}{p-1}} \left(\sum_{j=1}^{N_i} b_{ij}^p\right)^{\frac{1}{p-1}} \sum_{k=1}^{m} \frac{\left(\sum_{l=1}^{N_k} b_{kl}\right)^{\frac{p}{p-1}}}{r_k^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_k} b_{kl}^p\right)^{\frac{1}{p-1}}}}, \tag{143}$$

when $0 < p < 1$.

Proof: Revenue F has the Lagrangian form

$$\begin{aligned} F &= \sum_{i=1}^{m}\sum_{j=1}^{N_i} F(B_{ij}) + \lambda\left(1 - \sum_{i=1}^{m} w_i\right) \\ &= \sum_{i=1}^{m}\sum_{j=1}^{N_i} r_i \left(\frac{b_{ij} w_i}{\sum_{l=1}^{N_i} b_{il}}\right)^p + \lambda\left(1 - \sum_{i=1}^{m} w_i\right) \\ &= \sum_{i=1}^{m} \frac{r_i}{\left(\sum_{l=1}^{N_i} b_{il}\right)^p} w_i^p \sum_{j=1}^{N_i} b_{ij}^p + \lambda\left(1 - \sum_{i=1}^{m} w_i\right). \end{aligned} \tag{144}$$

Optimal weights w_i are obtained by taking first order derivative of F as follows:

$$\frac{\partial F}{\partial w_i} = \frac{pr_i}{(\sum_{l=1}^{N_i} b_{il})^p} w_i^{p-1} \sum_{j=1}^{N_i} b_{ij}^p - \lambda = 0. \tag{145}$$

Because Lagrangian solution

$$\frac{\partial F}{\partial \lambda} = 0 \tag{146}$$

yields $\sum_{j=1}^{m} w_j = 1$, one obtains

$$\begin{aligned} w_i &= \frac{\lambda^{\frac{1}{p-1}} (\sum_{j=1}^{N_i} b_{ij})^{\frac{p}{p-1}}}{(pr_i)^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_i} b_{il}^p\right)^{\frac{1}{p-1}}} \\ &= \frac{\lambda^{\frac{1}{p-1}} (\sum_{j=1}^{N_i} b_{ij})^{\frac{p}{p-1}}}{(pr_i)^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_i} b_{il}^p\right)^{\frac{1}{p-1}} \sum_{k=1}^{m} w_k} \\ &= \frac{\lambda^{\frac{1}{p-1}} (\sum_{j=1}^{N_i} b_{ij})^{\frac{p}{p-1}}}{(pr_i)^{\frac{1}{p-1}} \left(\sum_{j=1}^{N_i} b_{ij}^p\right)^{\frac{1}{p-1}} \sum_{k=1}^{m} \frac{\lambda^{\frac{1}{p-1}} (\sum_{l=1}^{N_k} b_{kl})^{\frac{p}{p-1}}}{(pr_k)^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_k} b_{kl}^p\right)^{\frac{1}{p-1}}}} \\ &= \frac{(\sum_{j=1}^{N_i} b_{ij})^{\frac{p}{p-1}}}{r_i^{\frac{1}{p-1}} \left(\sum_{j=1}^{N_i} b_{ij}^p\right)^{\frac{1}{p-1}} a} \end{aligned} \tag{147}$$

where

$$a = \sum_{k=1}^{m} \frac{(\sum_{l=1}^{N_k} b_{kl})^{\frac{p}{p-1}}}{r_k^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_k} b_{kl}^p\right)^{\frac{1}{p-1}}} \tag{148}$$

This is just the expression of the weight claimed in the theorem. From the expression (147) it is seen that the penalty term λ has the form

$$\lambda^{\frac{1}{p-1}} = \frac{1}{a}, \tag{149}$$

or

$$\lambda = \frac{1}{a^{p-1}} \tag{150}$$

Thus the first order derivative is

$$\begin{aligned} \frac{\partial F}{\partial w_i} &= \frac{pr_i}{(E(b_i) N_i)^p} w_i^{p-1} \sum_{j=1}^{N_i} b_{ij}^p \\ &- \frac{1}{\left(\sum_{k=1}^{m} \frac{\sum_{l=1}^{N_k} b_{kl})^{\frac{p}{p-1}}}{r_k^{\frac{1}{p-1}} \left(\sum_{l=1}^{N_k} b_{kl}^p\right)^{\frac{1}{p-1}}}\right)^{p-1}}. \end{aligned} \tag{151}$$

The right hand side term after minus sign does not depend on the weight, and so the second order derivative is

$$\frac{\partial^2 F}{\partial w_i^2} = \frac{(p-1)pr_i}{(\sum_{j=1}^{N_i} b_{ij})^p} w_i^{p-2} \sum_{j=1}^{N_i} b_{ij}^p < 0, \tag{152}$$

when $0 < p < 1$. In that condition, F is concave with respect to the weights w_i, and the optimal solution is unique. That completes the proof. **Q.E.D.**

As a special case, let us consider $p = \frac{1}{2}$ in (142). That leads to a square root pricing model, and to the optimal solution

$$w_i = \frac{r_i^2 \left(\sum_{j=1}^{N_i} b_{ij}^{\frac{1}{2}} \right)^2}{\sum_{j=1}^{N_i} b_{ij} \sum_{k=1}^{m} \frac{r_k^2 \left(\sum_{l=1}^{N_k} b_{kl}^{\frac{1}{2}} \right)^2}{\sum_{l=1}^{N_k} b_{kl}}}. \tag{153}$$

From this expression it is noticed that it is more cost-effective to allocate more bandwidth to those classes having larger price factor r_i. In addition, because the parameters r_i are positive for all classes, all classes obtain service. Call Admission Control (CAC) mechanism can be made by simple hypothesis testing without any assumptions about call or dropping rates. *CAC is performed only in the connection level.* In hypothesis testing, revenue formula is applied as

$$F(t) = F[N_i(t)] \tag{154}$$

$$\tilde{F}(t+1) = F[N_i(t+1)] \tag{155}$$

where the number of connections at time t is $N_i(t)$, $t = 1, \ldots$ and the new hypothetical number of connections at the time $t+1$ be $\tilde{N}_i(t+1)$, $t = 1, \ldots$, when one or several connections appear in some class / classes. If $F(t) > \tilde{F}(t+1)$, then the connection is rejected, otherwise it is accepted. The mean bandwidth of all classes can be guaranteed by the following mechanism. From (133) and (153) an expression for the bandwidth of connection j in class i at time t is obtained

$$B_{ij}(t) = \frac{b_{ij} r_i^2 \left(\sum_{l=1}^{N_i(t)} b_{il}^{\frac{1}{2}} \right)^2}{\left(\sum_{l=1}^{N_i(t)} b_{il} \right)^2 \sum_{k=1}^{m} \frac{r_k^2 \left(\sum_{l=1}^{N_k(t)} b_{kl}^{\frac{1}{2}} \right)^2}{\sum_{l=1}^{N_k(t)} b_{kl}}} \tag{156}$$

The new hypothetical bandwidth at time $t+1$ is

$$B_{ij}(t+1) = \frac{b_{ij} r_i^2 \left(\sum_{l=1}^{N_i(t+1)} b_{il}^{\frac{1}{2}} \right)^2}{\left(\sum_{l=1}^{N_i(t+1)} b_{il} \right)^2 \sum_{k=1}^{m} \frac{r_k^2 \left(\sum_{l=1}^{N_k(t+1)} b_{kl}^{\frac{1}{2}} \right)^2}{\sum_{l=1}^{N_k(t+1)} b_{kl}}}. \tag{157}$$

So, if B^i_{lim} is the guaranteed mean bandwidth for class i, then the CAC mechanism rejects the connection if $F(t) > \tilde{F}(t+1)$ OR

$$\frac{1}{N_i(t+1)} \sum_{j=1}^{N_i(t+1)} B_{ij}(t+1) < B^i_{lim}, \tag{158}$$

for any $i = 1, \ldots, m$, otherwise it is accepted.

Experiments

In this section we demonstrate by simulation the operation of the square root pricing algorithm (i.e. $p = \frac{1}{2}$ in (142)) with the optimal weights (153) and compare it with a constant weight ($w_1 = 1/2, w_2 = 1/3$ and $w_3 = 1/6$) version of the algorithm. The number of classes is $m = 3$ (i.e. gold, silver and bronze, which are described by subindexes 1,2 and 3, respectively). The connections in the different service classes have different average data packet sizes $E(b_1) = 50$, $E(b_2) = 25$ and $E(b_3) = 10$ with a standard deviation of 1 (i.e. in a specific service class the connections have similar demand for bandwidth). The processing time of the server is chosen as $T = 1/1000$ s/kbyte. The life time of a connection (i.e. the time the connection is served and has packets in the queue) is exponentially distributed and weighted by the connection's packet size. The arrival rates of the connections are Poisson distributed. The connection rates per unit time for the gold, silver, and bronze classes are $\alpha_1 = 0.30$, $\alpha_2 = 0.40$ and $\alpha_3 = 0.50$, respectively. Therefore, customers' demand depends on the charging in such a way that for gold class, there are least connections, while for bronze class, there are most connections. The duration parameters (i.e. "decay rates") for the connections are $\beta_1 = 0.3$, $\beta_2 = 0.15$ and $\beta_3 = 0.03$, where the probability density functions for the durations are

$$p_i(t) = \beta_i e^{-\beta_i t}, \quad , i = 1, 2, 3 \quad t \geq 0. \tag{159}$$

The number of unit times in the experiments was $\tau = 2000$. In the experiments the gold, silver and bronze classes have gain factors $r_1 = 400$, $r_2 = 200$ and $r_3 = 50$, respectively.

Experiment 1. In this experiment, the CAC mechanism described by (154) and (155) is used to limit the service of the scheduler. Figs. 35 and 36 represent the mean bandwidths for constant and adaptive weight version of the algorithm, respectively. The adaptive weights (and thus the different gain factors r_i) guarantee better bit rate for the gold class, but limit the access of some connections to achieve this.

From Figs. 37 and 38 it is seen that with the constant weights the number of connections is greater than with adaptive weights as the adaptive weight algorithm is guarantying optimal revenue and good throughput to higher classes.

From Fig. 39 it is seen that the revenues are close, with the adaptive weight revenue being slightly greater than the constant weight revenue, even though there is more connections with the constant weight version. The mean of the revenues are 3005.6 and 3050.9 for constant and adaptive weights, respectively.

Experiment 2. In this experiment, the CAC mechanism is omitted, so a same number of connections is accepted in both cases (Figs. 40 and 41).

Now the difference in the revenues is greater to the benefit of the adaptive weight algorithm (Fig. 42) than in the first experiment (Fig. 39). The mean of the revenues are 3686.2 and 3963.0 for constant and adaptive weights, respectively. If we compare Figs. 39 and 42 we see that the moments when the constant weight revenue is greater in Fig. 39 is due that there are more connections than with

adaptive weights.

The mean bandwidths (Figs. 43 and 44) are similar to those in the first experiment, i.e. the adaptive weights guarantee better bit rate for the gold class.

Experiment 3. In this experiment, the CAC mechanism is accompanied by an additional guarantee (158) that the mean bandwidth of for each class does not decrease under a specified limit. In this experiments the limits were 15, 3 and 0.1 kB, for the gold, silver and bronze classes, respectively. With these values, the bandwidth limit for the gold class is restricting the access of new connections

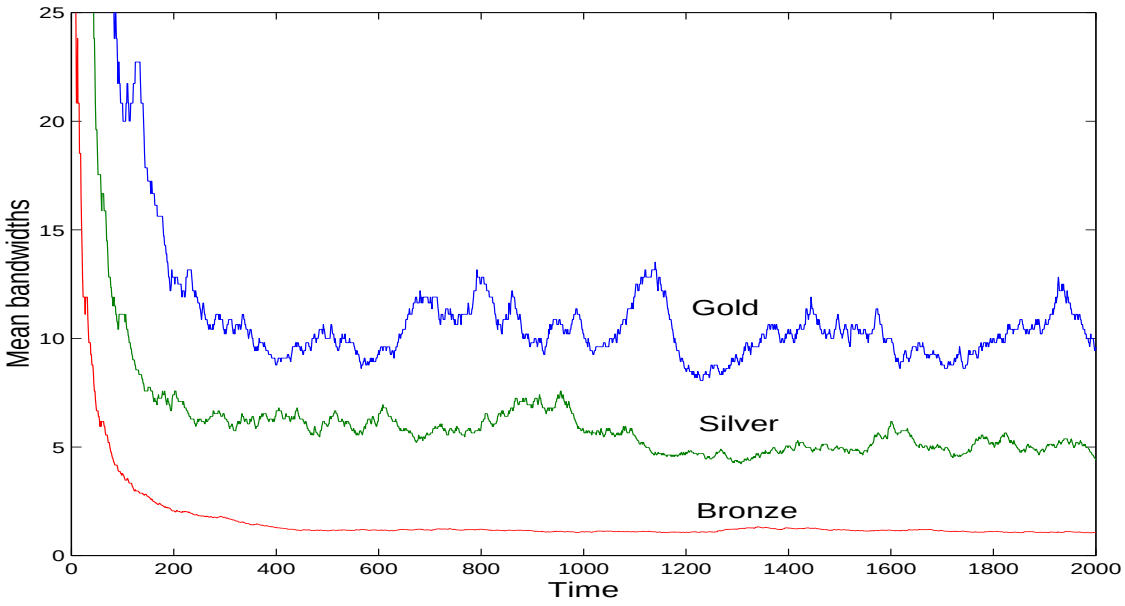


FIGURE 35 Constant weights - Evolution of the mean bandwidths as a function of time in the first experiment.

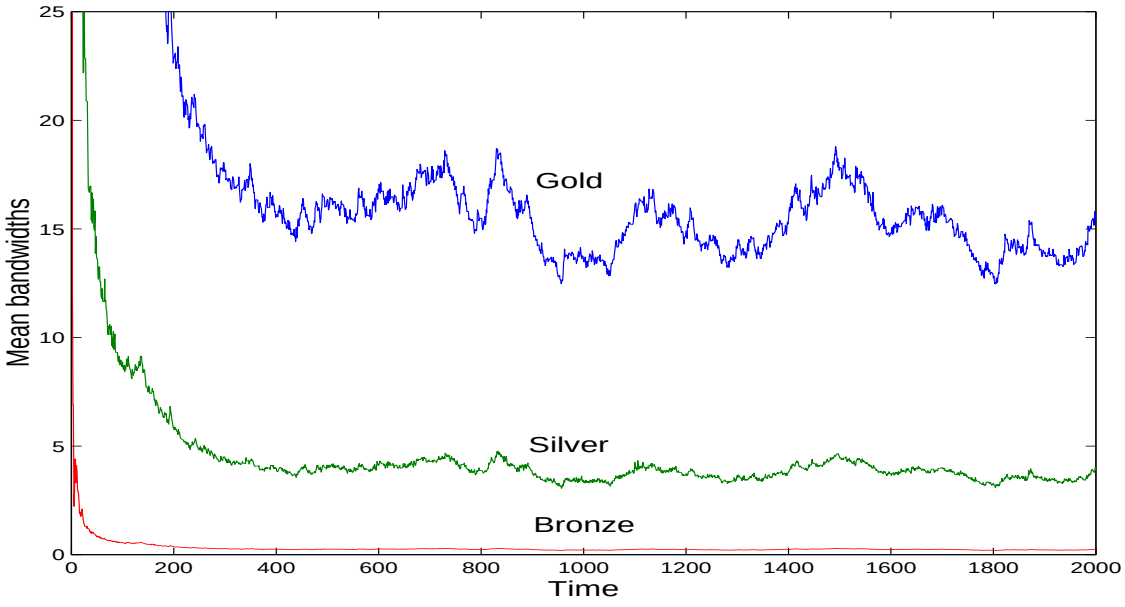


FIGURE 36 Adaptive weights - Evolution of the mean bandwidths as a function of time in the first experiment.

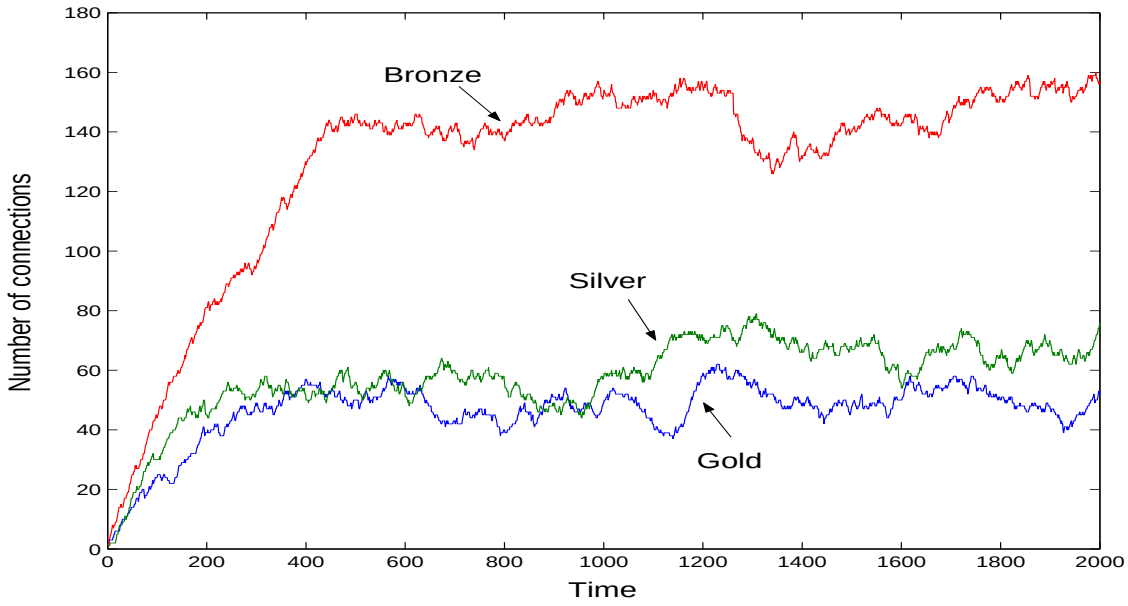


FIGURE 37 Constant weights - Evolution of the number of connections as a function of time in the first experiment.

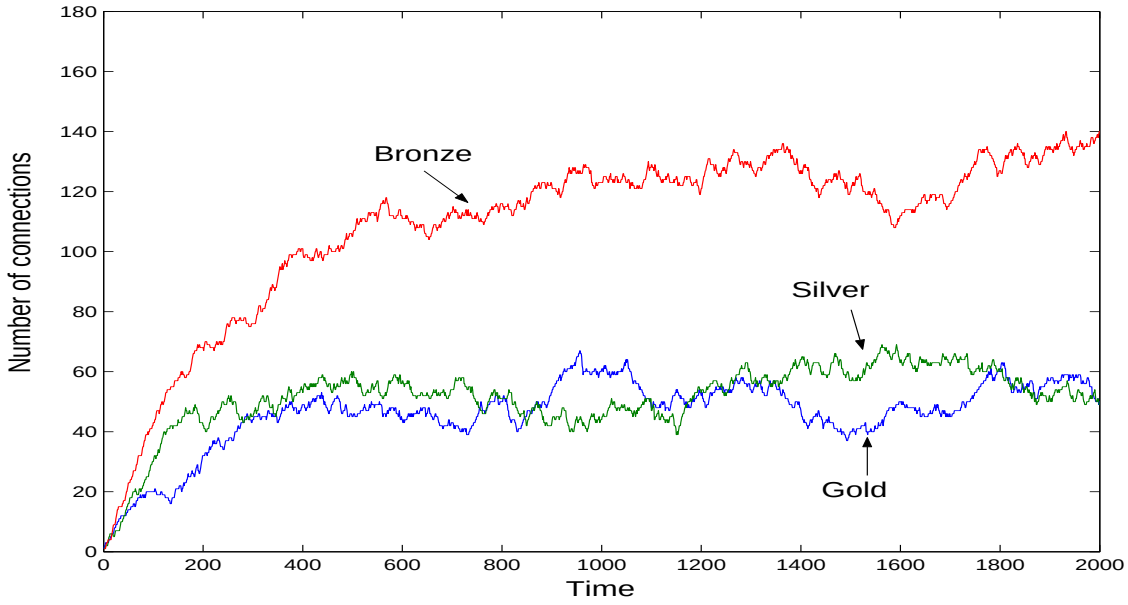


FIGURE 38 Adaptive weights - Evolution of the number of connections as a function of time in the first experiment.

(Figs. 45 and 46).

Now, the constant weight algorithm restricts access for the users in the gold class (Fig. 47), but the adaptive weight algorithm grants access to the gold class customers and restricts service from the bronze class customers (Fig. 48). This is a very plausible result as now the CAC with the adaptive weight algorithm favors the connections in the gold class, guarantees a minimum bandwidth for the connections and also provides optimal revenue, although it is smaller then in the previous experiments as because of the bandwidth limits more connections

are rejected in this experiment.

The revenues for the constant and adaptive weight algorithms are seen in Fig. 49. In this experiment the mean of the revenues are 2652.1 and 2970.7 for constant and adaptive weights, respectively.

These kinds of results give valuable information on tuning the model, between the use of bandwidth limits and the gain factors. The network operator might e.g. want a best effort class to be available to the users. Then, it would be possible to use the minimum bandwidth requirements for the gold and silver classes, while the bronze class would have no guarantee on the obtained bandwidth. With the price factors the operator can favor some class to perhaps influence the customers to change between classes, thus optimizing the use of network resources.

Discussion

In this section, we discuss the algorithm from the point of view of theory and experiments, as well as implementation and computational complexity.

Theory and experiments

The conclusions shown by theory and experiments are:

- In the polynomial pricing scenario, we have derived analytic form to the revenue and the weights w_i, which allocate data traffic to the connections of different service classes.

- The updating procedure is deterministic and nonparametric i.e. it does not

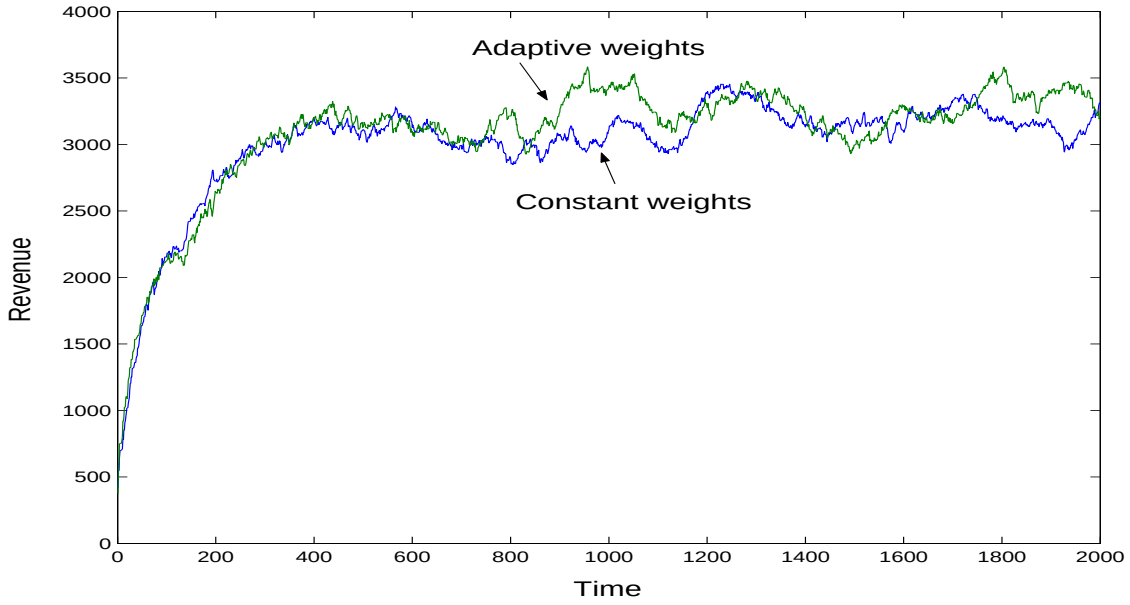


FIGURE 39 Evolution of the revenue F with constant and adaptive weights, as a function of time in the first experiment.

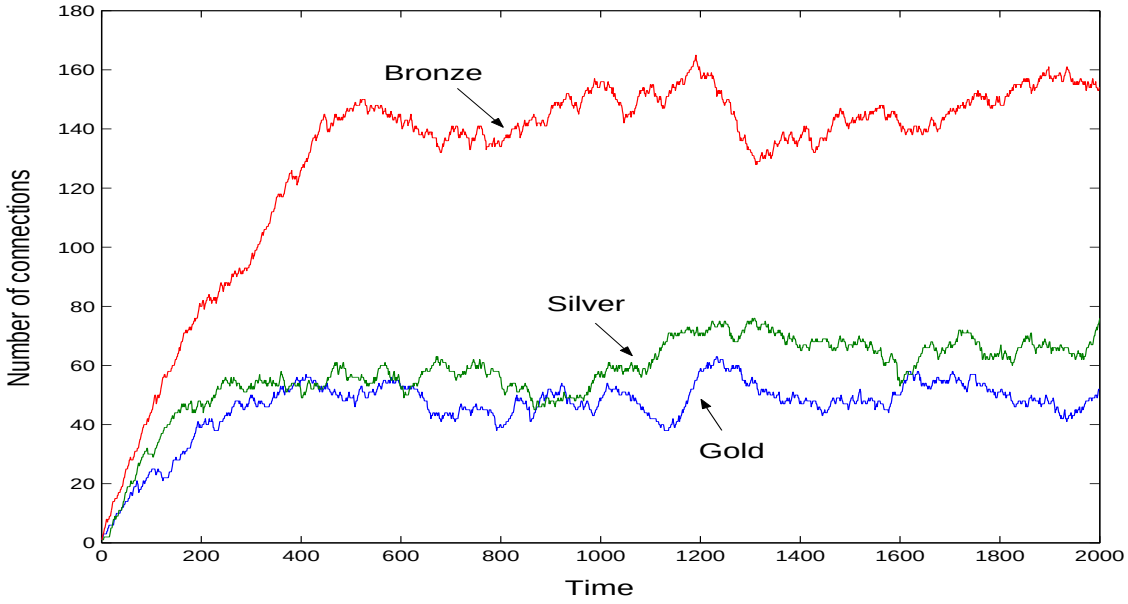


FIGURE 40 Constant weights - Evolution of the number of connections as a function of time in the second experiment.

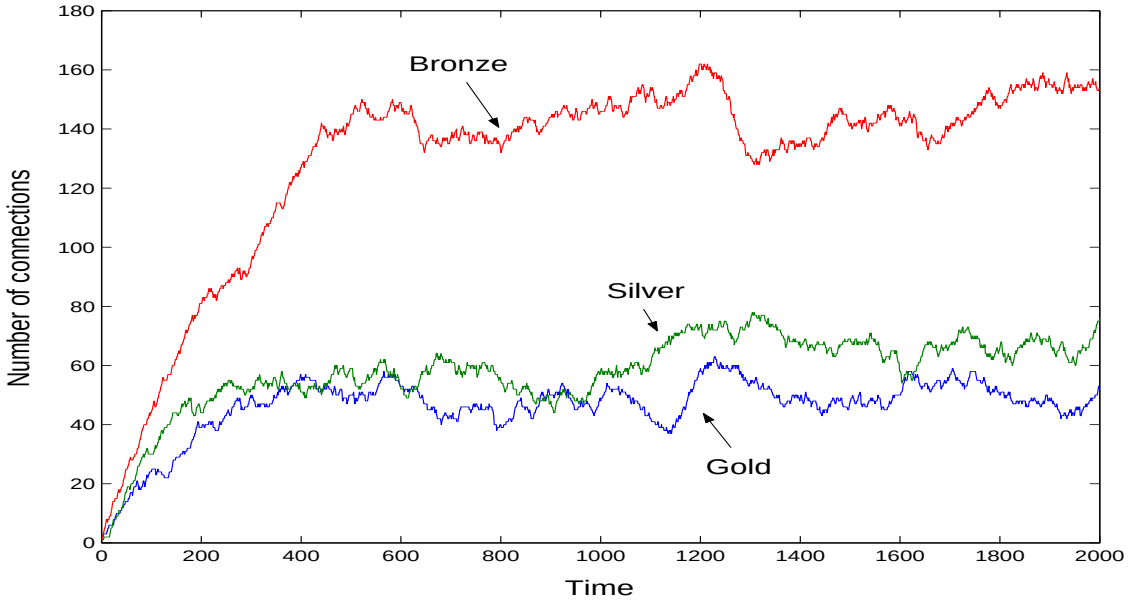


FIGURE 41 Adaptive weights - Evolution of the number of connections as a function of time in the second experiment.

make any assumptions of the statistical behavior of the traffic and connections. Thus it is robust against erroneous models.

- Algorithm is unique and optimal, which has been theoretically proved by Lagrangian optimization method.
- Closed form solution makes algorithm quite simple.
- Minimum bandwidth can be guaranteed in call admission mechanism by

adding a specific constraint to the updating rule. In the experiments, bandwidths always stay above the minimum limits.

- When the gain pricing factors r_i are high, the corresponding connections obtain more bandwidth.

- Because all gain factors r_i are positive, all classes obtain service in fair way.

- Algorithm outperforms the fixed weight algorithm in the sense that it gives larger revenue.

- Mechanism with no CAC performs well. It is temptating from the point of the customers, because the assumption in telephones, Internet etc. is that all requests are accepted.

- Concave pricing scenario has a *load balancing* feature. To show this, consider e.g. the square root pricing function $\sqrt{B}$. B denotes bandwidth. Price paid for 1 Mbit/second is 1 unit of money/second. This corresponds to the case where the network is highly loaded. When e.g. 30 Mbits are transferred, total time is 30 seconds, and the price is 30 units of money. On the other hand, price paid for 5 Mbits/second is 2.24 units of money/second. This corresponds to the case where the network is low loaded - say at night. When 30 Mbits are transferred, total time is 6 seconds, and the price is 13.44 units of money. Therefore, it is cost-effective from the point of view of the customers to use the network, when it is low loaded. This effect balances the load.

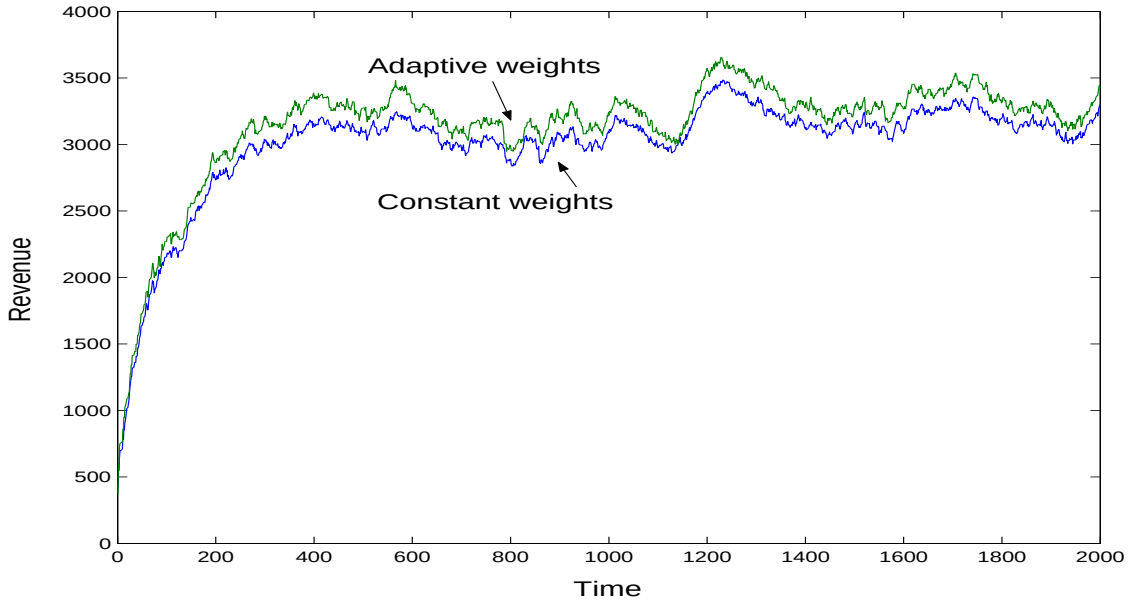


FIGURE 42 Evolution of the revenue F with constant and adaptive weights, as a function of time in the second experiment.

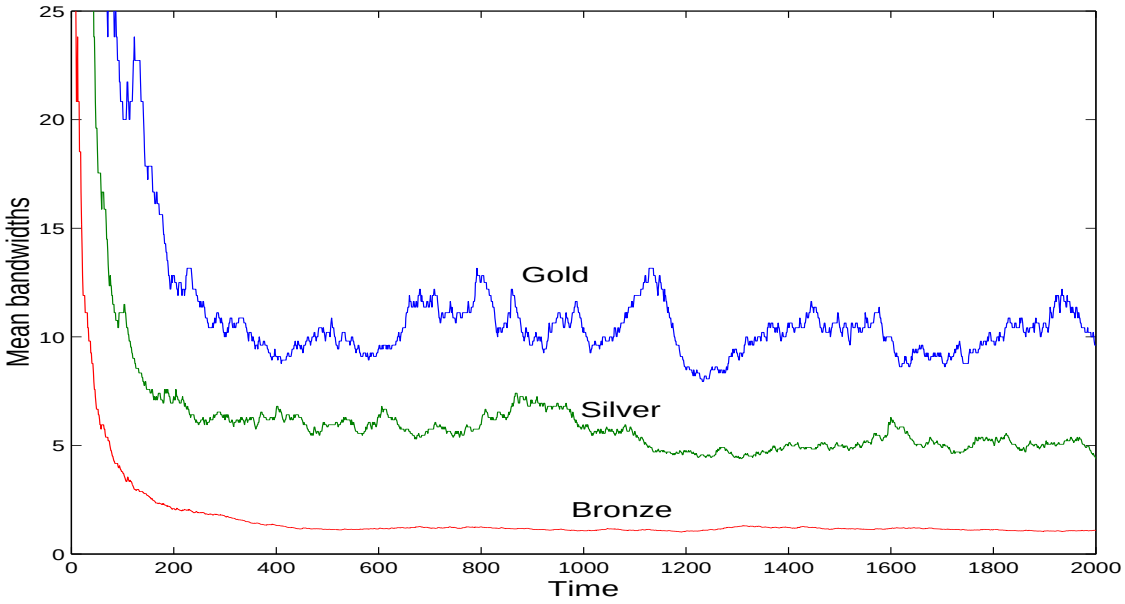


FIGURE 43 Constant weights - Evolution of the mean bandwidths as a function of time in the second experiment.

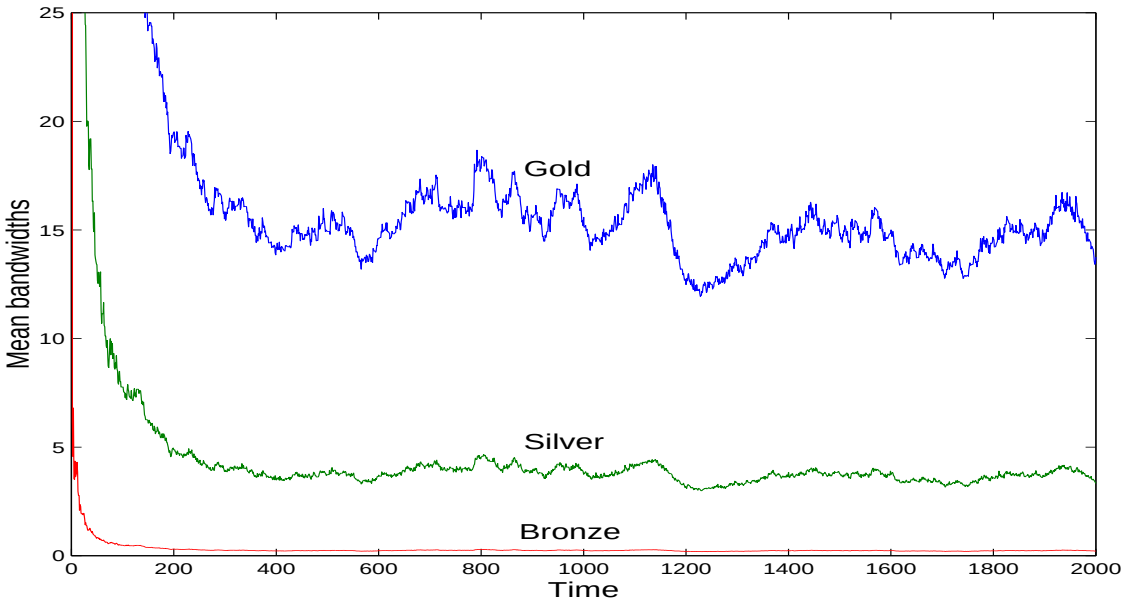


FIGURE 44 Adaptive weights - Evolution of the mean bandwidths as a function of time in the second experiment.

Implementation Issues

One possible implementation target of the proposed model are the edge routers of the DiffServ architecture [16]. In this case, all major adaptive issues will be implemented at the edge of a domain. Unlike the core routers, the edge routers have the per-flow information that enables them to perform classification and policing. Since data about each traffic flow is available, it is more convenient to perform the adaptive resource allocation in this part of a DiffServ domain. In

this framework, the core routers could remain intact and are not overburdened with additional adaptive software that might slow the packet forwarding process. Moreover, such an approach fits into the original idea of the DiffServ technology, which states that the edge routers perform sophisticated functions, while the core routers perform only simple forwarding. In the proposed adaptive framework, the key role of the adaptive egress routers is to control the amount of traffic injected into a DiffServ domain. By tracking the number of active data flows and their QoS parameters, the adaptive edge routers can optimally allocate the output bandwidth between the different traffic aggregates. Of course, this solution does not diminish the use of other adaptive solutions that could be implemented in the core routers to provide finer resource allocation and to achieve better utilization [167].

Computational Complexity

Let us consider the computational complexity of updating the weights (143). They can be updated in *connection level*, instead of packet level. The evaluation of the numerator of w_i needs two multiplications and one power calculation. Computation of the denominator needs about $O(N_i) + O(m)$ operations. When m is small - say three (gold, silver, bronze) - the total number of calculations of the sum $\sum_{j=1}^{m}(\cdot)$ remains quite small. The dominating procedure is to calculate $\sum_{j=1}^{N_j}(\cdot)$, since the number of connections may be large in all the classes. Thus the consequence is that the updating of all the weights need about $O(\sum_{i=1}^{m} N_i)$ operations. Therefore, it is important that they are updated only when connection appears or disappears. However, when the number of connections is small, even packet level updating might be possible; or e.g. one updating per five packets. Practi-

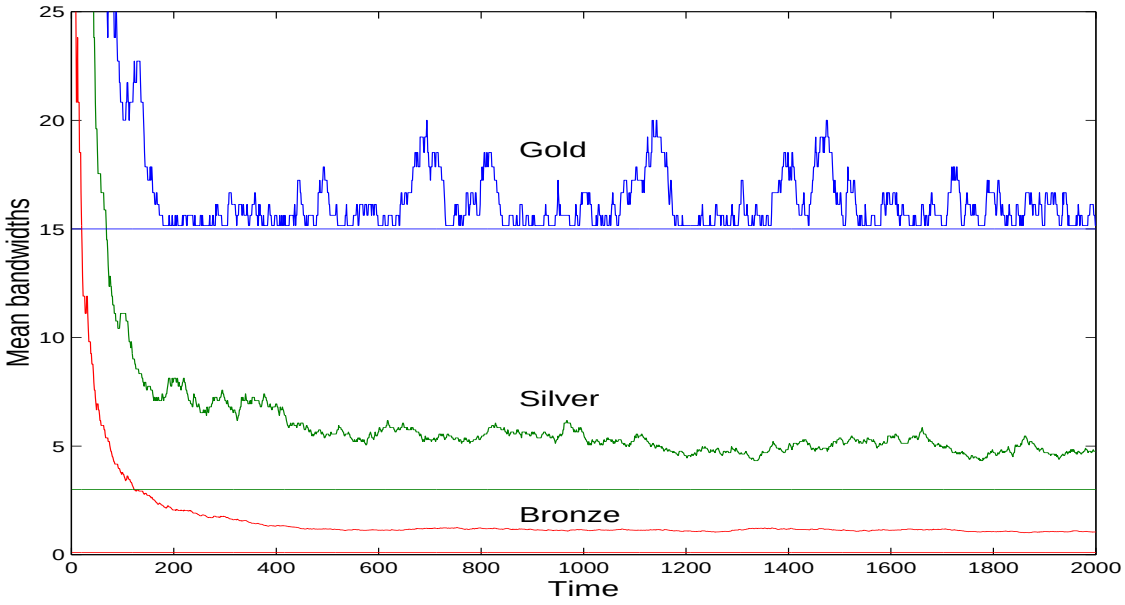


FIGURE 45 Constant weights - Evolution of the mean bandwidths as a function of time in the third experiment.

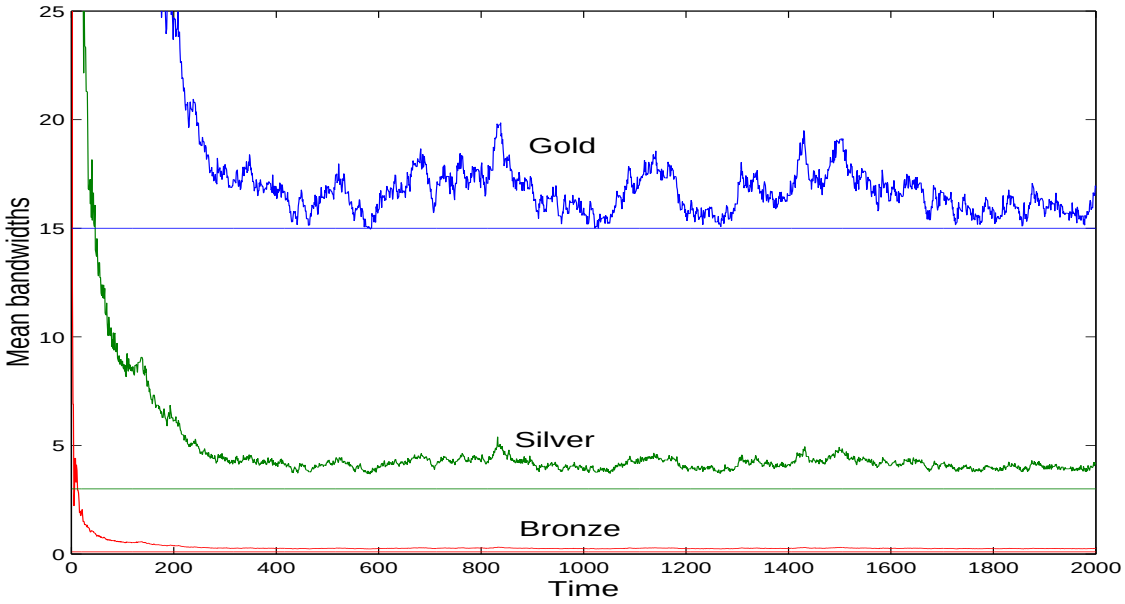


FIGURE 46 Adaptive weights - Evolution of the mean bandwidths as a function of time in the third experiment.

cal implementation determines the updating strategy, and it is open topic in this work.

Conclusions

In this section, we presented an adaptive algorithm for optimizing the network operator revenue and for ensuring bandwidth as a Quality of Service (QoS) re-

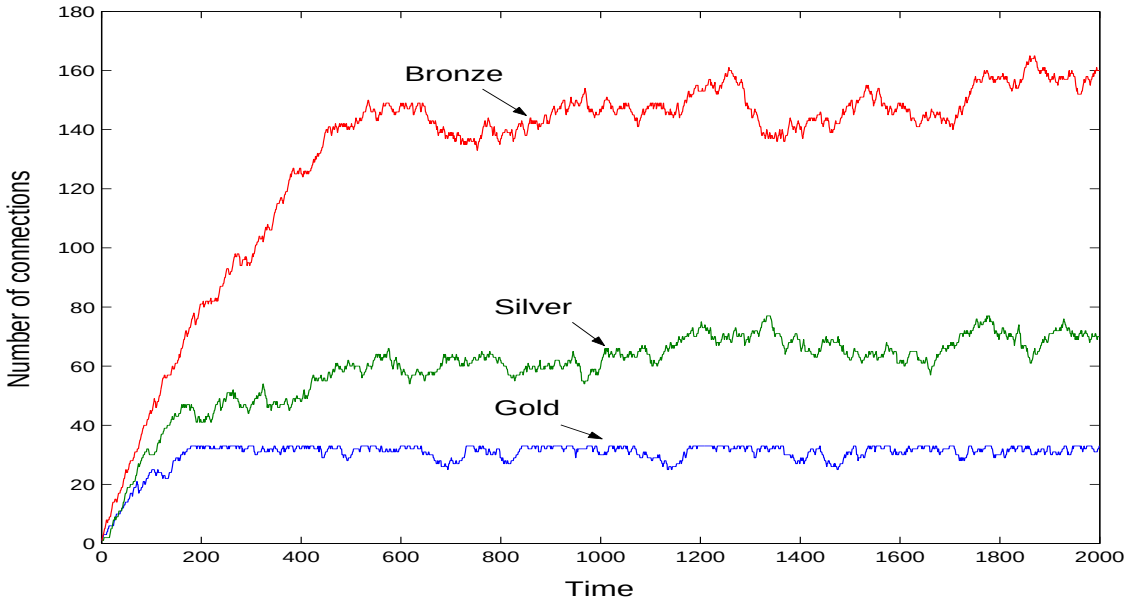


FIGURE 47 Constant weights - Evolution of the number of connections as a function of time in the third experiment.

quirement. We derived the closed form algorithm from a revenue-based optimization problem. The proposed weight updating algorithm was found to be computationally inexpensive in our scope of study. In the experiments we simulated the operation of the adaptive algorithm and compared it with a constant weight version of the same algorithm. The obtained results show that by using the adaptive weights the revenue is larger than with the constant weights. Also, the revenue is maximized while ensuring that the gold class customers get most of the bandwidth, while bronze class customers get least of the bandwidth. Our algorithm is deterministic and non-parametric, and thus we believe that in practical environments it is a competitive candidate due to its robustness. Most important conclusion is that *we have combined bandwidth allocation, weight updating, pricing, and revenue maximization in a unique manner.*

Computational Complexity

2.3.11 General algorithm for scheduling

Here we present very general approach for QoS scheduling approach with some questions and analysis.

In our scenario, there are several service classes for communications network services. Price paid by the customers depends directly on the QoS parameters like delay, bit rate etc. Thus it can vary in real time on the contrary to the classical approach where price is constant. We try to maximize the revenue of the service provider while still giving fair service to the customers. Thus this approach is tempting both from the point of view of service provider and customers. Consider the packet scheduler for two service classes. Gold class customers pay

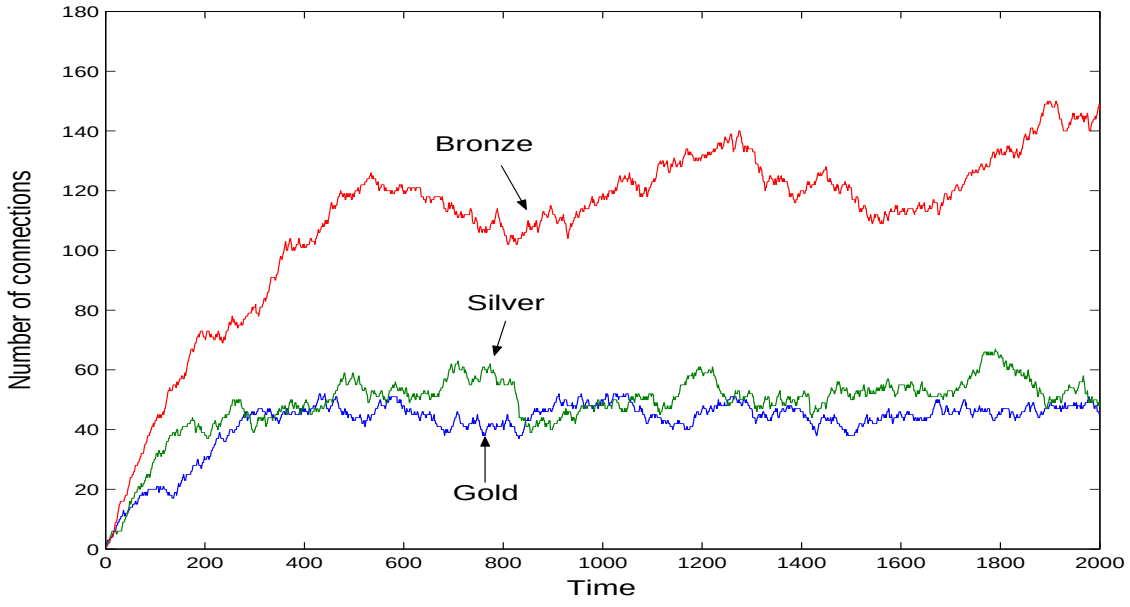


FIGURE 48 Adaptive weights - Evolution of the number of connections as a function of time in the third experiment.

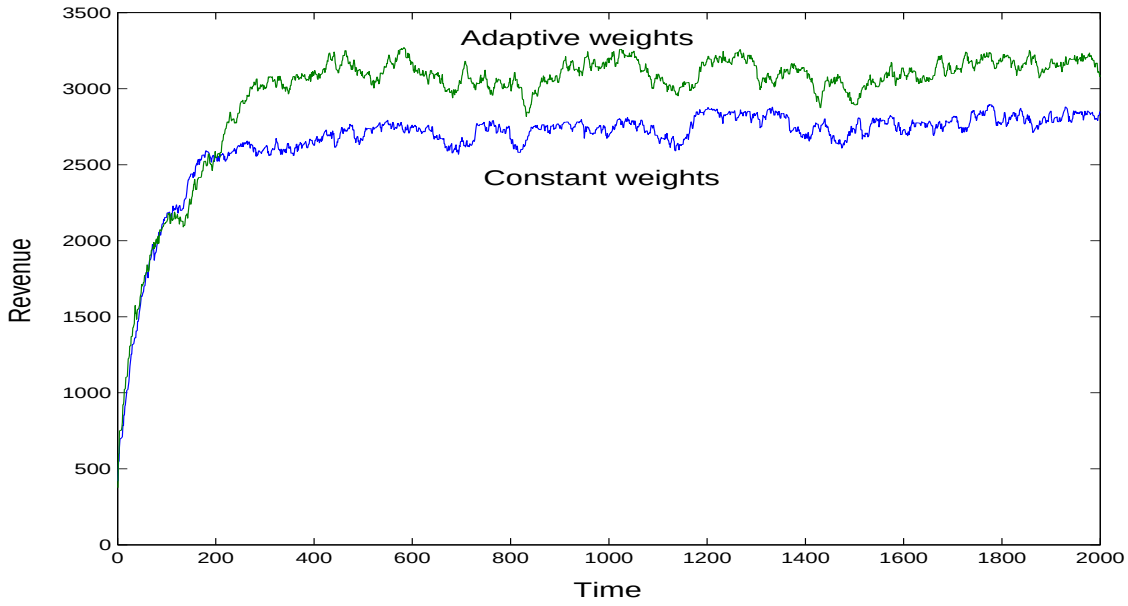


FIGURE 49 Evolution of the revenue F with constant and adaptive weights, as a function of time in the third experiment.

most of money while getting best service, and silver class pay less of money. Weights w_i determine how long time each queue is served. Natural constraint for the weights is

$$\sum_{i=1}^{m} w_i = 1 \tag{160}$$

Weights w_i have an influence to the QoS parameters, like delay d_i and bandwidth (bit rate) B_i. Price function is

$$f = f(d_1, ..., d_m, B_1, ..., B_m) \tag{161}$$

Delays and bit rates depend on the weights:

$$d_i = d_i(w_i) \tag{162}$$

$$B_i = B_i(w_i) \tag{163}$$

Therefore

$$f = f(w_1, ..., w_m) \tag{164}$$

Because delay decreases with respect to w_i and bit rate increases with respect to w_i, then $f(w_i)$ is increasing with respect to w_i. Pricing function $f(w_i)$ is strictly concave. (For example, bit rate is 1000 bit / sec for moving image and 1 bit / sec for text message, and price is 5 units for moving image and 1 units for text message. Then price is concave.) Then we can conclude that

$$\frac{\partial f}{\partial w_i} > 0, \tag{165}$$

$$\frac{\partial^2 f}{\partial w_i^2} < 0. \tag{166}$$

Revenue is Lagrangian problem

$$R = Revenue = f(w_1, ..., w_m) + \lambda(1 - \sum_{i=1}^{m} w_i) \tag{167}$$

Question 1: In what condition we obtain the closed form solution?

For example:

$$f(w_i) = \sum_{i=1}^{m} a_i w_i^p \tag{168}$$

has closed form solution. Proof is based on the Kuhn-Tucker conditions, which we will give here.

Revenue has the form

$$R = \sum_{i=1}^{m} a_i w_i^p + \lambda(1 - \sum_{i=1}^{m} w_i) \tag{169}$$

Then derivatives are as follows:

$$\frac{\partial R}{\partial w_i} = p a_i w_i^{p-1} - \lambda = 0, \tag{170}$$

from which follows that

$$\lambda = p a_i w_i^{p-1} \tag{171}$$

On the other hand

$$\frac{\partial R}{\partial \lambda} = 0 \tag{172}$$

yields

$$\sum_{i=1}^{m} w_i = 1. \tag{173}$$

To proceed:

$$w_i = \frac{\lambda^{\frac{1}{p-1}}}{(p a_i)^{\frac{1}{p-1}}} \tag{174}$$

Then

$$\begin{aligned} w_i &= \frac{\lambda^{\frac{1}{p-1}}}{(p a_i)^{\frac{1}{p-1}} \sum_{k=1}^{m} w_k} \\ &= \frac{\lambda^{\frac{1}{p-1}}}{(p a_i)^{\frac{1}{p-1}} \sum_{k=1}^{m} \frac{\lambda^{\frac{1}{p-1}}}{(p a_k)^{\frac{1}{p-1}}}} \\ &= \frac{\sum_{k=1}^{m} \frac{1}{(p a_k)^{\frac{1}{p-1}}}}{(p a_i)^{\frac{1}{p-1}}}. \end{aligned} \tag{175}$$

This is the closed form for the weights, and it is fast to implement.

To study uniqueness and optimality of the solution, let us consider the second order derivative of R. It has the form (notice that the penalty term λ is not a function of the weights w_i)

$$\frac{\partial^2 R}{\partial w_i^2} = p(p-1)a_i w_i^{p-2} < 0, \tag{176}$$

when $a_i > 0$ and $0 < p < 1$. But this proves that the solution is globally optimal, when the pricing function is increasing and strictly concave.

Next we consider the case of the solution, where closed form optimal approach is not possible to achieve. Let us consider the gradient of the revenue.

$$\frac{\partial R}{\partial w_i} = \frac{\partial f}{\partial w_i} - \lambda \tag{177}$$

Then

$$\lambda = \frac{\partial f}{\partial w_i} \tag{178}$$

So

$$\lambda = \lambda \sum_{i=1}^{m} w_i = \sum_{i=1}^{m} \lambda w_i = \sum_{i=1}^{m} \frac{\partial f}{\partial w_i} w_i \tag{179}$$

Then the first order derivative of the revenue is

$$\frac{\partial R}{\partial w_i} = \frac{\partial f}{\partial w_i} - \sum_{k=1}^{m} \frac{\partial f}{\partial w_k} w_k \tag{180}$$

Let us check the uniqueness of the solution. The second orde derivative of the revenue is

$$\begin{aligned} \frac{\partial^2 R}{\partial w_i^2} &= \frac{\partial^2 f}{\partial w_i^2} - \frac{\partial^2 f}{\partial w_i^2} w_i - \frac{\partial f}{\partial w_i} \\ &= \frac{\partial^2 f}{\partial w_i^2}(1 - w_i) - \frac{\partial f}{\partial w_i} < 0, \end{aligned} \tag{181}$$

when

$$\frac{\partial^2 f}{\partial w_i^2} < 0, \tag{182}$$

$$0 < w_i < 1, \tag{183}$$

$$\frac{\partial f}{\partial w_i} > 0. \tag{184}$$

Thus the solution is globally optimal in the gradient updating rule.

Because gradient shows the direction of largest increase of the function R with respect to the variables w_i, the weights are updated according to the gradient algorithm. During one iteration step, weights are updated according to the rule

1. Update weights

$$v_i(t) = w_i(t) + \mu \left. \frac{\partial R(N_1(t), \ldots, N_m(t), w_i)}{\partial w_i} \right|_{w_i = w_i(t)}. \tag{185}$$

2. Perform scaling

$$w_i(t+1) = \frac{v_i(t)}{\sum_{j=1}^{m} v_i(t)}. \tag{186}$$

Here t denotes time, and $N_i(t)$ shows that the number of connections vary as a function of time. Parameter μ is a forgetting factor, which is either constant or depend inversely on the norm of the gradient. It ensures time-varying nature of the weights due to the nonstationary data traffic.

Example. Let us show an example for this kind of approach. Let the pricing function be linear with respect to the delays, and square root with respect to the bit rate. Then, by simplifying the notation, we can present the revenue as follows:

$$R = \sum_{i=1}^{m} a_i w_i^{-1} + 2\sum_{i=1}^{m} b_i w_i^{\frac{1}{2}} + \lambda(1 - \sum_{i=1}^{m} w_i) \tag{187}$$

Here the parameters a_i and b_i denote pricing factors corresponding to the delays and bit rates, respectively. Then

$$\frac{\partial R}{\partial w_i} = -\frac{a_i}{w_i^2} + \frac{b_i}{w_i^{\frac{1}{2}}} - \sum_{k=1}^{m} \left(-\frac{a_k}{w_k} + b_k w_k^{\frac{1}{2}} \right). \tag{188}$$

Question 2: What is the general solution of the general pricing approach?

General solution for (167) is

$$w_i = \frac{\frac{\partial f}{\partial w_i} w_i}{\sum_k \frac{\partial f}{\partial w_i} w_i} \tag{189}$$

Here we give the derivation of that claim.

Derivative with respect to the weights in the multinode case is

$$\frac{\partial R}{\partial w_{ij}} = \frac{\partial f}{\partial w_{ij}} - \lambda_i = 0. \tag{190}$$

Then

$$\lambda_i = \lambda_i \sum_j w_{ij} = \sum_j \lambda_i w_{ij} = \sum_j \frac{\partial f}{\partial w_{ij}} w_{ij}. \tag{191}$$

Thus derivative of the revenue is

$$\frac{\partial R}{\partial w_{ij}} = \frac{\partial f}{\partial w_{ij}} - \sum_k \frac{\partial f}{\partial w_{ik}} w_{ik}. \tag{192}$$

Let us verify the derivative by direct substitution. Substitute Eq. (191) into Eq. (**??**). Then we obtain

$$R = f + \sum_s \sum_k \frac{\partial f}{\partial w_{sk}} w_{sk}(1 - \sum_l w_{sl}), \tag{193}$$

and remember the constraint (**??**). Then we obtain

$$\begin{aligned}
\frac{\partial R}{\partial w_{ij}} &= \frac{\partial f}{\partial w_{ij}} \\
&+ \sum_s \sum_k \frac{\partial^2 f}{\partial w_{ij} \partial w_{sk}} w_{sk} \\
&+ \frac{\partial f}{\partial w_{ij}} \\
&- \sum_s \sum_k \frac{\partial^2 f}{\partial w_{ij} \partial w_{sk}} w_{sk} \sum_l w_{sl} \\
&- \frac{\partial f}{\partial w_{ij}} \sum_l w_{il} \\
&- \sum_k \frac{\partial f}{\partial w_{ik}} w_{ik} \\
&= \frac{\partial f}{\partial w_{ij}} - \sum_k \frac{\partial f}{\partial w_{ik}} w_{ik}. \qquad (194)
\end{aligned}$$

But this is the same as (192), when $\{\sum_i w_i = 1\}$.

General solution is derived as follows:

$$\frac{\partial R}{\partial w_{ij}} = \frac{\partial f}{\partial w_{ij}} - \sum_k \frac{\partial f}{\partial w_{ik}} w_{ik} = 0 \qquad (195)$$

$$\frac{\partial f}{\partial w_{ij}} \frac{w_{ij}}{w_{ij}} = \sum_k \frac{\partial f}{\partial w_{ik}} w_{ik} \qquad (196)$$

$$w_{ij} = \frac{\frac{\partial f}{\partial w_{ij}} w_{ij}}{\sum_k \frac{\partial f}{\partial w_{ik}} w_{ik}} \qquad (197)$$

Notice from Eq. (198), that it scales automatically the weights, i.e. $\sum_j w_{ij} = 1$.

The algorithm for updating the weights is as follows. Update weights

$$v_{ij}(t) = \left. \frac{\frac{\partial f}{\partial w_{ij}} w_{ij}}{\sum_k \frac{\partial f}{\partial w_{ik}} w_{ik}} \right|_{w_i = w_i(t)} \qquad (198)$$

Question 3: In what general conditions this is a fixed point rule? This is an open question. **Question 4:** What is the speed of convergence of the fixed point algorithm? This is an open question, and it may be very hard to solve in the general case.

Question 5: What is the *natural* gradient algorithm for maximizing (167). When a parameter space has a certain underlying structure, the ordinary gradient of a function does not represent its steepest direction, but the natural gradient does. Information geometry is used for calculating the natural gradients in the parameter space of perceptrons, the space of matrices (for blind source

TABLE 1 Number of constraints for the adaptive models.

Class of schedulers	Number of the upper constraints	
	no	maximum
FQ	$m+1$	$2m+1$
RR	m $2m-1$	$2m$ $3m-1$

separation), and the space of linear dynamical systems (for blind source deconvolution). The dynamical behavior of natural gradient on-line learning can be proved to be Fisher efficient, implying that it has asymptotically the same performance as the optimal batch estimation of parameters. This suggests that the plateau phenomenon which appears in the backpropagation learning algorithm of multilayer perceptrons might disappear or might be not so serious when the natural gradient is used. Adaptive methods of updating the learning rate have been proposed and analyzed in the literature.

2.4 Computational complexity

The computational complexity is an important issue in the context of the presented adaptive models and in the context of the QoS networks. As a router should run the adaptive model every time the input parameters change, it is important that the model is fast and does not consume a significant amount of computational resources. Otherwise, rather than sending packets, the router will solve the optimization tasks.

Since most optimization tasks are solved using iterative methods, it is possible to express the complexity of the adaptive models by the number of iterations. Unfortunately, there is no simple formula that can estimate it. Depending on the used method and the type of constraints, the number of iterations differs. However, it is known that for the simplex method the worst-case number of iterations is 2^m-1, where m is the number of variables. It gives quite reasonable results even for the DiffServ framework, in which there can be six service classes.[7] Besides, in practice, the number of iterations is less. It depends on the number of constraints and their type. In the case of the adaptive model that runs on the top of the WFQ scheduler, there are $m+1$ constraints, each of which is of a simple form. In the case of the WRR and DRR schedulers, there are m constraints. In the LLQ mode, the number of constraints is $2m-1$. If the upper constraints are present, the overall number of constraints increases appropriately. Table 1 summarizes the number of constraints for each class of schedulers. Note that the RR schedulers can work either in the normal or in the LLQ mode.

As follows from the table, the number of constraints is almost the same for

[7] The six bit DSCP value can encode up to 64 classes. However, at the moment the IETF has standardized only six of them: EF, AF that includes four subclasses, and BE.

all schedulers when only the bandwidth guarantees must be provided. However, if there are the delay guarantees and the upper constraints, then the RR schedulers have the larger number of constraints. Furthermore, the adaptive models for the WRR and DRR schedulers are ILP problems, and there is no general worst-case estimation for such optimization problems. So, as a part of simulation, we will present the maximum number of iterations it takes each adaptive model to find the optimal solution.

It is interesting to note that while WFQ is the most demanding in implementation among the other considered queuing disciplines, the adaptive model for WFQ is the simplest one. So, there is a tradeoff between the complicated scheduler and the complicated adaptive model. It is a challenging issue that can be exploited to achieve optimal functioning in a certain environment.

2.5 Implementation requirements

The presence of the adaptive models requires that the schedulers apply new weight values as soon as possible. Otherwise, if the scheduler continues to work with the old parameters, it may impact the accuracy of the QoS provisioning. It should be noted that it is not a big issue for the bandwidth guarantees. However, if the scheduler also has to provide the delay guarantees, then failing to update working parameters in time may cause bigger queuing delays.

The task of applying new weight values can be thought of as a two-stage process. First, the adaptive model passes new values to the kernel of the operating system, in which the scheduling disciplines are implemented. Secondly, the scheduler starts to use them. For practical reasons, it is almost impossible to control the first stage as it depends on the design of the concrete operating system. However, by introducing small changes into the implementation of the schedulers, it is possible to reduce the time it takes new weight values to take effect.

The WFQ scheduler calculates the virtual finishing time of a packet based on the weight value and the list of active sessions, which is just a sum of the weights of active queues. This list is updated every time a packet arrives or leaves the scheduler. Thus, WFQ always works with the latest parameters set by the adaptive model and there is no need to change something in its behaviour.

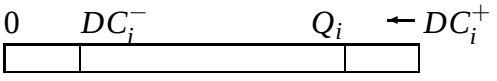


FIGURE 50 Deficit counter update procedure.

The original implementation of the DRR scheduler assumes that the deficit counters are updated only at the end of a round, which works fine for the static configuration. However, such a behaviour is unacceptable for the case when the

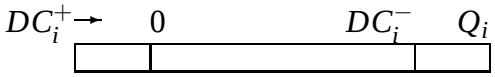


FIGURE 51 Deficit counter update procedure.

quantum values change all the time since they take effect only when all queues are served. Thus, it is proposed to update the deficit counter of a queue not at the of end a round, but after a queue has been served. It should be noted that this modification has no effect in the normal mode. However, in the LLQ mode, it allows to update the quantum value of the LLQ as soon as possible. Another modification of the DRR scheduler concerns the way the deficit counter is updated. Originally, it has been proposed to initialize it with the quantum value and then *subtract* the size of each transmitted packet. Then, the deficit counter is compared to 0. Such an approach fails to take into account the fact that the quantum value may change after the deficit counter has been initialized. Thus, to achieve more accurate resource allocation, it is proposed to initialize the deficit counter with 0, and then *add* the size of a transmitted packet. In this case, the deficit counter is compared to the quantum value. If the quantum value changes when a queue is served, it will be detected by the next comparison operation. Fig. 51 illustrates the difference between the original and the proposed implementation, where DC_i^+ stands for the value of the deficit counter at the beginning of a round, while DC_i^- is value of the deficit counter at the end. The arrow specifies the way the deficit counter changes as the DRR scheduler outputs packets.

The same concerns the WRR scheduler. Each queue must have a counter of the transmitted packets that is initialized to 0 and incremented by one after a packet has been served. If this counter equals the weight value associated with a queue, the next queue is served.

2.6 Adaptive router

Fig. 52 presents the structure of a router that implements the adaptive model. As follows from this figure, the structure does not undergo significant changes. Along with standard components, such as the packet classifier, set of queues, and a scheduler, there is a new module that is responsible for calculating parameters of the underlying scheduling discipline. Hence, this component will be referred to as the Dynamic Service Weight (DSW) calculator. It takes the configuration parameters, such as the number of active flows within each class and their QoS requirements, and calculates the optimal weight values that are sent to the scheduler. It is noticeable that the DSW calculator is decoupled from the scheduler. As a result, the complexity of the scheduler is not increased, and these two modules can function independently based on their settings. Practically, the adaptive model would be implemented as a system process running in the user space,

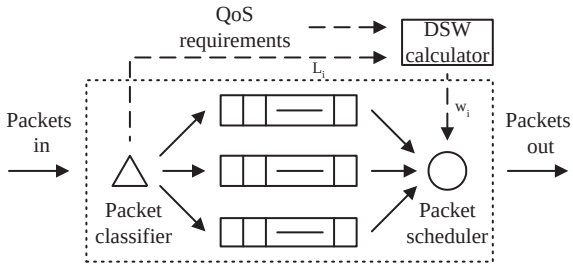


FIGURE 52 Structure of the adaptive router.

while the kernel of the operating system makes the scheduling decisions.

In the case of the adaptive model for the WRR scheduler, the mean, maximum, and minimum packet sizes are necessary. Though an administrator can specify these values manually, more accurate results will be achieved if a router updates this value [76]. This additional functionality can be implemented in the classifier as the latter examines the contents of a packet to put it into an appropriate queue.

The adaptive models are irrespective of the way the router obtains the QoS information concerning the data flows. Instead, the QoS signalling and configuration mechanisms are governed by a concrete QoS framework used by a provider. Section 4 considers the major QoS frameworks and their integration with the adaptive models.

2.7 Integration with other models

Measurement models

To achieve better allocation of resources, it is possible measure the effective parameters and adjust the QoS requirements sent by a user. For instance, by measuring the burstiness of the incoming flows, one can improve the estimation of the worst-case queuing delay. The same can be done for the bandwidth. For instance, if a certain flow requests 100 Kbps but the measurements show that it occupies only 80 Kbps, then we can allocate less resources thus providing resources for the more expensive one. The measurement models are very useful add-ons to the adaptive models since they can increase the total revenue.

The integration of the measurement models with the adaptive resource allocation seems to be quite straightforward. As the model detects that less resources can be allotted to a specific flow and / or service class, new QoS requirements are constructed and the adaptive models are rerun.

Pricing models

As mentioned earlier, we have assumed that the service prices are constant and do not change in the course of time. However, pricing of network services may

depend on various external parameters, such as time of the day, network state, request bandwidth etc. The congestion-dependent pricing has been presented in [118]. In [157], it is proposed to set prices based on the level of service usage and congestion. Adaptive users adapt to price changes by adjusting their sending rate or selecting a different service class. A similar approach is described in [20] where user applications and the network negotiate continuously the price based on the requested QoS guarantees. In [53], a distributed flow control technique has been proposed that is based on microeconomics. Switches price their link bandwidth based on supply and demand, and users purchase bandwidth so as to maximize their QoS.

The integration of the adaptive resource sharing with the pricing models is quite simple. As these models determine new prices, they can be passed to the models in the same way as the new QoS parameters are set.

2.8 Summary

In this chapter, the adaptive models for the WFQ, WRR, and DRR scheduling disciplines have been presented. The models are per se the optimization tasks that work as a superstructure over the considered scheduling disciplines. They use the prices for network services to find the optimal scheduler configuration that enables a provider to increase the total revenue. It has been shown that the model for WFQ can be changed easily to support other FQ disciplines, while the models for WRR and DRR reflect the basic properties of the RR schedulers.

To achieve small queuing delay in the RR schedulers, the LLQ has been introduced. When it works in the alternate priority mode, it serves the delay critical queue in between other queues. For this mode, this chapter has presented a new worst-case delay estimation that differs from the ones presented in [145] and [83]. To support the adaptive resource sharing, it has been proposed to introduce several changes in the implementation of the DRR scheduler. First, the deficit counter is updated not at the end of a round, but after a queue has been served. Secondly, the deficit counter value moves from zero to the quantum value. By this, a new implementation ensures that the DRR scheduler takes account of the new quantum value for the queue even when that queue is being processed. Such an implementation allows to avoid situations when the packets may experience larger queuing delays.

The chapter has presented the theoretical analysis of the computational complexity of the proposed adaptive models. It is anticipated that the model for WFQ will be the fastest one, while the models for WRR and DRR will need more iterations to the find the optimal solution.

The flexibility of the adaptive models allow them to be integrated easily with other models, such as the measurement and pricing model. By this, a provider can achieve more optimal resource allocation. Furthermore, a provider can modify the target function so that the adaptive models take other parameters into

account. It is also worth mentioning that the adaptive models can be modified in such a way, that the optimal weight values are calculated for the flows. It might be necessary if the scheduler allocates resources on the per-flow basis and each session corresponds to a flow, not a class. For these purposes, it is enough to assume that i refers to the ith data flow and substitute B_i^f to $B_i^f N_i$ in all the QoS constraints. The same is for the burst size. However, as considered in sections 2.1.2 and 2.3.2, the allocation of resources on the per-flow basis may lead to the increased load on the scheduler if the number of sessions is huge.

3 STOCHASTIC ANALYSIS OF UPPER DELAY BOUND OF GPS- BASED PACKETIZED FAIR QUEUING ALGORITHMS

The provision of Quality-of-Service (QoS) guarantees such as bandwidth, delay, jitter and cell loss to applications with widely different characteristics is a primary objective in the design of next-generation networks. One important issue in the provision of QoS guarantees is the study of the queueing algorithms employed at network nodes. Among the queuing algorithms that have been proposed, the class of algorithms which aim at approximating the Generalized Processor Sharing (GPS) policy are most popular. GPS [114, 115] is an idealized fluid discipline with a number of very desirable properties: (i) it provides minimum QoS guarantees to each traffic session [1], regardless of the behavior of other sessions; (ii) it provides the deterministic worse-case delay bound to each session whose traffics are leaky-bucket constrained; and (iii) it ensures fairness in the amount of service provided by a network node to competing sessions. Since this policy is a fluid model, it is not adapted for packet-by-packet transmission. Therefore, its packet-based extensions that we call GPS-based Fair Queueing algorithms have been proposed, well-known examples of which include Weighted Fair Queuing (WFQ) [44, 114], Self-Clocked Fair Queuing (SCFQ) [58], Frame-based Fair Queuing (FFQ) [149] and Starting Potential-based Fair Queuing (SPFQ) [149].

A significant volume of work in the literature [44, 114, 58, 59, 62, 148, 149, 33] has been concerned with evaluating the deterministic worst-case delay guarantees that GPS-based Fair Queueing (FQ) algorithms can provide when the burstiness of the traffic feeding them is bounded (mostly shaped by a leaky bucket). However, little work has been reported on analyzing the stochastic delay bounds of such packetized policies under a general probabilistic traffic model. This has been mainly due to the difficulty of stochastically modelling the complex behavior of a GPS-based FQ algorithm. Indeed an important advantage of stochastic

[1] in this Chapter and publication [177], the notion of session actually means the aggregate of packet streams which require the same QoS level and thus it is exchangeable with the notion of service class.

modelling of FQ systems as compared to worst-case deterministic analysis is that statistical analysis takes into account the actual dynamics of the packet arrival process, thus being more accurate in predicting the QoS guarantees provided to each traffic session and also being able to derive more efficient revenue-aware resource allocation scheme for maximizing SLA revenues in a network node deploying GPS-based FQ algorithms.

Zhang *et al* [174] studied the statistical behavior of GPS discipline using exponentially bounded burstiness processes [170] as the session source traffic model and derived upper bounds on the tail distributions of session backlog and delay. However, no simulation results were provided to verify the quality of those bounds. Pekergin [121] derived stochastic bounds on the delay distribution of GPS-related FQ algorithms fed by a Switched Bernoulli Batch process. The analysis in [121] is quite complex and does not result in explicit analytical equations, thus limiting its usefulness for back-of-the-envelope calculations and comparisons. The analysis also makes some limiting assumptions such as the use of fixed packet lengths and the need to set all sessions other than the tagged one to be greedy all the time. M. Hawa *et al* used the bounded fairness criterion of FQ algorithms to derive the upper and lower bounds on mean packet delay of those algorithms under Poisson arrivals [68]. However, the analysis in [68] assumes that all sessions have the exact same packet length distribution, which limits its application scope.

In publication [177], we extend the notion of *feasible partition* introduced by Zhang *et al* [174] for the analysis of idealized GPS discipline and apply it to the stochastic bound analysis of GPS-based FQ algorithms. With the help of this notion, we derive our new upper bound on mean packet delay under the general probabilistic traffic model of Poisson arrivals and any general packet length distribution. The resulted upper bound is much simpler and tighter than the ones by M. Hawa *et al* [68]. Moreover, it fits a class of GPS-based FQ algorithms including WFQ, SCFQ and SPFQ, which aims at approximating GPS discipline and thus validate the notion of *feasible partition*. Finally, in this Chapter our new upper bound on mean packet delay is utilized to derive the suboptimal resource allocation scheme under a given amount of network resources and flat pricing strategy for maximizing SLA revenues in a network node deploying a GPS-based packetized FQ algorithm.

3.1 Upper Bound on Mean Packet Delay of GPS-based Packetized FQ Algorithms

Consider a single-server GPS-based FQ system with capacity C bits/s, which multiplexes N sessions with Poisson arrival rates $\lambda_1, \lambda_2, ..., \lambda_N$ (packets/s). In the FQ system, each session has its own queue. Assume that the queues corresponding to different sessions are infinite in length and the packets in the same queue are served in the order they arrive. We use L_i to denote session i packet length

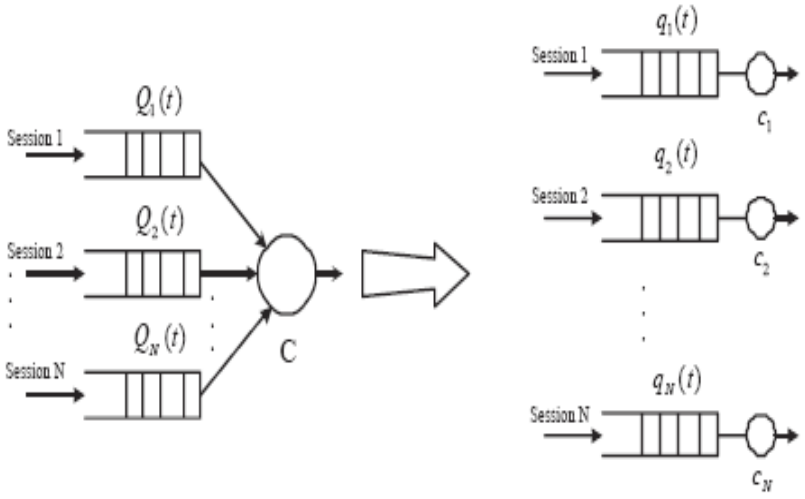


FIGURE 53 Decomposition of a GPS-based FQ system.

(in bits). As mentioned above, the distribution of L_i can be any general distribution and $\bar{L}_i$ is used to denote session i mean packet length, i.e., $E[L_i] = \bar{L}_i$. As a necessary stability condition, $\sum_{i=1}^{N} \lambda_i \bar{L}_i < C$ is required. Furthermore, the service weight w_i determines the minimum guaranteed share of capacity assigned to session i when session i is backlogged. Obviously, $\sum_{i=1}^{N} w_i = 1$ and $0 < w_i \leq 1$, $i \in [1, N]$.

Let $\mathcal{N} = \{1, 2, ..., N\}$. In publication [177], first a sequence of disjoint sets, $\mathcal{H} = \{H_k\}_{1 \leq k \leq m}$, where $m \geq 1$ and $H_1 \cup \cdots \cup H_m = \{1, 2, ..., N\}$, is defined as the *feasible partition* of $\mathcal{N}$ with respect to the given sets of arrival rates $\{\lambda_i \bar{L}_i\}_{i \in \mathcal{N}}$ (bits/s) and service weights $\{w_i\}_{i \in \mathcal{N}}$. Then, using the *decomposition approach* introduced in [174], the above GPS-based FQ system can be analyzed through a set of N separate single-queue systems, each of which has a dedicated server with capacity c_i (referred to as the *virtual decomposed system*) (see Figure 53).

Let $X_i = L_i / (w_i C)$ denote the service time of session i packets in the corresponding virtual decomposed system where a dedicated server with capacity $c_i = w_i C$ serves session i packets and $E[X_i] = \bar{X}_i$ and $E[X_i^2] = \bar{X}_i^2$. Moreover, the mean delay of session i packets in the FQ system is denoted by d_i, which equals the mean waiting time in queue plus the mean service time. Publication [177] derives the following upper bound on mean packet delay $\bar{d}_i$ as long as $\lambda_i \bar{L}_i < w_i C$ (i.e., session i in the GPS-based FQ system is a session in H_1):

$$\bar{d}_i \leq \bar{X}_i + \frac{\lambda_i \bar{X}_i^2}{2(1 - \lambda_i X_i)}. \tag{199}$$

It can be noticed that the above upper bound on mean packet delay is derived under the general probabilistic traffic model of Poisson arrivals and any general packet length distribution. Moreover, the simulation results in Publication [177] demonstrates that the derived upper bound is much tighter than the ones by M. Hawa *et al* [68] and it fits a class of GPS-based packetized FQ algorithms including WFQ [44, 114], SCFQ [58] and SPFQ [149].

3.2 Revenue-aware resource allocation scheme in a GPS-based network node under flat pricing strategy

Similarly, when *mean packet delay* is chosen as the QoS metric in a SLA, a network server provider may receive SLA revenues or penalties in a GPS-based network node based on the offered QoS (mean packet delay here) guarantees and the deployed pricing strategy. In this part, the above upper bound on mean packet delay in Eq. (199) is utilized to derive the revenue-aware resource allocation scheme in a GPS-based network node under flat pricing strategy. Specifically, first the flat pricing function which characterizes the flat pricing strategy is generally defined. Then the suboptimal resource allocation scheme is presented for maximizing SLA revenues in a GPS-based network node under flat pricing strategy, whose performances are investigated in the following simulation part.

3.2.1 Flat pricing strategy

Consider the above GPS-based FQ system. As *mean packet delay* is deployed as the QoS metric in the SLA, the flat pricing strategy for class [2] i is characterized by the following definition of a flat pricing function.

Definition 3: *The function*

$$r_i(\bar{d}_i) = \begin{cases} R_i & \texttt{if } \bar{d}_i \le D_i \\ -P_i & \texttt{if } \bar{d}_i > D_i \end{cases}, \quad i = 1, 2, ..., N \tag{200}$$

is called the *flat pricing function* of class i, where D_i is the QoS (mean packet delay) guarantee requested by class i packets and R_i and P_i are both positive constants. Obviously, the above flat pricing function specifies that if the real mean packet delay of class i is less than D_i, the network service provider receives a revenue R_i, otherwise a penalty P_i is incurred for failing to meet that QoS guarantee D_i. Moreover, $R_i \ge R_j$ and $P_i \ge P_j$ should hold to ensure differentiated pricing if class i has higher priority than class j, which are actually what we expect based on the SLA requirement. Figure 54 presents a example of the flat pricing functions of Gold, Silver and Bronze classes.

[2] The notion of class is used instead of session hereafter in this Chapter as they are exchangeable.

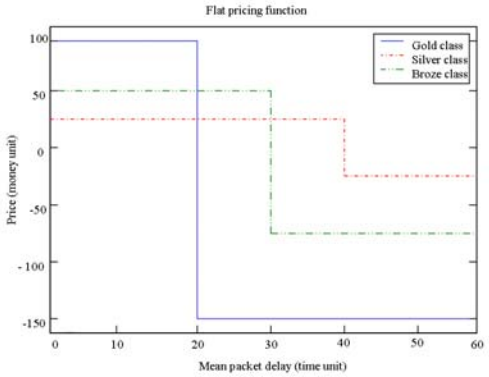


FIGURE 54 The flat pricing functions of Gold, Silver and Bronze classes.

3.2.2 Suboptimal resource allocation scheme in a GPS-based network node under flat pricing strategy

Consider a GPS-based network node with capacity C bits/s and a total of N service classes supported. As the QoS metric in the SLA is *mean packet delay*, obviously the SLA revenue F obtained in the GPS-based network node under one charging period is defined as follows.

$$F = \sum_{i=1}^{N} r_i(\bar{d}_i). \tag{201}$$

Based on the definition of flat pricing function in Eq.(200), it is clear that if $\bar{d}_i \leq D_i$ holds for each service class $i \in [1, N]$, F achieves its maximum value $\sum_{i=1}^{N} R_i$. Since the mean packet delay of class i is bounded tightly by our derived upper delay bound as long as $\lambda_i \bar{L}_i < w_i C$, the problem of deriving the suboptimal resource allocation scheme for maximizing F under flat pricing strategy can be formulated as follows by linking parameter D_i in the flat pricing function of class i with the upper delay bound in Eq. (199).

$$\texttt{Solve} \qquad \bar{d}_i \leq \bar{X}_i + \frac{\lambda_i \bar{X_i^2}}{2(1-\lambda_i X_i)} \leq D_i \tag{202}$$

$$\texttt{s.t.} \qquad \sum_{i=1}^{N} w_i \leq 1, \ 0 < w_i \leq 1, \tag{203}$$

$$w_i C > \lambda_i \bar{L}_i. \tag{204}$$

The formula of our upper delay bound in Eq. (199) is presented based on a general packet length distribution. Below, we illustrate how to derive the suboptimal resource allocation scheme by solving the right inequality in (202) under some practical packet length distributions in IP networks. First, under exponential packet length distribution, $E[L_i] = \bar{L}_i$ and $E[X_i^2] = 2(\bar{L}_i)^2$, resulting in $E[X_i] = \bar{X}_i = \bar{L}_i/(w_i C)$ and $E[X_i^2] = \bar{X_i^2} = 2[\bar{L}_i/(w_i C)]^2$. Substitute the above $\bar{X}_i$

and $\bar{X_i^2}$ into (202), then the right inequality in (202) becomes:

$$\frac{\bar{L_i}}{w_i C - \lambda_i L_i} \leq D_i$$

leading to the following solution under the exponential packet length distribution:

$$w_i \geq \frac{\bar{L_i}}{C}(\lambda_i + \frac{1}{D_i}), \ \ i = 1, 2, ..., N. \tag{205}$$

That is to say that we should allocate at least the capacity of $C_{i,min} = w_{i,min}C = \bar{L_i}(\lambda_i + 1/D_i)$ to class i so that its mean packet delay can be guaranteed to be less than D_i during the charging period. As a result of the above solution to w_i in (205), we present the suboptimal resource allocation scheme for maximizing the SLA revenue obtained in a GPS-based network node under a given amount of network resources and flat pricing strategy as follows:

1. Set $C_{i,min} = w_{i,min}C = \bar{L_i}(\lambda_i + 1/D_i)$, i=1,2,...,N,

2. If $\sum_{i=1}^{N} C_{i,min} \leq C$, to reserve capacity $C_{i,min}$ for any class $i \in [1, N]$ is the suboptimal resource allocation scheme in this case. By the suboptimal scheme, the network service provider will receive the maximum SLA revenue $\sum_{i=1}^{N} R_i$ during one charging period under flat pricing strategy,

3. Otherwise, it means that the total capacity C of the GPS-based network node is not enough to satisfy the reservation of capacity $C_{i,min}$ for all the supported service classes. Hence, we should first guarantee the reservation of capacity $C_{i,min}$ for a set of selected service classes so that the obtained SLA revenue is the highest in this situation. Then the remaining resources are allocated to all the supported service classes other than the above selected ones uniformly.

Note that the above set of selected service classes in Step 3 is acquired by the comparison of the analytic SLA revenues under all possible resource allocation schemes, which makes its calculation complexity increases quickly with larger values of N. Hence, instead we may first reserve capacity $C_{1,min}$ for the highest priority class, then reserve capacity $C_{2,min}$ for the second highest priority class until the remaining capacity is not enough to satisfy the reservation of capacity $C_{i,min}$ for class i (we have assumed that class 1 is the highest priority class and class N is the lowest one), which also tries to achieve the highest SLA revenue in this case as the incurred penalty due to failing to meet the QoS guarantee of higher class is larger.

For the network traffics which exhibit self-similarity nature, we use *Bounded Pareto* to model the heavy-tailed distribution as used in [38]. In this Chapter, $BP(p_i, q_i, \alpha_i)$ is used to denote Bounded Pareto packet length distribution of class i, where p_i is the smallest length of class i packets, q_i the largest ($p_i \leq L_i \leq q_i$), and α_i the shape parameter. Then,

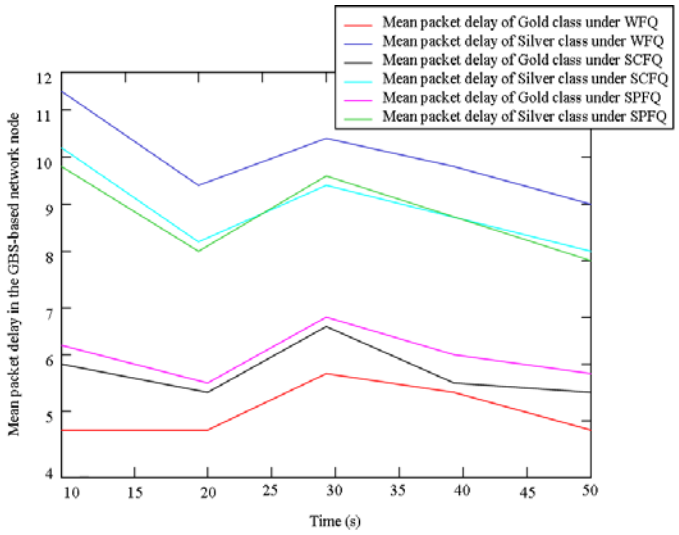


FIGURE 55 Mean packet delay in the first simulation where exponential packet length distribution is deployed.

$E[L_i] = \bar{L}_i = \alpha_i p_i^{\alpha_i}(p_i^{1-\alpha_i} - q_i^{1-\alpha_i}) / [(\alpha_i - 1)(1 - (p_i / q_i)^{\alpha_i})]$ and $E[L_i^2] = \bar{L}_i^2 = \alpha_i p_i^{\alpha_i}(p_i^{2-\alpha_i} - q_i^{2-\alpha_i}) / [(\alpha_i - 2)(1 - (p_i / q_i)^{\alpha_i})]$, resulting in $E[X_i] = \bar{X}_i = \bar{L}_i / (w_i C)$, $E[X_i^2] = \bar{X}_i^2 = \bar{L}_i^2 / (w_i C)^2$. Thus, the right inequality in (202) becomes as follows:

$$\frac{2\bar{L}_i(w_i C - \lambda_i \bar{L}_i) + \lambda_i \bar{L}_i^2}{2w_i C(w_i C - \lambda_i \bar{L}_i)} \leq D_i$$

leading to the solution under the Bounded Pareto packet length distribution:

$$w_i \geq \frac{(1 + \lambda_i D_i)\bar{L}_i + \sqrt{[(1 + \lambda_i D_i)\bar{L}_i]^2 - 2\lambda_i D_i(2\bar{L}_i^2 - \bar{L}_i^2)}}{2D_i C}, \; i = 1, 2, ..., N, \quad (206)$$

i.e., $C_{i,min} = w_{i,min} C = \left[(1 + \lambda_i D_i)\bar{L}_i + \sqrt{[(1 + \lambda_i D_i)\bar{L}_i]^2 - 2\lambda_i D_i(2\bar{L}_i^2 - \bar{L}_i^2)}\right] / 2D_i$ in this case. Then the suboptimal resource allocation scheme under the Bounded Pareto packet length distribution can also be derived based on the above approach.

3.2.3 Simulation results

In this section, we present some simulation results to illustrate the effectiveness of the above derived suboptimal resource allocation scheme in GPS-based network node under flat pricing strategy. Throughout the following simulations, three representative GPS-based FQ algorithms were used in a GPS-based network node: WFQ [44, 114], SCFQ [58] and SPFQ [149].

In the first simulation, we consider a GPS-based network node which has initial capacity C=1Mb/s and supports two service classes (namely Gold class and Silver class) with Poisson arrivals. The arrival rates of Gold and Silver classes

are $\lambda_1 = 350$ packets/s and $\lambda_2 = 200$ packets/s, respectively (class 1 indicates Gold class and class 2 means Silver class). The packet length distributions of the two classes are both exponential with $\bar{L}_1 = 1000$ bits and $\bar{L}_2 = 2000$ bits. Moveover, the parameters deployed in the two flat pricing functions are summarized as follows: $D_1 = 10$ms, $R_1 = 10$ money units, $P_1 = 15$ money units and $D_2 = 20$ms, $R_2 = 5$ money units, $P_2 = 8$ money units.

By the calculation of the inequality in (205), it is shown that at least capacity 0.45Mb/s should be reserved for Gold class (i.e., $C_{1,min}$=0.45Mb/s) and at least capacity 0.5Mb/s for Silver class (i.e., $C_{2,min}$=0.5Mb/s) to guarantee the satisfaction of both $\bar{d}_1 \leq D_1 = 10$ms and $\bar{d}_2 \leq D_2 = 20$ms. As $C_{1,min} + C_{2,min} < C$, the suboptimal resource allocation scheme is that the reserved capacity for Gold class is 0.45Mb/s, the reserved capacity for Silver class is 0.5Mb/s and only 0.95Mb/s of capacity is needed. Hence, the real capacity of the GPS-based node is set to 0.95Mb/s in the first simulation and the simulation results are presented in Figure 55, where it can be seen that $\bar{d}_i \leq D_i$ holds for each class $i \in [1,2]$ during each charging period (20s here) when any one of WFQ, SCFQ and SPFQ is deployed. Hence, it is demonstrated that the derived suboptimal resource allocation scheme achieves the maximum value $\sum_{i=1}^{2} R_i$=15 money unit of SLA revenue in the GPS-based network node under flat pricing strategy.

In the second simulation, the above GPS-based network node is fed by two classes of Poisson packet streams with arrival rates $\lambda_1 = 100$ packets/s, $\lambda_2 = 120$ packets/s and a heavy-tailed packet length distributions. The packet length distributions of the two classes are both modelled by Bounded Pareto with parameters $b_1 = 40$ bits, $p_1 = 12800$ bits, $\alpha_1 = 0.137$ (thus $\bar{L}_1 = 2000$ bits), and $b_2 = 320$ bits, $p_1 = 16000$ bits, $\alpha_2 = 0.164$ (thus $\bar{L}_2 = 3360$ bits). The parameters of the two flat pricing functions in this case are: $D_1 = 15$ms, $R_1 = 20$ money units, $P_1 = 30$ money units and $D_2 = 20$ms, $R_2 = 10$ money units, $P_2 = 15$ money units.

Similarly, by the calculation of the inequality in (206), it is gotten that $C_{1,min} = 0.3694Mb/s$, $C_{2,min} = 0.5857Mb/s$ and $C_{1,min} + C_{2,min} = 0.9551Mb/s < C = 1Mb/s$. Hence, the suboptimal resource allocation scheme in this case is that the reserved capacity for Gold class is 0.3694Mb/s, the reserved capacity for Silver class is 0.5857Mb/s and only 0.9551Mb/s of capacity is needed. Hence, the real capacity of the GPS-based node is set to 0.9551Mb/s in the second simulation and Figure 56 shows the simulation results.

The results in Figure 56 also indicate that $\bar{d}_i \leq D_i$ is satisfied for each class $i \in [1,2]$ during each charging period (50s in this case), leading to the achievement of the maximum SLA revenue $\sum_{i=1}^{2} R_i$=30 money unit obtained within one charging period in this case.

Finally, the third simulation was made to evaluate the performance of the suboptimal resource allocation scheme derived by the above approach for the case that the capacity C of a GPS-based network node is not enough to guarantee the reservation of $C_{i,min}$ for all its supported service classes. Hence, the same parameter settings and exponential packet length distributions as the ones in the first simulation are deployed in the third simulation except that the arrival rates of Gold and Silver classes are $\lambda_1 = 450$ packets/s and $\lambda_2 = 200$ pack-

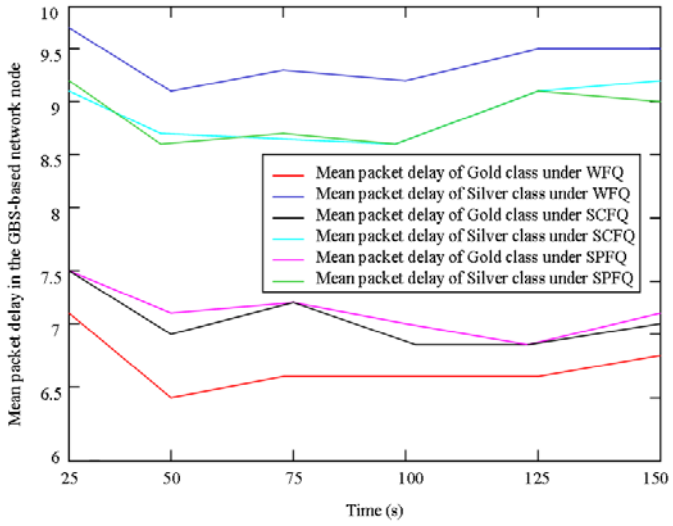


FIGURE 56 Mean packet delay in the second simulation where Bounded Pareto packet length distribution is deployed.

ets/s, respectively. By the inequality in (205), we know that $C_{1,min}$=0.55Mb/s, $C_{2,min}$=0.50Mb/s and $C_{1,min} + C_{2,min} = 1.05\text{Mb/s} > C = 1\text{Mb/s}$. Thus, according to our proposed approach, the suboptimal resource allocation scheme under flat pricing strategy in this case is as follows: the reserved capacity for Gold class is 0.55Mb/s, the reserved one for Silver class is 0.45Mb/s and all of the capacity C=1Mb/s of the GPS-based node is used. The simulation results are presented in Figure 57. It is shown in Figure 57 that both $\bar{d}_1 \leq D_1$ and $\bar{d}_2 \leq D_2$ are satisfied when any one of WFQ, SCFQ and SPFQ is deployed, which also results in the achievement of the maximum SLA revenue $\sum_{i=1}^{2} R_i$=15 money units, although the reserved capacity for Silver class is less than $C_{2,min}$ in this case. The reason is that in a network node which deploys any GPS-based FQ algorithm, the reserved resources (the capacity in this case) for a certain class indicate only the minimum guaranteed resources allotted to that class and the actual real resources received by that class may exceed the reserved one. However, as we do not know the exact amount of resources received by a certain class in the real situation due to the characteristic of GPS-based FQ algorithms, it is the best to derive the suboptimal resource allocation scheme by the tightest upper delay bound of GPS-based FQ algorithms.

Therefore, based on the above simulation results, we can conclude that the above approach of deriving the suboptimal resource allocation scheme in a GPS-based network node is effective and the derived resource allocation scheme can achieve the highest SLA revenue under a given amount of network resources and flat pricing strategy. More importantly, it also fits a class of GPS-based packetized FQ algorithms.

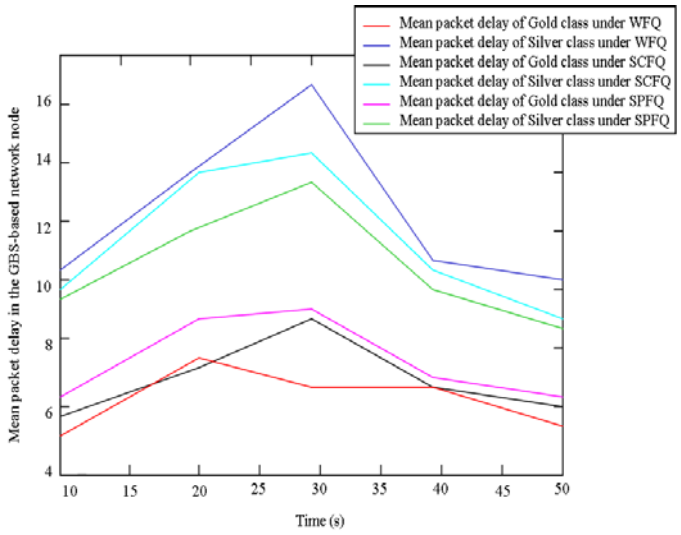


FIGURE 57 Mean packet delay in the third simulation where exponential packet length distribution is deployed.

4 QOS FRAMEWORKS

Up to this point, we have been considering the adaptive models as the means to ensure the QoS requirements at one node. However, as a packet moves through the network, it traverses multiple nodes that may belong to different administrative domains. Thus, each intermediate node must provide some level of QoS so that the network provides some level of the end-to-end QoS parameters. In most cases, the problem is formulated so: "having the set of the end-to-end QoS requirements, what should the router configuration be?" It is understandable, that the adaptive models cannot solve this problem since they do not possess the network-wide picture. Instead, they solve the problem of the local optimal configuration. Thus, other technologies are necessary, which will make decisions based on the state of the whole network, while the adaptive models will achieve the optimal resource allocation within routers.

As an example, it is possible to consider the following simple scenario. Suppose the end-to-end service requirements of a data flow are 100 Kbps and the worst-case end-to-end delay of 100 ms. It means that all the intermediate routers must allocate 100 Kbps for the flow. However, some entity must transform the end-to-end delay into the worst-case queuing delay of each intermediate router based on their number and the characteristics of links that connect them. After that the bandwidth and the worst-case queuing delay can be passed to the adaptive model that will recalculate the optimal parameters for the scheduler.

At the moment, the IETF has proposed two QoS frameworks for the Internet: Integrated Service and Differentiated Services. The subsequent sections present the overview of these frameworks and the way the proposed adaptive models can be integrated into them.

4.1 Integrated Services

The Integrated Services (IntServ) [21] represents the application-signalled approach of the QoS architecture. The application frames its service request within

the RSVP protocol [22] and then passes this request into the network. The network can either respond positively in terms of its agreement to commit to this service profile, or it can reject the request. If the network commits to the request with a resource reservation, the application can send traffic into the network. The reservation remains in force until the application explicitly requests termination of the reservation, or the network signals to the application that it is unable to continue with a service commitment. The essential feature of the IntServ model is "all or nothing" nature. In other words, the IntServ framework either provides all the QoS guarantees or replies negatively.

The IntServ QoS model requires that the network must maintain the remembered state to describe the resources that have been reserved and the network path over which the reserved service will operate. In addition, each active network element within the network must maintain a local state that allows incoming IP packets to be correctly classified into a reservation class.

4.1.1 Service classes

In addition to the BE class, the IntServ framework specifies two kinds of services: the Guaranteed Service (GS) and the Controlled Load (CL). It should be noted that a term class does not refer to the way the scheduler allocates resources, but rather to the treatment that packets, which belong to a certain service class, receive. Table 2 presents a brief summary of the IntServ classes and the QoS parameters that each class can provide.

TABLE 2 Summary of the IntServ classes.

Class	QoS requirements			Policy action
	Bandwidth	Delay	Loss	
GS	√	√	√	drop, mark, shape
CL	√		√	mark
BE / NULL				

Guaranteed Service

The GS class [142] is intended for applications that need a firm guarantee that datagrams will arrive within the guaranteed delivery time and will not be discarded due to the queue overflows. For these purposes, the IntServ routers estimate the worst-case delay of the network and allocate such amount of bandwidth, that this delay estimation is met. The GS service does not attempt to minimize the jitter, but rather controls the maximum queuing delay. If the maximum queuing delay is bounded by a certain value, then the jitter cannot exceed it.

Though each application informs network about the anticipated traffic load, it can happen that for some reason the traffic load increases. In fact, in certain circumstances a large number of packets may fail the conformance test as a matter of normal operation. Thus, the IntServ routers have to meter the incoming traffic

to classify it into the in- and out-profile. There are several forms of policing in GS for the out-profile packets. They can be treated as the best-effort traffic and forwarded appropriately. However, such a behaviour may lead to the packet re-ordering. Thus, another, and the more preferred, action for the out-profile packets is to drop them. A provider may also implement a shaper, which restores traffic shape to conform to its profile.

Controlled Load

The CL class [161] is intended to support a broad class of applications that are highly sensitive to overloaded conditions. It approximates the behaviour of network visible to applications receiving the best-effort service under unloaded conditions. The aim of this service is to provide sufficient bandwidth and buffer resource to ensure the bandwidth requirement and to minimize the packet loss. The CL class does not try to ensure any delay requirements.

The CL service has no strict requirements concerning the non-conforming packets. It is permissible to degrade the service delivered to all of the flow's packets equally. In other words, if an application start to send data at higher data rates, then some packets may be dropped including the ones that conform to the profile. Another solution is to sort packets into a conformant and non-conformant sets and deliver different levels of service for each set.

Best Effort/NULL service

As follows from the name, the BE class does not provide any QoS requirements for the applications, which, in turn, do not use the RSVP signalling. Since each RSVP router keeps a list of the GS and CL flows, all flows that are not on this list will be mapped automatically to the BE class.

Along with ordinary network services, there are also mission critical applications that require some form of prioritization, but cannot readily specify their resource requirements. To serve such applications, the notion of the NULL service was introduced [14]. The NULL service allows applications to identify themselves to a network using the RSVP protocol, however, it does not require them to specify the resource requirements. Such a behaviour is useful for those cases when an application wants better than the BE treatment and the network is responsible for allocating resources based on the application type. A customer and/or an application can identify themselves to the network by including the policy object [72] into the RSVP message, as described in [165]. This mode of usage is particularly applicable to networks that combine differentiated services QoS mechanisms with the RSVP signalling.

4.1.2 Adaptive model

The proposed adaptive models can be integrated into the IntServ QoS framework easily. The adaptive model is capable of providing the bandwidth and delay guarantees that are necessary for the GS and CL service classes. At the same time,

some fixed portion of the output bandwidth may be reserved for the BE class. One source of complexity is that the IntServ router uses to allocate bandwidth for individual flows, while the adaptive models allocate resources for classes. This problem can be overcome if a router groups all flows that belong to the same class and shares resources between the classes. As mentioned earlier, the fairness between data flows within a service class can be achieved by the per-flow buffer mechanism or by the per-flow traffic regulation mechanisms used in the network. Another problem is that the adaptive model assumes that all flows belonging to the same class have the same QoS requirements, while in the IntServ framework applications belonging to same class may request different amount of bandwidth resources. This problem can be solved if we modify all the expressions and substitute $\sum B_i^k$ to $N_i B_i^f$, where B_i^k is the bandwidth requirements of the kth flow within the ith class. The same considerations can be applied to the burst size.

Ideally, the adaptive model should be placed at every RSVP router that has the bottleneck links. Since it is not feasible, at the first stage, the adaptive model should be deployed to the routers that aggregate a significant number of flows, i.e. either to the core routers or to the routers that connect access networks to the high-speed cores. Such an approach can ensure that the adaptive resource allocation will affect the larger number of flows and that it will not be diminished by some router with the static configuration.

According to the IntServ framework, the RSVP protocol informs routers along the data path when a new flow arrives or an existent one stops functioning. Thus, each RSVP-capable router always possesses information about the number of active data flows. Besides, each PATH message carries the traffic specification in the from of the TSpec object, which consists of the token bucket rate, the burst size, the minimum policed unit, and the maximum datagram size. All these parameters have the direct correspondence to the input parameters of the adaptive models. Besides, L_i^{min} is the minimum datagram size over all values received in TSpec, and L_i^{max} is the maximum value over all maximum datagram sizes. The TSpec also carries the peak rate, which can be used to create the upper constraints.

The RSVP protocol does not carry the pricing information. However, a network provider may have the static price associated with each service class, which eliminates the need to include this data into the RSVP messages. On the other hand, due to the extendable nature of the RSVP protocol, a new object may be introduced easily.

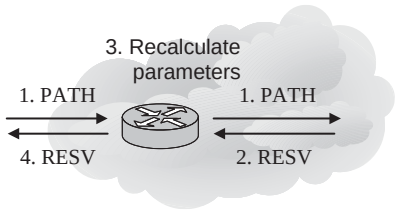


FIGURE 58 Adaptive model in the IntServ framework.

Fig. 58 illustrates a typical usage of the adaptive model in the IntServ framework. As the PATH message travels from the sender to the receiver, all the RSVP routers process this message according to the standard rules. At this stage, no interaction with the adaptive model is necessary. When the RESV message arrives at the RSVP router with the adaptive model, new QoS requirements for each service class are constructed and passed to the adaptive model, which calculates new optimal parameters for the scheduler. If the router has enough resources, then a new scheduler configuration takes effect and the RESV message is sent along the data path to the sender. Otherwise, the RESVERROR message is sent to the receiver, and the parameters of the scheduler remain intact. When a flow stops functioning, the PATHTEAR message is sent that causes the removal of the state information, recalculation and the update of the scheduler parameters. The same happens when either the RESVTEAR message arrives or the reservation state expires. The RESVCONFIRM message, which a sender may send to confirm the reservation, will be just ignored.

One of the disadvantages of the IntServ QoS framework is that there is no node that possesses the network-wide picture concerning the reservation states, which might be necessary to perform some network-wide optimizations or to track the available resources. The solution for this problem is to *outsource* RSVP messages. In such a framework, the RSVP router sends the received message to a known network management station (NMS).[1] For these purposes, the RSVP message is encapsulated according to the rules specified in [73] and transmitted over the COPS protocol [46]. This framework can be extended in such a way that the NMS also makes decisions on behalf of the RSVP routers and performs the admission control functions. In such a case, the routers will support only the basic processing of the RSVP messages. It eliminates the need to install the complicated software at every node. The outsourcing does not dictate the presence of one NMS for all the RSVP routers. Instead, each administrative domain may have its NMS.

Since the RSVP router has to make the policy control decisions concerning the PATH message and the admission control decisions concerning the RESV messages, it can outsource either all the message types, or only the RESV messages. It is often the case that the policy control module is simpler than the admission control algorithms, thus, the former can be implemented locally. According to [73], the RSVP router can also outsource the PATHERROR and RESVERROR messages. It should be noted that outsourcing RSVP messages does not change the interaction between the adaptive model and the RSVP protocol. As the router receives a positive answer from the NMS, it triggers the appropriate RSVP components, which in turn trigger the adaptive model. This scheme is illustrated in Fig. 59.

The price for such an architecture is the increased network response for the QoS requests since it takes time before a request is sent to the network manager and a decision is received. As mentioned above, one way to decrease this impact

[1] In the terminology of the policy-based management [159] it is referred to as the Policy Decision Point.

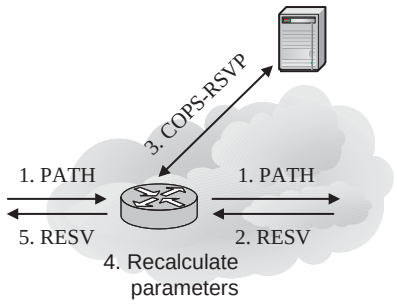


FIGURE 59 IntServ framework with the COPS protocol.

is to outsource only the RESV messages.

4.2 Differentiated Services

The Differentiated Services (DiffServ) [16] framework has a different approach comparing to the IntServ. Instead of maintaining the per-flow information in each router, flows with similar characteristics are grouped into a class, which is referred to as an *aggregate*. As a packet enters the DiffServ domain, the appropriate DiffServ codepoint (DSCP) is written into the IP header, which determines an aggregate a packet belongs to. Now, routers classify and schedule packets based on their DSCP values, not on their source / destination IP addresses, protocol, and the port numbers.

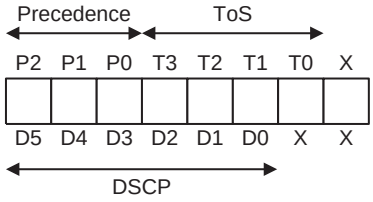


FIGURE 60 Structure of DSCP.

The DSCP field occupies six bits and overlap with the ToS octet [2] of the IPv4 packets. Fig. 60 illustrates the structure of the IPv4 ToS octet and its interpretation for the DiffServ [108]. The remaining two bits are reserved for the ECN technology [123]. It bears mention that the DSCP value is backward compatible with the IPv4 precedence field. As follows from the figure, three higher bits of DSCP overlap with the IP precedence bits. It must be noted that the higher bits of DSCP determine a class, while the three lower bits determine a subclass, if any. Thus, packets marked according to the DSCP value will be given the sufficient level of QoS if a provider has the old switching equipment that relies upon the IP precedence value. Consequently, packets marked according to the IP precedence field will be forwarded appropriately within the DiffServ domain. The structure of the IPv6 header is quite different [41]. It has a dedicated field called Traffic

Class that can be used to carry the DSCP value. There are cases when applications should exchange the DSCP value(s) before data communication can start. Examples include communication between bandwidth brokers and the MPLS support. In this case, the actual encoding of the six-bit DSCP value follows the rules specified in [26].

Since the DiffServ routers classify and schedule packets based on their DSCP values, resources are allocated not for a distinctive flow, like in the IntServ framework, but for the aggregates. It may lead to the fairness problem when one flow in an aggregate starts to consume more resources thus reducing the throughput of other flows. To cope with this problem, the DiffServ framework uses the per-flow metering and policing. So as not to overburden all the DiffServ routers with this task, it is performed only when a flow enters the DiffServ domain. This approach results in two types of the DiffServ routers: edge and core. While the edge routers perform sophisticated classification, metering, and marking functions, the core routers perform a simple packet forwarding. Such a framework simplifies the configuration of schedulers, reduces significantly the need for resources, and increases the overall scalability of the framework. It is worth noting that the performance requirements mandate the use of dedicated hardware routers in the core networks because the software-based routers may face some serious problems while handling large amount of data [15]. Instead, the QoS support in the edge routers is often software based because of the need for flexibility and upgradability [66].

4.2.1 Per-hop behaviour

To provide different packet treatments, the DiffServ framework introduces several types of traffic aggregates. Each type specifies the available QoS guarantees and their concrete values. To ensure the QoS requirements, each DiffServ router must provide a certain level of packet forwarding. Thus, if a packet belongs to a certain aggregate, one can say about the treatment the packet will experience while travelling through the router. In the terminology of the DiffServ framework, it is referred to as the per-hop behaviour (PHB). In other words, PHB may be thought of as the externally observable behaviour of packets [64]. Though this definition may sound odd, it makes sense from the implementation point of view. PHB specifies the desired behaviour, while a provider is free to choose mechanisms to achieve it.

TABLE 3 Summary of the DiffServ aggregates.

Aggregate	QoS requirements			Policy action
	Bandwidth	Delay / Jitter	Loss	
EF / DB	√	√	√	drop
AF	√		√	mark
BE				

The DiffServ framework defines three major types of the per-hop behaviours:

Expedited Forwarding (EF), Assured Forwarding (AF), and Best Effort (BE). Table 3 summarizes briefly the differences between them. The major differences are the QoS guarantees and the policy actions applied to packets. It must be noted that AF is a traffic class that defines four traffic aggregates: AF1, AF2 and so on. They have different values for the QoS parameters, but the general treatment remains the same.

Expedited Forwarding/Delay Bound

The EF PHB [78] is intended to provide an end-to-end service with low loss, low latency, low jitter, and assured bandwidth through the DiffServ domain. It is very similar to the GS service of the IntServ framework. Such a service appears to endpoints like a point-to-point connection or a "virtual leased line". Since loss, latency, and jitter are mainly caused by queuing, which packets encounter while transiting the network, the main purpose of the EF PHB is to offer packet transport, in which appropriately marked packets usually enter short or empty queues. Since buffering occurs when the short-term traffic arrival rate exceeds the departure rate of a router, the minimum departure rate of an aggregate may be set to be larger than its maximum arrival rate at every transit router. This can guarantee that packets encounter no or very small queues. Another way to implement EF PHB is to associate it with the highest priority of the PQ scheduler at each router.

The original definition of the EF PHB stated that packets must be forwarded as fast as possible through the domain. Though this definition corresponds to the nature of the "virtual leased line" services, it results in an inefficient resource allocation, especially when a service can tolerate some delays. Thus, the further standardization process has refined requirements for this behaviour aggregate [40]. The main difference is that a new specification adds mathematical formalism to give more rigorous definitions to the QoS parameters, such as delay and jitter. Practical issues for implementing the new interpretation of EF PHB have been considered in [32]. There is also another interpretation of the EF PHB that is referred to as the Delay Bound (DB) PHB [6]. The goal of the DB PHB is a strict bound on the delay variation of packets through a hop. Though the EF and DB PHBs have similar purposes, they have different implementation approaches. The EF PHB achieves low delay by allocating bandwidth that is equal or bigger than the input rate. In the case of the DB PHB, the scheduler’s output rate does not need to be specified since it will be whatever is needed to achieve the target delay variation.

As in the case of the IntServ GS class, a provider should not assume that a client application always transmits data at the required rate. If a source application for any reason increases its transmission rate, it can affect the behaviour of all EF/DB streams. Such situations are prevented by applying a policer to flows at the DiffServ edge nodes. The policing action for an EF/DB flow is to drop excessive packets to bring this flow in conformance with its profile.

The EF/DB PHB is intended for the applications that uses the UDP protocol

to send data because the absence of the acknowledgements allows to send real-time audio and video data at the desired rate even if some packets are dropped. Since the UDP protocol does not react to the packet drops, there is no need to use the sophisticated active queue management (AQM) algorithms. Simple FIFO queues are the best choice in this case.

Assured Forwarding

The AF PHB [69] is intended to offer different levels of forwarding assurance for IP packets that need better service than is available for BE packets. It is similar to the CL service since it aims to provide the bandwidth requirements. There are no quantifiable timing requirements, such as delay or jitter, associated with the forwarding of AF packets. In turn, four AF classes have been defined, which are referred to as AF1, AF2, AF3, and AF4. For each AF class a certain amount of forwarding resources, which can be expressed with bandwidth and buffer space, is allocated. Within each AF class IP packets are marked with one of three possible drop precedence values. To differentiate them, an appropriate number is added to the AF aggregate. For instance, aggregate AF2 has the following drop precedence levels: AF21, AF22, and AF23. When a congestion occurs, the drop precedence value determines the relative importance of the packet within the AF class. Implementation of the AF PHB responds to the long-term congestion by dropping packets, while the short-term congestion is handled by queuing them.

Unlike the EF PHB, source applications may exceed the subscribed profile with the understanding that the excess traffic is not delivered with as high probability as in-profile traffic. The policing action for the out-profile AF packets is to mark them to a higher drop probability. When a DiffServ router gets congested, it avoids loss of packets with a lower drop precedence by preferably discarding packets with a higher drop precedence value. It can guarantee low loss for in-profile packets while improving the network utilization. Since the AF PHB requests only the minimum bandwidth allocation, the over-subscribed AF traffic can be transmitted when there is enough bandwidth available either from other AF classes or from other PHB groups.

The AF PHB suits the TCP applications that can increase transmission rate and react to the packet losses by reducing the window size. Since the TCP applications react to the packet losses by reducing the window size, the AF PHB uses the AQM algorithms to avoid the drop-tail behaviour of the queues and, as a result, the effect of the global synchronization. Most of these algorithms are based on RED technique that is extended to support several drop precedence levels. They are often referred to as Multilevel RED (MRED). A good overview of the existent MRED algorithms and their comparison is given in [101]. We mention the names of the most important ones: Weighted RED (WRED) and RED with in / out (RIO) [34], which is also referred to as RIO-C. The reason is that there is a modification, called RIO with decoupled queues (RIO-DC) [141], in which the average queue for packets of each colour is calculated using the number of packets of that colour in the queue.

Best Effort

As in the case of the IntServ framework, the BE PHB is used to transmit non-critical user data, which has no particular QoS requirements at all. A provider can allocate some fixed bandwidth for this aggregate that is shared between all active data flows. The absence of the QoS requirements eliminates the need to perform per-flow metering. Technically speaking, a so-called NULL meter [12] is used that does not analyse traffic characteristics. Flow rates within the BE aggregate are controlled implicitly by the available bandwidth and by the buffer space provided at each DiffServ router.

Along with the BE aggregate, the IETF has tried to standardize another PHB called Lower Effort (LE) [18]. There are a lot of applications without the QoS requirements that are mapped automatically to the BE aggregate. However, some of them are more important than others. For instance, the mail data is of more importance than some broadcast traffic. Thus, a certain differentiation between the BE applications were necessary. It resulted in the idea of the lower-effort service that may transmit data only if there are unused resources from other aggregates. Based on this formulation, a possible implementation of the LE PHB is to allow to transmit LE traffic if and only if all other sessions including the BE PHB are idle. Fig. 61 depicts the structure of the scheduler that implements such a behaviour. PQ will output LE data only if the WRR scheduler is idle, i.e. if the EF, AF, and BE queues are empty.

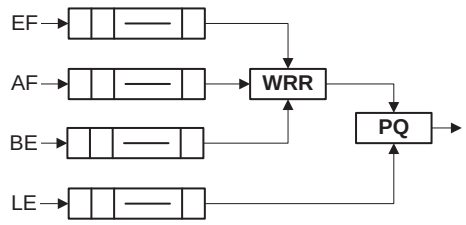


FIGURE 61 Implementation of the LE PHB.

Unfortunately, no strict differences between the BE and LE aggregates in terms of the quantifiable parameters were given. Thus, at the moment, the LE PHB is an optional solution that a provider can use while classifying the incoming BE traffic. As the LE traffic leaves a domain, it must be re-marked back into BE to achieve compatibility with the downstream domains that do not support the LE PHB.

4.2.2 Meters

To classify the incoming traffic into the in- and out-profile packets, the edge routers must implement meters that are responsible for analysing traffic characteristics. Depending on a traffic aggregate and its requirements, various meters are used. Table 4 presents several major meters used in the DiffServ framework.

The Token Bucket meter is based on the Token Bucket model of the traffic

TABLE 4 Meters used in the DiffServ framework.

Meter	Parameters	Aggregate
Token Bucket	CIR CBS	EF
TSWTCM	CIR PIR	AF
srTCM	CIR CBS PBS	AF
trTCM	CIR CBS PIR PBS	AF
NULL	–	BE

profile. It takes two parameters into account: Committed Information Rate (CIR) and Committed Burst Size (CBS). This meter is suitable for the EF aggregate since the latter does not need to determine the drop precedence of a packet. It is enough to decide whether a packet is within or out of a profile. Furthermore, by using the burst size of the Token Bucket meter a provider can estimate the worst-case queuing delay and allocate the necessary amount of buffer resources.

A provider can meter the incoming AF traffic with several different algorithms. The time sliding window three-colour meter (TSWTCM) [47] determines the drop precedence of the AF packets based on two configuration parameters: Committed Information Rate and Peak Information Rate (PIR). If the flow rate is less than CIR, a packet is marked with green colour. If it is higher than CIR and less than PIR, then a packet is marked with yellow colour. Otherwise, it is marked as red.[2] The TSWTCM meter is simpler in configuration than the TockenBucket since the former does not require the burst size. It may be a challenging task to estimate the anticipated burst size for an unknown application. On the other hand, TSWTCM fails to control the burstiness of the input traffic and it may mark some packets wrongly due to the time-sliding window that calculates the mean rate. The single-rate three-colour meter (srTCM) [70] is very similar to the Token Bucket policer. The difference is that it uses two burst sizes - Committed Burst Size and Peak Burst Size (PBS) - to decide whether to mark a packet with yellow or red colour. The two-rate three-colour meter (trTCM) [71] is even more complicated as it takes also Peak Information Rate (PIR) into account.

As mentioned above, the NULL meter does not analyse any traffic characteristics at all. Such a behaviour is necessary for the BE aggregate. In some cases, a provider can apply the NULL meter to the other aggregates if he trusts the traffic pattern sent from the upstream domain or from a customer.

4.2.3 Per-domain behaviour

Along with PHB, the IETF has proposed the concept of the per-domain behaviour (PDB) [109]. While PHB specifies how packets should be forwarded through the DiffServ router, the PDB describes the behaviour experienced by a particular set

[2] The three-colour meter can be turned into the two-colour one easily by setting CIR and PIR to the same value. It may be necessary if a provider wants to classify the incoming AF packets only into green and red ones. Sometimes two- and three-colour meters are also referred to as TSW2CM and TSW3CM respectively.

of packets as they cross the DiffServ domain. In other words, the DiffServ domain can be treated as a node that is capable of providing certain QoS guarantees. Such an approach simplifies the provision of the end-to-end services. Networks of the DiffServ domains can be connected to provide the end-to-end QoS guarantees by building on the PDB characteristics without regard to the particular PHBs used. For instance, each PDB can correspond to some PHB. At the same time, one PHB can implement several PDBs. This level of abstraction makes it easier to compose cross-domain services as well as making it possible to hide details of a network's internals while exposing information sufficient to enable QoS.

Along with the BE PDB, the IETF has standardized only the Lower Effort (LE)[3] PDB [17]. It is used to transmit uncritical data, such as netnews, bulk mail, content distribution, peer-to-peer file sharing, world-wide search engines etc. Some work has also been done on the Virtual Wire (VW) [79] and Assured Rate (AR) [140] PDBs.

4.2.4 Adaptive model

The proposed adaptive models can be used in the DiffServ framework as they are capable of providing bandwidth and delay guarantees. Each PHB would correspond to a certain service class in the adaptive model. It fully conforms to the DiffServ architecture in which each PHB is associated with a separate queue. Thus, the adaptive model will calculate optimal weight values for each PHB.

Taking into account types of traffic aggregates, defined in the DiffServ framework, it is possible to arrive at the conclusion that it is better to allocate free bandwidth to the AF classes. There is no need to allocate additional resources for the EF aggregate, since it is represented predominantly with UDP sources that transfer data at a constant rate. Even if there are TCP flows, then all the excess data will be dropped in the edge routers due to the EF policing action. Since the BE aggregate has no QoS requirements at all, the free resources can be provided for the AF aggregates for which assured QoS guarantees are necessary and for which end-users are willing to pay. The TCP flows, belonging to the AF aggregates, will increase their window sizes thus consuming the allocated bandwidth. This is not contrary to the concept of the DiffServ framework because the AF aggregates are capable of transferring excess amount of data if a network has enough resources.

It is suggested to implement all major adaptive issues at the edge of a DiffServ domain. It is often the case that the edge routers connect client networks to the high speed core network. In this case, the edge routers have more impact on the resulting bandwidth allocation between the traffic aggregates. Besides, unlike the core routers, the edge routers have the per-flow information that enables them to perform classification and policing. Thus, it is much more convenient to perform the adaptive resource allocation at these routers. By tracking the number of active data flows and their QoS parameters, the adaptive edge routers can allocate optimally the output bandwidth. At the same time, the core routers will remain intact and will not be overburdened with additional adaptive software

[3] The previous name for this PDB was bulk handling (BH).

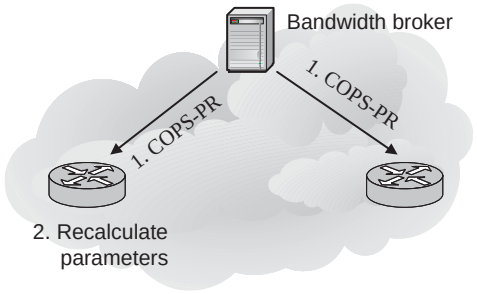


FIGURE 62 Adaptive model in the DiffServ framework.

that can slow the packet forwarding process. Such an approach fits into the Diff-Serv concept, which states that the edge routers perform sophisticated functions, while the core routers perform only the simple forwarding. It also bears mention that though the edge routers perform the policing and charging of the incoming traffic on the per-flow basis, the output bandwidth is allocated between the traffic aggregates, i.e. on the per-class basis.

The DiffServ framework does not have a dedicated protocol to exchange information about the number of active flows and their QoS parameters. Instead, the concept of the SLA was introduced. Its technical-level part, which is referred to as the SLS, specifies the amount of resources to be allotted to each PHB. Meanwhile, the COPS [46] protocol and its extension for the provisioning (COPS-PR) [31] can carry the actual configuration information. SLA can by dynamic, i.e. it can state that depending on the time the different amount of resources should be allocated. So, as the configuration changes and is transmitted to the DiffServ routers, the routers can re-run the adaptive model that updates the parameters of the scheduler.

Fig. 62 illustrates the adaptive model in the DiffServ environment. If the centralized management entity, for instance bandwidth broker (BB) [110], wants to change the configuration of the DiffServ routers, it sends a new configuration over the COPS-PR protocol. The latter, in turn, can trigger the adaptive model that will calculate new weight values and update the configuration of the scheduler. The configuration information is represented in the form of the Policy Information Base (PIB) that has been standardized for the DiffServ framework in [30]. This PIB specifies actions that must be applied to the incoming packets based on the model of the DiffServ router specified in [12].

Of course, the framework outlined above is not as efficient as IntServ from the viewpoint of the presence of the RSVP protocol, which can carry all the necessary signalling information in the horizontal plane. In other words, the DiffServ framework is irrespective of the number of the active flows and relies upon the configuration provided by SLA and carried by the COPS-PR protocol.

4.3 Hybrid architecture

The DiffServ approach to service management is more approximate in the nature of its outcome. There is no requirement for the network to inform the application that the request cannot be admitted, and it is left to the application to determine if the service has not been delivered. What appears to be required within the DiffServ framework is the resource availability signalling. It is implemented in the IntServ framework. However, there are several areas of concern about the deployment of the IntServ architecture. The resource requirements for running the per-flow resource reservations on routers increase in direct proportion to the number of separate reservations. Besides, router forwarding performance may be impacted by the packet classification and scheduling mechanisms that work on the per-flow basis. IntServ also poses some challenges to the scheduling mechanisms, where there is the requirement to allocate absolute levels of the output bandwidth to individual flows.

One approach is to attempt to combine both architectures into an end-to-end model, using the IntServ as the architecture that allows applications to interact with the network, and DiffServ as the architecture to forward data [10]. This approach combines the per-application view of the IntServ architecture and the network boundary-centric view of the DiffServ architecture. If the RSVP request is accepted, it would imply that there is a committed resource reservation within the IntServ-capable components of the network, and that the service requirements have been mapped into a compatible aggregate within the DiffServ-capable network. The DiffServ core must be capable of carrying the RSVP messages across the DiffServ network, so that further resource reservation is possible within the IntServ network upon egress from the DiffServ environment.

Based on the interaction between the IntServ and DiffServ parts, it is possible to present three major scenarios [13, 1]:

- static allocation in the DiffServ domain
- dynamic allocation by RSVP in the DiffServ domain
- dynamic allocation by other means

The first case assumes that the DiffServ domain has the static configuration. From the viewpoint of the IntServ routers, it can be treated as a link with a certain bandwidth. As a result, the IntServ router at the ingress point performs the admission control function for the whole DiffServ domain. The functioning of the adaptive model in this scenario is very similar to the one considered in section 4.1.2. It will be running in the IntServ router and react to the RSVP messages. The second allocation scheme assumes that some or all DiffServ routers participate in the RSVP signalling. From the viewpoint of RSVP, they act as the IntServ routers, but they use the DiffServ mechanisms in the forwarding plane. Once again, the functioning of the adaptive model will be the same as for the IntServ framework.

Fig. 63 illustrates the third allocation scheme, in which the RSVP protocol informs the DiffServ domain about the required resources, but the COPS-PR protocol carries a new configuration. Such an approach has several important advantages. All routers, which participate in the signalling, outsource messages to

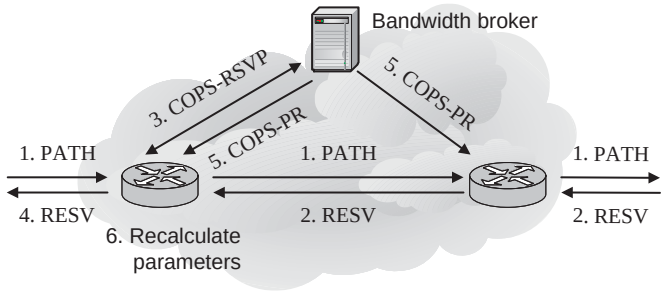


FIGURE 63 Adaptive model in the hybrid framework.

BB that makes decisions. It eliminates the need to install the admission control modules at every router. Furthermore, the overall amount of required memory and the processing load is less when compared to the case when all the routers participate in the RSVP signalling [42]. What is also important is that in such a framework the BB possesses the domain-wide picture which enables a provider to perform various optimizations and load balancing.

Referring back to Fig. 63, the DiffServ routers do not process the PATH message. However, as the RESV message leaves the DiffServ domain, the border router sends it over the COPS-RSVP protocol [73] to BB that is eligible for the admission control. If there are no resources, BB answers negatively, and the RESVERROR message is sent to the receiver. If the DiffServ domain has enough resources, then the BB answers positively and the RESV message is sent to the sender. After that the BB can update the configuration of all the necessary DiffServ routers. When the DiffServ router receives a new configuration, it can re-run the adaptive model. It is worth noting that in this case the COPS protocol, not the RSVP protocol, triggers the adaptive model. So, the functioning of the adaptive model is the same as considered for the pure DiffServ framework.

Since the BB has to mimic the reservation states of each RSVP router that outsources the RSVP messages, the number of states can be really huge. To reduce this overhead, a new COPS client type has been proposed in [102]. It is also worth mentioning that the amount of the RSVP signalling information can be reduced significantly in the core networks by aggregating the request and reservation messages, as described in [7]. In this case, it will depend on the number of aggregates rather than on the number of active data flows. Such an aggregate can be mapped to one of the DiffServ PHBs by including the DCLASS object [11] into the RSVP messages. However, it is a challenging task to map the RSVP reservations into the existent set of PHBs. While the GS class can be mapped to the EF PHB, the CL class is a more complicated task due to the availability of four AF PHBs. Since a provider is not obliged to implement all the AF PHBs and may support only a subset of them, it may be a complicated task to map automatically the CL flow to one of the available AF aggregates. The IETF has made an attempt to standardize it [163]. In [122], the dynamic approach for mapping the IntServ CL flows into the AF PHB has been presented.

Apart from RSVP, other protocols developed by the IETF can be used to

signal the network about the active data flows. As considered in [129], the framework described in Fig. 63 can be modified easily so that the user applications use the Session Initiation Protocol (SIP) [128]. As in the case of RSVP, the COPS protocol can carry the encapsulated SIP messages to the centralized management entity [63].

4.4 Next steps in signalling

Having analysed the IntServ and DiffServ approaches, it is possible to arrive at the conclusion that it is almost impossible to build a proper QoS framework without any signalling mechanisms [125]. Indeed, disappointed by the low scalability of IntServ and the signalling overhead, the designers of the DiffServ framework have not introduced any signalling mechanisms at all. Later, many authors have pointed to this weakness of DiffServ in their scientific works. Though the COPS protocol is capable of carrying the configuration information, it is used merely for the vertical management, while the RSVP protocol aims to perform the management functions in the horizontal plane. Having realized the low efficiency of the static configuration, the designers of the IntServ and DiffServ frameworks have found a compromise in combining the RSVP signalling with the DiffServ forwarding mechanisms. However, such an architecture was dictated predominantly by the available software in the routers, supplied by the major manufactures, not by the fact that it provides an efficient functioning.

The main source of problems is the RSVP protocol. Having been developed for the IntServ framework, it has several serious disadvantages that are described in detail in [103]. One of them is the incapability of the RSVP protocol to deliver safely the signalling data. The RSVP messages are sent over the IP datagrams and, as a result, can be dropped by the intermediate routers that do not prioritize the incoming data. Another limitation of the RSVP protocol is that a receiver initiates a reservation when the PATH message arrives from a sender. It makes the deployment of the RSVP protocol more complicated since two endpoints must support it. Furthermore, the RSVP protocol is incapable of carrying the signalling information between the routers, which may be necessary for the DiffServ framework.

To overcome these limitations, the IETF has established the Next Steps In Signalling (NSIS) working group, whose aims are to develop a new general signalling protocol for the Internet. In fact, these tasks are reflected in the name of this protocol - General Internet Signalling Transport (GIST) [139]. Unlike RSVP, GIST does not depend on a concrete transport environment and can carry data over various protocols, such as TCP and SCTP [111]. It is noticeable, that the reliable transport protocol are used which makes GIST more robust under network congestions. The GIST protocol is still under development including its application for the DiffServ and IntServ QoS frameworks.

4.5 Policy-based management

Ensuring that all routers function properly is not a trivial task. As the network grows and the number of nodes increases, the need for sophisticated technologies that can assist in configuration becomes more evident. Traditional approaches to network management focus on individual devices and often rely upon proven technologies, such as the SNMP protocol [28]. However, it can be quite a time consuming operation if the number of nodes to be managed is great. The introduction of a policy-based management can significantly ease the configuration of multiple routers in the network through the use of policy expressions [155]. An administrator has to provide all of the necessary data only once, enabling any number of routers to retrieve configuration data. The central idea behind the policy-based management is to provide a set of rules to manage and control the changing and / or maintaining of the state of one or more managed objects.

According to [166], the policy-based management system has the following components: a policy enforcement point (PEP), a policy decision point (PDP), a policy repository, and a policy management tool. The task of the PDP is to retrieve and interpret the policy information, and pass it to the PEP. The policy repository is the place where all policies are stored and from which they are taken by PDPs. The policy repository is populated with configuration data by an administrator, who uses the policy management tool to specify the desired network behaviour. Referring back to Fig. 63, the DiffServ routers may perform the functions of the

PEP, while the BB may perform the functions of the PDP. To achieve the interoperability across various devices from different vendors, the PEP and PDP exchange the policy information in the form of PIB, which is very similar to Management Information Base used in the SNMP protocol. The policy repository stores data by using the Common Information Model (CIM) and its extensions for the policy data [107, 106] and QoS [144].

However, the current information model for the QoS routers lacks the concept of the adaptive router. In other words, it is impossible to specify settings for the adaptive models by using concepts introduced in CIM. In [134], we have proposed a solution to support the adaptive mechanisms. Since the policy repository stores the rules in the form *if <set of conditions> then do <set of actions>*, where conditions and actions are represented with instances of classes logically interconnected by a policy rule, it has been proposed to extend the set of policy actions so that the adaptive models can function properly.

4.6 Summary

In this section, two major QoS frameworks - IntServ and DiffServ - have been presented and parameters of their classes have been analysed. Since the adaptive models are capable of ensuring the bandwidth and delay guarantees, they can be used to implement the IntServ GS and CL classes and the DiffServ EF and AF aggregates. Furthermore, the adaptive models are flexible in that they can interact with the signalling protocols, such as RSVP and COPS, that communicate the QoS requirements to the routers. As mentioned while considering the IntServ QoS framework, it may be necessary to change the resource allocation method of the IntServ router from the flow-based to the class-based. It is not contrary to the concept of IntServ. Furthermore, recent standardization documents, such as the aggregated RSVP, propose implicitly the class-based resource allocation for the IntServ networks. On the other hand, as considered in section 2.8, the adaptive models can be changed easily to calculate parameters for the per-flow scheduler.

Though the adaptive models increase the total revenue by allocating more resource to some data flows, it is not contrary to the IntServ and DiffServ specifications that allow to transmit excess data if a network has resources. However, unlike the AF aggregate, the specification of the CL class is vague concerning the transmission of the excess traffic. Thus, the DiffServ framework is more suitable for increasing the total revenue. On the other hand, the IntServ framework relies upon the RSVP protocol that informs routers about the required resources. Thus, the ideal framework for the adaptive models is the network that uses the DiffServ forwarding mechanisms in conjunction with the signalling protocols.

5 SIMULATION SCENARIOS WITH THE ADAPTIVE SCHEDULING MODELS

The simulation is carried out by the NS-2 simulator [153]. It is a part of the Virtual InterNetwork Testbed project, which aimed to provide improved simulation tools to use in the design and deployment of new protocols and solutions. Reasons for choosing the NS-2 simulator are [25]: a) a rich infrastructure for developing new protocols, b) the opportunity to study large-scale protocol interactions in a controlled environment, and c) easier comparison of results across research efforts. The NS-2 simulator allows two levels of programming. Simple scripts, topology layout, and parameters variation can be done in OTcl [160], which is the object-oriented extension of the Tcl scripting language [112]. At the same time, core components and protocols are implemented in C++ to achieve high efficiency and to reduce the simulation time.

The implementation of the adaptive models followed the framework described above. Fig. 64 illustrates the interconnection between the NS-2 simulator, simulation scripts, and the adaptive models. The adaptive models are implemented in C++ to achieve high efficiency and fast execution. They have a clear Application Programming Interface (see appendix **??**) that hides the implementation details and exposes a set of classes. As a result, the models can be used

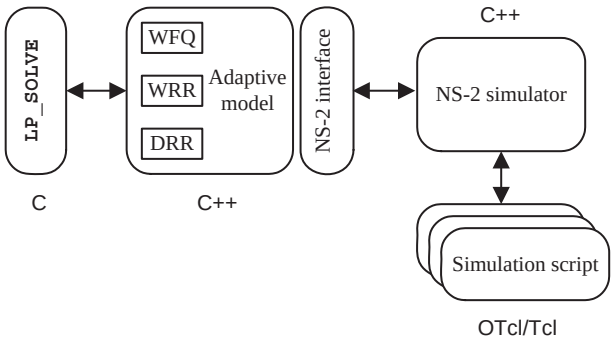


FIGURE 64 Interconnection between the adaptive model and NS-2.

in any application, not only in the NS-2 simulator. To access the adaptive models from the simulation scripts written in OTcl, appropriate interface classes have been introduced (see appendix **??**) that forward function calls from within the NS-2 simulator to the adaptive models and back to the NS-2 components. The implementation of the adaptive models relies upon the `LP_SOLVE` library [95] that is capable of solving LP and ILP tasks. Such a modular framework has several advantages:

- it is possible to introduce subsequent improvements into the adaptive models without touching the NS-2 simulator and the simulation scripts
- it is possible to use different LP and ILP solvers to study the overall performance of the adaptive models
- it is possible to create a set of simulation scripts to analyse the behaviour of the same model within various environments and sets of parameters

Apart from the adaptive models, additional modules have been written for the NS-2 simulator. Since the official release lacks WFQ, WRR and DRR, these scheduling disciplines have been implemented (see Appendix **??** for more details). Besides, while implementing WRR and DRR, the considerations presented in section 2.5 were taken into account.

5.1 Simulation parameters

To study the adaptive resource allocation, simulation scenarios use several service classes, which hence will be referred to as *Gold*, *Silver*, and *Bronze*. Since each QoS architecture introduces its own terminology, we decided to use the olympic names because they allow us to refer uniquely to a particular service class regardless of the considered QoS framework. The details of each service class are presented in Table 5. The Gold class corresponds to the real-time audio services. It is the most demanding class that has bandwidth and delay guarantees. This class is simulated by the G.711 audio codec [77] that transmits data over the Real Time Protocol (RTP) [138]. The G.711 codec outputs data at the constant rate of 64,000 bps with the frame size of 240 bytes. In the case of the IP networks, frames with audio data are encapsulated by the IP/UDP/RTP protocols that augment the basic frame size with their headers. The overhead is 60 bytes for IPv6 and 40 bytes for IPv4.[1] Since NS-2 simulates IPv6 by default, the resulting packet size for the G.711 codec is 300 bytes. In turn, the resulting transmission rate is 80,000 bps. Practically, it is also necessary to take the L2 header size into account, however NS-2 does not simulate the L2 transmission.

[1] The header size can be reduced significantly for the low-speed serial links by using the IP/UDP/RTP header compression as proposed in [29].

The Silver and Bronze classes represent general purpose services that have only bandwidth requirements. These classes are simulated by FTP-like applications that generate bulk data transferred over the TCP protocol. The reason for choosing such a type of application is that it always tries to send data thus behaving very aggressively. If the adaptive models are capable of ensuring QoS guarantees in an aggressive environment, then they will work in other environments as well. The bandwidth in Table 5 specifies the minimum rate that a flow must obtain within its service class.

Service classes are assigned different packet sizes to check that bandwidth is shared accurately regardless of the packet size value. It is especially important for the adaptive model for the WRR scheduler since the latter does not take the packet size into account and it is the responsibility of the model to calculate the correct weight values. While choosing packet sizes, the analysis of the Internet traffic patterns presented in [152] was taken into account.

At the beginning of a simulation run, all data streams from client applications are injected gradually into the network environment. To produce a random number of active flows, the simulation uses the ON / OFF model, parameters of which are given in Table 5. The ON-time represents a uniformly distributed random number, taken from an appropriate interval, and the OFF-time follows the exponential distribution with an appropriate mean value.

TABLE 5 Parameters of service classes.

Class	Price for 1Mb	Max flows	Flow parameters			ON / OFF time (s)
			Band. (Kbps)	Delay (ms)	P.size (bytes)	
Gold	2	10	78	20	300	20–60 / 10
Silver	1	15	50	–	840	30–70 / 5
Bronze	0.5	25	10	–	640	10–40 / 10

During the simulation runs, parameter γ_i is set to 1 for the Silver and Bronze classes, thus enabling allocation of an additional bandwidth only for them. As considered earlier, there is no need to allocate additional resources for the Gold class because it consists of the constant rate flows that will not increase their transmission rates if additional bandwidth is available.

To analyse the behaviour of the adaptive models in various environments, the following simulation cases will be considered:

- No QoS framework. No particular QoS framework and no signalling will be used in this simulation case. It allows to compare the basic characteristics of the adaptive models without the influence of the QoS mechanisms.

- IntServ framework. The RSVP protocol will be used to inform a router with the adaptive models about the number of active data flows and their QoS requirements.

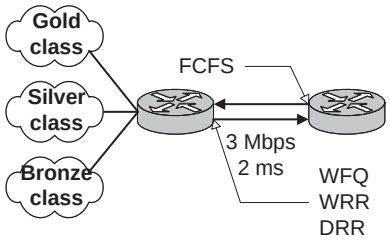


FIGURE 65 Simulation environment.

- DiffServ framework. The DiffServ classification and forwarding mechanisms will be used to transmit data. At the same time, additional signalling protocols will inform routers about the required resources.

To simplify further explanations, we will prefix the name of a scheduling discipline with RA^2 meaning that this is the adaptive model running on the top of the appropriate scheduler. For instance, RA-DRR stands for the adaptive model for the DRR scheduler. If either WRR or DRR works in the LLQ mode, then we will refer to it as WRR+ and DRR+ respectively. Thus, the adaptive model that works on the top of the WRR scheduler in the LLQ mode will be denoted as RA-WRR+.

5.2 No QoS framework

In this simulation case, we study the proposed adaptive models and compare them to the queuing disciplines with a static configuration in terms of parameters, such as the provided bandwidth and the obtained revenue. At first, we analyse the adaptive resource allocation when only bandwidth guarantees are necessary as if the Gold class has no delay requirements at all. For these purposes, the adaptive models (17), (45) and (57) are used.

As mentioned above, this simulation case does not consider any signalling protocol that the client nodes, the router, and the destination node can use to exchange information about the required resources. Instead, the inner possibilities of the simulation environment are used to keep track of the number of active flows at the routing node. On the one hand, it does not correspond to a real-life scenario. On the other hand, the amount of additional signalling information, which the nodes would exchange in the presence of such a protocol, is not great. Furthermore, the absence of the signalling protocol allows the generation of the same pattern of behaviour of flows regardless of the used scheduling discipline, which ensures the accurate comparison of the results.

Fig. 65 illustrates an environment used in this simulation case. It consists of a bottleneck router, a destination node, and a set of client nodes with applications. Depending on a simulation run, the bottleneck router implements either

[2] In our previous articles, we have been using the A prefix. However, since other authors use the same prefix quite often, we have changed it to RA meaning the revenue-based adaptive model.

TABLE 6 Static configuration of the schedulers.

Discipline	Parameters		
	Gold	Silver	Bronze
WFQ	0.3	0.6	0.1
WRR	6	4	1
DRR	3000	3500	640

WFQ, or WRR, or DRR. While testing the adaptive models, the router runs the appropriate model for the underlying scheduler. To simplify the simulation and avoid mutual interference of the applications, each client node hosts exactly one application that generates exactly one stream of data, addressed to the destination node. Every node is connected to the router with a link, whose bandwidth and delay are set to 1 Mbps and 2 ms, respectively. It should be noted that the router classifies and schedules the packets only when they move to the destination node. All responses, if any, are sent back to the source applications unclassified in the first-come-first-served order.

To compare the adaptive models and the scheduling disciplines with a static configuration, independent simulation runs were made for every discipline in the static and adaptive mode. In order to make this comparison a fair one, the same behaviour pattern of data flows, which is presented in Fig. 66, was submitted in each case. It is generated using the ON / OFF model, the parameters of which are given in Table 5.

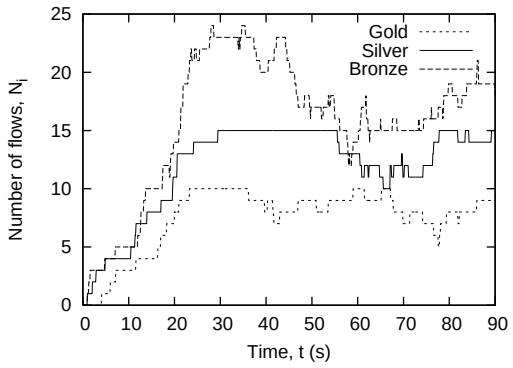
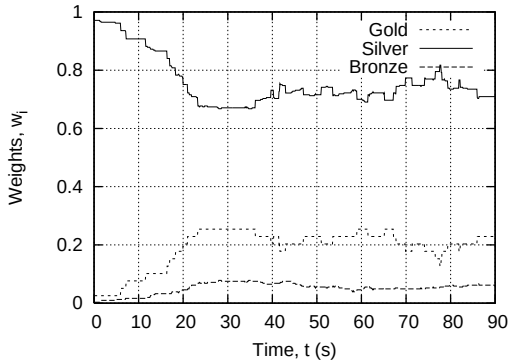


FIGURE 66 Dynamic of the number of active flows.

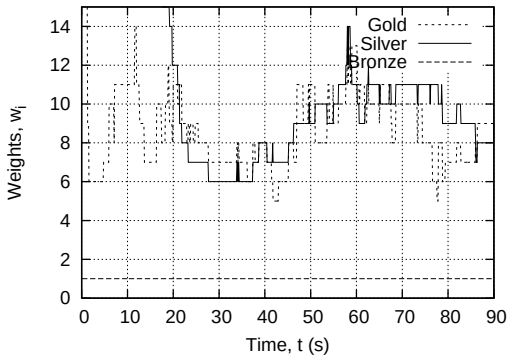
In a static case, the scheduler configuration is chosen so that each service class always has enough bandwidth, *regardless* of the number of active data flows. It corresponds to the case when the network is *overprovisioned*. Table 6 presents the static weight values for the schedulers.

As an opposite to this, the adaptive models recalculate the weight values when the number of active flows changes. Depending on the underlying scheduler, the weights are calculated differently (see Fig. 67). In the case of WFQ, the weights are the floating-point numbers in a range of 0 to 1, which enables the adaptive model to calculate them precisely. In the case of WRR, weights are integer numbers. So, the adaptive model has to find the optimal configuration that

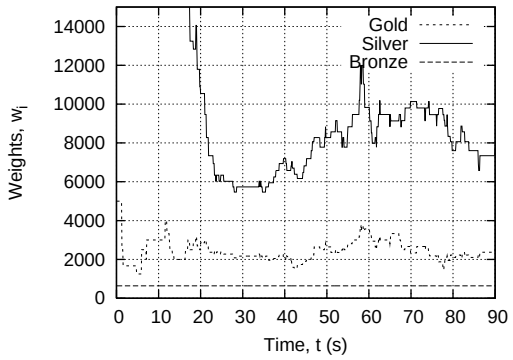
is based on the integer weight values. It results in small "jumps" that can be seen from Fig. 67(b). Though the quantum values of the DRR scheduler are also integer numbers, they specify the number of bytes to output during one round. As a result, calculated quantum values for the DRR scheduler look smoother than weight values of WRR, but still they are not as accurate as weights of WFQ.



(a) RA-WFQ



(b) RA-WRR



(c) RA-DRR

FIGURE 67 Dynamics of weights.

It is worth noting that regardless of the underlying scheduling discipline, the general dynamics of the calculated weights is the same. Adaptive models try to allocate as much as possible resources for the Silver class, at the same time providing enough bandwidth for the Gold and Bronze classes. Though the Gold class is the most expensive one, it consists of UDP flows, for which there is no sense in providing excess resources (the value of γ_i for this class is 0). Since the Silver class is more expensive than Bronze, the latter is allotted only minimal bandwidth resources. Thus, the remaining bandwidth is consumed by the Silver packets. Such a resource allocation increases the total revenue.

Table 7 shows statistical data collected during the simulation runs. The following parameters are presented for every tested queuing discipline: the number of transmitted packets, mean per-flow rate (measured in Kbps), and the total revenue, which is calculated based on the amount of transmitted data by using (1). The mean per-flow rate is obtained by using the exponentially weighted moving average. As presented in Table 7, the same number of Gold packets are transmitted under all scheduling disciplines and modes. The difference in a few packets

TABLE 7 Simulation results (no QoS framework, bandwidth guarantees).

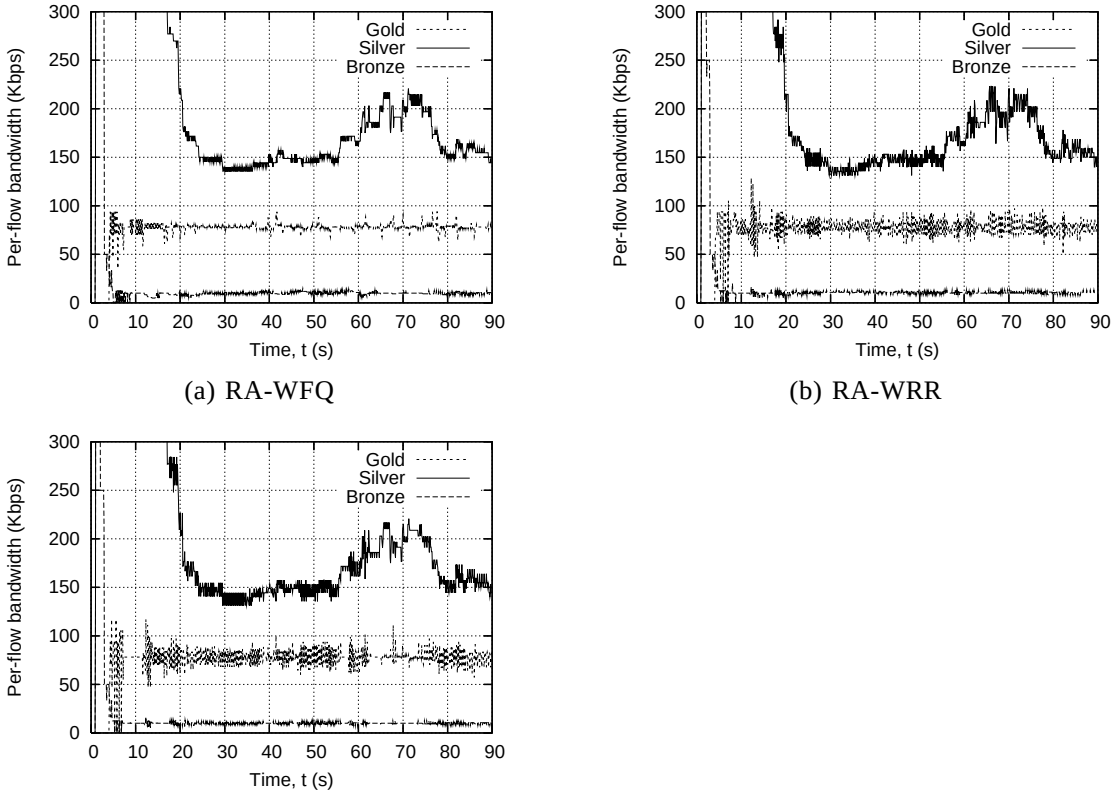
Quantity	Discipline	Classes		
		Gold	Silver	Bronze
Departed packets	WFQ	22320	28548	6727
	WRR	22319	27996	7232
	DRR	22319	28222	6976
	RA-WFQ	22287	31128	3355
	RA-WRR	22315	30946	3362
	RA-DRR	22304	31069	3246
Mean per-flow rate (Kbps)	WFQ	77.05	159.05	21.26
	WRR	77.06	155.91	23.48
	DRR	77.06	156.98	22.67
	RA-WFQ	77.07	172.77	10.47
	RA-WRR	77.06	171.89	10.88
	RA-DRR	77.06	172.59	10.40
Total revenue	WFQ	316		
	WRR	313.6		
	DRR	314.5		
	RA-WFQ	324.6		
	RA-WRR	323.5		
	RA-DRR	323.9		

is explained by the fact that some packets were being processed when the simulation stopped. The number of the transmitted packets of the Silver and Bronze classes are different. Since WFQ, WRR, and DRR rely upon the static weight values, resources are not allocated optimally. Though all the bandwidth requirements are satisfied, the Bronze class has a higher mean per-flow rate than actually required. As a result, the total revenue is lower than in the adaptive case. Since the adaptive models calculate the parameters based on the number of flows and their QoS requirements, a better resource allocation is achieved. The Silver class has a larger number of transmitted packets, and a smaller number of packets are transmitted within the Bronze class. However, it is noticeable that its mean per-flow rate is not smaller than 10 Kbps.

If we compare the results obtained for different adaptive models then, not surprisingly, we can notice that RA-WFQ provides the best results in terms of the total revenue, while RA-WRR has the lowest total revenue among RA-WFQ and RA-DRR. As explained earlier, weights for the WFQ scheduler are the floating-point numbers. Thus, WFQ can provide higher accuracy in allocating resources compared to the WRR and DRR schedulers. It is interesting to note that the RA-DRR places the intermediate position. Its total revenue is less than the one obtained under RA-WFQ but is larger than RA-WRR. In any case, as follows from the simulation results, all the adaptive models give better results than the schedulers with the static configuration.

Fig. 68 illustrates the dynamics of the per-flow rate within each service class in the adaptive case. The per-flow rate is calculated by dividing the amount of

data transmitted during a sufficiently small time interval by the number of active data flows within a service class. As can be seen, there is a warm-up period that lasts approximately 10-20 seconds. During that period new TCP flows of the Silver and Bronze classes are injected into the network. As the network state stabilizes, the Gold class has a per-flow rate that fluctuates near the value of 78 Kbps. The rate fluctuations are explained by the nature of the transmitted data packets and by the fact that the flows appear and disappear. Furthermore, it is noticeable that under RA-WRR and RA-DRR the fluctuations are bigger. This is due to the round-robin nature of these scheduling disciplines. Depending on whether the static configuration or the adaptive model is used, different per-flow rates are provided for the Silver and Bronze classes. When the scheduler parameters are static, some Bronze flows obtain more bandwidth than necessary which results in an inefficient allocation of resources. As can be seen from Table 7, the per-flow rate is bigger than 10 Kbps for WFQ, WRR, and DRR. As an opposite to this, the adaptive model tracks the number of active flows and allocates only the necessary amount of bandwidth. As shown in Fig. 68, less bandwidth is allocated for the Bronze class. Nonetheless, despite the fact that the number of active flows varies all the time, the adaptive models calculate weights in such a way, the per-flow rate of the Bronze class never goes below the value of 10 Kbps and the per-flow rate within the Gold class is 78 Kbps.



(a) RA-WFQ

(b) RA-WRR

(c) RA-DRR

FIGURE 68 Dynamics of the per-flow rate.

The behaviour of the Silver per-flow rate is explained by the available band-

width and by the number of active flows within this class. For instance, from 20 to 50 seconds of the simulation time, the number of active flows within the Silver class has reached the maximum of 15 flows (see Fig. 66). As a result, the per-flow rate was 150 Kbps on average. From 60 to 70 seconds, the number of active flows within all the classes declined. Thus, a considerably bigger amount of bandwidth was available for the Silver class and the per-flow rate has reached the value of 200 Kbps.

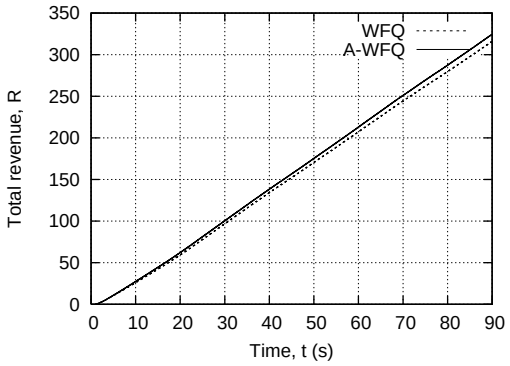


FIGURE 69 Progress of the total revenue.

Fig. 69 presents the progress of the total revenue under different queuing disciplines, used in this simulation scenario. As considered earlier, the adaptive models give higher revenue when compared to the disciplines with the static configuration. Although it may seem that the total revenue increases insignificantly, Fig. 69 illustrates that as the time goes, the gap between the total revenue obtained under the adaptive models and under the other queuing mechanisms increases. In other words, if the difference between the total revenue under WRR and RA-WFQ is 11 monetary units after 90 seconds of simulation, then the difference will double after 180 seconds. Of course, it is true when the behaviour of flows remains the same.

It is also worth mentioning that relatively small revenue gain is explained by the fact that the static configuration of the schedulers was quite close the mean weight values calculated by the adaptive models. In fact, the static configuration was determined based on the results obtained from the simulation of the adaptive models. In the real-life scenario, a provider has no such a possibility. As a result, the static configuration is far from being close to the optimal solution, and the adaptive models will provide a significantly bigger revenue comparing to the non-adaptive case.

5.3 Integrated Services

In this simulation case we consider a more realistic scenario to analyse the behaviour of the adaptive models. Now, the RSVP protocol[3] is used to inform the

[3] We have used the implementation of the RSVP protocol made for the NS-2 simulator by Marc Greis. Since that implementation includes the admission control module and a simple

TABLE 8 Traffic profile sent in the PATH message.

Class	IntServ class	Bucket rate (Kbps)	Bucket size (B)
Gold	GS	78	300
Silver	CL	50	1680
Bronze	CL	10	1280

router about the required resources. Such a framework corresponds to the IntServ QoS model, in which the RSVP protocol signals the network about the required resources. The general interaction between the adaptive models and the RSVP protocol is the ssme as described in section 4.1.2, however, only PATH, RESV, and RESVTEAR messages are sent between the nodes. We do not consider the case when the router does not have the sufficient amount of resources and, as a result, sends the RESVERROR message. Neither do we consider a case when the router rejects the PATH message and sends the PATHERROR message.

The mapping of the service classes into the IntServ classes is quite straightforward. The Gold class corresponds to GS, while the flows of the Silver and Bronze classes correspond to the CL service with the different level of required resources.

Table 8 presents the parameters for the traffic profile sent in the PATH message by the client applications. Note that the burst size of the Silver and Bronze classes is not analysed by the adaptive models since these classes do not have the delay guarantees. Practically, a router uses the bucket size to reserve enough buffer space, but during the simulation runs buffers are configured statically.

Since the RSVP protocol sends messages along the forwarding path, appropriate bandwidth resources must be provided for the RSVP packets as well. Otherwise, important signalling information may be lost. The straightforward solution is to send the RSVP packets with user data. However, according to our preliminary simulations, it is not efficient since the RSVP packets can be delayed in queues of the bottleneck router and, as a result, it does not receive notifications about required or freed resources immediately. Furthermore, due to the finite size of buffers, the RSVP packets can be even dropped. Another solution is to introduce the strict priority queue for the signalling packets. Unfortunately, it may result in the decreased accuracy of the QoS in user classes. In the core networks, the signalling information may consist of data sent by routing, network management, and configuration protocols. Most of them, and especially the routing protocols, are of the bursty nature [97], i.e. long periods of inactivity are followed by the data transmission. As a result, they can impact the delay experienced by user packets. Thus, to ensure all the QoS requirements, it is better to add a new service class to the adaptive model for the signalling packets. The model will ensure that the router has enough resources and that all the QoS requirements will be guaranteed.

WFQ scheduler, we have decoupled those components from RSVP to use the latter for pure signalling purposes. See Appendix **??** for more details.

All the adaptive models allocate 56 Kbps for the RSVP packets. Besides, since the adaptive model for the WRR scheduler requires the mean packet size, it has been set to 164 bytes, which corresponds to the size of the PATH message.[4] Though the client applications also send the PATHTEAR messages, each of which occupies 120 bytes, the number of the PATH messages is bigger because the client applications have to refresh the reservation states periodically. As mentioned earlier, the router does not classify packets as they more from the destination node to the sources applications. Thus, the size of the RESV messages is not critical.

To provide the detailed simulation of the adaptive models in the IntServ framework, two simulation subcases are introduced. The first one corresponds to the case when a provider has to ensure only the bandwidth guarantees, while in the second subcase, both the bandwidth and delay guarantees must be ensured.

5.3.1 Bandwidth

The simulation environment for this subcase is exactly the same as presented in Fig. 65. The only difference is that all the nodes participate in the RSVP signalling. Before starting to transmit data, a client application sends the PATH message and waits until the RESV message arrives. When an application stops data transmission, it sends the PATHTEAR message. Since the bottleneck router participates in the RSVP signalling, it tracks the number of active flows and updates the parameters of the schedulers by using the adaptive models. As explained in section 4.1.2, the adaptive models are triggered by the RSVP protocol when the router receives either the RESV or PATHTEAR message.

Since all the adaptive models provide the bigger total revenue comparing to the scheduling disciplines with the static configuration, we will present the simulation results with the RSVP protocol only for the adaptive models. Besides, figures for the dynamics of weights and the per-flow rate will be omitted since they are almost the same as for the previous simulation case.

Table 9 presents the results for this simulation case. Along with user classes, the table also contains information for the RSVP protocol. As follows from the results, all the adaptive models have ensured the QoS requirements of all the service classes, including the class for the RSVP packets. The mean per-flow rate of the Bronze packet is not lower than 10 Kbps, and the mean per-flow rate within the Gold class is approximately 77 Kbps. It is not 78 Kbps since the router with the adaptive models increments the number of active flows when the RESV message arrives. However, the client application starts to send data a little bit later when the RESV message reaches the client node. Thus, there is a small mistiming between the number of active flows at the router and the number of flows that actually send data.

---

[4] The implementation of the RSVP protocol for the NS-2 simulator does not support the ADSPEC object [162] that selects the service type. This object occupies 44 bytes in the case of the CL and NULL service types, and 76 bytes in the case of GS. As a result, the size of the PATH message is smaller.

TABLE 9 Simulation results (IntServ, bandwidth guarantees).

Quantity	Discipline	Classes				
		Gold (GS)	Silver (CL)	Bronze (CL)	RSVP	
					PATH	TEAR
Departed packets	RA-WFQ	22308	31013	3426	197	76
	RA-WRR	22290	30447	3925	232	74
	RA-DRR	22288	31044	3324	231	74
Mean per-flow rate (Kbps)	RA-WFQ	77.12	172.26	10.71	4.30	
	RA-WRR	77.09	171.22	11.69	4.78	
	RA-DRR	77.11	172.81	10.68	4.73	
Total revenue	RA-WFQ	324.1			–	
	RA-WRR	321.5			–	
	RA-DRR	323.7			–	

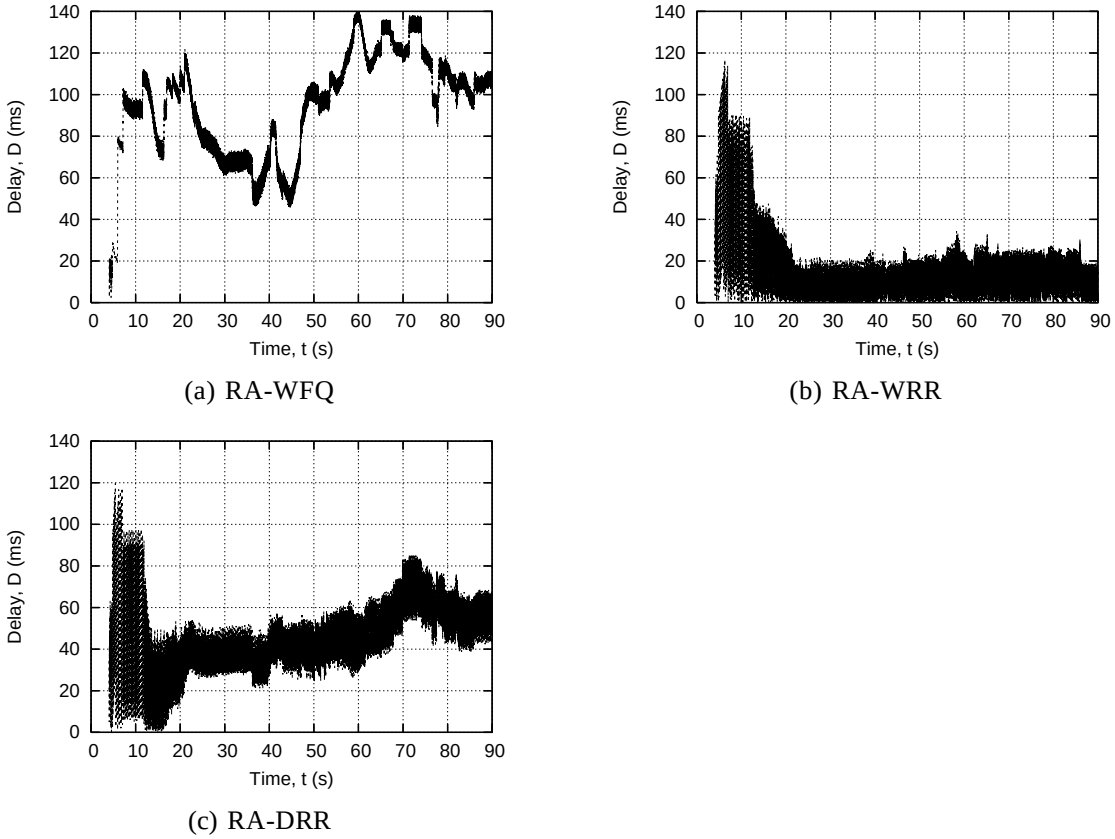
The number of the RSVP packets is significantly smaller comparing to the number of packets sent within the user classes. Indeed, the client applications do not have to send signalling data all the time. Rather, they refresh their reservation states periodically. For these reasons, the number of sent PATH messages is bigger than the number of the PATHTEAR messages, which are sent when the resources are not needed anymore.

The presence of the RSVP protocol has not changed the results significantly. Comparing the results presented in Table 7 and Table 9, it is possible to notice that they are almost the same. However, the total revenue is a little bit smaller. It is explained by the fact the adaptive models have allocated some bandwidth for the RSVP packets. As a result, less bandwidth was available for the Silver class. It is also interesting to note that the difference in revenue between different adaptive models was quite small in the previous simulation case, while after we had introduced a new class for the RSVP packets, the gap has increased. The explanation is that since RA-WFQ relies upon the floating-point weight values, it is capable of allocating resources accurately regardless of the number of service classes and their requirements. RA-WRR and RA-DRR have to rely upon the integer weight values. As a result, the more service classes with diverse QoS requirements we have, the bigger is the anticipated gap in the total revenue between RA-WFQ and the adaptive models based on the round-robin schedulers.

5.3.2 Bandwidth & delay

The purpose of this simulation subcase is to analyse how the adaptive models provide simultaneously the bandwidth and delay guarantees. Fig. 70 presents the queuing delay of the Gold packets under the adaptive models when only the bandwidth guarantees are ensured. As follows from this figure, the queuing delay is much bigger than 20 ms, as specified in Table 5. Thus, the adaptive models have to allocate enough bandwidth resources to ensure also the delay guarantees.

The simulation environment is the same as presented in Fig. 65. As in the



(a) RA-WFQ

(b) RA-WRR

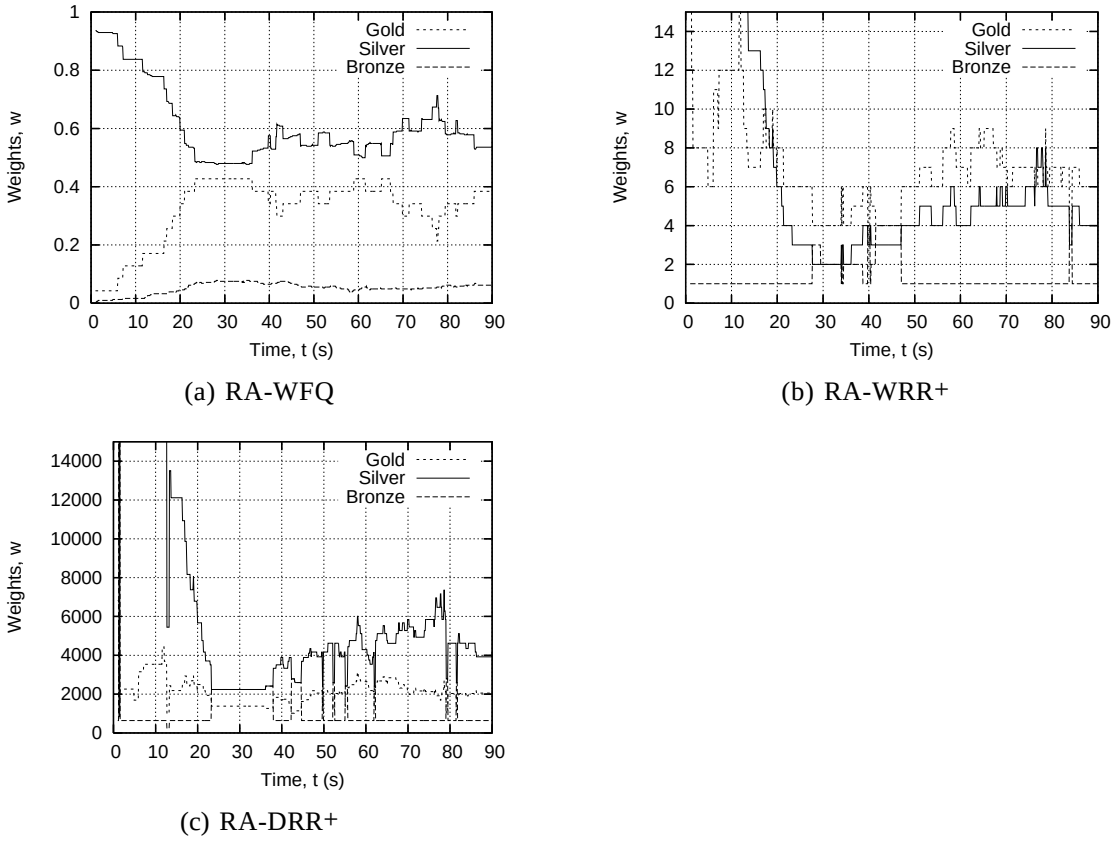
(c) RA-DRR

FIGURE 70 Queuing delay of the Gold class packets.

previous subcase, the RSVP protocol is used by all the client applications to signal network about the required resources. The adaptive models running in the bottleneck router calculate the optimal weight values to ensure the QoS guarantees for all the service classes including RSVP. It should be noted that since we have to ensure the delay guarantees, the adaptive models (44) and (56) are used. Besides, the WRR and DRR schedulers have to work in the LLQ mode so that the delay critical queue is served in between other queues.

Fig. 71 presents the dynamics of the weight values calculated by the adaptive models (for the sake of clarity, weight values of the RSVP class are omitted). As follows from this figure, the general dynamics of weights is quite similar to the one presented in Fig. 67 and is governed by the fact that the adaptive models try to allocate as much as possible resources for the Silver class at the expense of the other service classes. However, it is noticeable that this time the adaptive models allocate more bandwidth to the Gold class to ensure the required delay guarantees. For instance, as follows from Fig. 71(a), the weight of the Gold class fluctuates near the value of 0.4, while in the previous simulation cases its mean value was 0.2 (see Fig. 67(a)). The same is for the weight values of the WRR and DRR schedulers. To provide the necessary delay guarantees, the adaptive models for these schedulers have to assign smaller weights for the Silver class, thus providing more bandwidth for the Gold class.

Table 10 presents the results collected during these simulation runs. As fol-



(a) RA-WFQ

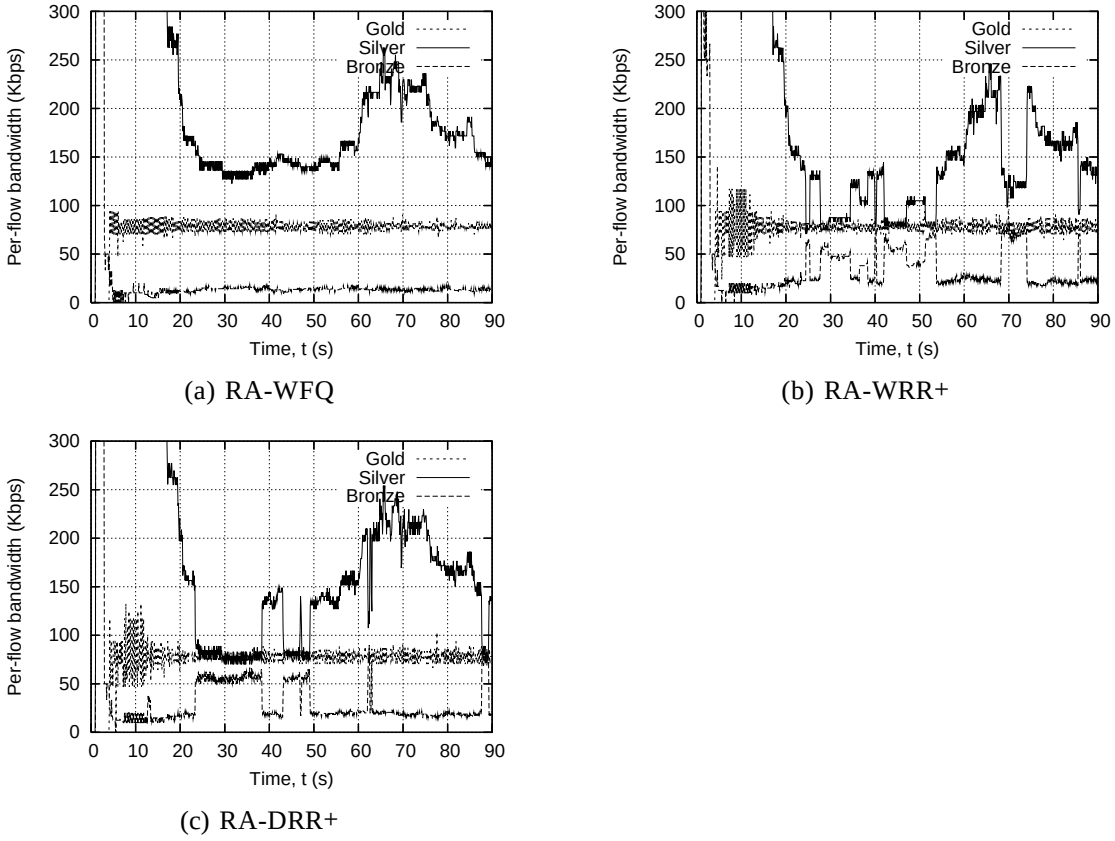
(b) RA-WRR+

(c) RA-DRR+

FIGURE 71 Dynamics of weights (IntServ).

TABLE 10 Simulation results (IntServ, bandwidth and delay guarantees).

Quantity	Discipline	Classes				
		Gold (GS)	Silver (CL)	Bronze (CL)	RSVP PATH	RSVP TEAR
Departed packets	RA-WFQ	22308	30489	4113	197	76
	RA-WRR+	22296	27199	8185	232	74
	RA-DRR+	22307	27167	8376	232	74
Mean per-flow rate (Kbps)	RA-WFQ	77.12	169.29	13.01	4.30	
	RA-WRR+	77.02	156.13	23.45	4.73	
	RA-DRR+	77.04	158.52	21.90	4.73	
Mean delay (ms)	RA-WFQ	1	204	1829	–	
	RA-WRR+	4	218	1003	–	
	RA-DRR+	4	212	1216	–	
Total revenue	RA-WFQ	322.3			–	
	RA-WRR+	310.6			–	
	RA-DRR+	310.9			–	



(a) RA-WFQ

(b) RA-WRR+

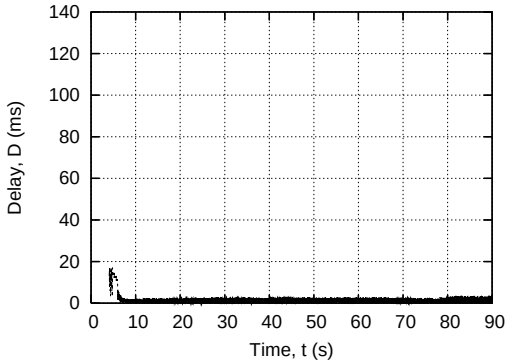
(c) RA-DRR+

FIGURE 72 Dynamics of the per-flow rate (IntServ).

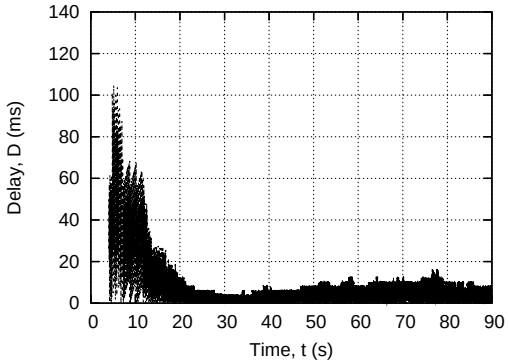
lows from the results, all the adaptive models provide sufficient amount of bandwidth resources for all the service classes. It is interesting to note that the mean per-flow rate of the Bronze class is higher than in the previous simulation subcase. It is explained by the fact that the adaptive models cannot allocate much bandwidth for the Silver class as it will cause a larger queuing delay of the Gold packets. Thus, the models try to increase the total revenue by allocating more resources for the Bronze class. Exactly for these reasons, the total revenue is smaller than in the previous simulation case, in which only the bandwidth guarantees must be ensured. It is noticeable that the RA-WRR+ gives worse results in terms of the total revenue among the other adaptive models. RA-DRR+ provides better results than RA-WRR+, however it is not as good as RA-WFQ. So, if there are tight delay requirements, it is better to use RA-WFQ. As in the previous simulation subcase, the number of the RSVP packets is not large.

Fig. 72 presents the dynamics of the per-flow rate within each service class under different adaptive models. All the classes are allotted sufficient bandwidth resources so that the per-flow rate is equal to or bigger than the bandwidth specified in Table 5. The general behaviour of the per-flow rate under the RA-WFQ model is quite similar to the one presented in Fig. 68(a), however, the per-flow rate under RA-WRR and RA-DRR is different. As can be seen from Fig. 72(b) and Fig. 72(c), there are cases in which the per-flow rate within the Bronze class is considerably larger than 10 Kbps and reaches the value of 50-70 Kbps. As explained

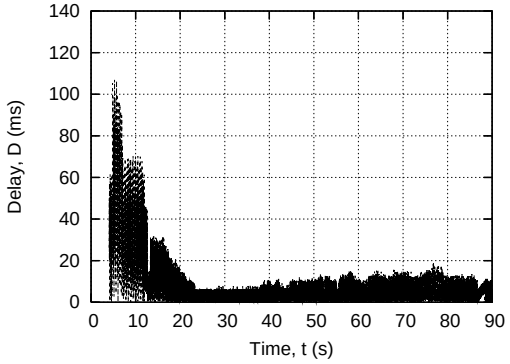
above, the adaptive models for the WRR and DRR schedulers cannot assign large weight values for the Silver class because it will cause bigger queueing delays within the Gold class. As a result, the adaptive models try to compensate it by assigning larger weight values for the Bronze class which results in a bigger per-flow rate. Another reason is the integer weight values of WRR and DRR.



(a) RA-WFQ



(b) RA-WRR+



(c) RA-DRR+

FIGURE 73 Queuing delay of the Gold class packets.

According to Table 10, regardless of the adaptive model, the mean delay of the Gold class packets is less than 20 ms. While calculating this parameter, we have removed the first 20 seconds of simulation when the network warms up. Along with the mean delay, it is also important to ensure that the *peak delay* is not more than 20 ms. Fig. 73 presents the queuing delay experienced by all Gold packets under different adaptive models. This data also includes the first 20 seconds of the simulation when the network state stabilizes. It is interesting to note that RA-WFQ provides the required delay guarantees even at the beginning of the simulation, while RA-WRR and RA-DRR fail to do it. As explained in section 2.3.5, we can rely upon expression (35) only if $N_l\sigma \gg w_i L_i^{min}$, in other words, when there is a significant amount of active flows within a class. However, there are only a few flows within the Gold class at the beginning of the simulation run. As a result, the queuing delay is bigger than 20 ms.

Having analysed Fig. 73 thoroughly, it is possible to arrive at the conclusion that the adaptive models allocate a little bit more resources than necessary for the Gold class. Indeed, having thrown off the first 20 seconds of the simulation, it

is possible to notice that the peak delay is sometimes considerably smaller than 20 ms, which is especially the case for RA-WFQ. As considered in section 2.3.2, the worst-case estimation used to calculate the burst size relies upon a simple assumption that the resulting burst size of a class is just a sum of burst sizes of all the flows that belong to this class. Though it ensures the delay guarantees, it does not provide the efficient resource allocation. Thus, by using the more sophisticated model for estimating the burst size, the better resource allocation can be achieved and, as a result, the total revenue will increase.

5.4 Differentiated Services

In this simulation case, the adaptive models are analysed in the DiffServ QoS framework. Among the models presented in chapter 2, only RA-WFQ and RA-WRR will be considered because the official release of the NS-2 simulator does not provide the implementation of the DRR scheduler for the DiffServ architecture. As presented in the previous simulation cases, RA-WFQ provides the best results from the viewpoint of the total revenue, while RA-WRR has the worst results comparing to RA-WFQ and RA-DRR. Thus, it is enough to analyse RA-WFQ and RA-WRR because the results for RA-DRR will place an intermediate position.

The service classes are exactly the same as presented in Table 5. In addition to them, a new class for the BE data is introduced. It represents non-critical user data, such as the mail system, that has no particular requirements. The inclusion of this traffic class has several purposes. First, it is used to demonstrate how the adaptive models protect other classes from the best-effort traffic. Second, it is also important to ensure that certain amount of resources are allocated for the BE data. Third, the BE class demonstrates the way the adaptive model allocates resources between service classes with different pricing strategies. While the Gold, Silver, and Bronze classes have the associated usage-based price, the BE class has no price at all, or may have the flat price. The reason for not including the BE class in the previous simulation cases was the desire to present the basic properties of the adaptive models.

Since the BE flows do not have any QoS requirements, the bandwidth of 100 Kbps is reserved for the whole class and all the data streams compete for the available resources. The Telnet application is chosen to generate general purpose BE data that is carried over the TCP protocol. The packet size is set to 340 bytes. It should be noted that the number of active BE flows do not change during a simulation run. After all the 40 streams are injected, they create a constant load on the network.

Fig. 74 illustrates the simulation environment used to test the adaptive models in the DiffServ framework. It is noticeable that it is an extended version of the environment presented in Fig. 65. As in the previous simulation cases, all the client applications are connected to the bottleneck router that performs the adaptive sharing of resources. This router acts as the *ingress* node of the DiffServ

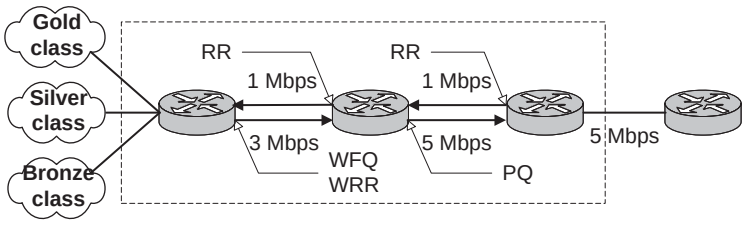


FIGURE 74 Simulation environment for the DiffServ framework.

domain that consists of the core router and two border routers. Though the core router does not have a bottleneck link, it is decided to use the PQ scheduling discipline there to provide some basic QoS differentiation between the service classes. The matter is that while the router outputs one big packet, two smaller packets can arrive. Thus, the PQ discipline will ensure that the more important data is sent first. The *egress* router implements a simple FCFS discipline. As in the previous simulation cases, when packets move from the destination node to the source applications, they are not classified and sent as the BE class. However, to limit the impact of the user data on the signalling information, a separate class is allocated for the signalling packets. The DiffServ routers use the RR scheduler to share bandwidth between the user and signalling data.

To inform the DiffServ domain about the required resources, the client applications use the RSVP protocol. Traffic profiles carried in the PATH message are exactly the same as specified in Table 8. Since the BE applications do not have the QoS requirements, they do not participate in the signalling at all. Such a framework corresponds to the hybrid QoS model described in section 4.3. The difference between the IntServ framework is that only the client applications, the ingress router, and the destination node exchange the RSVP messages. The core and egress DiffServ routers forward the RSVP packets as the ordinary IP datagrams. From the viewpoint of the RSVP protocol, the DiffServ domain appears as one RSVP node. As mentioned in section 4.3, in such a framework the ingress router must perform the admission control for the whole DiffServ domain when the RESV message arrives. However, we assume that the domain always has enough resources. Thus, the ingress router only performs the adaptive allocation of resources.

The mapping of the service classes to the DiffServ PHB aggregates is quite straightforward. The Gold class corresponds to the EF PHB because the former has the bandwidth and delay guarantees. The Silver class corresponds to the AF2 PHB since it has more demanding QoS requirements than the Bronze class, which is mapped to the AF1 PHB. The RSVP signalling information is mapped to the class-selector PHB value of 48, which means inter-domain management data. Table 11 summarizes the DiffServ aggregates used in the simulation. Hence, we will be using a term *aggregate*, not class, to conform to the DiffServ terminology [64].

Table 11 also presents the configuration parameters for the aggregates. For each behaviour aggregate, the in- and out-profile DSCP values and information

TABLE 11 Parameters of the DiffServ aggregates.

Aggregate	DSCP		Meter	
	in	out	Type	Parameters
RSVP	48	–	NULL	
EF	46	–	Token Bucket	78 Kbps, 300 B
AF2	18	20	TSW2CM	50 Kbps
AF1	10	12	TSW2CM	10 Kbps
BE	0	–	NULL	

on meters are given. It should be noted that the RSVP, EF, and BE aggregates do not have the out-profile codepoints. The policing action for the EF packets is to drop all the excessive traffic, thus the out-profile packets will not ever leave the DiffServ router. The BE aggregate has no associated QoS parameters at all. As a result, there is no need to measure its characteristics and to classify the incoming data into in- and out-profile packets. For exactly these reasons, the NULL meter is used for the BE aggregate. The RSVP aggregate has the same configuration. The AF aggregates rely upon the TSW2CM meter that requires only one parameter – the mean rate. It simplifies the policing of the incoming data since a provider can set the mean rate to the bandwidth that has to be provided for each flow (compare requirements in Table 5 and parameters in Table 11). The EF aggregate uses the Token Bucket meter that polices the incoming packets by using the mean rate and the burst size. By knowing the maximum burst size, the adaptive models are capable of providing the delay guarantees.

As can be noticed from Table 11, two drop precedences are used for the AF aggregates. As considered in [61], it makes sense to introduce three drop precedences only if a provider has excess bandwidth. In such a case, the highest drop precedence level will act as an early notifier for the TCP flows to slow down the data transmission. If the bandwidth requirements approach the available bandwidth resources, then the highest drop precedence queue will get full immediately. Another reason to introduce three drop precedence levels is to isolate the out-profile congestion sensitive packets from the out-profile congestion insensitive ones. In this simulation case, all the out-profile packets belong to the TCP protocol and the bandwidth requirements approach the available bandwidth. Thus, it is enough to use two levels.

It is also worth noting that two drop precedence levels simplify the interaction between the RSVP protocol and the DiffServ policing mechanisms. The RSVP traffic profile contains the token bucket rate and size, but it does not carry the peak burst size. Thus, if the RSVP messages are used to set up automatically the DiffServ ingress routers, then it is impossible to use meters, such as srTCM and trTCM. However, since the traffic profile object contains the peak rate, it is still possible to classify packets into the three drop precedence levels by using the TSW3CM meter.

To cope with the severe congestions and the effect of the global synchronization, and to provide lower delay for the interactive services [24], the DiffServ

TABLE 12 Parameters of the AQM mechanism.

Aggregate	AQM mechanism				
	Type		Min thres.	Max thres.	Probability
RSVP	DropTail		–	10	–
EF	DropTail		–	15	–
AF2	RIO-C	in	30	60	0.05
		out	15	30	0.2
AF1	RIO-C	in	45	90	0.07
		out	20	45	0.2
BE	RED		40	100	0.15

routers use the AQM techniques that are based on the RED mechanism. Table 12 presents the type and the parameters of the AQM mechanism used for each behaviour aggregate (threshold values are specified in the number of packets). The RSVP and EF aggregates rely upon the simple drop-tail mechanism. The reason is that neither RSVP nor the UDP packets of the EF aggregate can react to the packet drops. Thus, simple drop-tail queues are the best choice for this non-responsive traffic. Since the AF and BE aggregates have the applications that send data over the TCP protocol, it is better to use RED to avoid the drop-tail behaviour of queues. According to [101], RIO-C provides definite assurance for packets of the lower precedence comparing to WRED. Furthermore, the *staggered* settings provide better assurance for the in-profile packets than the overlapped or partially overlapped parameters.

To provide a thorough analysis of the adaptive models in the DiffServ framework, two simulation subcases will be considered. First, the adaptive models will have to ensure only the bandwidth guarantees, and in the second simulation subcase, both the bandwidth and delay requirements must be ensured.

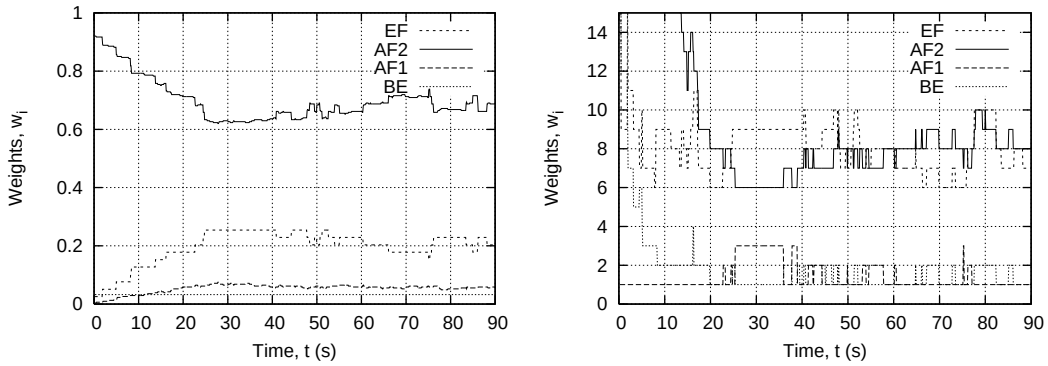
Referring back to Fig. 74, it should be mentioned that some data is gathered at the ingress router, while some parameters are calculated at the egress router. The number of transmitted packets, the mean per-flow rate, and the total revenue are gathered at the egress router. Such an approach ensures that we measure and analyse packets' characteristics as they leave the DiffServ domain. For instance, a provider considers a packet to be transmitted as it leaves a domain, and not as a packet enters it. At the same time, the queuing delay and packet drops are gathered at the ingress router, in which the adaptive model resides.

5.4.1 Bandwidth

In this simulation subcase we present the results for the adaptive models in the DiffServ framework when a provider has to ensure only the bandwidth requirements. In such a case, the EF aggregate can be treated as AF3 with the more demanding requirements when compared to the AF1 and AF2 aggregates.

Fig. 76 illustrates the calculated weight values. As in the previous simulation cases (see Fig. 67), they are calculated so that the total revenue is maximized

and all the QoS guarantees are ensured. As can be seen, the BE aggregate has a fixed value of 0.03, which corresponds to 100 Kbps reserved for this aggregate. In the case of RA-WRR, the BE weight value is either 1 or 2. Thus, the adaptive models calculate weight values accurately regardless of the fact whether an aggregate has the per-flow or per-aggregate bandwidth requirements.



(a) RA-WFQ (b) RA-WRR

FIGURE 75 Dynamics of the weights (DiffServ).

TABLE 13 Simulation results (DiffServ, bandwidth guarantees).

Quantity	Discipline	Classes								
		Gold (EF)	Silver (AF2)		Bronze (AF1)		BE	RSVP	
			in	out	in	out		PATH	TEAR
Departed packets	RA-WFQ	23188	8308	21094	2112	1750	2902	233	80
			29402		3862				
	RA-WRR	23177	8361	19388	2358	3394	3245	233	80
			27749		5852				
Dropped packets	RA-WFQ	0	0	3138	0	440	236	0	0
	RA-WRR	0	0	3040	0	673	92	0	0
Per-flow rate (Kbps)	RA-WFQ	76.94	49.55	113.56	7.54	3.72	2.52	5.00	
			163.11		11.26				
	RA-WRR	76.88	50.28	107.93	7.96	5.98	2.88	5.01	
			158.21		13.94				
Total revenue	RA-WFQ	318.6						–	
	RA-WRR	312.5						–	

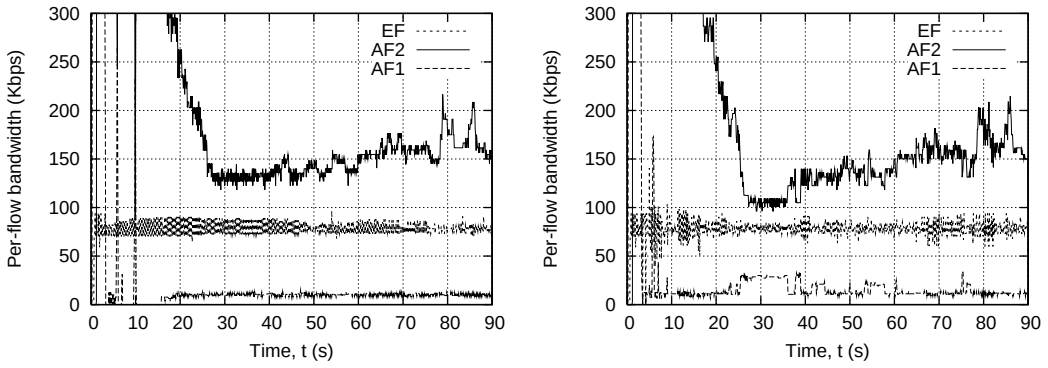
Table 13 presents the results for this simulation subcase. Like in the IntServ simulation, the table includes data, such as the number of transmitted packets, the mean per-flow rate, and the total revenue. In addition, the number of transmitted in- and out-profile packets is presented for the AF aggregates. In the same way, the mean per-flow rate has been measured separately for the in- and out-profile packets. As follows from Table 13, RA-WFQ and RA-WRR transmitted approximately the same amount of EF packets. The difference of ten packets is explained by the fact that under WRR these packets were queued at the end of

the simulation run. While the amount of transmitted in-profile AF packets is comparably the same for all the models, the number of the out-profile packets differs. Being capable of sharing resources precisely, RA-WFQ allocates as much bandwidth as possible for the AF2 aggregate. Consequently, only the minimum amount of resources are allocated for the AF1 aggregate. As a result, under RA-WFQ the AF2 aggregate has a larger number of out-profile packets. Although RA-WFQ allocates such an amount of resources for the AF1 aggregate that only in-profile packets are transmitted, there is always a certain number of forwarded out-profile packets. This is due to the fact that there can be inactive sessions and available bandwidth resources from other aggregates.

The mean per-flow rate of the AF2 in-profile packets approaches 50 Kbps which corresponds to the bandwidth requirements for this aggregate. Since the adaptive models allocate free bandwidth for the AF2 aggregate, the excess data is transmitted as the out-profile packets. At the same time, the per-flow rate of the AF1 in-profile packets is less than 10 Kbps. It is explained by the accuracy of TSW2CM that marked some packets as out-profile. In support of this explanation, one can notice that the mean per-flow rate for the whole AF1 aggregate is slightly larger than 10 Kbps. Thus, the scheduler has allocated enough bandwidth, but the TSW2CM meter has marked some packets wrongly. Though the BE flows do not have the per-flow bandwidth requirements, the mean per-flow rate within that aggregate is also presented. By multiplying this value by the number of the BE flows we obtain 100.8 Kbps under RA-WFQ and 102.8 Kbps under RA-WRR. Thus, the adaptive models reserved the sufficient amount of resources also for the BE aggregate.

By analysing the number of the dropped packets it is possible to arrive at the conclusion that the QoS requirements are ensured completely for the EF and the AF in-profile packets since there were no dropped packets. There is always a certain amount of dropped out-profile packets because by this the router informs the client applications to slow down the data transmission. The number of dropped AF2 out-profile packets is larger under RA-WFQ since the latter allocates more bandwidth for the AF2 aggregate. Having more bandwidth, the client applications send more data, increase the transmission window size, and are policed more aggressively by the DiffServ router. For exactly these reasons, the number of dropped AF1 out-profile packets is larger under RA-WRR, which allocates a little bit more bandwidth for AF1.

Fig. 76 illustrates the dynamics of the per-flow rate within each behaviour aggregate at the egress router (for the sake of clarity, the BE and RSVP aggregates are not included). There is a warm-up period that lasts approximately 20-30 seconds. As explained for the previous simulation cases, during that period new TCP flows of the AF and BE aggregates are injected into the network. Under RA-WFQ and RA-WRR, the EF aggregate has the per-flow rate that fluctuates near the value of 78 Kbps. For the AF aggregates, the adaptive models provide different per-flow rates. RA-WFQ provides the AF2 aggregate with the highest rate. Now and then, it reaches the value of 150 Kbps and even higher. Since RA-WRR relies upon the integer weight values, it allocates sometimes more bandwidth to



(a) RA-WFQ (b) RA-WRR

FIGURE 76 Dynamics of the per-flow rate (DiffServ).

the AF1 aggregate, the per-flow rate of which reaches the value of 30 Kbps.

The total revenue is a little bit less when compared to the results presented in section 5.3.1, in which the IntServ QoS framework and the bandwidth guarantees have been analysed. The reason is that we have introduced a new traffic aggregate for the BE data. Since it requires some bandwidth resources, less bandwidth were allotted to the AF2 aggregate.

The previous simulation run was made with the ECN technology turned off. However, since the ECN technology allows to reduce the number of dropped packets, it was interesting to know its impact on the simulation results and, in particular, on the total revenue. For these purposes, we have rerun the simulation scenario with the ECN technology turned on. Table 14 presents results for the modified simulation subcase.

TABLE 14 Simulation results (DiffServ, bandwidth guarantees, ECN).

Quantity	Discipline	Classes								
		Gold (EF)	Silver (AF2)		Bronze (AF1)		BE	RSVP	
			in	out	in	out		PATH	TEAR
Departed packets	RA-WFQ	23186	8341	21163	1988	1712	2962	233	80
			29504		3700				
	RA-WRR	23175	8352	19649	2419	3179	3093	237	82
			28001		5598				
Per-flow rate (Kbps)	RA-WFQ	76.98	49.59	113.62	7.31	3.86	2.53	5.00	
			163.21		11.17				
	RA-WRR	76.86	49.87	107.68	8.39	6.34	2.54	5.21	
			157.55		14.73				
Total revenue	RA-WFQ	318.9						–	
	RA-WRR	313.6						–	

As follows from the simulations results, it is possible to achieve zero packet loss with the help of the ECN technology (that is why the number of dropped packets has been omitted). Indeed, if the adaptive models provide enough bandwidth resources for the in-profile packets, then all excess data will be classified out-profile. Since RED in conjunction with ECN does not drop packets, but rather marks them appropriately, the zero packet loss can be achieved. It is also important to note that the total revenue is slightly bigger. The explanation is that since RED and ECN do not drop packets, network bandwidth is utilized better.

5.4.2 Bandwidth & delay

In this simulation subcase, we present the results for the RA-WFQ adaptive model when a provider has to ensure both the bandwidth and delay guarantees. Unfortunately, at the moment, the DiffServ framework of NS-2 does not support the LLQ mode of the WRR scheduler and it is not possible to present results for RA-WRR+. Thus, only RA-WFQ will be analysed.

TABLE 15 Simulation results (DiffServ, bandwidth and delay guarantees).

Quantity	Classes							
	Gold (EF)	Silver (AF2)		Bronze (AF1)		BE	RSVP	
		in	out	in	out		PATH	TEAR
Departed packets	23188	8372	20202	2364	2498	3063	233	80
		28574		4862				
Dropped packets	0	0	3027	0	560	0	0	0
Per-flow rate (Kbps)	76.94	49.95	109.12	8.44	5.34	2.73	5.00	
		159.07		13.79				
Mean delay (ms)	1.1	75.3		918.2		1.9	–	
Total revenue	315.6						–	

Table 15 presents the information gathered during this simulation subcase. As expected, the total revenue is smaller comparing to the previous subcase because the adaptive model has to allocate more bandwidth resources for the EF aggregate at the expense of AF2. Since the EF aggregate may have unused bandwidth, it is allotted to the other aggregates with respect to their weight values. As a result, the mean per-flow rate of the AF1 aggregate is slightly bigger than in the previous simulation subcase and equals 13.79 Kbps. As follows from Table 15, RA-WFQ ensures the required queuing delay the mean value of which is less than 20 ms. The mean queuing delay of the AF1 and AF2 aggregates is significantly larger, which is explained by the greedy nature of the TCP flows and the large buffer space at the ingress router. The BE aggregate has a relatively small mean queuing delay since the underlying Telnet applications do not send data constantly, as the FTP applications of the AF1 and AF2 aggregates do.

Fig. 77 presents the queuing delay experienced by all the EF packets during the simulation run. As can be seen, the peak delay is also less than 20 ms. Thus, being deployed to the DiffServ framework, RA-WFQ is capable of ensuring the

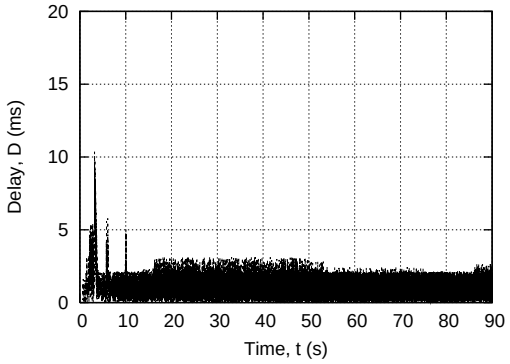
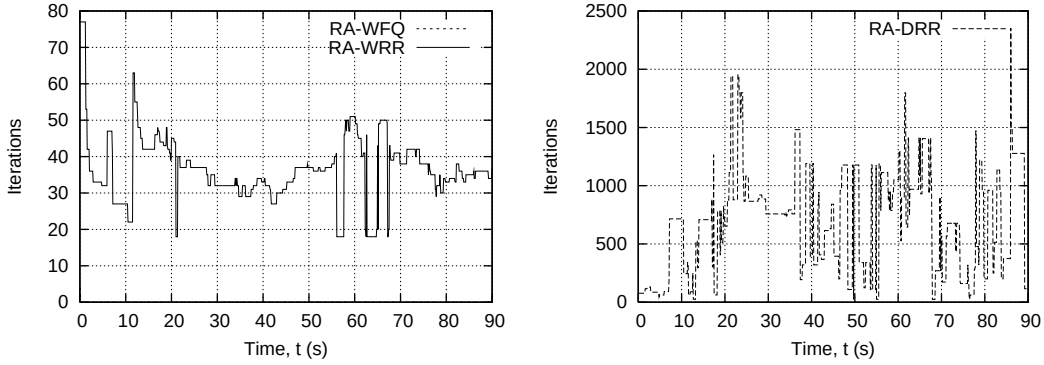


FIGURE 77 Queuing delay of the EF packets.



(a) RA-WFQ & RA-WRR (b) RA-DRR

FIGURE 78 Number of iterations.

bandwidth and delay guarantees simultaneously.

5.5 Analysis of computational complexity

As considered in section 2.4, it is important to ensure that the adaptive models do not consume a lot of computational resources. Otherwise, they may slow down the transmission of packets because the router will be busy with finding the optimal solution. It is especially important for those routers that have only one central processor and do not have additional processors on the interface cards.

Fig. 78 presents the number of iterations it took the adaptive models to calculate the optimal parameters for the schedulers. This data was gathered during the first simulation case presented and analysed before. As follows from this figure, only a few iterations are necessary by RA-WFQ. In the case of RA-WRR, 30-40 iterations are necessary on average which is more than acceptable in many cases. However, RA-DRR needs a considerably bigger number of iterations. Though it is also ILP, it has to deal with big integer numbers while finding the optimal combination of quantum values, which is not the case for RA-WRR that operates small integer numbers.

One solution to reduce the number of iterations for RA-DRR is to use the optimized ILP solvers that can take into account the form of the constrained area. Another approach is to solve the optimization task as LP and then round calculated parameters. In the case of the large floating-point numbers, rounding to the closest integer will not affect the results significantly. Let us suppose that the LP solver finds the following optimal solution: (599.21 2500.98 3600.54). If we round these numbers, then the solution is (599 2501 3601). It is understandable that it may violate some QoS constraints because the ILP solver would give another solution, for instance (600 2500 3600). One the one hand, if a provider has to ensure only the bandwidth guarantees, then we can neglect easily this small difference. On the other hand, if a provider has to ensure also the delay guarantees, then even a small difference can impact the queuing delay. For instance, suppose that the packet size is 300 bytes. If a queue is assigned the quantum value of 600 bytes, the scheduler can output two packets per round. However, if the quantum value is 599 bytes, the scheduler will output only one packet while the second packet will be transmitted during the next round. As a result, the queuing delay will increase. To avoid such a situation, it is possible to round the quantum value of the LLQ to the *largest* integer value and the quantum values of the other queues to the *smallest* integer value. It can ensure that the LLQ will be allotted sufficient bandwidth resources.

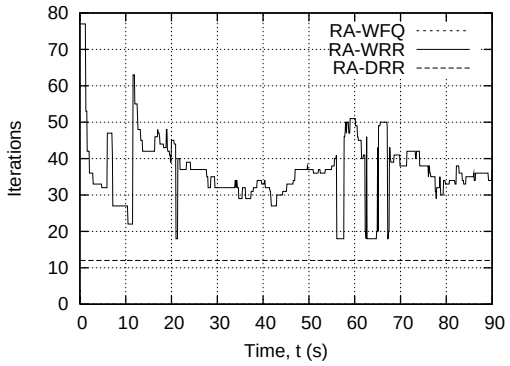


FIGURE 79 Number of iterations (improved RA-DRR).

To check whether it is as it is, we have modified the underlying adaptive model so that the quantum values are calculated as the LP, not as the ILP. Fig. 79 illustrates the number of iterations for the modified solver. As follows from the figure, now RA-DRR needs 12 iterations, which is significantly less than in the previous case. It is also worth mentioning that the simulation results are exactly the same as for the RA-DRR that relies upon the ILP solver.

5.6 Summary

The simulations results presented in this section have confirmed the correctness of the adaptive models. They are capable of ensuring the QoS requirements of several service classes with various requirements and pricing schemes. At the

same time, the models enable a provider to increase the total revenue by allocating the unused bandwidth to the more expensive service classes. The simulation results have also shown that the models can be applied easily to the DiffServ and IntServ QoS frameworks.

Based on the simulation results, it is possible to arrive at the following conclusion concerning the choice of the adaptive model. If there is a high-speed router in the core network, and the QoS requirements consist only of the bandwidth guarantees, then RA-WRR is the best choice. In the case of the core network, the router does not have much time for the scheduling decision. Thus, the underlying scheduling discipline has to be as fast as possible. At the same time, as the router has the significant output bandwidth, it is not a problem to find the optimal integer values of weights. RA-WRR can be used efficiently in the ATM networks where the cells always have the same size. The ideal place for the RA-WFQ is the router in the access network that aggregates several incoming links and has one output link, bandwidth of which is not great. In this case, the router has time to make a scheduling decision and the accurate allocation of resources is of a much bigger importance. Furthermore, the delay requirements of several classes can be guaranteed efficiently. The RA-DRR model places the intermediate position. On the one hand, its complexity is the same as for WRR, on the other hand, it provides results that are very close to those obtained under RA-WFQ. The only important limitation is that only one class can be provided with the delay guarantees.

6 REVENUE-AWARE RESOURCE ALLOCATION SCHEMES IN A MULTICLASS-SUPPORTED NETWORK NODE

Resource allocation in the multiservice communication networks presents a very important problem in the design of the future Internet. The main motivation for the research in this field lies in the necessity for structural changes in the way the Internet is designed. The current Internet offers a single class of 'best-effort' service. However, the Internet is changing and a diverse set of services should be provided in the future Internet to support the requirements of various applications and customers, which result in the definition of different service classes with different requirements of QoS levels. For instance, new sophisticated real-time applications (video conferencing, video on demand, distance learning, etc) demand a better and more reliable network performance. Moreover, these applications require firm performance guarantees from the network where certain resources should be reserved for them. On the other hand, in the future multiclass Internet, each class of customers may have to pay network service providers for their received level of QoS based on the pricing strategy agreed upon in the Service-Level-Agreements between them. A Service-Level-Agreement (SLA) defines the QoS parameters for each class of service, the anticipated per-class workload intensity and the pricing strategy by which the service payment will be determined. Obviously, the pricing strategy will specify the relationship between the QoS level offered to each class of customers and the relevant price which should be paid by them. For example, the service provider will receive a certain amount of revenue from a class of customers if the offered QoS level is more than the minimal requirement of that class and suffer a certain amount of penalty for failing to meet that. Thus, from service providers' point of view, the optimal resource allocation scheme, which can achieve the maximization of SLA revenues under a given amount of network resources (e.g., bandwidth) and a given pricing strategy, is very desirable.

In this chapter, we focus on the delay performance of each service class and *mean packet delay* and *packet delay* are chosen as the QoS metric in a SLA, respec-

tively, in this Chapter. When the *mean packet delay* is used as the QoS metric in a SLA, the measurement period of packet delays should also be specified in the SLA so that the SLA revenues can be collected periodically based on the deployed pricing strategies and the periodical QoS performance measurements (in this case, the *mean packet delay*). Whereas, with *packet delay* as the QoS metric, the SLA revenues are collected based on the used pricing strategies and the delay of each inbound packet; in other words, a certain amount of revenue or penalty is obtained as long as an inbound packet is served.

6.1 Linear pricing strategy

In this section, the linear pricing strategy, which has been demonstrated to be of practical importance in the real world [169], is deployed in the derivation of optimal resource allocation schemes for maximizing SLA revenues. Consider a multiclass-supported network node with network bandwidth C bits/s, where m service classes are supported and the inbound packets are queued in a multi-queue system (each queue corresponds to one service class). Moreover, the delay of class i packets in the network node is denoted by d_i and the mean delay of class i packets during one measurement period is denoted by $\bar{d}_i$, $i \in [1, m]$. Then, when *mean packet delay* is chosen as the QoS metric, the linear pricing strategy for class i is characterized by the following definition of the linear pricing function.

Definition 1: *The function*

$$r_i(\bar{d}_i) = b_i - k_i \bar{d}_i, \quad i = 1, 2, ..., m, \ b_i > 0, \ k_i > 0 \tag{207}$$

is called the *linear pricing function* of class i under the QoS metric "*mean packet delay*", where b_i and k_i are positive constants and $b_i \geq b_j$ and $k_i \geq k_j$ should hold to ensure differentiated pricing if class i has higher priority than class j (in this book, we assume that class 1 is the highest priority and class m is the lowest one). With *packet delay* as the QoS metric, the linear pricing strategy of class i is characterized by the linear pricing function defined below.

Definition 2: *The function*

$$r_i(d_i) = b_i - k_i d_i, \quad i = 1, 2, ..., m, \ b_i > 0, \ k_i > 0 \tag{208}$$

is called the *linear pricing function* of class i under the QoS metric "*packet delay*", where b_i and k_i are positive constants and $b_i \geq b_j$ and $k_i \geq k_j$ should also hold to ensure differentiated pricing if class i has higher priority than class j.

6.2 Optimal resource allocation schemes for maximizing SLA revenues under linear pricing strategy

Consider the above multiclass-supported network node fed by m Poisson packet streams with arrival rates λ_1, λ_2, ..., λ_m, respectively. We assume that in this section the packet length distribution is exponential and use $\bar{L_i}$ to denote the mean packet length of class i in bits. Let the weight allotted to class i be w_i, i=1,2,...,m, which means that the reserved bandwidth for class i packets is $w_i C$ (bits / s). Without loss of generality, only non-empty queues are considered, and thus $w_i \neq 0$. Therefore, the natural constraint for the weights is $\sum_{i=1}^{m} w_i = 1, w_i \in (0,1]$. As class i is guaranteed to use a portion of the network resource $w_i C$ and the packets of class i arrive at queue i with rate λ_i, the analytic mean delay of class i packets in the network node (referred to as $\hat{d_i}$) can be denoted as:

$$\hat{d_i} = \frac{1}{\frac{w_i C}{\bar{L_i}} - \lambda_i} = \frac{\bar{L_i}}{w_i C - \lambda_i \bar{L_i}} \tag{209}$$

based on queueing theory. The natural constraint of Eq. (209) is $w_i C > \lambda_i \bar{L_i}$ due to the fact that delay can not be negative.

With the *mean packet delay* chosen as the QoS metric in a SLA, we use $\hat{d_i}$ to estimate the real mean delay of class i packets $\bar{d_i}$ in the network node during one measurement period. Thus, the SLA revenues F obtained in the network node during one measurement period is defined by the following equation under linear pricing strategy:

$$F = \sum_{i=1}^{m} r_i(\bar{d_i}) = \sum_{i=1}^{m} r_i(\hat{d_i}) = \sum_{i=1}^{m} (b_i - \frac{k_i \bar{L_i}}{w_i C - \lambda_i L_i}). \tag{210}$$

Based on the above definition, publication [179] derived the closed-form solution to the optimal resource allocation scheme (the optimal weight) for this case under the linear pricing strategy:

$$w_i = \frac{\sqrt{k_i \bar{L_i}}(C + \frac{\sum_{j=1}^{m} \sqrt{k_j \bar{L_j}}}{\sqrt{k_i \bar{L_i}}} \lambda_i \bar{L_i} - \sum_{j=1}^{m} \lambda_j \bar{L_j})}{C \sum_{j=1}^{m} \sqrt{k_j \bar{L_j}}}, \quad i = 1, 2, ..., m. \tag{211}$$

When *packet delay* is the QoS metric in a SLA, the real delay of class i packets d_i in the network node is also estimated by $\hat{d_i}$. As in this case a certain amount of revenue or penalty is attained as long as one inbound packet is served, we try to maximize the SLA revenues obtained in the network node per time unit (also denoted by F). Thus, in this case, F is defined as follows under linear pricing strategy:

$$F = \sum_{i=1}^{m} \lambda_i r_i(d_i) = \sum_{i=1}^{m} \lambda_i r_i(\hat{d_i}) = \sum_{i=1}^{m} \lambda_i (b_i - \frac{k_i \bar{L_i}}{w_i C - \lambda_i L_i}). \tag{212}$$

And the closed-form solution to the optimal resource allocation scheme for this case is presented in publications [180] and [178]:

$$w_i = \frac{\sqrt{\lambda_i k_i \bar{L_i}}(C + \frac{\sum_{j=1}^{m}\sqrt{\lambda_j k_j \bar{L_j}}}{\sqrt{\lambda_i k_i \bar{L_i}}}\lambda_i \bar{L_i} - \sum_{j=1}^{m}\lambda_j \bar{L_j})}{C\sum_{j=1}^{m}\sqrt{\lambda_j k_j \bar{L_j}}}, \quad i = 1, 2, ..., m. \tag{213}$$

Furthermore, publication [176] proposed the suboptimal resource allocation scheme for the case that all the supported service classes have their firm QoS (mean delay) requirements, which can satisfy those required QoS guarantees while still being able to achieve very high revenue close to the analytic maximum one under linear pricing strategy.

7 MAXIMIZING SLA REVENUES IN CLUSTER-BASED WEB SERVER SYSTEMS

The Web is changing from a sole communication and browsing infrastructure to an important medium for conducting personal business and e-commerce, which makes the Quality of Service (QoS) an increasingly critical issue. A fundamental characteristic of e-commerce environments is the diverse set of services provided to support the requirements of various businesses and customers, which result in the definition of different service classes. In a typical e-commerce environment, an e-business operator contracts with a Web service provider to provide applications and services to its business customers, which can be consumers (B2C) or other businesses (B2B); in other words, a Web service provider hosts an e-commerce Web site via a contract with the e-business operator. In many e-commerce contracts, the Web service provider agrees to offer a certain level of QoS to each class of service in the hosting of the e-commerce site, and in return the e-business operator agrees to pay the service provider based on the QoS levels received by its customers. These contracts are based on a SLA (Service-Level-Agreement) between the e-business operator and the Web service provider that defines the QoS metrics for each class of service, the anticipated workload intensity of per-class requests from the e-business and the pricing strategy by which the SLA payment will be determined.

The exponential growth in Internet usage, much of which is fueled by the growth and requirements of various aspects of e-commerce, has created the demand for more and faster Web servers capable of serving over 100 million Internet users. As mentioned in Chapter 2, server clustering has emerged as a promising technique to build faster, scalable and cost-effective Web servers [137] during recent years, which has made cluster-based Web server systems a major means to hosting e-commerce sites. A state-of-the-art cluster-based Web server system consist of a number of back-end server nodes and a specialized front-end node, which acts as the single input point of customer requests and is responsible for distributing the inbound requests among the back-end nodes. That is to say that the back-end server nodes of a cluster-based Web server system which hosts an e-commerce Web site are shared by the inbound requests of different service

classes.

In this Chapter, we analyze the problem of maximizing the revenue attained in the hosting of an e-commerce site with a SLA contract under a given amount of server resources by optimally partitioning the server resources among all the supported classes. Little work has been reported on this topic. The issue of maximizing SLA revenues in the cluster platform of Web server farms was recently studied by Liu *et al* in [93]. A Web server farm is typically deployed to host several Web sites simultaneously on the same platform. Moreover, they assumed that each back-end server node in a Web server farm can serve multiple service classes; they then tried to optimally allocate the resource (e.g., processing capacity) of each server node among its supported service classes to maximize the resulted SLA revenues. However, the closed-form solution to the optimal resource allocation scheme (i.e., the optimal service weights) in each back-end node was not derived in [93]. Diao *et al* [45] proposed a profit-oriented feedback control system for maximizing SLA profits in Web server systems, which automated the admission control decisions in a way that balances the loss of revenue due to rejected work against the penalties incurred if admitted work has excessive response times by a fuzzy control algorithm. Additionally, the issue of maximizing the expected value of a given cluster utility function by allocating server resources of a cluster-based Web server system dynamically was studied in [94], where the closed-form solution was also not derived.

7.1 Target Web cluster architecture for the hosting of an e-commerce Web site

The target Web cluster architecture consists of a front-end component called Web switch and a number of homogeneous back-end server nodes connected by a high-speed LAN. The Web switch acts as the network representative for an e-commerce Web site built upon the target cluster architecture, making the distributed nature of the site architecture completely transparent to its customers. In such a way, the authoritative DNS server for the e-commerce site translates the site name into the IP address of its Web switch, which receives all inbound requests destined for the site and then distribute them across the back-end nodes. Moreover, to enable QoS support in the e-commerce site, the Web switch must be able to examine the content of a HTTP request and identify its requested service class, i.e., it is the so-called layer-7 Web switch [137]. The above Web cluster architecture can be further classified on the basis of whether the data from the back-end server nodes to clients (outgoing data) go through the Web switch. In our target cluster architecture, the TCP handoff mechanism [113] is deployed to enable the back-end nodes respond to the clients directly without passing through the front-end nodes as an intermediary. Thus the target cluster architecture can be abstracted as a queuing system shown in Figure 80.

Suppose that an e-commerce Web site built upon the target cluster archi-

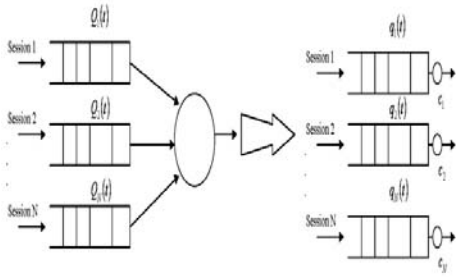


FIGURE 80 Queuing model of the target Web cluster architecture upon which an e-commerce Web site is built.

tecture consists of a layer-7 Web switch and N homogeneous back-end server nodes, each of which has the processing capacity C bits/s; there are a total of m service classes supported in the site. The idea of partitioning server resources among the supported service classes is to partition the N back-end server nodes into m disjoint server subsets so that each class of requests will be served only by its own server subset assigned. Specifically, the server subset assigned to class i is denoted by S_i and the number of server nodes in S_i is denoted by n_i, then $S_i \cap S_j = \oslash$, for $i \neq j$ and $i, j \in [1, m]$, and $\sum_{i=1}^{m} n_i = N$. Thus, the problem of deriving the optimal/suboptimal resource partitioning scheme is actually to find the optimal/suboptimal value of n_i, $i \in [1, m]$. Note that n_i does not have to be an integer, which means that a back-end server node may actually be assigned to multiple service classes with each class taking a portion of it. In this case, we have that back-end node serve those service classes by WFQ algorithm and the WFQ weights equal their shares in that node, respectively.

Based on the analysis in [175], the service time of a class i request at a back-end node is proportional to the size of its requested Web object, i.e., $X_i = L_i / C$, where L_i denotes the size (Bytes) of the Web object requested by class i customers and C (Bytes/s) is the processing capacity of the back-end node. Thus, $\bar{L}_i = E[L_i]$, $\bar{L_i}^2 = E[L_i^2]$ and $\bar{X}_i = E[X_i] = \bar{L}_i / C$, $\bar{X_i}^2 = E[X_i^2] = \bar{L_i}^2 / C^2$. In our scheme, the layer-7 Web switch distributes the inbound requests from class i uniformly among the back-end server nodes within server subset S_i to make the server loads balanced. In other words, if the overall arrival rate of class i requests to the e-commerce site is λ_i requests/s and a back-end server node in S_i is used exclusively by class i requests, the mean arrival rate of class i requests to that back-end node can be estimated as λ_i / n_i. Furthermore, in this book, the processing delay at the layer-7 switch is neglected due to the fact that in a Web environment the client-to-server packets are typically much less than the server-to-client packets and the chosen QoS metric in a SLA is the *mean request delay* in the e-commerce Web site. Hence, according to the queuing theory of M/G/1, the analytic mean

delay of class i requests $\hat{\bar{d}}_i$ in the e-commerce site can be calculated as follows.

$$\hat{\bar{d}}_i = \bar{X}_i + \frac{\frac{\lambda_i}{n_i}\bar{X_i}^2}{2(1-\frac{\lambda_i}{n_i}\bar{X}_i)} = \frac{\bar{L}_i}{C} + \frac{\lambda_i \bar{L_i}^2}{2C(n_iC-\lambda_i)L_i}. \tag{214}$$

The natural constraint of Eq. (214) is $n_iC > \lambda_i\bar{L}_i$ due to the fact that delay can not be negative.

7.2 Optimal resource partitioning scheme for the hosting of an e-commerce site under linear pricing strategy

Consider an e-commerce Web site built upon the target cluster architecture with N back-end server nodes and m service classes supported. The processing capacity of each back-end node is C bytes/s. Additionally, the arrival rate of class i requests is denoted by λ_i requests/s and the mean delay of class i requests in the e-commerce site denoted by $\bar{d}_i$. The linear pricing strategy for class i requests is characterized by the following definition of the linear pricing function $r_i(\bar{d}_i)$.

Definition 4: *The function*

$$r_i(\bar{d}_i) = b_i - k_i\bar{d}_i, \;\; i = 1,2,...,m, \;\; b_i > 0, \;\; k_i > 0 \tag{215}$$

is called the *linear pricing function* of class i, where $\bar{d}_i$ is the mean request delay of class i, b_i and k_i are both positive constants and $b_i \geq b_j$ and $k_i \geq k_j$ hold to ensure differentiated pricing if class i has a higher priority than class j.

As the QoS metric considered in the SLA is the *mean request delay*, the mean delay $\bar{d}_i$ of class i requests in the e-commerce site will be measured periodically and the SLA revenue due to serving class i requests can be determined also periodically based on class i pricing function and the above QoS (mean request delay) measurement. Specifically, we use Eq. (214) to estimate $\bar{d}_i$. Thus, the SLA revenue F obtained in the hosting of the e-commerce site during one measurement period is defined as follows under the linear pricing function in Eq. (215):

$$F = \sum_{i=1}^{m} r_i(\bar{d}_i) = \sum_{i=1}^{m}[b_i - k_i(\frac{\bar{L}_i}{C} + \frac{\lambda_i \bar{L_i}^2}{2C(n_iC-\lambda_iL_i)})]. \tag{216}$$

Moreover, the issue of maximizing SLA revenue in the hosting of an e-commerce Web site under linear pricing strategy can be formulated as follows:

$$\max \quad F = \sum_{i=1}^{m}[b_i - k_i(\frac{\bar{L}_i}{C} + \frac{\lambda_i \bar{L_i}^2}{2C(n_iC-\lambda_iL_i)})] \tag{217}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} n_i = N, \; 0 < n_i < N, \tag{218}$$

$$n_iC > \lambda_i\bar{L}_i. \tag{219}$$

By the Lagrangian optimization approach, we derived the following optimal resource partitioning scheme in publication [180], which can achieve the maximum SLA revenue F under linear pricing strategy when hosting an e-commerce site built upon the above target cluster architecture.

$$n_i = \frac{(CN - \sum_{j=1}^{m} \lambda_j \bar{L_j}) \sqrt{\frac{k_i \lambda_i \bar{L_i}^2}{2}}}{C \sum_{j=1}^{m} \sqrt{\frac{k_j \lambda_j \bar{L_j}^2}{2}}} + \frac{\lambda_i \bar{L_i}}{C}, \ i \in [1, m]. \tag{220}$$

Publication [180] made two sets of simulations to evaluate the effectiveness of the derived optimal resource partitioning scheme for maximizing the SLA revenue under linear pricing strategy, where the Bounded Pareto distribution ($BP(p, q, \alpha)$) [38] is used to model the heavy-tailed characteristic of Web objects. The simulation results there demonstrate that our derived optimal resource partitioning scheme can succeed in implementing the maximization of SLA revenues under a given amount of server resources and linear pricing strategy when a Web service provider hosts an e-commerce Web site by the above target Web cluster architecture.

7.2.1 Suboptimal resource partitioning scheme for the hosting of an e-commerce site under flat pricing strategy

The suboptimal resource partitioning scheme is proposed here for maximizing SLA revenue in the hosting of an e-commerce Web site under flat pricing strategy.

7.2.2 Flat pricing strategy for the hosting of an e-commerce site

As *mean request delay* is deployed as the QoS metric in the SLA, the definition of flat pricing strategy in this case is almost the same as the one defined in Chapter 5.2.1 except that $\bar{d_i}$ denotes the mean request delay of class i here. Specifically, consider an e-commerce Web site built upon the above target cluster architecture with N back-end server nodes and m service classes supported. The flat pricing strategy for class i is characterized by the following definition of flat pricing function.

Definition 5: *The function*

$$r_i(\bar{d_i}) = \begin{cases} R_i & \text{if } \bar{d_i} \leq D_i \\ -P_i & \text{if } \bar{d_i} > D_i \end{cases}, \quad i = 1, 2, ..., m \tag{221}$$

is called the *flat pricing function* of class i in the hosting of an e-commerce Web site, where D_i is the QoS (mean request delay) guarantee required by class i requests and R_i and P_i are both positive constants. The above flat pricing function specifies that if the real mean delay offered to class i requests is less than D_i during one

charging period, the Web service provider will receive a revenue R_i, otherwise a penalty P_i is incurred for failing to meet that D_i. Moreover, $R_i \geq R_j$ and $P_i \geq P_j$ should hold to ensure differentiated pricing if class i has higher priority than class j, which are expected under the SLA requirement.

7.2.3 Suboptimal resource partitioning scheme under the flat pricing strategy

Suppose that an e-commerce Web site built upon the above target cluster architecture consists of N homogeneous back-end server node, each of which has the processing capacity C bits/s, and supports a total of m service classes with Poisson arrival rate λ_1, λ_2, ..., λ_m, respectively. As the analytic mean delay $\hat{d}_i$ of class i requests in Eq. (214) can be used to estimate the real mean request delay $\bar{d}_i$, $\hat{d}_i$ has to be less than D_i so that the SLA revenue R_i can be obtained during one charging period for serving class i requests. The minimum number of back-end server nodes which has to be assigned to class i to meet its QoS guarantee D_i can be derived from the inequality $\hat{d}_i \leq D_i$. However, the real mean request delay $\bar{d}_i$ of class i will definitely differ by a small amount from $\hat{d}_i$ as shown in publication [180]. Hence, to guarantee $\bar{d}_i \leq D_i$ in the real situation, we may deploy a small constant parameter ϵ_i for class i and then construct the following inequality:

$$\hat{d}_i + \epsilon_i = \frac{\bar{L}_i}{C} + \frac{\lambda_i \bar{L_i^2}}{2C(n_i C - \lambda_i)L_i} + \epsilon_i \leq D_i. \tag{222}$$

By solving this inequality, the solution to n_i for a general distribution of Web object size can be calculated as follows:

$$n_i \geq \frac{\lambda_i \bar{L_i^2}}{2C[(D_i - \epsilon_i)C - L_i]} + \frac{\lambda_i \bar{L}_i}{C} \tag{223}$$

$$\text{under the constraint } \sum_{i=1}^{m} n_i \leq N,\ 0 < n_i \leq N. \tag{224}$$

Note that the parameter ϵ_i determines how well the inequality $\bar{d}_i \leq D_i$ will be guaranteed. We should set the value of ϵ_i carefully based on the system parameter settings and the burstiness of class i traffic. As a result of the above solution to n_i in (223), we present the suboptimal resource partitioning scheme for maximizing the SLA revenue in the hosting of an e-commerce site built upon the target cluster architecture under a given amount (N back-end server nodes here) of server resources and flat pricing strategy as follows:

1. Set $n_{i,min} = \frac{\lambda_i \bar{L_i^2}}{2C[(D_i - \epsilon_i)C - L_i]} + \frac{\lambda_i \bar{L}_i}{C}$, i=1,2,...,m,
2. If $\sum_{i=1}^{m} n_{i,min} \leq N$, then the above $n_{i,min}$, i=1,2,...,m, is the suboptimal resource partitioning scheme in this case,
3. Otherwise, it means that the N back-end server nodes are not enough to guarantee that $\bar{d}_i \leq D_i, i \in [1, m]$ for all the supported service classes.

Hence, we should first satisfy the QoS guarantees of a set of selected service classes so that the obtained SLA revenue (actually revenue plus penalty) is the highest in this situation. The remaining server resources are allocated to all the supported service classes other than those selected ones uniformly.

Note that the above set of selected service classes in Step 3 is acquired by the comparison of the resulted SLA revenues under all possible resource partitioning schemes, which makes its calculation complexity increases quickly with larger values of m. Hence, instead we may first assign $n_{1,min}$ back-end server nodes to class 1, $n_{2,min}$ nodes to class 2,..., and $n_{j,min}$ nodes to class j ($j \in [1, m-1]$) until $\sum_{i=1}^{j+1} n_{i,min} > N$. The derivation of suboptimal resource partitioning scheme is illustrated in the next section.

7.2.4 Simulation results

Here some simulation results are presented to illustrate the effectiveness of our above approach by which the suboptimal resource partitioning scheme can be derived for the hosting of an e-commerce Web site under flat pricing strategy. Throughout this section, we study an e-commerce site built upon the target Web cluster architecture which consists of a layer-7 Web switch and 16 homogeneous back-end server nodes (N=16) that support three service classes (m=3, namely, Gold, Silver and Bronze classes). Moreover, each back-end server node has the processing capacity of C=5.95MB/s and the parameters of the three flat pricing functions for Gold, Silver and Bronze classes are summarized below: D_1=18ms, R_1=10 money units, $P1$=15 money units, D_2=25ms, R_2=5 money units, $P2$ =8 money units, and D_3=45ms, R_3=2 money units, $P3$=4 money units. Furthermore, ϵ_1=3ms, ϵ_2=4ms and ϵ_3=6ms are set for Gold, Silver and Bronze classes, respectively.

For actual Web workloads, it is recognized that Web object sizes are distributed with a heavy tail. Here the Bounded Pareto distribution ($BP(p, q, \alpha)$) [38] is used to model the heavy-tailed characteristic of Web objects. Specifically, the mean size of Web objects is set to 21KB as measured in [5] with p=1KB and q=10MB are chosen as the reasonable minimum and maximum Web object size, respectively. The resulting α=0.8037 is within the range of α values measured in [4] and [37]. The arrival process of client requests destined for the e-commerce site was modelled by Poisson distribution. Additionally, for each of the following simulations, we first derive the suboptimal resource partitioning scheme by our above approach and then deploy it as well as another proportional resource partitioning scheme in the simulation for comparison. Specifically, the *proportional* resource partitioning scheme assigns the following number of back-end server nodes to class i requests: $n_{i,proportional} = \lambda_i \bar{L}_i / \sum_{j=1}^{m} (\lambda_j \bar{L}_j), i \in [1, m]$.

In the first simulation, λ_1=100 requests/s, λ_2=150 requests/s and λ_3=250 requests/s. Thus, we see that $n_{1,min}$=5.3899, $n_{2,min}$=5.4919 and $n_{3,min}$=4.9559 by Eq. (223). As $\sum_{i=1}^{3} n_{i,min}$=15.8377 $< N$=16, the e-commerce site has enough server resources to satisfy the QoS guarantees of Gold, Silver and Bronze classes and the above set of $n_{1,min}$, $n_{2,min}$ and $n_{3,min}$ is the suboptimal resource parti-

TABLE 16 For the first simulation: mean request delay in the e-commerce Web site.

	Mean request delay of Gold class ($\bar{d}_1$)	Mean request delay of Silver class ($\bar{d}_2$)	Mean request delay of Bronze class ($\bar{d}_3$)
By the suboptimal scheme	16.2579 ms	22.4538 ms	36.1440 ms
By the proportional scheme	25.7174 ms	24.1147 ms	23.8124 ms

tioning scheme in this case. Moreover, because the remaining server resources ($N-\sum_{i=1}^{3} n_{i,min}$=0.1623) is less than 1 and in a back-end server node which will serve multiple service classes, the requests from the multiple classes share the back-end node by WFQ algorithm, we allotted the remaining server resources to Gold class requests in the simulation. Hence, the deployed suboptimal resource partitioning scheme is as follows: $n_{1,suboptimal} = 5.5522$, $n_{2,suboptimal} = 5.4919$ and $n_{3,suboptimal} = 4.9559$. The simulation results of the mean request delays in the e-commerce site are presented in Table 16.

It can be seen from Table 16 that $\bar{d}_1 \leq D_1 = 18ms$, $\bar{d}_2 \leq D_2 = 25ms$ and $\bar{d}_3 \leq D_3 = 45ms$ are all satisfied by the derived suboptimal resource partitioning scheme. Hence, the suboptimal resource partitioning scheme achieves the maximum value $\sum_{i=1}^{3} R_i$=17 money units of the SLA revenue during one charging period (400s here) for the hosting of the e-commerce site under the flat pricing strategy. Whereas, the proportional resource partitioning scheme can not have $\bar{d}_1 \leq D_1$ hold, which results in $-P_1 + R_2 + R_3 = -8$ money units of the SLA revenue during the same charging period. In other words, the service provider will get the loss of 8 money units due to failing to satisfy the QoS guarantee of Gold class requests when using the proportional scheme.

Next, the workload intensity with λ_1=120 requests/s, λ_2=180 requests/s and λ_3=300 requests/s is fed into the e-commerce Web site in the second simulation, which leads to $n_{1,min}$=6.4679, $n_{2,min}$=6.5903 and $n_{3,min}$=5.9471 by Eq. (223). As $\sum_{i=1}^{3} n_{i,min} = 19.0053 > N$=16, it means that the e-commerce site does not have enough server resources available to satisfy the QoS guarantees of all supported service classes (Gold, Silver and Bronze classes here). However, it is noticed that $n_{1,min} + n_{2,min} < 16$ holds in this case. Hence, we can derive the following suboptimal resource partitioning scheme by first guaranteeing the satisfaction of QoS requirements of both Gold and Silver classes: $n_{1,suboptimal}$=$n_{1,min}$=6.4679, $n_{2,suboptimal}$=$n_{2,min}$=6.5903 and $n_{3,suboptimal}$=$N - n_{1,min} - n_{2,min}$=2.9418. The simulation results for this case are presented in Table 17.

Table 17 shows that $\bar{d}_1 \leq D_1 = 18ms$ and $\bar{d}_2 \leq D_2 = 25ms$ both hold although $\bar{d}_3 > D_3 = 45ms$ when the above suboptimal resource partitioning scheme is deployed. Hence, the suboptimal scheme can achieve the highest revenue in this case, i.e., $R_1 + R_2 - P3$=11 money units. Whereas, although the proportional resource partitioning scheme satisfies the QoS guarantee of Bronze class ($\bar{d}_3 < 45$ ms), it sacrifices the QoS performances of Gold and Silver classes (i.e., $\bar{d}_1 > 18$ ms and $\bar{d}_2 > 25$ ms). Thus, the Web service provider will get the loss

TABLE 17 For the second simulation: mean request delay in the e-commerce Web site.

	Mean request delay of Gold class ($\bar{d}_1$)	Mean request delay of Silver class ($\bar{d}_2$)	Mean request delay of Bronze class ($\bar{d}_3$)
By the suboptimal scheme	16.7959 ms	21.8557 ms	90.8417 ms
By the proportional scheme	26.3930 ms	27.4331 ms	30.4962 ms

of $P_1 + P_2 - R_3$=21 money units during each charging period for the hosting of the e-commerce site by the proportional resource partitioning scheme. Therefore, based on the above simulation results, it can be concluded that the suboptimal resource partitioning scheme derived by our proposed approach is able to achieve the highest SLA revenue for the hosting of an e-commerce site under flat pricing strategy.

7.3 Summary

Cluster-based Web server systems have become a major means to hosting e-commerce sites. In this Chapter, we linked the issue of resource partitioning scheme with the pricing strategy in a Service-Level-Agreement (SLA) and analyzed the problem of maximizing the revenues obtained in the hosting of an e-commerce site with a SLA contract by optimally partitioning the server resources among all supported service classes. In publication [180], the optimal resource partitioning scheme is derived under a given amount of server resources and linear pricing strategy when the QoS metric in the SLA is *mean request delay*, which has the closed-form solution to the optimal number of the back-end server nodes assigned to each service class. Moreover, the closed-form solution can apply to any general distribution of requested Web object size. Finally, the suboptimal resource partitioning scheme is proposed based on the analysis of the target cluster architecture when the *mean request delay* is chosen as the QoS metric in the SLA, which can achieve the highest SLA revenue obtained for the hosting an e-commerce site under a given amount of back-end server nodes and flat pricing strategy. The suboptimal resource partitioning scheme can also apply to any general size distribution of requested Web object.

8 CONCLUSIONS

The presence of services, such as VoIP and VoD, create a strong impetus to introduce service differentiation in modern networks. In such a framework, a provider has to use appropriate mechanisms to share the limited bandwidth resources. Though the existent scheduling mechanisms enable a provider to allocate resources between the service classes, they fail to react appropriately to the varying number of active flows and, as a result, to the varying bandwidth requirements. Thus, it is anticipated that the static configuration of routers will be replaced with more sophisticated algorithms. Though the static configuration can ensure that the certain class will obtain a better treatment, it is not capable of ensuring that flows within that class will be provided with the desired QoS guarantees.

The models presented in this book are capable of translating individual flow bandwidth and delay requirements into the configuration for the given scheduler so that all the QoS guarantees are ensured. The QoS constraints of Adaptive models specify the set of feasible configurations for the scheduler. If there is no feasible configuration, then a router does not have enough resources. By this, the models perform the admission control. Finally, by using the optimization criteria, such as the price for network services, the models choose a configuration that allocates free resources in such a way that the given criterion is maximized. By this, the proposed models can help a provider to remove the gap between the high-level SLA information and the low-level configuration of a particular scheduler. The input parameters of the models relate to the SLA and its technical level part, SLS. The output of the models is the weight values for the given scheduler type.

The simulation results have confirmed the correctness of the presented adaptive models. They are capable of ensuring the QoS guarantees of several service classes with various requirements and pricing schemes. At the same time, free resources are allotted in such a way that the total revenue is maximized. As considered theoretically and presented in the simulation results, the models can be applied easily to the DiffServ and IntServ QoS frameworks. Depending on the link speed and the available network equipment, a provider can choose the model for the particular underlying scheduler.

Since Adaptive models depend neither on a concrete signalling protocol,

nor on specific traffic characteristics, they can be used almost in any QoS framework. In particular, Adaptive model based on the WRR scheduler can be used in the ATM networks that have cells of a fixed size. It also worth mentioning that the adaptive models can also be adopted for sharing the resources on the link layer. For instance, they can be used at the hybrid coordinator of the 802.11e WLAN networks. Since the hybrid coordinator has to allocate resources between the WLAN nodes in such a way, that all the QoS requirements are satisfied within the contention-free period, Adaptive models can calculate the optimal distribution of transmission opportunities. In the same way, Adaptive models can calculate the number of slots within the 802.16 WiMax frame to ensure the QoS requirements of each station.

REFERENCES

[1] W. Almesberger, S. Giordano, R. Mameli, S. Salsano, and F. Salvatore. A prototype implementation for the IntServ operation over DiffServ networks. In *IEEE Globecomm*, volume 1, pages 439–444, Nov / Dec 2000.

[2] P. Almquist. Type of service in the internet protocol suite. IETF RFC 1349, July 1992.

[3] J. Altmann, B. Rupp, and P. Varaiya. Internet user reactions to usage-based pricing. In *Internet Economics Workshop (IEW'99)*, May 1999.

[4] M. F. Arlitt and T. Jin. A workload characterization study of the 1998 World Cup Web site. IEEE Network, 14(3):30-37, May / June 2000.

[5] M. Arlitt and C. Williamson. Web server workload characterization: the search for invariants. In Proceedings of ACM SIGMETRICS, pp. 126-137, 1996.

[6] G. Armitage, B. Carpenter, A. Casati, J. Crowcroft, J. Halpern, B. Kumar, and J. Schnizlein. A delay bound alternative revision of RFC 2598. IETF RFC 3248, Mar 2002.

[7] F. Baker, C. Iturralde, F. Le Faucher, and B. Davie. Aggregation of RSVP for IPv4 and IPv6 reservations. IETF RFC 3175, Sep 2001.

[8] J. C. R. Bennett and H. Zhang. W^2FQ: Worst-case fair weighted fair queueing. In *INFOCOM*, pages 120–128, Mar 1996.

[9] J. C. R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. *IEEE/ACM Transactions on Networking*, 5(5):675–689, Oct 1997.

[10] Y. Bernet. The complementary roles of RSVP and Differentiated Services in the full-service QoS network. *IEEE Communications*, 38:154–162, Feb 2000.

[11] Y. Bernet. Format of the RSVP DCLASS object. IETF RFC 2996, Nov 2000.

[12] Y. Bernet, S. Blake, D. Grossman, and A. Smith. An informal management model for Diffserv routers. IETF RFC 3290, May 2002.

[13] Y. Bernet, P. Ford, R. Yavatkar, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, and E. Felstaine. A framework for Integrated Services operation over DiffServ networks. IETF RFC 2998, Nov 2000.

[14] Y. Bernet, A. Smith, and B. Davie. Specfication of the NULL service type. IETF RFC 2997, Nov 2000.

[15] A. Bianco, J. Finochietto, G. Galante, M. Meillia, and F. Neri. Open-source PC-based software routers: a viable approach to high-performance packet switching. In *3rd International Workshop on QoS in multiservice IP networks (QoSIP-2005)*, pages 351–366, Feb 2005.

[16] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. IETF RFC 2475, Dec 1998.

[17] R. Bless, K. Nichols, and K. Wherle. A lower effort per-domain behaviour (PDB) for differentiated services. IETF RFC 3662, Dec 2003.

[18] R. Bless and K. Wherle. A limited effort per-hop behavior. INTERNET DRAFT (expired), Feb 2001.

[19] V. Bollapragada, C. Murphy, and R. White. *Inside Cisco IOS software architecture*. Cisco Press, 1st edition, 2001.

[20] C. Bouras and A. Sevasti. SLA-based QoS pricing in DiffServ networks. *Computer Communications*, 27(18):1868–1880, Dec 2004.

[21] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. IETF RFC 1633, Jun 1994.

[22] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jasmin. Resource reservation protocol (RSVP) – version 1 functional specification. IETF RFC 2205, Sep 1997.

[23] P. Brady. A techniques for investigating on-off patterns in speech. *Bell Systems Technical Journal*, 44:1–22, Jan 1965.

[24] B. Braden et al. Recommendations on queue management and congestion avoidance in the internet. IETF RFC 2309, Apr 1998.

[25] L. Breslau, D. Estrin, K. Fall, S. Floyd, J.Heidemann, A.Helmy, P. Huang, S. McCanne, K. Varadhan, Y.Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[26] S. Brim, B. Carpenter, and F. Le Faucheur. Per hop behavior identification codes. IETF RFC 2836, May 2000.

[27] B. Carpenter and K. Nichols. Differentiated services in the internet. *IEEE Network*, 90(9):1479–1494, Sep 2002.

[28] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction and applicability statements for internet standard management framework. IETF RFC 3410, Dec 2002.

[29] S. Casner and V. Jacobson. Compressing IP/UDP/RTP headers for low-speed serial links. IETF RFC 2508, Feb 1999.

[30] K. Chan, R. Sahita, S. Hahn, and K. McCloghrie. Differentiated services Quality of Service policy information base. IETF RFC 3317, Mar 2003.

[31] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. COPS usage for policy provisioning (COPS-PR). IETF RFC 3084, Mar 2001.

[32] A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, A. Chiu, W. Courtney, S. Davari, V. Firoiu, C. Kalmanek, and K.K. Ramakrishnan. Supplemental information for the new definition of the EF PHB (expedited forwarding per-hop behavior). IETF RFC 3247, Mar 2002.

[33] F. M. Chiussi and A. Francini, Implementing Fair Queueing in ATM switches - Paart 1: A Practical Methodology for the Analysis of Delay Bounds, In Proc. of GLOBECOM97, Vol. 1, *pp. 509-518*, November 1997.

[34] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Transactions on Networking*, 6(4):362–373, Aug 1998.

[35] R. Cocchi, S. Shenker, D. Estrin, and L. Zhang. Pricing in computer networks: motivation, formulation, and example. *IEEE/ACM Transaction on Networking*, 1(6):614–627, Dec 1993.

[36] C. Courcoubetis, F. P. Kelly, and R. Weber: Measurement based usage charges in communication networks, Oper. Res., Vol.48, No.4, pp. 535–548, 2000.

[37] M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. IEEE / ACM Trans. on Networking, 5(6):835-846, Dec. 1997.

[38] M. Crovella, M. Harchol-Balter, and C. Murta. Task assignment in a distributed system: improving performance by unbalancing load. In Performance Evaluation Review, Vol. 26, No. 1, pp. 268-269, 1998.

[39] R. Cruz. A calculus for network delay, Part I: Network elements in isolation. *IEEE Transaction on Information Theory*, 37(1):114–131, Jan 1991.

[40] B. Davis, A. Charny, J.C.R. Bennett, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, and D. Stiliadis. An expedited forwarding PHB (per-hop behavior). IETF RFC 3246, Mar 2002.

[41] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. IETF RFC 2460, Dec 1998.

[42] A. Deitti, M. Listanti, S. Salsano, and L. Veltri. Supporting RSVP in a differentiated service domain: an architectural framework and a scalability analysis. In *IEEE International Conference on Communication*, volume 1, pages 204–210, Jun 1999.

[43] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *Journal of Internetworking: Research and Experience*, pages 3–26, Sep 1990.

[44] A. Demers, S. Keshav and S. Shenker, Analysis and Simulation of a Fair Queuing Algorithm, Journal of Internetworking Research and Experience, pp. 3-26, October 1990.

[45] Y. Diao, J. L. Hellerstein and S. Parekh. Using fuzzy control to maximize profits in service level management. In IBM Systems Journal, Vol. 41, No. 3, pp. 403-420, 2002.

[46] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry. The COPS (common open policy service) protocol. IETF RFC 2748, Jan 2000.

[47] W. Fang, N. Seddigh, and B. Nandy. A time sliding window three colour marker (TSWTCM). IETF RFC 2859, June 2000.

[48] D. Ferrari and D. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communcation*, 8(3):368–379, Apr 1990.

[49] T. Ferrari and P. Chimento. A measurement-based analysis of expedited forwarding PHB mechanisms. In *IWQoS'00, Pittsburgh*, June 2000.

[50] R. Fletcher. *Practical Methods of Optimization*. Wiley, 1987.

[51] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993.

[52] S. Floyd and V. Jacobson. Link-sharing and resource management models for packet networks. *IEEE/ACM Transactions on Networking*, 3(4):365–386, 1995.

[53] E. Fulp, M. Ott, D. Reininger, and D. Reeves. Paying for QoS: an optimal distributed algorithm for pricing network resources. In *Sixth International Workshop on Quality of Service (IWQoS'98)*, pages 75–84, 1998.

[54] R. J. Gibbens and F. P. Kelly, Resource pricing and the evolution of congestion control, Automatica, Vol.35, No.12, pp. 1969–1985, 1999.

[55] S. J. Golestani. Congestion-free transmission of real-time traffic in packet networks. In *INFOCOM*, pages 527–542, Jun 1990.

[56] S. J. Golestani. A self-clocked fair queueing scheme for broadband applications. In *INFOCOM*, pages 636–646, Jun 1994.

[57] S. J. Golestani. Network delay analysis of a class of fair queueing algorithms. *IEEE Journal on Selected Areas in Communcations*, 13(6):1057–1070, Aug 1995.

[58] S. J. Golestani, A Self-Clocked Fair Queueing Scheme for Broadband Applications, In Proc. *of INFOCOM94,* pp. 636-646, April 1994.

[59] S. J. Golestani, Network Delay Analysis of a Class of Fair Queueing Algorithms, *IEEE Journal on Selected Areas in Communications, Vol. 13, No. 6,* pp. 1057-1070, August 1995.

[60] P. Goyal, H. Chen, and H. Vin. Start-time fair queueing: a scheduling algorithm for integrated servieces packet switching networks. In *SIGCOMM* , pages 157–168, Aug 1996.

[61] M. Goyal, A. Duressi, P. Misra, C. Liu, and R. Jain. Effect of number of drop precedences in assured forwarding. In *Proceedings of GLOBECOMM, Vol. 1A*, pages 188–193, Dec 1999.

[62] P. Goyal, S. S. Lam and H. M. Vin, Determining End-to-End Delay Bounds in Heterogeneous Networks, In *Proc. of 5th International Workshop on Network and Operating System Support for Digital Audio and Video,* pp.287-298, April 1995.

[63] G. Gross, D. Rawlins, H. Sinnreich, and S. Thomas. COPS usage for SIP. INTERNET DRAFT (expired), Jun 2001.

[64] D. Grossman. New terminology and clarifications for DiffServ. IETF RFC 3260, Apr 2002.

[65] R. Guérin, S. Kamat, V. Peris, and R. Rajan. Scalable QoS provision through buffer management. In *SIGCOMM, pages 29–40, Aug 1998.*

[66] R. Guérin, L. Li, S. Nadas, P. Pan, and V. Peris. The cost of QoS support in edge devices an experimental study. In 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 99)*, volume 2, pages 873–882, Mar 1999.*

[67] R. Guérin and V. Peris. Quality-of-service in packet networks: basic mechanisms and directions. Computer Networks*, 31(3):169–189, Feb 1999.*

[68] M. Hawa and D. W. Petr, M/G/FQ: Stochastic Analysis of Fair Queueing Systems, In Proc. of IEEE 2nd International Conference on Networking*, pp.368-381, Aug. 2002.*

[69] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. IETF RFC 2597, Jun 1999.

[70] J. Heinanen and R. Guérin. A single rate three color marker. IETF RFC 2697, Sep 1999.

[71] J. Heinanen and R. Guérin. A two rate three color marker. IETF RFC 2698, Sep 1999.

[72] *S. Herzog. RSVP extensions for policy control. IETF RFC 2750, Jan 2000.*

[73] *S. Herzog, J. Boyle, R. Cohen, D. Durham, R. Rajan, and A. Satry. COPS usage for RSVP. IETF RFC 2749, Jan 2000.*

[74] *M-F. Horng, W-T. Lee, K-R. Lee, and Y-H. Kuo. An adaptive approach to Weighted Fair Queue with QoS enhanced on IP network. In* IEEE Region 10 International Conference on Electrical and Electronic Technology, *volume 1, pages 181–186, Aug 2001.*

[75] *G. Huston. Next steps for the IP QoS architecture. IETF RFC 2990, Nov 2000.*

[76] *Y. Ito, S. Tasaka, and Y. Ishibashi. Variably weighted round robin for core IP networks.* Performance, Computing, and Communications Conference, *pages 159–166, 2002.*

[77] *ITU-T G.711, Pulse code modulation (PCM) of voice frequencies. ITU-T recommendation G.711, 1988.*

[78] *V. Jacobson, K. Nichols, and K. Poduri. An expedited forwarding PHB. IETF RFC 2598, Jun 1999.*

[79] *V. Jacobson, K. Nichols, and K. Poduri. The 'virtual wire' per-domain behavior. INTERNET DRAFT (expired), Jul 2000.*

[80] *J. Joutsensalo, T. Hämäläinen, and J. Zhang. Revenue maximization-based adaptive WFQ. In* Proceedings of APOC 2002*, pages 108–117, Oct 2002.*

[81] *J. Joutsensalo, T. Hämäläinen, K. Luostarinen, and J. Siltanen: Adaptive scheduling method for maximizing revenue in flat pricing scenario.* International Journal of Electronics and Communications 60 (Feb. 2006).

[82] *J. Joutsensalo, T. Hämäläinen, M. Pääkkönen, and A. Sayenko: QoS- and revenue aware adaptive scheduling algorithm.* Journal of Communications and Networks 6 (March 2004), *68–77.*

[83] *S. S. Kanhere and H. Sethu. On the latency bound of deficit round robin. In* 11th International Conference on Computer Communications and Networks, *pages 548 – 553, Oct 2002.*

[84] *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, *9(8):1265 –1279, Oct 1991.*

[85] *F. P. Kelly, Notes on effective bandwidths, in Stochastic Networks: Theory and Applications, S. Zachary, I. B. Ziedins, and F. P. Kelly, Eds. London, U.K.: Oxford Univ. Press, Vol.9, pp. 141–168, 1996.*

[86] *F. P. Kelly, On tariffs, policing and admission control for multiservice networks, Oper. Res. Lett., Vol. 15, pp. 1–9, 1994.*

[87] *F. P. Kelly, Charging and rate control for elastic traffic, European Transaction on Telecommunication, Vol.8, pp. 33–37, 1997.*

[88] *F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, Rate control in communication networks: Shadow prices, proportional fairness and stability, Journal of the Operational Research Society, Vol.49, pp. 237–252, 1998.*

[89] *L. Kleinrock.* Queueing systems. *New York: John Wiley & Sons, 1975.*

[90] *E. Knightly and H. Zhang. Traffic characterization and switch utilization using determenistic bounding interval dependent traffic models. In* INFOCOM, *Apr 1995.*

[91] *R. J. La and V. Anantharam, Utility-based Rate Control in the Internet for Elastic Traffic,* IEEE / ACM Transactions on Networking, Vol.10, Issue: 2, *pp. 272–286, April 2002.*

[92] *H. Li, C. Huang, M. Devetsikiotis, and G. Damm. Extending the concept of effective bandwidths to DiffServ networks. In* Electrical and Computer Engineering, *volume 3, pages 1669–1672, May 2004.*

[93] *Z. Liu, M. Squillante, and J.Wolf. On Maximizing Service Level Agreement Profits. In* Proceedings of the 3rd ACM conference on Electronic Commerce, *pp. 213C223, 2001.*

[94] *R. Levy, J. Nagarajao, G. Pacifici, M. Spreitzer, A. Tantawi, and A. Youssef. Performance management For Cluster BasedWeb Services. In* Proceedings of 8th IFIP / IEEE International Symposium on Integreted Network Management, *Mar. 24-28, 2003.*

[95] *LP_SOLVE. http://www.geocities.com/lpsolve/.*

[96] *J.K. Mackie Mason, H.R. Varian, Pricing the Internet: Public Access to the Internet, Prentice-Hall, 1995.*

[97] *O. Maennel and A. Feldmann. Realistic BGP traffic for test labs. In* ACM SIGCOMM, *pages 31–44, Aug 2002.*

[98] *Magaña, E.; Morató, D.; Varaiya, P.: Router scheduling configuration based on the maximization of benefit and carried best effort traffic. Telecommunication Systems 23 (October- December 2003), 275292.*

[99] *E. Magaña, D. Morató, and P. Varaiya. Router scheduling configuration based on the maximization of benefit and carried best effort traffic.* Telecommunication systems, *24(2-4):275–292, Oct/Dec 2003.*

[100] *B. Maglaris, D. Anastassiou, P. Sen, G. Karlsson, and J. Robins. Performance models of statistical multiplexing in packet video communications.* IEEE Transactions on Communication, *36(7):834–844, Jul 1988.*

[101] *R. Makkar, I. Lambadaris, J.H. Salim, N. Seddigh, B. Nandy, and J. Babiarz. Empirical study of buffer management scheme for DiffServ assured forwarding PHB. In* IEEE 9th International Conference on Computer Communications and Networks*, pages 632–637, Nov 2000.*

[102] *M. Mameli and S. Salsano. Use of COPS for IntServ operations over DiffServ: architectural issues, protocol design, and test-bed implementation. In* IEEE International Conference on Communication*, pages 3265 – 3270, Jun 2001.*

[103] *J. Manner and X. Fu. Analysis of existing quality-of-service signaling protocols. IETF RFC 4094, May 2005.*

[104] *P.E. McKenny. Stochastic fairness queueing. In* 9th Annual Joint Conference of the IEEE Computer and Communication Societies*, volume 2, pages 733–740, Jun 1990.*

[105] *C. Mills, D. Hirsh, and G. Ruth. Internet accounting: background. IETF RFC 1272, Nov 1991.*

[106] *B. Moore. Policy core information model (PCIM) extensions. IETF RFC 3460, Jan 2003.*

[107] *B. Moore, E. Elleson, E. Strassner, and A. Westerinen. Policy core information model – version 1 specification. IETF RFC 3060, Feb 2001.*

[108] *K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services field (DS field) in the IPv4 and IPv6 headers. IETF RFC 2474, Dec 1998.*

[109] *K. Nichols and B. Carpenter. Definition of Differentiated Services per domain behaviors and rules and their specifications. IETF RFC 3086, April 2001.*

[110] *K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. IETF RFC 2638, July 1999.*

[111] *L. Ong and J. Yoakum. An introduction to the stream control transmission protocol (SCTP). IETF RFC 3286, May 2002.*

[112] *J.K. Ousterhout.* Tcl and the Tk Toolkit*. Addison-Wesley, 1994.*

[113] *V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel,W. Zwaenepoel, and E. Nahum. Locality-Aware Request Distribution in Cluster-based Network Servers. In* Proceedings of the 8th Conference on Architectural Support for Programming Languages and Operating Systems*, San Jose, CA, Oct. 1998.*

[114] *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*, 1(3):344–357, Jun 1993.*

[115] *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 Trans. on Networking*, Vol.2, No.2, pp. 137-150, April 1994.*

[116] *I. Ch. Paschalidis, Class-specific quality of service guarantees in multimedia communication networks,* Automatica, Vol.35, No.12*, pp. 1951-1968, 1999.*

[117] *I. Ch. Paschalidis and J. N. Tsitsiklis, Congestion-dependent pricing of network services,* IEEE / ACMTransactions on Networking, Vol.8*, pp. 171–184, April 2000.*

[118] *I. Ch. Paschalidis and N. Tsitsiklis. Congestion-dependent pricing of network services.* IEEE ACM Transactions on Networking, 8(2)*, pp. 171–184, April 2000.*

[119] *I. Ch. Paschalidis and Yong Liu, Pricing in multiservice loss networks: static pricing, asymptotic optimality and demand subsitution effects,* IEEE / ACM Transactions on Networking, Vol.10(3)*, pp. 425–438, June 2002.*

[120] *V. Paxson and S. Floyd. Wide area traffic: the failure of poisson modelling.* IEEE Transactions on Networking*, 3:226–244, Jun 1995.*

[121] *N. Pekergin, Stochastic Bounds on Delays of Fair Queueing Algorithms, In* Proceedings of INFOCOM99*, pp. 1212-1219, 1999.*

[122] *A. Pereira and E. Monteiro. Bandwidth management in IntServ to DiffServ mapping. In* 3rd International Workshop on QoS in multiservice IP networks (QoSIP-2005)*, pages 433–444, Feb 2005.*

[123] *K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. IETF RFC 3168, Sep 2001.*

[124] *P. Reichl, S. Leinen, and B. Stiller. A practical review of pricing and cost recovery for internet services. In* Internet Economics Workshop (IEW'99)*, May 1999.*

[125] *Requirements for signaling protocols. IETF RFC 3726, Apr 2004.*

[126] *A. Romanow and S. Floyd. Dynamics of TCP traffic over ATM networks.* IEEE Journal on Selected Areas in Communication*, 13(4):633–641, May 1995.*

[127] *E. Rosen, A. Viswanathan, and R. Callon and. Multiprotocol label switching architecture. IETF RFC 3031, Jan 2001.*

[128] *J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session initiation protocol. IETF RFC 3261, Jun 2002.*

[129] *S. Salsano and L. Veltri. QoS control by means of COPS to support SIP-based applications.* IEEE Network, 16(2) *pp. 27–33, Mar/Apr 2002.*

[130] *A. Sayenko, T. Hämäläinen, J. Siltanen, and J. Joutsensalo. An adaptive approach for Weighted Fair Queueing with revenue as the optimization criterion.* In The 8th IEEE Symposium on Computers and Communications (ISCC 2003)*, pages 181–186, Jul 2003.*

[131] *A. Sayenko, T. Hämäläinen, J. Siltanen, and J. Joutsensalo. The simulation and analysis of the revenue criterion based adaptive WFQ. In* 18th International Teletraffic Congress (ITC 18), volume 5b*, pages 1071–1080, Sep 2003.*

[132] *A. Sayenko, T. Hämäläinen, J. Siltanen, and J. Joutsensalo. On providing bandwidth and delay guarantees using the revenue criterion based adaptive WFQ. In* The 9th Asia Pacific Conference on Communication (APCC 2003), volume 2*, pages 745–750, Sep 2003.*

[133] *A. Sayenko, T.Hämäläinen, J. Joutsensalo, and P. Raatikainen. Adaptive scheduling using the revenue-basedWeighted Round Robin. In* The 12th IEEE International Conference On Networks (ICON 2004)*, volume 2, pages 743–749, Nov 2004.*

[134] *A. Sayenko, T. Hämäläinen, and J. Joutsensalo. Extension of the QoS policy information model to support adaptive queueing techniques. In* The 12th IEEE International Conference On Networks (ICON 2004), Workshop on Coordinated QoS in Distributed Systems (COQODS), volume 1*, pages 375-380, Nov 2004.*

[135] *A. Sayenko, T.Hämäläinen, J. Joutsensalo, and P. Raatikainen. Revenue based adaptive Deficit Round Robin. In* 3rd International Workshop on QoS in multiservice IP networks*, pages 600–612, Feb 2005.*

[136] *A. Sayenko, T. Hämäläinen, J. Joutsensalo, L. Kannisto. Comparison and analysis of the revenue-based adaptive queuing models. In* Journal of Computer Networks (Special Issue on Quality of Service)*, 2005.*

[137] *T. Schroeder, S. Goddard, and B. Ramamurthy. ScalableWeb server clustering technologies. In* IEEE Network, 14(3)*, pp.38-45, May/June 2000.*

[138] *H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. IETF RFC 1889, Jan 1996.*

[139] *H. Schulzrinne and R. Hancock. GIST: General Internet signalling transport. INETRNET DRAFT, Sep 2005. draft-ietf-nsis-ntlp-08.*

[140] *N. Seddigh, B. Nandy, and J. Heinanen. An assured rate per-domain behaviour for Differentiated Services. INTERNET DRAFT (expired), Jul 2001.*

[141] *N. Seddigh, B. Nandy, P. Pieda, J. Hadi Salim, and A. Chapman. An experimental study of assured services in a DiffServ IP QoS network. In* Proceeding of SPIE symposium on QoS issues related to the Internet*, Nov 1998.*

[142] *S. Shenker, C. Partridge, and R. Guérin. Specification of guaranteed quality of service. IETF RFC 2212, Sep 1997.*

[143] *M. Shreedhar and G. Varghese. Efficient fair queuing using deficit round-robin.* IEEE / ACM Transactions on Networking, 4(3)*, pp.375–385, Jun 1996.*

[144] *Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy Quality of Service (QoS) information model. IETF RFC 3644, November 2003.*

[145] *D. Stiliadis.* Traffic scheduling in packet-swtiched networks: analysis, design, and implementation. *PhD thesis, University of California, Santa Cruz, CA, 1996.*

[146] *D. Stiliadis and A. Varma, Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms, In* Proceedings of INFOCOM96*, pp. 111-119, March 1996.*

[147] *D. Stiliadis and A. Varma. Frame-based fair queuing: a new traffic scheduling algorithm for packet-switched networks. In* Proceeding of SIGMETRICS*, 1996.*

[148] *D. Stiliadis and A. Varma, Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms, In* Proceedings of INFOCOM96*, pp. 111-119, March 1996.*

[149] *D. Stiliadis and A. Varma. Efficient fair queueing algorithms for packet-switched networks.* IEEE / ACM Transactions on Networking, 6(2)*, pp.175–185, Apr 1998.*

[150] *I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high-speed networks.* IEE / ACM Transactions on Networking, 11(1)*, pp.33–46, Feb 2003.*

[151] *P.R. Thie.* An Introduction to Linear Programming and Game Theory. *John Wiley & Sons, New York, second edition, 1988.*

[152] *K. Thompson, G. Miller, and R. Wilder. Wide-area internet traffic patterns and characteristics.* IEEE network, 11*, pp.10–23, Nov/Dec 1997.*

[153] *UCB/LBNL/VINT. Network simulator ns-2, 1997.* `http://www.isi.edu/nsnam/ns`.

[154] *S. Vegesna.* IP Quality of Service. *Cisco Press, 1st edition, 2001.*

[155] *D. Verma. Simplifying network administration using policy-based management.* IEEE Network Magazine, 16(2)*, pp.20–26, Mar/Apr 2002.*

[156] *H. Wang, C. Shen, and Shin K. G. Adaptive-weighted packet scheduling for premium service. In* IEEE International Conference on Communications (ICC2001)*, volume 6, pages 1846–1850, Jun 2001.*

[157] *X. Wang and H. Schulzrinne. Pricing network resources for adaptive applications in a differentiated services network. In* IEEE INFOCOM*, pages 943–952, Apr 2001.*

[158] *Z. Wang and J. Crowcroft. Analysis of burstiness and jitter in real-time communications. In* IEEE Global Telecommunications Conference*, volume 3, pages 1496–1500, Nov/Dec 1993.*

[159] *A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser. Terminology for policy-based management. IETF RFC 3198, Nov 2001.*

[160] *D. Wetherall and C.J. Linblad. Extending Tcl for dynamic object-oriented programming. In* Usenix Tcl / Tk Workshop*, page 288. Usenix, Berkley, California, 1995.*

[161] *J. Wroclawski. Specification of the controlled-load network element service. IETF RFC 2211, Sep 1997.*

[162] *J. Wroclawski. The use of RSVP with IETF integrated services. IETF RFC 2210, Sep 1997.*

[163] *J. Wroclawski and A. Charny. Integrated service mappings for differentiated services networks. INTERNET DRAFT (expired), Feb 2001.*

[164] *X. Xiao and L.M. Ni. Internet QoS: a big picture.* IEEE network, 13(2)*, pp.8–18, Mar/Apr 1999.*

[165] *S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore, S. Herzog, and R. Hess. Identity representation for RSVP. IETF RFC 3182, Oct 2001.*

[166] *R. Yavatkar, D. Pendakaris, and R. Guérin. A framework for policy based admission control. IETF RFC 2753, Jan 2000.*

[167] *Yi, S.; Deng, X.; Kesidis, G.; Das, C. R.: Providing fairness in diffserv architecture. In* Proceedings IEEE Global Telecommunications Conference (GLOBECOM02)*, 2002. 14351439.*

[168] *Sungwon Yi, Xidong Deng, G. Kesidis, and C. R. Das. Providing fairness in DiffServ architecture. In* IEEE Global Telecommunications Conference, Vol.2*, pages 1435–1439, Nov 2002.*

[169] *K. Yamori and Y. Tanaka. Relation between Willingness to Pay and Guaranteed Minimum Bandwidth in Multiple-Priority Service. In* Proceedings of Joint Conference of 10th Asia Pacific Conf. on Communications and 5th International Symposium on Multi-Dimensional Mobile Communications APCC2004 and MDMC2004*, Aug. 29-Sep. 1, 2004.*

[170] *O. Yaron and M. Sidi, Performance and Stability of Communication Networks via Robust Exponential Bounds,* IEEE / ACM Trans. on Networking, Vol.1, No.3*, pp. 372-385, 1993.*

[171] *H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks.* Proceeding of IEEE, 83(10)*, pp.1374–1396, Oct 1995.*

[172] *H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In* IEEE INFOCOM, Vol.1*, pages 227–236, Apr 1993.*

[173] H. Zhang and D. Ferrari. Rate-controlled service disciplines. Journal of High Speed Networks, 3(4), *pp.389–412, 1994.*

[174] Z.-L. Zhang, D. Towsley and J. Kurose, Statistical Analysis of Generalized Processor Sharing Scheduling Discipline, IEEE Journal of Selected Areas in Communications, *13(6),* pp. 1071-1080, August 1995.

[175] J. Zhang, T. Hämäläinen and J. Joutsensalo. A New Mechanism for Supporting Differentiated Services in Cluster-based Network Servers. In *Proceedings of 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, MASCOTS02,* pp. 427-432, Oct. 12-16, 2002.

[176] J. Zhang, T. Hämäläinen and J. Joutsensalo. Revenue-aware Resource Allocation in the Future Multi-Service IP Networks. In *Proceedings of IFIP TC6 Conference on Network Control and Engineering for QoS, Security and Mobility, Net-Con2004,* Nov. 3-5, 2004.

[177] J. Zhang, T. Hämäläinen and J. Joutsensalo, Stochastic Analysis of Upper Delay Bound of GPS-based Packetized Fair Queueing Algorithms, In *Proceedings of 12th IEEE International Conference On Networks (ICON2004),* Nov. 16-19, 2004.

[178] J. Zhang, T. Hämäläinen and J. Joutsensalo. Optimal Resource Allocation Scheme for Maximizing Revenue in the Future IP Networks. In *Proceedings of Joint Conference of 10th Asia Pacific Conf. on Communications and 5th International Symposium on Multi-Dimensional Mobile Communications, APCC2004 and MDMC2004,* Aug. 29-Sep. 1, 2004.

[179] J. Zhang, T. Hämäläinen and J. Joutsensalo. Packet Scheduling for Maximizing Revenue in a Network Node. In *Proceedings of 1st International Conference on E-Business and Telecommunication Networks ICETE2004,* Aug. 25-28, 2004.

[180] J. Zhang, T.Hamalainen and J. Joutsensalo. Maximizing Service-Level-Agreement Revenues in Clustered-basedWeb Server Systems *WSEAS Transactions on Communications,* 2004.