Providing Quality of Service in Packet Switched Networks

Ph.D. dissertation

Tamás Marosits

Supervisor:

Sándor Molnár, Ph.D.

Doctoral School of Electrical Engineering Sciences

Department of Telecommunications and Media Informatics
Faculty of Electrical Engineering and Informatics
Budapest University of Technology and Economics

Budapest, 2011

Abstract

Services of infocommunication networks become more and more accessible for us. Using of these services is natural and permanent need for more and more people. An evident answer to this expectation by the vendors, access and content providers is the concentration and integration which can be observed in the physical networks as well as in the field of used protocols. Although the expansion of network capacities seems to keep up with the increasing amount of user traffic, perceptible quality degradation can be experienced due to the huge amount of transmitted data. To combat these problems network operators have several potential solutions. Among these the traffic control methods are emphasized in my dissertation.

In the first part of my work I proposed a scheduler primarily applicable in fixed size packet switching networks. This method provides optimal service in a given network node for all flows taking into account the individual flows' quality-of-service requirements given by their sources, and the quality of the service received by the flows previously. I verified the functionality of the server with simulation.

In the second part of my work I proposed an other packet scheduling method which is a variant of the well known round robin. The main difference is that the flows have multiple service opportunities in a single round. In this system called Advanced Round Robin we allow to set up a new connection only in the case if it is ensured that all active flows (including the new one) can achieve the required quality of service. Based on the architecture of the scheduler I gave the statistical and deterministic (worst case) service guarantees provided to the individual flows. I revealed that ARR scheduler is related to the servers based on the Generalized Processor Sharing, it can be characterized as a Latency Rate server and I calculated the fairness index of the ARR. I compared the Advanced Round Robin scheduler with the most popular similar servers treated in the literature. It can be stated that ARR is one of the best servers regarding latency and fairness parameters.

The mechanisms of the ARR scheduler and the provision of quality of service guar-

antees rely on a properly constructed theoretical service cycle. So I gave a procedure for the construction of the service cycle knowing the service opportunities assigned to the individual flows. The number of service opportunities received by a flow during a cycle should primarily depend on the quality of service requirements of that flow, however, we have to take into account the actual characteristics of the system. Based on this I gave an algorithm which calculates the number of service opportunities (orders) to be assigned to a newcomer connection (and even orders of the connections being already in the system if needed). From these two methods arise boundaries which decide unambiguously that a newcomer connection characterized by known traffic descriptors and quality of service requirements can be accepted or not assuming an ARR server. I verified the functionality of the two methods mentioned above with simulations, as well as I established practical bounds to make the acceptance decision within a reasonably short period of time.

Kivonat

Az infokommunikációs hálózati szolgáltatások egyre könnyebben hozzáférhetőek számunkra. Ezek használata egyre több ember számára természetes és állandó igény. Erre az igényre adott természetes gyártói, hozzáférés- illetve tartalomszolgáltatói válasz a fizikai hálózatok terén és az alkalmazott protokollok terén is megfigyelhető összefonódás és integrálódás. A hálózati kapacitások bővítése egyelőre lépést tart a forgalom növekedésével, de a nagy átvitt adattömeg miatt így is előfordulhat, hogy bizonyos szolgáltatások minősége észrevehetően romlik. Az ezzel kapcsolatos problémákra a hálózatüzemeltetők eszköztárában több megoldási lehetőséget találunk. A disszertáció ezek közül a hangsúlyt forgalomirányítási módszerekre helyezi.

A munkám első részében egy olyan – elsősorban fix méretű csomagokat továbbító hálózatokban alkalmazható – forgalomirányítási eljárást javasoltam, amely az adott csomópontban az összes átmenő folyam számára a lehetséges legoptimálisabb szolgáltatást nyújtja, figyelembe véve a folyamokra vonatkozó, a források által megadott minőségi követelményeket, illetve a folyamok által korábban kapott kiszolgálás minőségét is. A kiszolgáló működőképességét szimulációval támasztottam alá.

A munkám második részében egy másik csomagütemező eljárást javasoltam, amely az ismert körforgó prioritásos rendszer egy olyan változata, amelyben a folyamok egy cikluson belül többször is kiszolgálást nyerhetnek. Az Advanced Round Robinnak nevezett rendszerben a hívások felépítését abban az esetben engedjük meg, amennyiben biztosítható, hogy a csomagütemező az újjal együtt az összes aktív folyamot a megkívánt minőségben tudja kiszolgálni. A csomagütemező architektúrája alapján megadtam az egyes folyamok számára nyújtott statisztikus illetve determinisztikus értelemben számítható szolgáltatás minőségi jellemzőket. Megmutattam, hogy az általam javasolt ARR csomagütemező bizonyos korlátozásokkal rokonítható a Generalized Processor Sharing eljárásra alapozott ütemezőkkel. Az ARR leírható, mint Latency Rate kiszolgáló, valamint megadtam az ARR csomagütemező fairness értékét. Ezek alapján az általam javasolt ARR csomagütemezőt összehasonlítottam a szakirodalomban gyakrabban tárgyalt hason-

ló jellegű csomagütemezőkkel. Az összevetés eredményeként megállapítható, hogy az ARR ütemező a várakozási idő (*latency*) és a méltányosság (*fairness*) paramétereket tekintve a legjobbak közé tartozik.

A csomagütemező működése és a szolgáltatásminőségi garanciák nyújtása azon a feltevésen alapul, hogy a rendelkezésünkre áll a korábban összeállított elvi kiszolgálási ciklus. Adtam tehát egy eljárást arra, hogy az egyes folyamokhoz rendelt kiszolgálási gyakoriságokat felhasználva miként építjük fel az elvi kiszolgálási ciklust. A kiszolgálási gyakoriságnak elsősorban az adott folyamhoz kapcsolódó szolgáltatásminőségi követelményektől kell függnie, azonban kiszámításánál figyelembe kell venni a rendszer aktuális jellemzőit is. Erre alapozva megadtam egy eljárást, amely egy újonnan beérkező folyam esetén meghatározza a hozzá rendelendő kiszolgálási gyakoriságot (és szükség esetén a rendszerben lévő többi folyamét is újra). Ebből a két eljárásból kiadódnak olyan korlátok, amelyek ARR kiszolgálót feltételezve egyértelműen megadják, hogy egy újonnan érkező, ismert forgalomleírókkal és szolgáltatásminőségi követelményekkel jellemezhető folyamot elfogadhatunk-e vagy sem. Az említett két eljárás működőképességét szimulációs vizsgálatokkal igazoltam, valamint ezek alapján olyan gyakorlati korlátokat állítottam fel, amelyeken belül az új hívás beengedéséről kellően rövid idő alatt dönteni tudunk.

Acknowledgement

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The telecommunication networks of the 21^{st} century are intended to provide high speed, secure and cost effective connections with guaranteed quality of service (QoS) for their users. During the evolution of services and packet switched networks the increase of the number of users and the diversity of resource requirements can be experienced. The need for simple and robust traffic control methods is obvious. These methods should satisfy the following general requirements:

- high utilization, due to the interest of service providers;
- fine granularity, because networks should adapt many different quality of service requirements and traffic characteristics and high utilization could not be achieved if there are only rough categories for connections;
- being capable to provide both worst case and average quality of service guarantees, because delay and jitter sensitive and best effort-like applications will coexist in the integrated packet switched networks; and
- to protect the traffic generated in conformity with the traffic contract, because the misbehavior of a connection must not influence the service received by the others.

Behind the above general statements the transporting of the traffic generated by the wide variety of networking applications and services asks for much more consideration. The increasingly popular real time services e.g. video streaming generates a massive amount of data that cannot be handled using the traditional best effort forwarding. The quality of service of a flow cannot be corresponding only to packet loss rate suffered by that flow. Data loss rate get a much smaller role than before while the retransmission of

the missed packet is not only impossible because of the time constraint, but it is unnecessary because of the redundancy forwarded in a media stream and the error correction algorithms applied in the media players. On the other hand, jitter wins a distinguished importance, as never before[1] and the delay should be controlled even in the case of non real-time applications following the users' expectations. If high server and link capacities assured for a connection throughout its route it can achieve low end-to-end delay. However, when this bandwidth reservation is made in a static manner the network utilization will be lower which naturally results in higher transmission costs. Statistical multiplexing allows higher utilization but requires computational resources because of the complexity of these algorithms.

To control the traffic *inside* the network[2] we have the classic tools: buffer management and packet scheduling. If a connection gets more buffer space in the switches and its packets are forwarded before any others it would enjoy low packet loss and delay.

However, the set of traffic control tools is not complete without call acceptance functionality. The acceptance decision should be made at the *edges of the network*, however close cooperation should be exist with the tools used inside the network.

In my dissertation traffic control methods are discussed. I present the in depth performance analysis of such methods via extensive analytical investigations and simulations. The detailed performance analysis of these methods have done by analysis and/or simulation. There are solutions given to choose the appropriate parameter set of schedulers and a co-operative Call Admission Control (CAC) function is presented to the Advanced Round Robin scheduler presented in Section 4.1 scheduler.

1.1 Research objectives

The fundamental motivation behind my research was to introduce and analyze packet scheduling methods for fast packet switched networks which can provide statistical and/or deterministic quality of service guarantees taking into account not only the packet loss but delay and jitter which receive more and more importance thanks to the increasing ratio of time sensitive traffic in the nowadays' infocommunication networks. Solutions should be given to determine the parameter set of these schedulers.

The proposed schedulers should be simple and computationally feasible. To achieve this goal seems to be straightforward because of the rapid increase of the computational

[1] In the traffic mixture of Internet the part of UDP is increasing thanks to the real time applications.
[2] To control the traffic before it enters to the network we can use traffic shapers e.g. token bucket.

capabilities in the elements of the recent infocommunication networks.

However, traffic control is much more than scheduling: it includes traffic shaping, packet marking, buffer management, call admission control, etc (see e.g. in [10] and [9]). The latter was involved in my research work to supplement the scheduler proposals, i.e. my second goal was to find an appropriate call admission control algorithm which will co-operate with the scheduler. The presented CAC function accepts or rejects a newcomer connection considering traffic characteristics and QoS requirements of the existing connections and the newcomer connection and the utilization of network resources. If a new connection is accepted the CAC algorithm will output the parameters for the reconfiguration of scheduler.

1.2 Methodology

For the analysis of schedulers (in Section 3.2 and in Section 4.1.7) I have developed a simple simulator program to achieve as appropriate characterization as possible. The call admission control function associated to the Advanced Round Robin scheduler have been implemented under Wolfram Mathematica, which were also the framework for the in depth performance evaluation in Section 4.2.3.

For the Advanced Round Robin scheduler and its co-operative CAC algorithm I also present analytical results in Section 4.1 and Section 4.2.

1.3 Models

1.3.1 Traffic Control Parameters

Traffic control is usually built upon the following three parameter sets: description of traffic, network resources and quality of service requirements. From CAC to scheduling every network function should take into account these groups. Starting out from the standardization work of ATM Forum [3] [4] and ITU-T [32] [33], I considered the following parameters in my traffic control framework:

- **Connection Traffic Descriptors**: Peak Cell Rate (PCR), Cell Delay Variation Tolerance (CDVT), Sustainable Cell Rate (SCR), Maximum Burst Size (MBS), Minimum Cell Rate (MCR) and the conformance definition: the Generic Cell Rate Algorithm (GCRA) [33]

- **Quality of Service Parameters**: Cell Loss Ratio (CLR), average Cell Transfer Delay (CTD), peak-to-peak Cell Delay Variation (CDV)
- **Network Resources**: link capacities (C), memory size for buffering (Q).

1.3.2 Quality of Service specification

There are five service classes[3] with different traffic descriptors and QoS requirements defined in Asynchronous Transfer Mode (ATM) [16]. Nice correlation can be discovered between the traffic descriptors and the QoS requirements, which are specified in Table 1.1. My assignment is mainly based on the ATM Forum specification [4].

Table 1.1. Parameters of traffic contract and QoS requirements

Attribute	CBR	rtVBR	nrVBR	UBR	ABR
Traffic Parameters					
PCR, CDVT	X	X	X	X	X
SCR, MBS, CDVT		X	X		
MCR					X
QoS Parameters					
CDV	X	X			
CTD	X	X			
CLR	X	X	X		X

An exhaustive performance study should have examined the dependence of the Quality of Service on the load and burstiness changing of five service classes. However, this work is not only immense but also unnecessary. CBR is a very regular traffic, and the impact of its load increase is calculable. ABR flows are difficult to handle because of feedback rate control [11], but this feature makes them resistant to network congestion and packet loss. UBR can also be disregarded because the QoS of guaranteed services must not depend on the behaviour of UBR flows in the case of any appropriate scheduling policies. The only thing to do with this class is to show that it has no effect on the other classes. VBR flows can change their rate during the connection. They are bursty and they have no feedback, but they apply for service guarantees in connection set up. Most of our simulation results deal with the real time VBR and non-real time VBR traffic.

[3]Constant Bit Rate (CBR), Variable Bit Rate for real time and non real time traffic (rtVBR and nrVBR, respectively), Available Bit Rate (ABR) and Unspecified Bit Rate (UBR)

1.3.3 Traffic generation

Speaking about network simulation appropriate test traffic generation is a fundamental question. The traffic model should be sophisticated enough to describe the behavior of the traffic transmitted on the recent worldwide info-communication networks and should be easy to implement because of the simulation time. In my simulation Interrupted Bernoulli Process (see e.g. in [46]) was used to generate variable rate bursty traffic. Constant bit rate flows are simply implemented by generating packets with uniform timing.

1.3.4 Buffer architecture

One of the pivots of packet switched networks is buffering. How much memory should be used in the nodes to store the packet bursts before they can be forwarded? How to divide the total amount of buffer space between the connections? Although, these are fundamental questions, they are out of the scope of this dissertation.

Buffering and other tools of traffic control could not be perfectly detached from each other but we should separate them as far as possible. In this work I deal with *complete buffer partitioning architecture* because this policy ensures us of the segregation of different connections in the network. Using this policy the total switch memory is divided to as many queues as the number of connections.

1.4 The structure of the dissertation

After this brief introduction an overview is given about the scheduling in Chapter 2. Among the huge number of schedulers more attention will be payed to the round robin type schedulers and to the Generalized Processor Sharing service discipline. In the second part of this chapter two characteristics related to schedulers are presented which will allow to compare the Advanced Round Robin server to well known schedulers.

Next, the results of my research work will be introduced. My statements form the following 3 theses:

- Theses 1: A new scheduler, called Weighting Function scheduler with parameter settings (Chapter 3).

- Theses 2: A novel round robin type scheduler, called Advanced Round Robin scheduler (Section 4.1).

- Theses 3: Service cycle construction and call acceptance control for the Advanced Round Robin scheduler (Section 4.2).

In Theses 1 I present a new scheduler architecture, in which a weighting function is associated with each flow and the flow with highest weight will be chosen to be scheduled in the next time slot. Based on the proposed structure of weighting functions statistical guarantees can be provided to the flows. The weighting function set described in the dissertation can be used in networks transmitting constant length packets but the scheme can be extended for variable length packets as I write in Section 3.3.

In Theses 2 I introduce the Advanced Round Robin scheduler and evaluate its deterministic and statistical quality of service guarantees. I also demonstrate that ARR shows up the characteristic properties of several well known server families and ARR related to the Generalized Processor Sharing service discipline. Although, the original Advanced Round Robin scheduler serves constant length packets I suggest two possibilities in Section 4.1.9 for the generalization.

Within Theses 3 a call admission functionality associated with the ARR scheduler will be presented. This means actually two algorithms, one for the evaluation of the numbers of service opportunities of the competing flows during a round, while the other one establishes the round (service cycle) with the full knowledge of the number of service opportunities of the flows. Based on these algorithms the necessary and sufficient conditions of the acceptance of a new call will be formulated in a theorem. Further, the performance analysis of the CAC functionality results in practical limits of the usage of the CAC.

In Chapter 5 I will write about the possible applications of my results and in Chapter 6 I will summarize my dissertation.

Chapter 2

Related works

Recent communication networks are intended to provide Quality of Service guarantees for their users. In the Internet world IP header contains a Type-of-Service (TOS) field for QoS differentiation but the real pioneer in the field of QoS support was the Asynchronous Transfer Mode (ATM) [16]. The recent developments regarding the Internet technology yielded a number of possibilities for QoS networks, e.g. Integrated Services [10] or Differentiated Services [9] with or without MPLS [50] infrastructure. Specifically, the emergence of applications with very different throughput, loss or delay requirements calls for a network capable of supporting different levels of services, as opposed to a single, best-effort service which is the rule in today's Internet.

2.1 Schedulers

To support QoS we have two classical opportunities: buffer management and scheduling [28]. The first one is associated mainly with packet loss and the delay characteristics of the flow mostly controlled by the latter, but the bounds are loose (e.g. if a flow has absolute priority over all the others it does not need a large buffer to get zero packet loss, and vice versa, a short buffer bounds from above the delay of the flows). Both the buffer allocation and the scheduling policies influence the total amount of buffer space required by the system. Generally the impact of buffer management and scheduling policy cannot be separated to delay and packet loss. In some recent works the separation was done for Generalized Processor Sharing (GPS) schedulers fed by leaky bucket shaped sources by Szabó et al. e.g. in [55]. However, these solutions yield overdimensioning of resources in practice.

A packet scheduler is used in general to arbitrate the transmission of the packets from

the queues on a link. There is a huge number of schedulers presented and analyzed in the literature. They can be generally separated into two classes: work-conserving and non-work-conserving disciplines [60]. While *work-conserving* schedulers can not remain idle if at least one packet is buffered in the system, with a *non-work conserving* service discipline a packet should be held back until its *eligibility time* assigned to it upon arrival [61]. Work-conserving schedulers provide always lower average delay and higher average throughput than non-work-conserving servers. Non-work-conserving schedulers are used in the cases when we should decrease the impact of traffic pattern distortion caused by the network load fluctuations. For example, we have to control the burstiness of the traffic inside the network which is important if the buffer space is limited in the inner servers, or we have jitter-sensitive traffic.

An other characterization of schedulers is based on their internal structure. Some of them (e.g. Weighted Fair Queueing [15], Virtual Clock [62], Start Time Fair Queuing [26]) can be characterized as *sorted priority algorithms*. During the working of the server they maintain a sorted queue which allows them to serve packets regardless of their arrival sequence. These algorithms simulate an ideal Generalized Processor Sharing [44, 45] discipline and try to transmit packets in the order in which they would leave a GPS server. These type of schedulers generally provide a delay bound respecting the weight associated with the queue but have considerable high computational complexity which depends on the number of flows that use the service. Other schedulers use the well known round robin order in the transmission of packets of the flows, they are called *frame based schedulers*, too. When using appropriate parameters these servers can achieve the per-packet work complexity of $O(1)$.

In [58] Xu and Lipton was presenting their interesting results about the correlation between the delay bounds which can be provided by a scheduler and the computational complexity of that scheduler. They had proven that under some conditions the computational complexity of a scheduler which guarantees a GPS relative delay[1] of $O(n^a)$ (where $0 < a < 1$ and n is the number of connections) is $\Omega(logn)$. They also extended this proposition to the members of Latency Rate server class. However, the above mentioned preconditions of their result (the so called *continuously backlogged fair sharing* (CBFS) assumption which means that the competing sessions have equal weight and they are continuously backlogged during the examined time interval) cannot be met in most realistic situations.

[1]The difference between the time the packet finishes service in the scheduler under analysis and its virtual finish time under an ideal fluid flow Generalized Processor Sharing server.

In [14] Cruz presented SCED+ network scheduling algorithm. Because of *aggregation* of best effort and guaranteed traffic, the method can provide only statistical packet loss guarantees. His method yields scalable provision of tight deterministic end-to-end delay bounds. However, aggregation entails a sacrifice in the granularity of the delay bounds.

Round Robin is a well-known method to serve systems with multiple queues. It has a considerable advantage: it is very easy to give delay bounds for the applications. However, this worst case delay bound can be too loose because of the huge number of switched connections. This is the reason that I modified the Round Robin scheme, such that some flows could have access to a more frequent service.

In the literature there are a number of proposals for Round Robin-type (RR) algorithms. Katevenis et al. in [37] proposed the Weighted Round Robin scheduling algorithm for the scheduler of an ATM switch. The WRR scheduler takes into account, that the flows with different traffic characteristics should not be treated equally, the server process "visits" flows which has higher weight more frequently, than flows with lower weight. They also thought that these visits should be spread "evenly" in the time axis and presented a method - with remarkable limitations - to resolve this issue. However, it is not clear where the weights come from, what is the connection between the quality-of-service requirements and the weights, and what is the performance of their system in the case of higher throughput (they calculated with a utilization of 0.5).

The reader can find in [21] a more ingenious algorithm designed for fixed size packets (cells). Recursive Round Robin (RRR) scheduler is based on the idea of the construction of a "scheduling tree" (which is a binary tree in the case of basic scheduler) in which the individual flows are served recursively. Connections may have multiple slots in a round. The allocation is made on the basis of the binary representation of the flow's rate relative to the whole server rate. A null stream bears the free resources of the system which means that the scheduling tree has leafs which induces no packet transmission. If this 0 stream is on the schedule link may remain idle[2], transmit a cell from the next stream, or from the best effort traffic. When a new flow arrives its insertion is isomorphic to the subtracting of binary numbers, i.e. the capacity needed by the newcomer connection should be subtracted from the resources belonging to the null stream. When a connection finishes it leads to addition of binary numbers, inversely. RRR provides delay and jitter bounds, and fairness is bounded, too. However, guarantees are depending on the number of ones in the binary representation of the flow's normalized rate which is bounded by the length of this fix point fractional number, the depth of the scheduling tree, .i.e. the

[2]Which indicates that the server is used in a non-workconserving manner.

granularity of the allocated rate. Besides this the building of the scheduling tree is based only on the rate of the sources and disregards the delay requirements. In addition, from the possibilities listed at the case of scheduling null stream one can conclude that authors of the Recursive Round Robin decided to handle best effort traffic outside their scheduler.

The idea of Mini Round Robin (MRR) presented in [2] can be tracked back to the nested version of Deficit Round Robin in the viewpoint of using mini rounds. On the other side it is similar to the Elastic Round Robin because the MRR scheduler does not need to know the maximum packet size that may arrive to the server, however, in Mini Round Robin there is not a fixed quantum assigned to the flow which has to be used up during an inner (mini) round. In the MRR two linked list of the flows should be handled. One for the active flows and one for the flows which has the right to send packet in the next mini round. In the beginning of each outer (major) round, the content of the first list is moved to the second list. Then in the first mini round each flow can transmit one packet. After transmission each flow's balance will be decreased by the length of the packet sent from the flow. If the balance of a flow becomes negative or zero it will be excluded from the next mini round and added to the tail of the list of active flows. When no flows are left in the mini round list a new major round is started and the flows' balance will be updated (increased by $(MaxD(r)+1)$, where $MaxD(r)$ is the maximum deficit collected in the previous round r).

Deficit Round Robin (DRR) was proposed by Shreedhar and Varghese in [52]. Their goal was to approximate fair queueing with small computational complexity for flows transmitting packets of different sizes. DRR is a simple modification of the well known round robin scheme. The queues served in a round robin manner according to the *quantum* (Q_i) and the value of *deficit counter* (DC_i) assigned to it. The bandwidth allocated to packets of queue i is proportional to Q_i. The deficit counter is started from zero when the queue become backlogged. Theoretically each queue can send a number of bytes equal to its quantum in each round. Practically, in the first round each queue allowed to transmit packets consist of bytes up to the value of its quantum. If the quantum not entirely consumed it will be added to the deficit counter. In the subsequent rounds the amount of bytes allowed to transmit by queue i will be $DC_i + Q_i$. If there are no packets remaining in the queue after service the deficit counter will reset to zero. The latency parameter of Deficit Round Robin was evaluated first in [53], but Kanhere and Sethu in [35] gave a tighter delay bound.

As a variable of the Deficit Round Robin Surplus Round Robin was originally proposed in [1] by Adiseshu et al., but it is identical with the scheduler proposed by Floyd in [18] and

[19]. SRR was evaluated in [42]. The main difference between DRR and SRR is that in the case of Surplus Round Robin the deficit counter (called here as *surplus counter* referring to the nature of the scheduler) may become negative after the round. This means that Surplus Round Robin does not need to know the size of the head-of-line packet of the queue.

Elastic Round Robin (ERR) presented by Kanhere et al. in [36] tried to overcome the drawbacks of the previously known schedulers, especially of Deficit Round Robin and Surplus Round Robin, which have a per-packet work complexity of $O(1)$ but we have to ensure that the size of the *quantum* is at least equal to the largest packet that may potentially arrive to the scheduler. In the case of the ERR we need not know the maximum packet size because the quantum which will be added in round r to the *allowance* $A_i(r)$ of flow i is not a constant value. It is chosen adaptively according to the maximum of surplus counts $SC_i(r-1)$ attached to the flows served in the previous round $(r-1)$. Elastic Round Robin has a weighted version too, in which the weight w_i associated to flow i will be taken into account during the calculation of the allowance. The relative fairness bound of the ERR evaluated in [36] is $3m$, where m is the size of the largest packet that actually arrives during the execution of the scheduler. However, this may seem lower than the fairness of DRR or SRR, which is $M+2m$, where M is the size of the largest packet which can *potentially* arrive to the scheduler, the absolute fairness bound is the same for all three schedulers.

When searching for a simple scheduler, which is suitable to provide service to delay-sensitive real time traffic DRR, SRR, and ERR seem to be inadequate, because the flows having most weights are scheduled in one burst in the round. This yields to a bursty arrival of these flows at the next hops in a multi-nodal scenario and ignores quality of service requirements regarding delay and jitter.

To address the output burstiness problem of the DRR-server family Guo developed some round robin type schedulers. Smoothed Round Robin (SRR) ([30]) provides a service sequence in which the number of visits allowed to flow i is equal to the weight w_i associated with that flow and this visits are distributed evenly. The working of the SRR is based on the Weight Matrix M which contains the binary coded w_is in its rows. The server is ruled by the Weight Spread Sequence (WSS) in which the value of actual term determines the column of M according to it the queues to be scheduled can be chosen. The fairness of the Smoothed Round Robin server is $(k+2)L_{max}/2min(w_i, w_j)$, where k is the order of M (the number of its columns[3]), L_{max} is the maximum of the packet size while w_i and w_j

[3]Which is the equivalent of the granularity of the server's bandwidth here, as one can observe, that

are the weights of the compared queues. In order to reduce the computational complexity of SRR we have to maintain double k linked lists corresponding to the columns of Weight Matrix which keep records about the index of the rows of M which contain 1 in the actual column.

In [31] Guo published two further improvements to Smoothed Round Robin based on the transformation of Weight Matrix (WM). The SRR$^+$, in which WM is transformed to an upper triangular matrix, will make independent the distribution of the schedule opportunities of flow i from the sequence of the flows arriving to service. This procedure overcomes the skewed weight distribution problem which can lead unevenly distributed service sequence and inhibits to provide tight delay bounds. In SRR$^\#$ the WM is transformed to a diagonal matrix which results in the further tightening of the server's delay bound.

In [29] also a RR-type scheduler called G3 for fixed size packet networks was presented by Guo. G3 is an O(1) time complexity packet server built up on the Recursive Round Robin described in [21] and Smoothed Round Robin [30]. This server model has a more strict delay bound than Smoothed Round Robin, but its fairness was not evaluated and published yet.

Ordered Round Robin (ORR) method presented in [59] aims to make contributions to the packet scheduling algorithms of Network Processors and to link striping problem and it assumes that the workload of the entire server is perfectly divisible in bytes. Channel stripping addressed previously in [1] means that we want to share load among the multiple channels between the sender and receiver. Originally it does not care about the ordered arrival of the data. There are many similarities between link stripping and packet scheduling, so Yao et al. present their model only in terms of Network Processors without the loss of generality ([59]).

The main goals of the Packetized Ordered Round Robin (P-ORR) which was designed on the basis of combination of DRR and SRR are to provide load balancing for processing *variable length* packets using a group of heterogeneous processors and to ensure in order delivery of packets without considering receiver's rearranging capability. To achieve these it wants to minimize the sum of squares of the difference between the ideal and actual load distributed to a processor.

The P-ORR method seems to be very simple and computationally feasible. However, it is compared only with the SRR and traditional RR, and there are no delay bounds given. According to the simulation results the service rate of the flows are proportional

the binary representation of the smallest possible weight is $0 \ldots 01$.

to their reservations[4], which means that the scheduler is fair, but we do not know from where these reservations come, and no fairness index is given.

Table 2.1 summarizes the round robin-type schedulers presented above. There are a huge number of further variants too, due to the simplicity of the model and the low computational complexity needed to implement it.

The Generalized Processor Sharing (GPS) service discipline [44] is an ideally fair fluid model in which the traffic is considered as infinitely divisible and every session is being served simultaneously sharing the server capacity. The description of GPS could be found in many papers, here we will use the notations introduced in [44, 45] by Parekh and Gallager.

A GPS server serving N sessions is characterized by N positive real numbers, $\phi_1, \phi_2, \ldots, \phi_N$. The server is work-conserving[5] and operates at a *constant rate* r. Let we denote the amount of session i traffic served in the interval $[t_1, t_2]$ with $W_i(t_1, t_2)$, then a GPS scheduler is defined as a server for which

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j}, \qquad j = 1, 2, \ldots, N, \tag{2.1}$$

for any session i that is continuously backlogged[6] in the interval $[t_1, t_2]$. It is followed from this definition immediately is that every session has a minimum *guaranteed service rate* which can be expressed as

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} \, r. \tag{2.2}$$

In the Generalized Processor Sharing model the input traffic of the sessions can be shaped by a *token bucket* (see Fig. 2.1) that is described by a token pool depth (σ) and a token generation rate (ρ).

An important advantage of using leaky buckets is that it allows the separation of packet delays into two components: delay in the leaky bucket and delay in the network. The first component is independent of other (active) sessions, while the second one is independent of the incoming traffic.

Further, the amount of traffic at the output of a token bucket shaped active source i in the interval $[t_1, t_2]$ assuming infinite capacity links[7] can be characterized by the function $A_i(t_1, t_2)$. If $A_i(t) = A_i(0, t) = \sigma_i + \rho_i t$, which means that session i starts with its maximal

[4] In this context we can think about reservations that they are in fact weights.

[5] A server is work-conserving if it is never idle whenever there are packets to send.

[6] the session buffer is not empty

[7] We can assume that the internal link between a queue and the leaky bucket associated with it do not constrain the service provided by the GPS server.

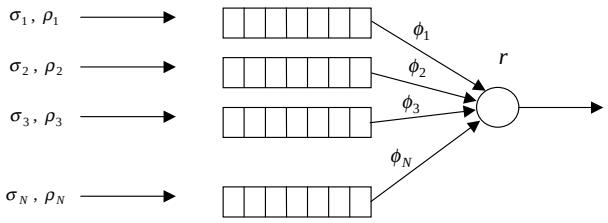

Figure 2.1. GPS server with leaky bucket inputs

burst σ_i at time zero and continues to transmit with its maximal rate ρ_i then by definition session i starts *greedy*. If all sessions start greedy one gets a *greedy GPS system*.

For every session i, the maximum delay D_i^* and the maximum backlog Q_i^* are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period[8]. Furthermore, assuming that for each session i $g_i \geq \rho_i$, then

$$Q_i^* \leq \sigma_i \quad \text{and} \quad D_i^* \leq \frac{\sigma_i}{g_i}. \tag{2.3}$$

In the previous decade there have been a vast work on developing and analyzing GPS schedulers, see e.g. [17], [43], [57]. Although such a GPS system can not be accomplished in practice, there are several schedulers emulating it at the background to determine packet serving orders (e.g. Weighted Fair Queueing and its variants: Virtual Clock, WF^{2}Q [6], Start Time Fair Queueing, etc.). These packet-by-packet versions of GPS were also analyzed establishing important relations between the fluid model and the packetized versions. In most cases analysis of the GPS model is sufficient since results can be transformed to packetized versions in a straightforward manner. By presenting the relationship between Generalized Processor Sharing and any other service discipline we will be able to take full advantages of results achieved in connection of GPS: for example, new worst case guarantees can be formulated for single node case and a multi-node scenario can be easily analyzed taking into account the results [55] and [45], respectively.

[8] an interval in which the server is continuously working

Table 2.1. Overview of the well-known RR-type schedulers

Name	Author(s)	Packet size	Delay bound and fairness index	Main properties
Weighted Round Robin (WRR)	Katevenis et. al. (1991) [37]	ATM	both	connection between weight and QoS requirements is not clear
Recursive Round Robin (RRR)	Garg and Chen (1999) [21]	fixed	both	resource allocation based on rate
Deficit Round Robin (DRR)	Shreedhar and Varghese (1995) [52]	variable	both	max packet length and HOL packet length should be known, weight based on input rate
Surplus Round Robin (SRR)	Adiseshu et al. (1996) [1]	variable	both	max packet length should be known, weight based on input rate
Elastic Round Robin (ERR)	Kanhere et al. (2002) [36]	variable	both	weight based on input rate
Mini Round Robin (MRR)	Al-Khasib et al. (2005) [2]	variable	both	resource allocation based on rate
Smoothed Round Robin (SRR)	Guo (2004) [30]	variable	fairness only	delay bound is proportional with the number of flows
G3	Guo (2007) [29]	fixed	both	tight delay bound, combination of the Smoothed RR and the RRR
Packetized Ordered Round Robin (P-ORR)	Yao et al. (2008) [59]	variable	none	weight based on input rate

2.2 Characterization of schedulers

A wide range of schedulers (e.g. Weighted Fair Queueing, Weighted Round Robin, Deficit Round Robin, etc.) can be described as a Latency Rate (LR) server [53]. Belonging to the class of Latency Rate servers means to correspond a general model used in the analysis of traffic scheduling algorithms in high speed packet switched networks. The behavior of a Latency Rate scheduler can be characterized by two parameters - the *latency* and the *allocated rate*. The significance of the theory of LR-servers is that we can derive tight upper bounds on the end-to-end delay, internal burstiness and buffer requirements of individual flows in case of the networks of even different type schedulers belonging to the LR-family, when the traffic of the flow is shaped by a leaky bucket. When we show that ARR belongs to this class we can use all of the results reached previously in connection with LR-servers. We can now easily calculate tight delay bounds for a series of ARR servers or use ARR-server with other LR-schedulers in a network.

The significance of the Guaranteed Rate (GR) server class is similar as of the Latency Rate servers mentioned above. Guaranteed Rate servers ([24, 25]) can be characterized by the property that they can guarantee a deadline for a packet in a flow by which this packet will be transmitted. Based on this model deterministic end-to-end delay bounds can be derived for packets originated by leaky bucket shaped sources and served by the series of schedulers belonging to the Guaranteed Rate class. In [24] Goyal et al. proven that Virtual Clock, Packet-by-Packet Generalized Processor Sharing and Self-Clocked Fair Queuing are Guaranteed Rate servers.

The significance of this result is that these two models have been widely used for the analysis of Integrated Services network as well as Differentiated Services networks. If it have been proven about a scheduling discipline that it belongs to either or both server class it can be considered as a scheduler useable in IntServ or DiffServ environment.

In [34] Jiang showed that if a scheduling algorithm belongs to Guaranteed Rate servers, it also belongs to Latency Rate class and vice versa and even the relation between *latency* (Θ) and *error term* (β) were investigated.

Analyzing different schedulers we can notice considerable differences in the service received by various connections over an interval of time. Moreover, this deviation can be observed during the operation of a single server. This property is described with *fairness*. Using a fair queueing service discipline is advantageous not only from the viewpoint of ensuring fairness in the amount of service received by the competing flows but providing worst-case performance guarantees, too. To formulate fair queueing in a general case the

notion of fairness was applied to an idealized fluid-flow environment by Demers et al. in [15]. The result was used then to specify fair queueing for a packet-based discipline. When using a fluid-flow model the amount of service offered to a flow is infinitesimally divisible or - equivalently - we can assume that multiple connections can be serviced in parallel. This model is referred to in the literature as *fluid-flow fair queueing* (FFQ).

Obviously, fluid-flow fair queueing cannot be applied directly in a real packet-based scenario, so the the definition of FFQ was extended in the way that packets have to be scheduled to service in the order that they would finish service according to the FFQ scheme in the fluid-flow server model. The same approach has been described and used by Parekh and Gallager in [44, 45]. Analogous to FFQ, this scheme is referred to as *packet-by-packet fair queueing* (PFQ). The actual FFQ and PFQ disciplines are referred to in [44, 45] as *generalized processor sharing* (GPS) and *packet-by-packet generalized processor sharing* (PGPS), respectively.

Because these early packet-based fair queueing disciplines were based on hypothetical fluid-flow reference systems to determine the fair order of traffic transmission, this approach leads to considerable computational complexity and anticipate the infeasibility of the scheme for high speed applications. Golestani in [23] defines fairness in a self contained manner as the maximum difference between the normalized service received by two backlogged flows over an interval of time in which both are continuously backlogged.

Chapter 3

Scheduling with weighting functions

I have proposed a new scheduler, called Weighting Function scheduler (WF scheduler) with a parameter setting. I have also carried out the performance evaluation of the scheduler in packet switching networks with constant packet length. The results related to WF scheduler and summarized in this chapter were described and published previously in [**J2**, **C3**, **C4**]. A typical example of constant packet length is the ATM.

The motivation behind the creation of the WF scheduler was to present a traffic control framework which provides at least statistical quality of service guarantees for flows accepted to service. These flows can be originated from several types of sources with diverse traffic descriptors and quality of service requirements. To achieve high throughput the server should allow to allocate network resources with an arbitrary degree of granularity. Moreover, this capacity reservation cannot be based only on the static quality of service requirements, it should be payed attention to the instantaneous quality of service parameters of each flow in the system. To the operation of this dynamic scheduler the continuous ("real time") QoS monitoring of Virtual Channel Connections (VCCs) is both required and feasible by current and next generation ATM switches.

It follows that a scheduling algorithm is needed, which decides (possibly at each time slot) which partition's cell gets served next. The basic requirements to this algorithm are that

1. it should guarantee statistically the negotiated QoS parameters to each VCC,
2. it should protect flows from the possible drawbacks caused by the misbehavior of one or more flows, and
3. it should optimize the "overall" network performance in the sense that each VCC gets the highest possible quality of service while network utilization is also kept

high.

In the proposed scheme of the WF scheduler the control of the traffic is realized via weighting functions associated to the individual flows. Through the calculation of the weights the QoS requirements given by the source of the traffic, the instantaneous (current) QoS parameters of the VCC under service, and the network resources allocated for the VCC under service are taken into account.

Since the algorithm is to be executed real time, it should be simple and feasible by current technologies. The basic idea of this traffic control algorithm was published in [**C3**], while the performance evaluation of this scheme was presented in [**C4**].

The main idea of the algorithm is straightforward: the QoS is handled as a 3-dimensional space in which the QoS requirements are surfaces and the instantaneous QoS parameters of services are represented by points. These points describe the individual VCCs exactly (or with arbitrary precision). If all of the points are inside the region bordered by the corresponding surface then QoS requirements are met. Behind this, a complete buffer partitioning coupled with complete link capacity sharing [40] architecture of ATM multiplexers is used, allowing an "individual handling" of VCCs requesting sharply different QoS measures from the network.

The traffic control algorithm works as follows: it selects the connection whose current QoS parameters are the worst compared to the corresponding requirements and has a packet to send and schedules it. Graphically we can say that the server like a "tracking beam" tries to draw the QoS-vectors from the distance towards the security regions (see Figure 3.1).

3.1 The architecture of the WF scheduler

The architecture of the Weighting Function scheduler is the following: Every connection has its own dedicated buffer space. In the buffers the constant length packets are queued according to their arriving time[1]. The sequence of the packets arrived in the same time slot is arbitrary. In every time slot a nonnegative real scalar value called *weight* is assigned to each non-empty queue. The weight of empty queues is set to minus infinity. The scheduler forwards packet from that queue which has the greatest weight (see Figure 3.2).

I have presented an appropriate set of weighting functions for the above described scheduler architecture according to the requirements of ATM quality of service. This set

[1]Actually according to the arrival time slot. We can assume discrete arrival and departure time because of the constant packet length.

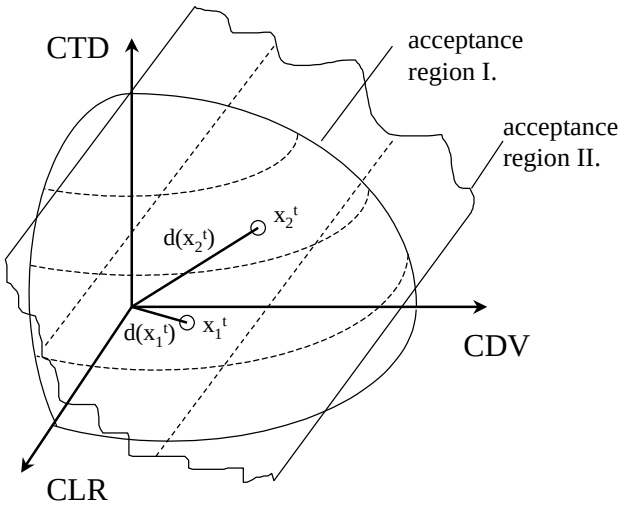

Figure 3.1. The 3D QoS space

incorporates all of the service classes defined in ATM and takes into account the most important three QoS parameters. The general form of the weighting functions for the different service classes is the following:

$$
\begin{aligned}
W_{CBR} &= a_{CBR} * \frac{LC_{CBR}}{SUM_{CBR} * CLR_{CBR}} + b_{CBR} * \frac{T_{CBR}}{CTD_{CBR}} + \\
&\quad + c_{CBR} * max(T_{CBR} - CTD_{CBR} - \frac{2}{3} * CDV_{CBR}, 0) \\
W_{rtVBR} &= a_{rtVBR} * \frac{LC_{rtVBR}}{SUM_{rtVBR} * CLR_{rtVBR}} + b_{rtVBR} * \frac{T_{rtVBR}}{CTD_{rtVBR}} + \\
&\quad + c_{rtVBR} * max(T_{rtVBR} - CTD_{rtVBR} - \frac{2}{3} * CDV_{rtVBR}, 0) \\
W_{nrVBR} &= a_{nrVBR} * \frac{LC_{nrVBR}}{SUM_{nrVBR} * CLR_{nrVBR}} \\
W_{ABR} &= a_{ABR} * \frac{LC_{ABR}}{SUM_{ABR} * CLR_{ABR}} \\
W_{UBR} &= \begin{cases} w_{UBR} & \text{if } K_{CBR},\ K_{rtVBR},\ K_{nrVBR},\ K_{ABR} \text{ are all} > 1 \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

where:

$$
\begin{aligned}
K_{CBR} &= \frac{d_{CBR} * (a_{CBR} + b_{CBR} + c_{CBR})}{W_{CBR}}; \\
K_{rtVBR} &= \frac{d_{rtVBR} * (a_{rtVBR} + b_{rtVBR} + c_{rtVBR})}{W_{rtVBR}};
\end{aligned}
$$

and

$$
K_{nrVBR} = \frac{d_{nrVBR} * a_{nrVBR}}{W_{nrVBR}}; K_{ABR} = \frac{d_{ABR} * a_{ABR}}{W_{ABR}}
$$

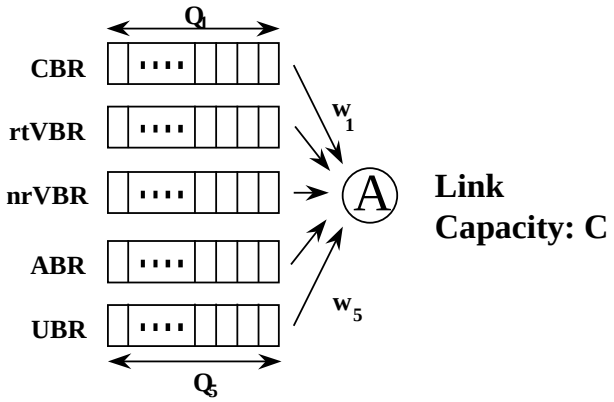

Figure 3.2. The buffer architecture

The value of a weighting function is equal to minus infinity if the queue is empty. Let LC_i be the number of lost cells of class i, SUM_i is the total number of cells of class i, and T_i is the waiting time of head-of-line (HOL) cell in the queue of class i.

This set of weighting functions was obtained based on Table 1.1, thus it reflects the different service classes' sensitivity to cell loss, delay and delay variation and also takes into account the required QoS parameters. Specifically, the weighting parameters a_i, b_i, c_i and d_i give the relative "importance" of a given QoS parameter in the weight of a given service class, while the constants CLR_i, CTD_i, and CDV_i are the negotiated (contracted) cell loss ratio, cell transfer delay and cell delay variation of the respective VCCs. These latter three parameters are referred to as QoS in this chapter.

Note that in our model the UBR service is not totally transparent to the other services. Even if there are cells of other classes in the buffer, a UBR packet may be delivered, because of the adaptability of our model. We can give a chance to the UBR if all other classes meet their QoS requirements with a given margin. However, the UBR has poor prestige in the network, if the other services needs the bandwidth. For example, the increase of the UBR load does not have any impacts on the QoS parameters of the other classes: the average cell transfer delay of UBR traffic significantly increases, while other classes have the same CTD.

I have developed a simulation program to do the performance analysis of the scheduling method. I have examined a single switch fed by the traffic of the 5 different ATM service classes.

My simulation results support the claim that the WF scheduler works as it was expected. In all realistic cases, the connections of guaranteed service classes met their

previously specified Quality of Service criteria.

The weighting function parameters used in the simulation scenarios were obtained by heuristics taking into account the possible requirements of connections served by the different service classes. These parameters cannot be used in general cases. It was not the aim of my research work to present an algorithm which can be used to determine the weighting function parameters. Note that the determination of the weighting function parameters can be considered as an optimization problem as it was presented in [**C3**].

3.2 Performance analysis

In the following simulation scenarios I have considered a link capacity of 45 Mbps, and a multiplexer with 5 input ports corresponding to the 5 service classes. The basic state of the traffic sources is the following: The CBR source is of 1.5 Mbps representing DS-1 circuit emulation. The rtVBR, nrVBR and UBR sources are all bursty and modeled as Interrupted Bernoulli Processes (IBPs) and are characterized by their peak and sustainable cell rates given in Mbps. The ABR source is assumed to be of rate based and is also modeled by an IBP. It is characterized by its peak and minimum cell rate (see Table 3.1). I have given the burstiness parameters of all services measured by the squared coefficient of variation of the interarrival time (i.e. the c^2 parameter).

Table 3.1. Basic input traffic characteristics

	PCR	SCR	MCR	c^2
CBR	1.5	-	-	0
rtVBR	15.0	3.0	-	9.44
nrVBR	22.5	1.0	-	20.75
UBR	45.0	5.0	-	26.06
ABR	22.5	-	4.5	-

Note that concerning the above mentioned link capacity a time slot in our discrete time model corresponds to 9.422 μs, which will be used as the time unit in the CTD and CDV values below. Tables 3.2-3.4 display the QoS requirements of different services, the buffer sizes available for different service classes and an appropriate parameter set for weighting functions, respectively. In Table 3.2 the CTD and CDV requirements are given in time unit. Note that no delay or delay variation parameters are negotiated for

the nrVBR or the ABR service classes and no QoS requirements are given for the UBR service.

Table 3.2. The QoS requirements for the service classes used in simulation

	CBR	rtVBR	nrVBR	UBR	ABR
CLR_i	10^{-5}	10^{-6}	10^{-7}	-	10^{-7}
CTD_i	3.0	5.0	-	-	-
CDV_i	1.0	2.0	-	-	-

Table 3.3. Buffer sizes in cells

Service class	CBR	rtVBR	nrVBR	UBR	ABR
Buffersize	5	8	12	250	80

Table 3.4. The parameter set of weighting functions

	a_i	b_i	c_i	d_i
CBR	0.1	0.6	0.9	0.5
rtVBR	0.2	0.3	0.4	0.5
nrVBR	0.6	-	-	0.7
UBR	-	-	-	-
ABR	0.4	-	-	0.6

and $w_5 = 6.0$

The weighting function parameters (Table 3.4) obtained by heuristics. These parameters have reference only to the traffic mixture, QoS requirements and network resources described by Tables 3.1-3.3. Minor variations (e.g. the increase of the traffic load) are allowed but sources with significantly different characteristics and requirements cannot be handled by the weighting function set parameterized according to Table 3.4.

Figures 3.3-3.5 display simulation result on CLR, CTD and CDV respectively, when I increased the CBR load from 1.5 Mbps up to 7.5 Mbps while other sources remained untouched. In this example I consider a single multiplexer with the weighting function parameter set described above. Due to the lower utilization of the connections (between

0.73 and 0.87) there is no considerable decrease in the QoS parameters of the traffic classes, which have a strict traffic contract with the network. We can see that all the negotiated QoS parameters met their requirements. The CLR and CDV of the CBR service class are slightly increasing according to the increasing load, but this increase effects the increase of the value of the weighting function of CBR class, i.e. the CBR service class gets more bandwidth and the QoS parameters finally stay within the negotiated region.

UBR service class has no any QoS requirements, so the load change causes changes only in the QoS parameters of this service, as it can be seen in the Figures 3.3-3.5.

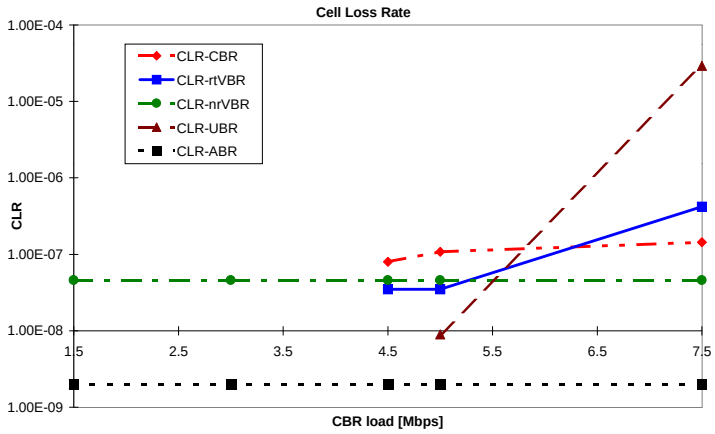

Figure 3.3. Cell Loss Ratio vs. CBR load

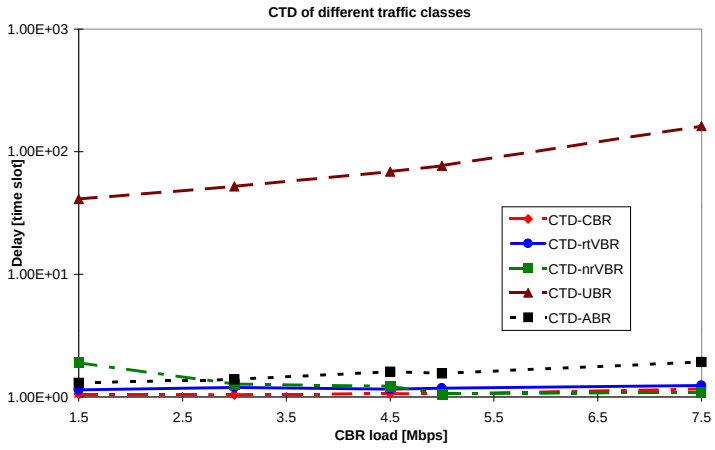

Figure 3.4. Cell Transfer Delay vs. CBR load

In the following scenario, we increase the sustainable cell rate of UBR traffic to 12 Mbps end we set the parameter d_3 to 0.9. The remaining three sources are in basic state

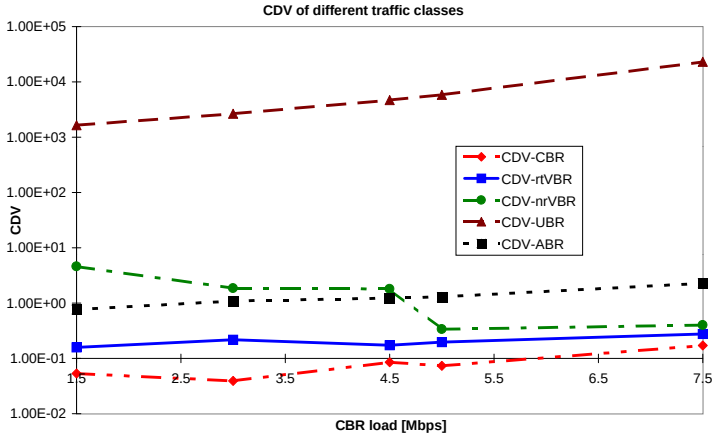

Figure 3.5. Cell Delay Variation vs. CBR load

and the other parameters are the same as in the previous scenario.

Figures 3.6-3.8 display the QoS parameters of a highly utilized link. The utilization goes from 88% up to 95%. The CLR parameters are similar to the previous case. The guaranteed services have constant cell loss except CBR, which has an increase by a decade. This resulted in the slow decreasing of the CDV parameter. The CDV of the other delay sensitive class (rtVBR) is normal. The nrVBR traffic class has no CDV assurance; the non-monotony of the curve comes from the abrupt step of its CLR at the same point.

Observe that the load increase affects the CLR of UBR only, as desired, since all other classes have strictly prescribed CLR values. The same behaviour can be observed for the CTD and CDV parameters of CBR and rtVBR classes. The ABR class is congestion controlled and sensitive to CLR only so its CTD and CDV behaviour is determined by the other classes.

In the following simulation studies we examine the dependence of QoS parameters on the increasing load of VBR traffic. In Figures 3.9-3.11 the load of rtVBR goes from 3 Mbps up to 12 Mbps. The sustainable cell rate of UBR source is set to 15 Mbps and the d_3 is set to 0.9; the other sources and parameters are in basic state.

The utilization is about 0.97 in the Figures 3.9-3.11. In these cases the CLR requirements of nrVBR and ABR classes are increased to 10^{-6} and 10^{-7}, respectively. Because rtVBR is a bursty traffic, there are more significant changes in the QoS parameters of the guaranteed classes. The CDV of CBR class gets in the near of QoS requirement (1.0). This, considering the increasing average delay of CBR, effects the decreasing of CLR at the last measuring point. The other curves meet their QoS requirements.

In Figures 3.12-3.14 the load of nrVBR goes from 1 Mbps up to 9 Mbps. The sustain-

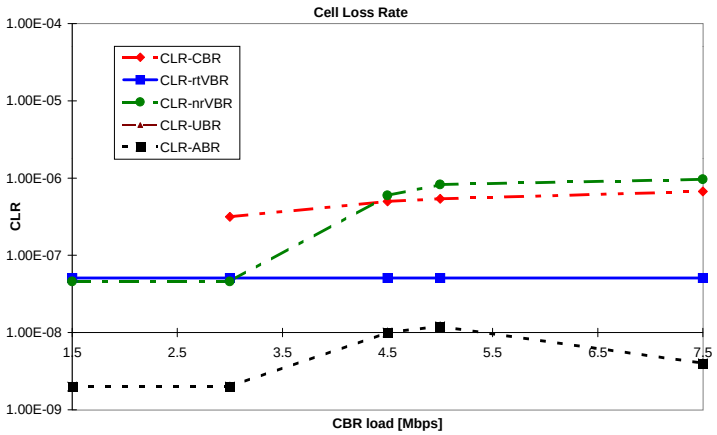

Figure 3.6. Cell Loss Rate vs. CBR load under heavy UBR traffic

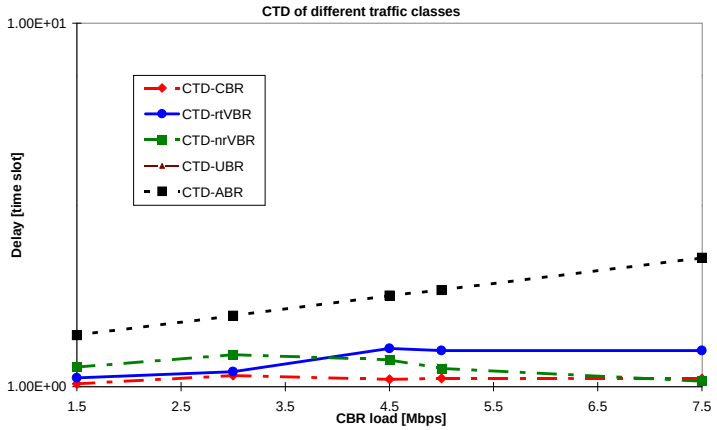

Figure 3.7. Cell Transfer Delay vs. CBR load under heavy UBR traffic

able cell rate of UBR source is set to 18 Mbps and the d_3 is set to 0.7; the other sources and parameters are in basic state.

In the following simulation studies we examine the dependence of QoS parameters on the increasing burstiness of VBR traffics. In Figures 3.15-3.17 the burstiness of rtVBR (measured by the squared coefficient of variation of the rtVBR interarrival time) goes from 5 up to 50. The sustainable cell rate of UBR source is set to 18 Mbps and the load of rtVBR source is 3 Mbps; the other sources and parameters are in basic state.

In the Figures 3.15-3.17 it can be seen excellently, that the weighting functions handle the different flows independent from each other. Real-time VBR traffic with increasing burstiness is arriving to the short buffer described in Table 3.3. The CLR of the rtVBR has linear increase with the burstiness. This causes a decrease in the CTD and CDV of

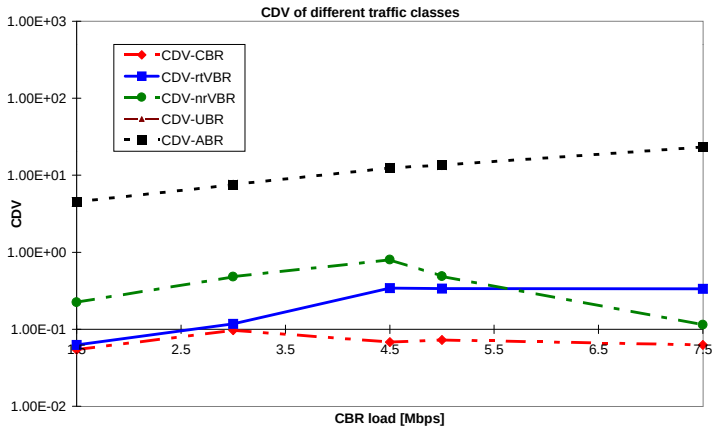

Figure 3.8. Cell Delay Variation vs. CBR load under heavy UBR traffic

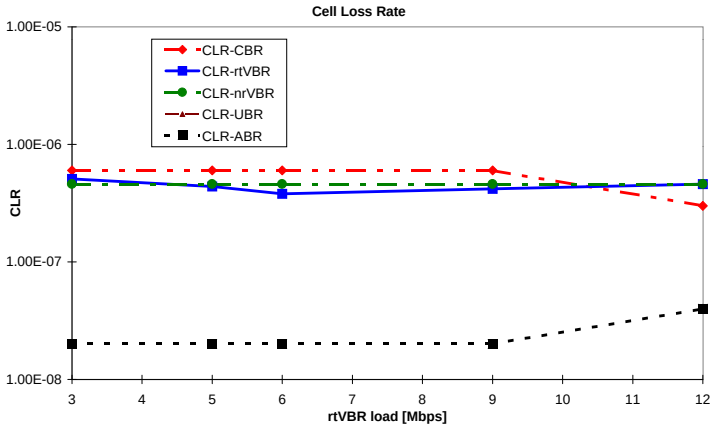

Figure 3.9. Cell Loss Rate vs. rtVBR load under heavy UBR traffic

the rtVBR, but for other classes it seems to be neutral.

At the last we show what is the dependence of the QoS parameters of service classes on the increasing load of UBR traffic. In Figures 3.18-3.20 the load of UBR goes from 5 Mbps up to 18 Mbps. Note that the burstiness of UBR traffic is constantly 26.06 in all cases. The other sources and parameters are in basic state.

The increase of the UBR load does not have any impacts on the QoS parameters of the other classes. It can be seen in Figure 3.19 that the average cell transfer delay of UBR traffic significantly increases, while other classes have the same CTD. Note that in our model the UBR service is not totally transparent for the other services. Even if there are cells of other classes in the buffer we may deliver an UBR cell, because of the adaptability of our model. We give a chance to the UBR if all other classes meet their QoS with a

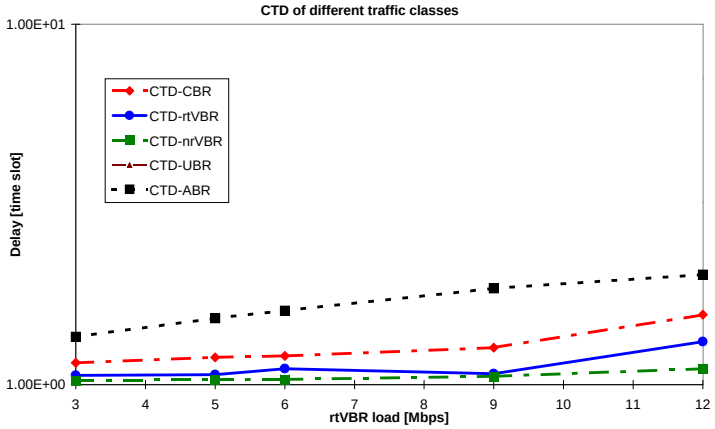

Figure 3.10. Cell Transfer Delay vs. rtVBR load under heavy UBR traffic

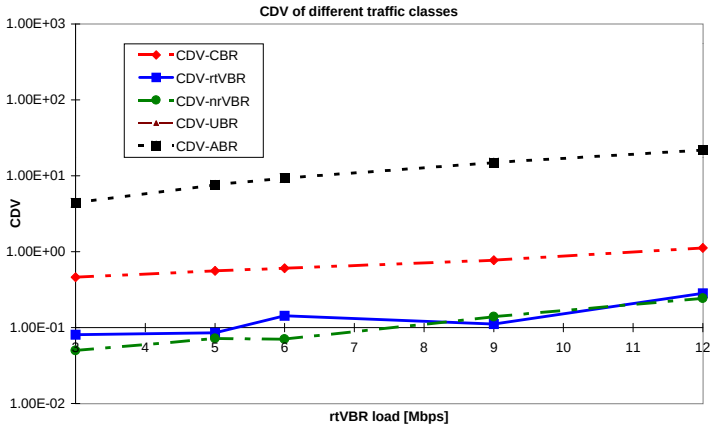

Figure 3.11. Cell Delay Variation vs. rtVBR load under heavy UBR traffic

given margin.

Comparing the performance of our algorithm with static scheduling rules - First Come First Served (FCFS) and Round Robin - it is notable that obviously no quality of service can be provided by the FCFS rule, so the delay and jitter requirement cannot be hold in the case of bursty background traffic[2]. Although, Round Round discipline provides worst case delay and jitter guarantees, these guarantees are not strict enough for the real time services[3]. I conducted simulations to compare the performance of our algorithm with Round Robin. Our results (see Figures 4.6-4.8 in Section 4.1.7) show that to achieve the same CLR the static traffic control schemes need 15-20% more buffer space for the

[2]Worst case delay is limited by the size of memory in the server.

[3]In the traditional Round Robin-case the delay is limited by the number of served queues.

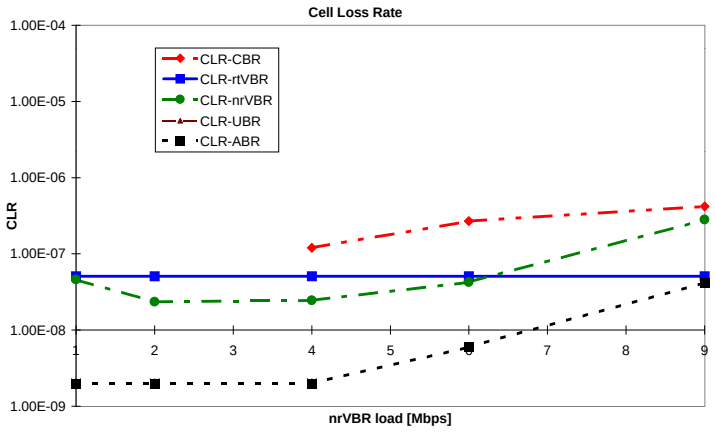

Figure 3.12. Cell Loss Rate vs. nrVBR load under heavy UBR traffic

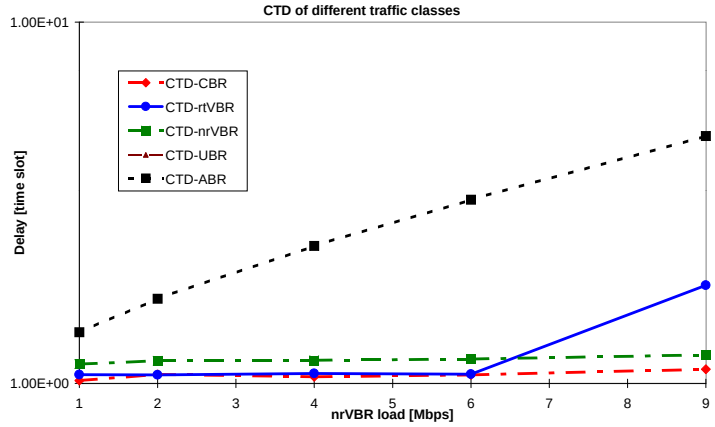

Figure 3.13. Cell Transfer Delay vs. nrVBR load under heavy UBR traffic

guaranteed services, moreover, the utilization of the network decreases.

3.3 Generalization of the Weighting Function scheduler

The weighting function set presented above can be used to operate a scheduler which transmits constant length packets. However, the architecture of the WF scheduler does not have such a limit, which means that the weighting function set can be extended to handle packets with variable length. If the packet length varies than the data loss can not be derived from the number of lost packets which means that the number of lost and successfully transmitted *bytes* should be counted instead of cells and in the loss-related element of the weighting functions the ratio of the lost bytes and the total number of

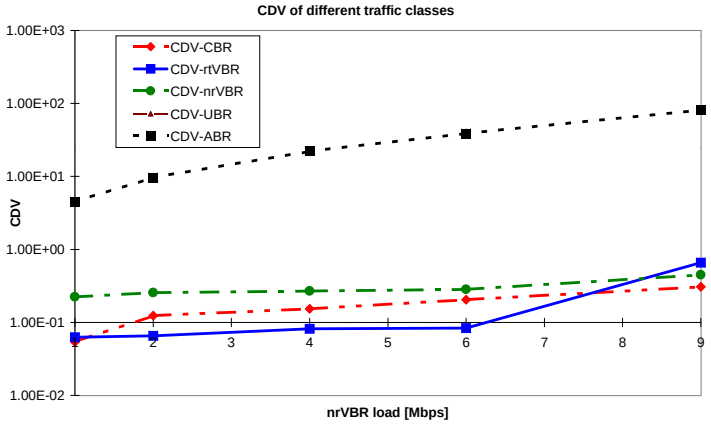

Figure 3.14. Cell Delay Variation vs. nrVBR load under heavy UBR traffic

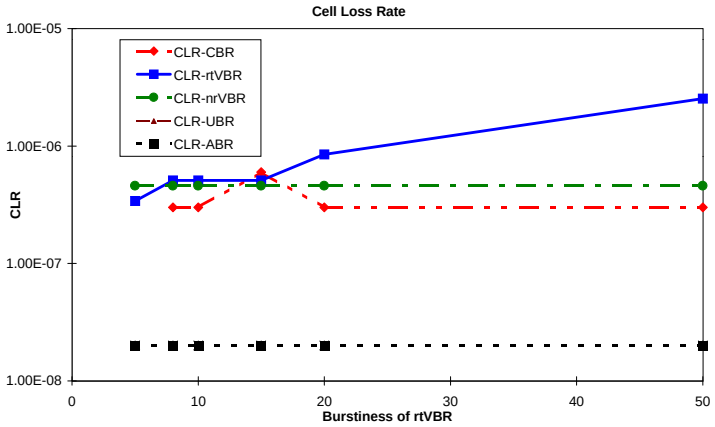

Figure 3.15. Cell Loss Rate vs. rtVBR burstiness under heavy UBR traffic

bytes should used. Further, the size of the HOL packets and the maximum packet size should be taken into account by adding a new element to the weighting functions, because the duration of the serving of a packet can influence the quality of service received by the other flows in the system.

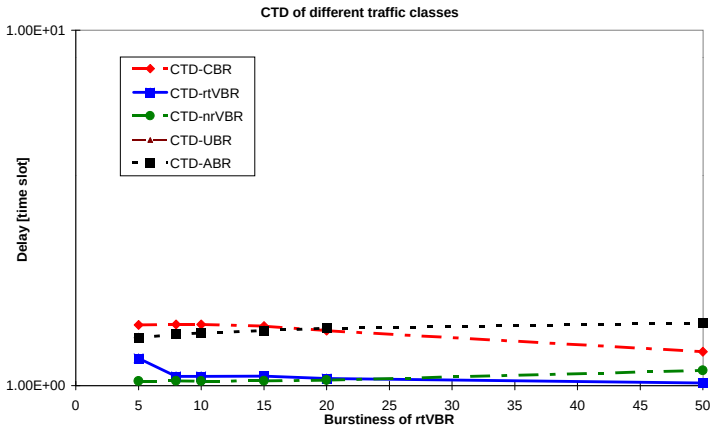

Figure 3.16. Cell Transfer Delay vs. rtVBR burstiness under heavy UBR traffic

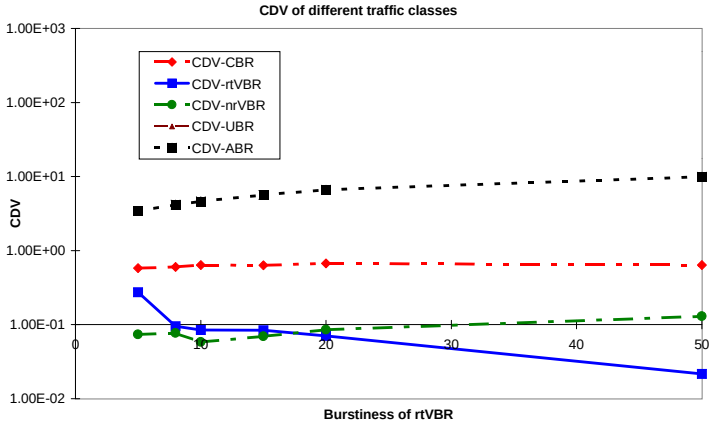

Figure 3.17. Cell Delay Variation vs. rtVBR burstiness under heavy UBR traffic

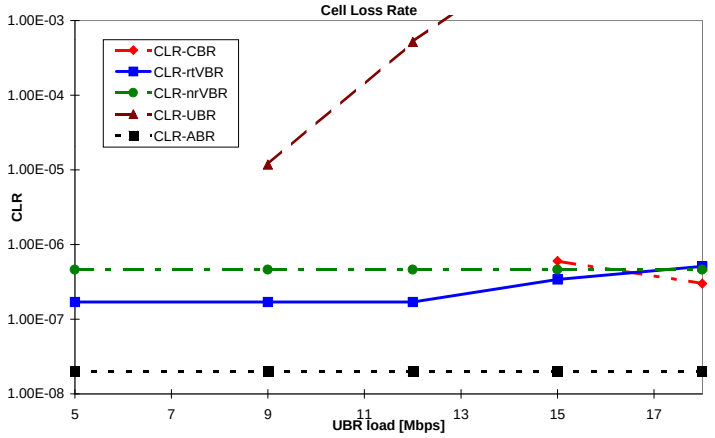

Figure 3.18. Cell Loss Rate vs. UBR load

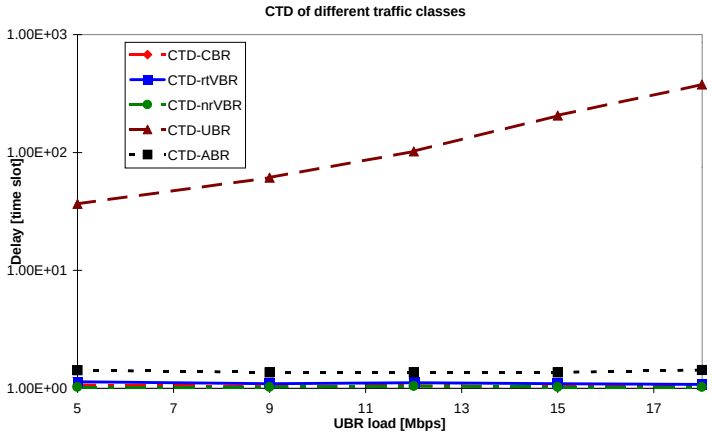

Figure 3.19. Cell Transfer Delay vs. UBR load

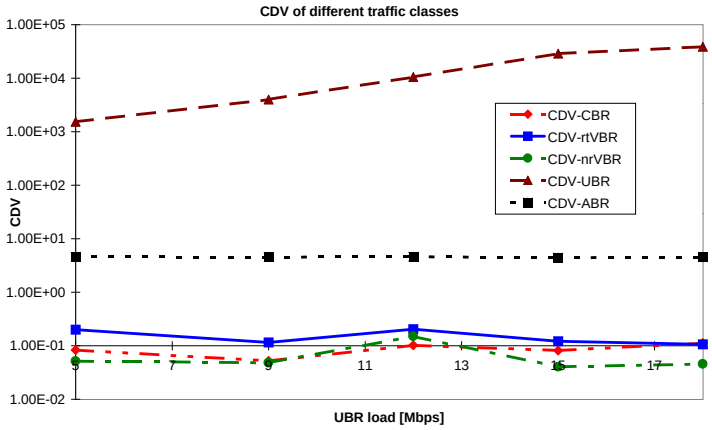

Figure 3.20. Cell Delay Variation vs. UBR load

3.4 Conclusion

In this chapter I presented a scheduler for high speed networks transmitting fix sized packets. The scheduler's operation is based on the evaluation of the weighting functions attached to the queues and it serves the first packet from the queue with highest actual weight. I proposed a comprehensive system of weighting functions for ATM networks. The elementary functions of this system can take into account the quality of service requirements of the individual flows, the quality of service received by that flow until the moment of the evaluation and - in the case of UBR traffic - the state of the other competing flows.

I used simulation to validate the proposed weighting function structure. In the presented scenarios the dependencies of the above mentioned Quality of Service parameters (Cell Transfer Delay, Cell Delay Variation, and Cell Loss Ratio) were investigated as a function of the rate and burstiness of the sources belonging to the different service classes. To accurately understand the working of the scheduler some of the simulations were done under very high link utilization (approx. 0.95).

It can be stated from the presented results that the scheduler meets my expectations: it transmits packets of different types of connections without any considerable degradation in quality of service regardless of the behavior of the other flows. The isolation of the traffic arising from different sources was also realized by this structure of weighting functions.

Chapter 4

Traffic control with Advanced Round Robin

As it was shown in Chapter 3 the scheduler based on weighting functions works very well in practical cases. However it is a hard task to analyze it and to give worst case bounds for it. This was the motivation to develop Advanced Round Robin scheduler.

4.1 Advanced Round Robin scheduler

The results of this chapter were described and published in [**C5**, **C6**, **C7**, **J3**].

I have developed and analyzed the Advanced Round Robin (ARR) scheduler for packet switching networks using constant length packets.

The Advanced Round Robin scheduler is a modified version of the well-known Round Robin scheme. The main goal of the modification was to make it capable to provide deterministic quality of service for guaranteed traffic classes.

4.1.1 The architecture of the ARR scheduler

I have introduced and validated a new Round Robin-type scheduling architecture, called Advanced Round Robin.

Round Robin is a well-known method to serve systems with multiple queues. It has a considerable advantage: it is very easy to give delay bounds for the applications. However, this worst case delay bound can be too loose because of the huge number of switched connections. This is the reason why I modified the Round Robin scheme, such that some flows could have access to a more frequent service.

To increase the service frequency I have constructed the Advanced Round Robin algorithm. In the following I define the basic quantities of the Advanced Round Robin algorithm.

Definition 1 The **service period** (or "access period") of a flow is the time interval between the consecutive packet transmission opportunities of that flow including one of this service opportunities. As a matter of fact the difference between the ideal and real service periods could be treated as a jitter-like quantity. The service period of a group is generally the same as the service period of any flow in that group.

For example, in Figure 4.1 the length of the service period is 6 for the flow whose service opportunities were indicated with blue pattern. □

Definition 2 The **service cycle**[1] is the lowest common multiple of the service periods. In other words the series the sequence numbers of the queues which receive service opportunities one after the other are periodic functions of the time[2]. The shortest period of this function can be treated as a service cycle[3].

In Figure 4.1 the length of the *planned* service cycle is 12. □

Definition 3 The **service preference order** (referred to as "order" hereafter) of flow is the number of opportunities that the flow could gain service during one service cycle. The order of a group is the same as the order of any flow in that group. The service period of the flows with the order of one is the length of the service cycle.

For example, in Figure 4.1 the above mentioned flow whose service opportunities were indicated with blue pattern has an order of 2, while the flow whose service opportunities were indicated with light green pattern has an order of 3. □

The Advanced Round Robin scheme is the following: We form groups from the flows according to the required maximum delay and decide how many times the server should serve flows in a certain group during the service cycle. Then the flows of the group should be scheduled in a service cycle according to the service preference order. The access periods of a group are uniformly distributed during the service cycle.

The planned service sequence enumerates flows in the order in which they can transmit packets. The scheduler is a work conserving one [60], the length of a cycle in the planned

[1] In the most round robin-type schedulers presented in the literature the analogous quantity is called round.

[2] In a stable system, where no call arrives or leaves.

[3] According to the choice of starting point we could have more service cycle, as many as the length of the service cycle is in packets.

sequence is the upper bound of the length of a cycle in the realized sequence (for notations see Table 4.1). Obviously the consecutive service cycles may not be the same. A flow will receive service only if it has at least one packet in its buffer. This causes that the service periods of a service group inside a service cycle can also differ. Figure 4.1 helps to understand the above mentioned scheme of the Advanced Round Robin server.

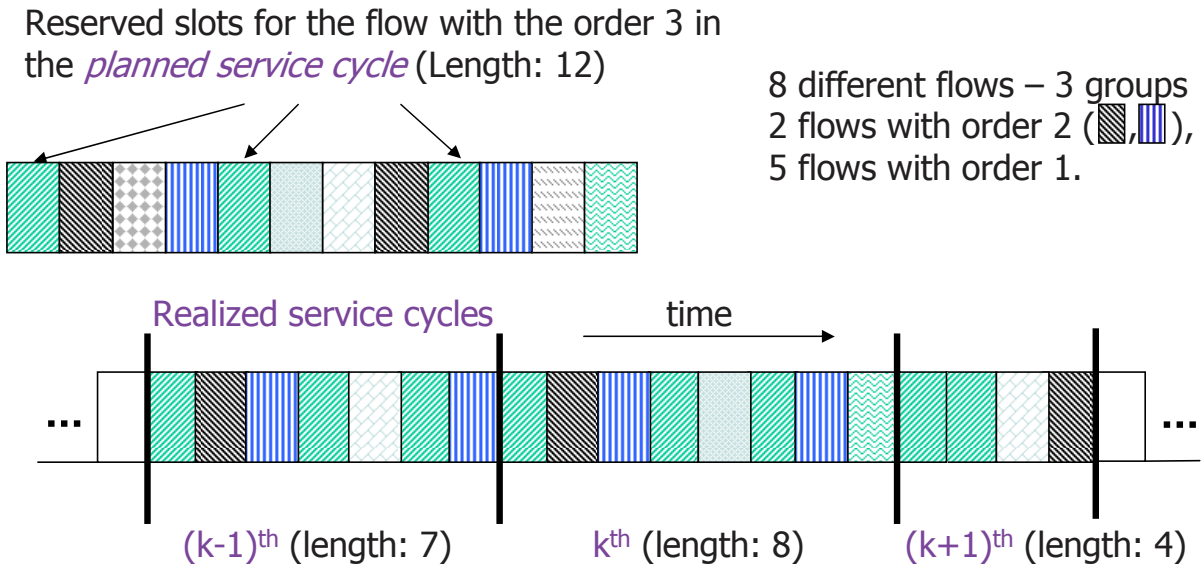

Figure 4.1. The architecture of ARR

Table 4.1 contains the notations about the Advanced Round Robin algorithm. We assume a fixed packet length system, e.g. ATM. We model the bursty traffic by an Interrupted Bernoulli Process with traffic intensity λ.

To support the characterization of the algorithm we define *utilization* (ρ) and *availability* ($\hat{\rho}$). The first one covers the traditional meaning of utilization, while the second one refers to the maximum permissible load of the scheduler taking into account the requested delay of connection j in group i ($D_{i,j\,req}$):

$$\rho = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i}\lambda_{i,j}}{C/l} \tag{4.1}$$

$$\hat{\rho} = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i}B_{i,j}/D_{i,j\,req}}{C/l} \tag{4.2}$$

I have carried out a performance analysis of the Advanced Round Robin scheduler. Note that the simulation results are related to a single switch scenario.

According to the goals of this work the simulation results can be divided into two groups:

Table 4.1. Notations of the Advanced Round Robin scheme

L	maximum length of the service cycle in packets
G	number of groups
N_i	number of flows in the ith group
k_i	the service preference order of group i
$B_{i,j}$	buffer length of the jth flow in group i in packets
$Q_{i,j}$	number of packets in the buffer of jth flow in group i
$\overline{Q}_{i,j}$	average number of packets in the buffer of jth flow in group i
$\lambda_{i,j}$	arrival intensity of the jth flow in group i [packet/sec]
l	length of a single packet in *bits*
C	capacity of the server in *bps*
g_i	minimum guaranteed service rate of any flow in group i in *bps*
$\overline{g}_i$	average service rate of any flow in group i in *bps*
$D_{i,j}$	maximum delay of the jth flow in group i [sec]
$\overline{D}_i$	average delay of the jth flow in group i [sec]
$J_{i,j}$	maximum difference between successive packet departures of the jth flow in group i [sec]
$\overline{J}_{i,j}$	average difference between successive packet departures of the jth flow in group i [sec]
ρ	utilization of the server
$\rho_{i,j}$	relative utilization of the queue of the jth flow in group i
$\hat{\rho}$	*availability* of the server

- first I show that QoS can be provided with our ARR algorithm for traffic flows which have applied for guaranteed service,
- then I examine how should be the network capacities increased to give the flows with the same traffic patterns the same service guarantees which they had in the reference system presented in [**J2**].

In a performance study I examined the impact of the traffic load and traffic burstiness increase. As the result of performance analysis I have experienced that the Advanced Round Robin algorithm needs more buffer space to achieve the same cell loss compared to the reference system, but because of the dedicated access right of a flow the different services are separated, so the extraordinary behaviour of a flow does not influence the

service quality of other connections. For the details see Section 4.1.7.

4.1.2 Theoretical limits and QoS guarantees provided by the ARR scheduler

I have provided the average delay and the jitter characteristics for the ARR scheduler. I have also given the delay and the jitter for the worst case scenario. Worst case bounds are used for guaranteed services while average values are typically considered at transmission of best effort services.

Worst case guarantees

For calculating the worst case delay ($D_{i,j}$), first we should express the length of the service cycle from the other quantities, then we proceed with the maximum delay of a flow. The maximum difference between successive packet departures ($J_{i,j}$) is a jitter-like quantity, and can be easily formulated assuming a backlogged queue (the queue of flow j in the ith group is not empty after the first departure). It is important to note that the access periods of service groups with an order 2 or more should be uniformly distributed in the service cycle. The mathematical formulation is the following:

$$L = \sum_{i=1}^{G} N_i k_i \tag{4.3}$$

$$D_{i,j} = \left\lceil \frac{LB_{i,j}}{k_i} \right\rceil \frac{l}{C} \tag{4.4}$$

$$J_{i,j} = \left\lceil \frac{L}{k_i} \right\rceil \frac{l}{C} . \tag{4.5}$$

Tighter delay bounds for special types of traffic can also be given, but this is not discussed in my work.

For the more picturesque analysis and better understanding the working of the scheduler I formulate here the *minimum guaranteed service rate* of any flow in group i:

$$g_i = \frac{k_i}{L} C . \tag{4.6}$$

Average delay guarantees

For the estimation of the average quantities I take the worst case guarantees as starting point. Two factors should be considered to capture average characteristics:

- the buffer of a flow is usually not full, which means that $B_{i,j}$ can be substituted by $\overline{Q}_{i,j}$, and
- the length of the realized service cycle is lower than L in general, which can be taken into account by multiplying L with ρ.

The estimation of the average difference between successive departures ($\overline{J}_{i,j}$) goes in a similar way:

$$\overline{D}_{i,j} = \frac{L\,\rho\,\overline{Q}_{i,j}}{k_i}\frac{l}{C} \tag{4.7}$$

$$\overline{J}_{i,j} = \frac{L\,\rho}{k_i}\frac{l}{C}. \tag{4.8}$$

$\overline{Q}_{i,j}$ can be approximated from the M/D/1 queue (see e.g. [38]), i.e.:

$$\overline{Q}_{i,j} = \frac{\rho_{i,j}}{2(1-\rho_{i,j})}. \tag{4.9}$$

4.1.3 Classification of the ARR scheduler

I have shown that ARR meets the classification of LR servers. I have carried out the latency parameter of the ARR scheduler.

As several well known scheduling algorithms such as WFQ, VC, SCFQ, WRR and DRR, do belong to the class of Latency Rate servers (described in [53]) also does ARR. In this section we will show how ARR meets the classification of LR servers.

According to the definition a server belongs to the LR class if and only if for all times t after τ that jth busy period started and until the packets that arrived during this period are serviced

$$W_{i,j}(\tau,t) \geq max[0, \overline{g_i}(t-\tau-\Theta_i], \tag{4.10}$$

where Θ_i is the minimum non-negative number that satisfies the above inequality. The parameters involved by the definition called latency (Θ_i) and rate ($\overline{g_i}$).

The right-hand side of (4.10) defines an envelop to bound the minimum service offered to any backlogged session in group i in the jth busy period. However, using ARR we can also give such an envelop for the minimum service.

The average service rate $\overline{g_i}$ can be obviously evaluated dividing the guaranteed service rate with utilization ρ. The latency Θ_i is determined by the architecture of ARR. We should answer the question: how does a queue became empty after a greedy system start. In this case all queues are continuously backlogged and $\rho = 1$. Any queue in group i has

only the minimum guaranteed service rate g_i. For the beginning of serving the queue of connection j in group i we should wait at most $L\rho/k_i l/C$. At the end, the last packet is served when its last bit is served, which means l/C. Finally we get

$$\Theta_i = \frac{L\rho l}{k_i C} + \frac{l}{C} = \frac{l}{g_i} + \frac{l}{C} \tag{4.11}$$

for the latency [**C7**, **J3**].

I gave the worst case delay, the maximum backlog and the leaky bucket model for the ARR scheduler as a member of the class of Latency-Rate servers.

Also Stiliadis and Varma gave the characterization of LR servers in [53]. Based on their results we can give new bounds for the ARR, however these bounds might be looser than those which were evaluated based on the architecture of ARR.

Using the previously introduced notation the maximum backlog is as follows:

$$Q_{i,j}(t) \leq \sigma_{i,j} + g_i\Theta_i = l(B_{i,j} + 1 + \frac{k_i}{L}). \tag{4.12}$$

$$D_{i,j}(t) \leq \frac{\sigma_i}{g_i} + \Theta_i = \frac{lL\rho}{k_i C}(B_{i,j} + 1 + \frac{k_i}{L}). \tag{4.13}$$

The output traffic conforms to the leaky bucket model with parameters $(\sigma_i + \Theta_i g_i, g_i) = (l(B_{i,j} + 1 + \frac{k_i}{L}), \frac{k_i C}{L\rho})$.

In [34] Jiang proves that if a server belongs to the Latency Rate class it also belongs to the Guaranteed Rate class [24, 25] and vice versa. According to definition a scheduler is a GR server for a flow with error term β if

$$f^j \leq GRC^j + \beta, \tag{4.14}$$

where $GRC^j = max(a^j, GRC^{j-1}) + l^j/g_j$, $GRC^0 = 0$, a^j is the time the jth packet of the flow arrives to the scheduler and f^j is the time the jth packet finishes service from the scheduler. According to the conversion rules proven in [34] ARR is a member of class of Guaranteed Rate servers with guaranteed rate g_i and error term

$$\beta = \Theta - \frac{L^{min}}{g_i} = \frac{l}{C}. \tag{4.15}$$

4.1.4 Parameter conversion between ARR and GPS

A Generalized Processor Sharing (GPS) server serving N sessions is characterized by N positive real numbers, $\phi_1, \phi_2, \ldots, \phi_N$. The server is work-conserving[4] and operates at a

[4] A server is work-conserving if it is never idle whenever there are packets to send.

constant rate r. Let the amount of session i traffic served in the interval $[t_1, t_2]$ denoted by $W_i(t_1, t_2)$, then a GPS scheduler is defined as a server for which

$$\frac{W_i(t_1, t_2)}{W_j(t_1, t_2)} \geq \frac{\phi_i}{\phi_j}, \qquad j = 1, 2, \ldots, N, \tag{4.16}$$

for any session i that is continuously backlogged[5] in the interval $[t_1, t_2]$. From this definition immediately follows that every session has a minimum *guaranteed service rate* which can be expressed as

$$g_i = \frac{\phi_i}{\sum_{j=1}^{N} \phi_j} \, r. \tag{4.17}$$

In the Generalized Processor Sharing model the input traffic of the sessions can be shaped by a *token bucket* that is described by a token pool depth (σ) and a token generation rate (ρ).

Furthermore, the amount of traffic at the output of a token bucket shaped active source i in the interval $[t_1, t_2]$ assuming infinite capacity links[6] can be characterized by the function $A_i(t_1, t_2)$. If $A_i(t) = A_i(0, t) = \sigma_i + \rho_i t$, which means that session i starts with its maximal burst σ_i at time zero and continues to transmit with its maximal rate ρ_i then by definition session i starts *greedy*. If all sessions start greedy one gets a *greedy GPS system*.

Then, for every session i, the maximum delay D_i^* and the maximum backlog Q_i^* are achieved (not necessarily at the same time) when every session is greedy starting at time zero, the beginning of a system busy period[7]. Furthermore, assuming that for each session i $g_i \geq \rho_i$, then

$$Q_i^* \leq \sigma_i \quad \text{and} \quad D_i^* \leq \frac{\sigma_i}{g_i}. \tag{4.18}$$

The significance of this result is that for worst case behavior one has to analyze a greedy system 'only', which makes the analysis more tractable compared to any arbitrary arrival pattern imposed to the system.

I have shown the relationship between the Advanced Round Robin scheduler and the Generalized Processor Sharing scheduling discipline.

Thinking about the definition of GPS represented by (4.16) we could remark, that the orders k_i of ARR has the same role as ϕ_is in GPS. The main difference is that $k_i \in Z$ and $\phi_i \in R$. However, one can easily observe, that any real number ϕ_i can be

[5] the session buffer is not empty

[6] We can assume that the internal link between a queue and the leaky bucket associated with it do not constrain the service provided by the GPS server.

[7] an interval in which the server is continuously working

approximated with optional accuracy by a number $q_i \in Q$, which can be correspond to k_i/L or $k_i/max(k_j)$.

Considering this we can rewrite (4.16). For the service received by session (i, j) which is continuously backlogged in the interval $[t_1, t_2]$

$$\frac{W_{i,j}(t_1, t_2)}{W_{x,y}(t_1, t_2)} \geq \frac{k_i}{k_x}, \tag{4.19}$$

where $x = 1, 2, \ldots, G;\ j = 1, 2, \ldots, N_i;\ y = 1, 2, \ldots, N_x$.

Similarly, the minimum *guaranteed service rate* of any session in group i can be evaluated as

$$g_i = \frac{k_i}{\sum_{j=1}^{G} N_j k_j} C = \frac{k_i}{L} C. \tag{4.20}$$

If we want ARR to provide the same worst case guarantees as the GPS then from the comparing of (4.4) and (4.18) we get

$$\frac{\sigma_i}{g_i} = \frac{LB_{ij}}{k_i} \frac{l}{C}. \tag{4.21}$$

Substituting g_i from (4.20) we get $\sigma_i = B_{i,j} l$ for the token pool depth of the token bucket shaper.

Assuming the same considerations as in GPS for the worst case delay and taking into account, that the maximum backlog $\max Q_{i,j} \leq \sigma_i = l \cdot B_{i,j}$, we get

$$D_{i,j} \leq \frac{\sigma_i}{g_i} = \frac{LB_{ij}}{k_i} \frac{l}{C}. \tag{4.22}$$

which is not greater than the worst case delay evaluated in (4.4) from the architecture of ARR.

By these simple evaluations I have shown that ARR can be approximated by GPS. Obviously, all of the refinements of the worst case guarantees of GPS (e.g. [55]) can be easily applied in the analysis of ARR.

4.1.5 Fairness of the Advanced Round Robin scheduler

The fairness index introduced by Golestani in [23] help us to compare different scheduling algorithms. From the point of fairness the ideal scheduling method is Generalized Processor Sharing. However, GPS is fluid flow scheduler providing service for all backlogged connections in any moment, in the practice we have schedulers which forward one packet from one of the backlogged connections in a moment. In [53] Stiliadis and Varma give the fairness of the most popular schedulers.

Using the fairness definition of Golestani described in [23] I have evaluated the fairness index of the Advanced Round Robin scheduling algorithm. According to the definition the fairness index of a scheduling algorithm is the maximum difference between the normalized service received two backlogged connections over an interval in which both are continuously backlogged.

$$\left|\frac{W_i(t_1,t_2)}{g_i} - \frac{W_k(t_1,t_2)}{g_k}\right| \leq F. \tag{4.23}$$

The interval (t_1, t_2) can be divided into two parts. The first part consists of many full[8] realized service cycles in which the normalized serviced received by two continuously backlogged connections is the same. The second part is the interval $(\tau, t_2) \leq \frac{Ll}{C}$. In a full cycle session (i, j) with an order of k_i receives its service opportunities of k_i uniformly distributed, consequently the minimum and maximum service received by connection (i, j) in this fragment of a service cycle described by (4.24) and (4.25), respectively. That means, that as a function of t_2 we have at least 0, at most k_i service opportunities in this fragment cycle.

$$\min W_{i,j}(\tau, t_2) = l \left\lfloor \frac{(t_2 - \tau)Ck_i}{Ll} \right\rfloor \tag{4.24}$$

and

$$\max W_{i,j}(\tau, t_2) = l \left\lceil \frac{(t_2 - \tau)Ck_i}{Ll} \right\rceil \tag{4.25}$$

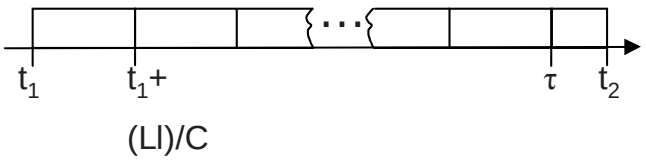

Figure 4.2. The calculation of fairness of Advanced Round Robin scheduler

Taking into account the other competing connections we can observe that they bounded the service received by each other because of their uniformly distributed service opportunities. If flow j in group i having an order of k_i receive s service in the last fragmental cycle $(0 \leq s \leq k_i)$ than connection (x, y) having the order of k_x will receive u service opportunities if we know that

$$\max\left\lfloor (s-1, 0)\frac{k_x}{k_i} \right\rfloor \leq u \leq \min\left\lceil (s+1, k_i)\frac{k_x}{k_i} \right\rceil. \tag{4.26}$$

[8] In this point of view a service cycle can be started at the finishing moment of the service of a packet of connection (x, y) having the order k_x and will be closed when the number of k_x service opportunities was provided to this connection by the ARR scheduler.

Moreover, in (4.26) we did not take into account that the service opportunities in the service cycle are fixed, which implies that in any real scenarios one of the two competing connections is always preferred i.e., it will get its service earlier. The importance of this is straightforward: we could omit the evaluation of one side of the above mentioned formula.

Finally, this leads us to the following fairness criteria [**C7**, **J3**]:

$$\mathcal{F}^{\mathcal{S}} = \frac{k_x + k_i}{k_x k_i} \frac{lL}{C} = \frac{l}{g_i} + \frac{l}{g_x}. \tag{4.27}$$

4.1.6 Comparison between the characteristics of ARR and other well known schedulers

A detailed analysis relating to the latency and fairness of several well known work-conserving schedulers can be found in [53]. In order to compare the latency and fairness of these servers with Advanced Round Robin we listed them in Table 4.2 *as they have been serving fixed-size packets.*

Based on these results we can find that the latency of Advanced Round Robin is not worse than the latency of any other method but GPS and in some limited scenarios WRR. Actually, considering a PGPS or a Frame-based Fair Queueing server in a fixed-size packet scenario we will have the same latency as with ARR. SCFQ and VirtualClock perform worse if the number of simultaneously backlogged sessions is more than 2. Deficit Round Robin and Weighted Round Robin work originally with fixed-size packets. WRR can achieve better results in limited scenes: if $k_i = 1$, in other words the flow has only one service opportunity in the cycle, the latency of ARR is higher with l/C which is the service time of one packet. The latency of DRR is higher in all possible cases.

Regarding the fairness our method performs as one of the bests. It is definitely better than PGPS, DRR, FFQ and VirtualClock, of course. It performs the same as the SCFQ. According to WRR in the case of both k_i and k_j are greater than 2 the fairness of ARR will be better.

4.1.7 Performance evaluation of ARR the scheduling method

In this section the ARR scheduling has been analyzed in realistic simulation scenarios. The performance of the ARR is compared to the performance of our reference system (see Section 3.2 and [**J2**]) and to the performance of the traditional Round Robin. Note that the simulation results of this section are related to a one switch scenario.

Table 4.2. Latency and fairness of several work-conserving servers

Server	Latency	Fairness
GPS	0	0
PGPS	$\frac{l}{g_i} + \frac{l}{C}$	$\max(\max(W_j + \frac{l}{g_i} + \frac{l}{g_j}, W_i + \frac{l}{g_j} + \frac{l}{g_i}))$, where $W_i = min((V-1)\frac{l}{g_i}, \max_{1 \le n \le V}(\frac{l}{g_n}))$.
SCFQ	$\frac{l}{g_i} + \frac{l}{C}(V-1)$	$\frac{l}{g_i} + \frac{l}{g_j}$
Virtual Clock	$\frac{l}{g_i} + \frac{l}{C}(V-1)$	∞
DRR	$\frac{3Ll-2lk_i}{C}$	$\frac{3Ll}{C}$
WRR	$\frac{Ll-lk_i+l}{C}$	$\frac{Ll}{C}$
FFQ	$\frac{l}{g_i} + \frac{l}{C}$	$\max(\frac{2Ll-lk_i}{C} + \frac{l}{g_i}, \frac{2Ll-lk_i}{C} + \frac{l}{g_j}, \frac{l}{g_i} + \frac{l}{g_j})$
ARR	$\frac{Ll}{Ck_i} + \frac{l}{C} = \frac{l}{g_i} + \frac{l}{C}$	$\frac{Ll}{k_iC} + \frac{Ll}{k_jC} = \frac{l}{g_i} + \frac{l}{g_j}$

The size of the fixed packet (cell) is l. We denote with g_i the rate allocated to connection i and with C the rate of the server. W_i is the maximum normalized service that session may receive in a PGPS server in excess of that in the GPS server and V is the maximum number of connections that can be backlogged in the server at the same time. In WRR and DRR, lL is the frame size and k_i is the amount of traffic in the frame allocated to session i.

Concerning the description of simulation scenarios the reader is requested to scroll back to Section 3.2. For the basic state test traffic parameters can be find in Table 3.1, quality of service requirements are given in Table 3.2, while Table 3.3 contains the buffer sizes.

In the following I concentrate only to the deviations from the basic scenario.

In Figures 4.3-4.11 the burstiness of rtVBR (measured by the squared coefficient of variation of the rtVBR interarrival time) goes from 5 up to 50. The sustainable cell rate of UBR source is set to 18 Mbps and 15 Mbps in the reference system and in the case of simplified algorithms, respectively, and the load of rtVBR source is 3 Mbps; the other sources and parameters are in basic state.

It can be seen in Figures 4.3-4.5 that the weighting functions handle the different services independent from each other. Real-time VBR traffic with increasing burstiness is arriving to the short buffer described in Table 3.3. The CLR of the rtVBR has linear increase with the burstiness. This causes a decrease in the CTD and CDV of the rtVBR,

but for other classes it seems to be neutral.

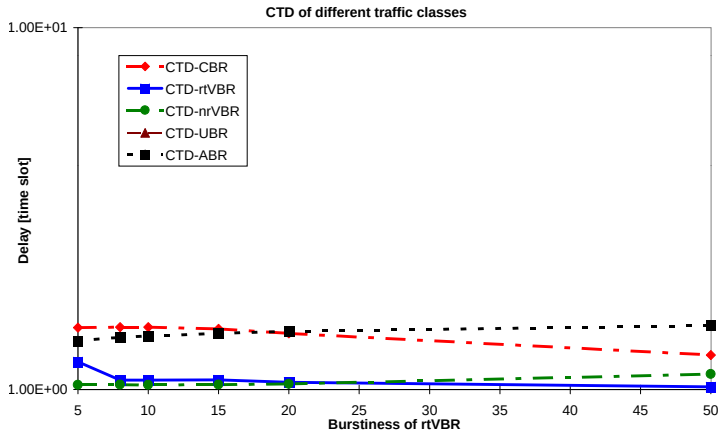

Figure 4.3. Delay vs. rtVBR burstiness (reference system)

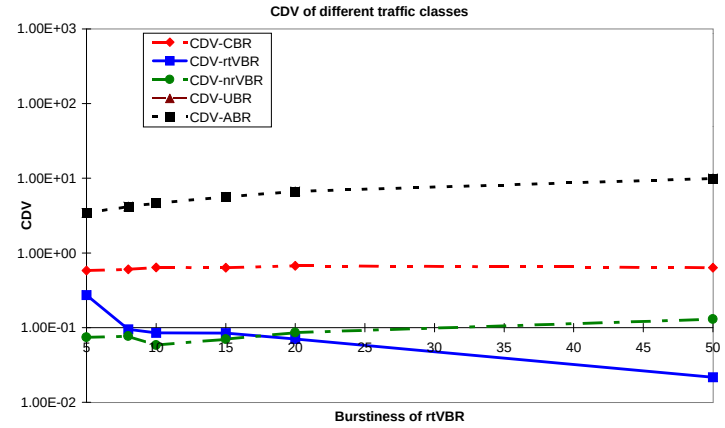

Figure 4.4. Jitter vs. rtVBR burstiness (reference system)

Figures 4.6-4.8 show the performance of the basic Round Robin scheduler. Our expectations have been justified: the QoS of real time VBR become worse with the increasing burstiness and there is no important change in the QoS of other classes. This rule makes the service of different classes independent. While curves are flat in the figures we can recognize that the requirements listed above are not always fulfilled. Neither the jitter of rtVBR nor the cell loss probability of VBR and ABR flows are within the acceptance region. In the latter case the increase of buffer space for VBR and ABR queues improves quality, but jitter can be even worse.

A solution to this problem is, for example, our ARR algorithm (see Figures 4.9-4.11). In this simulation study CBR and VBR queues are scheduled twice in a service cycle. The

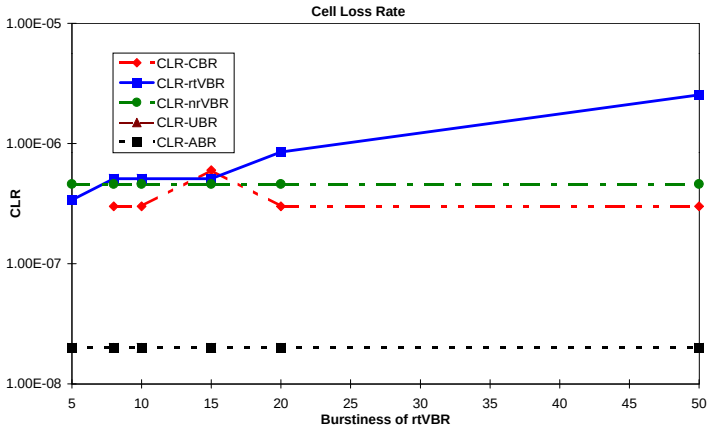

Figure 4.5. CLR vs. rtVBR burstiness (reference system)

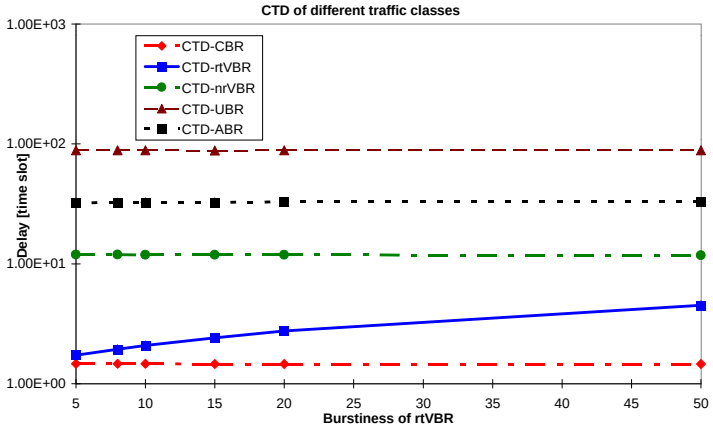

Figure 4.6. Delay vs. rtVBR burstiness (RR)

traffic pattern is the same as in the case of original Round Robin algorithm. Comparing the figures the reader can experience an evident advance. Not only the stressed three services have better QoS, but even the others should not suffer any significant drawbacks. The average delay and the jitter of rtVBR cells are hardly influenced by the increasing of their burstiness measured on the input side of the switch and the high cell loss probability of ABR traffic can be handled by increasing the buffer space or by changing the parameters of its rate control.

I examined also the impacts of increasing load of any traffic type. These results are not detailed here because they do not give further essential information to the appraisal of Advanced Round Robin, but we should note that the ARR worked according to the expectations.

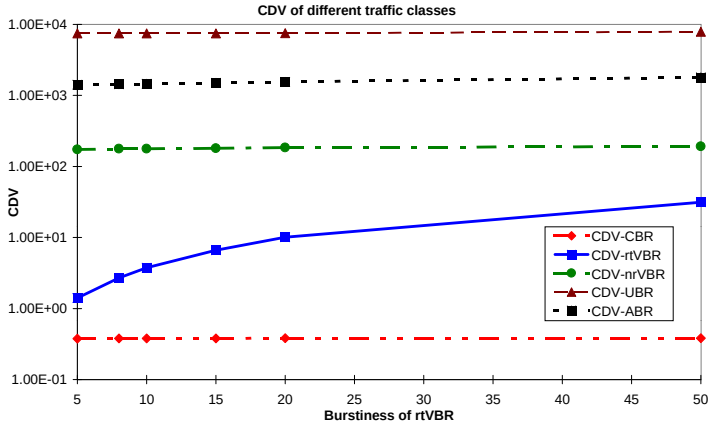

Figure 4.7. Jitter vs. rtVBR burstiness (RR)

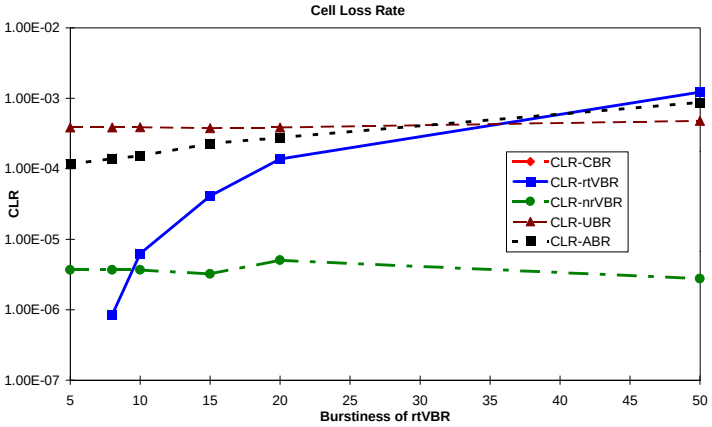

Figure 4.8. CLR vs. rtVBR burstiness (RR)

In the following simulation scenario the impacts of increasing burstiness of non real time VBR traffic are examined. To catch the real picture about the delay and jitter of VBR flows the buffers of rtVBR and nrVBR were extended to 10 and 50 slots, respectively. The load of nrVBR is 2 Mbps and its burstiness is set to be 10, 15, 20, 30, 50, and 100. The other connections are in basic state.

Comparing Figures 4.12-4.13 and Figures 4.14-4.16 we can observe that the Advanced Round Robin scheme sometimes gives better performance than our reference system. The main reason for this is that the reference system does not give any worst case guarantees, but only tries to do its best using the available resources. The Advanced Round Robin algorithm needs more buffer space to achieve the same cell loss compared to the reference system (see Figure 4.5 and 4.11 or Figures 4.13 and 4.16), but because of the dedicated

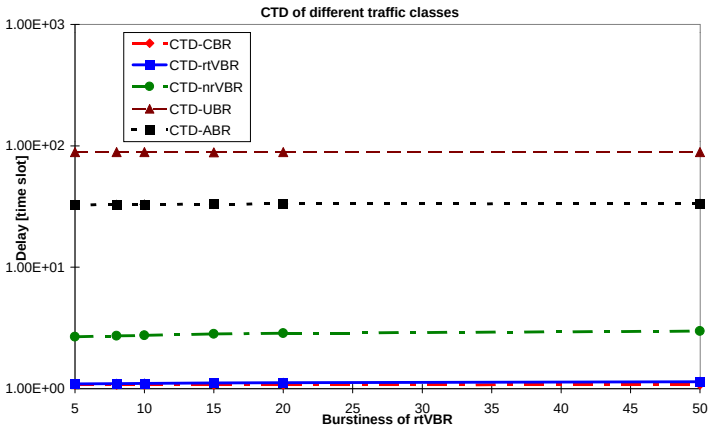

Figure 4.9. Delay vs. rtVBR burstiness (ARR)

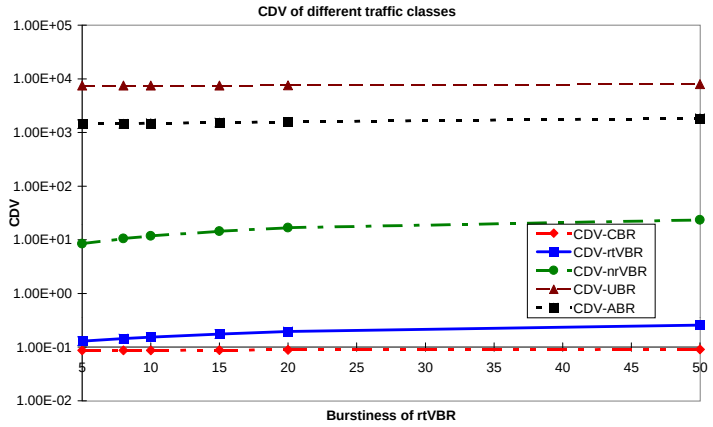

Figure 4.10. Jitter vs. rtVBR burstiness (ARR)

access right of a flow the different services are separated, so the extraordinary behavior of a flow does not influence the service quality of other connections (see e.g. Figure 4.9).

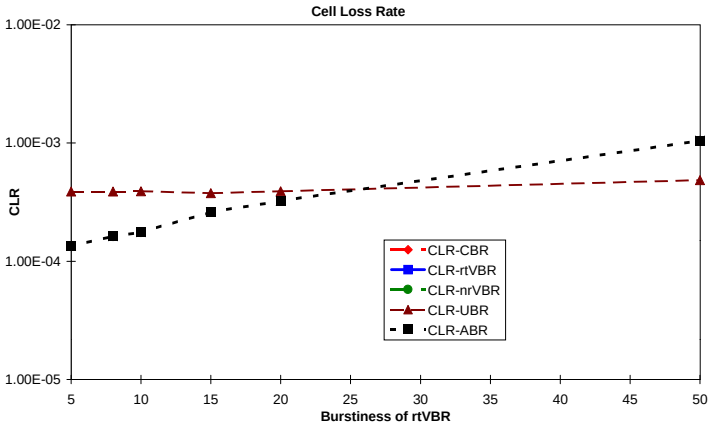

Figure 4.11. CLR vs. rtVBR burstiness (ARR)

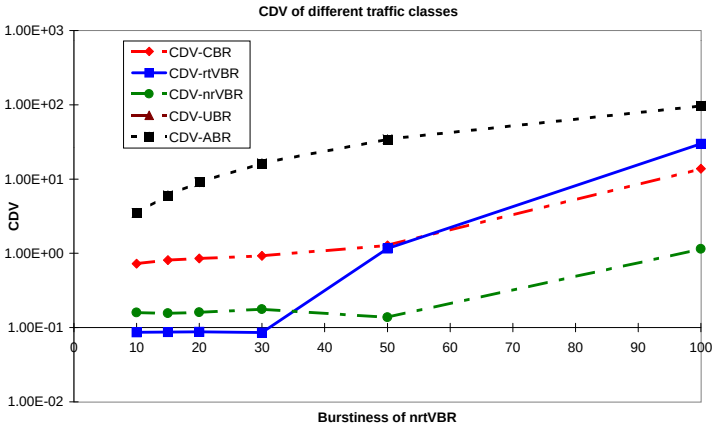

Figure 4.12. Jitter vs. nrVBR burstiness (reference system)

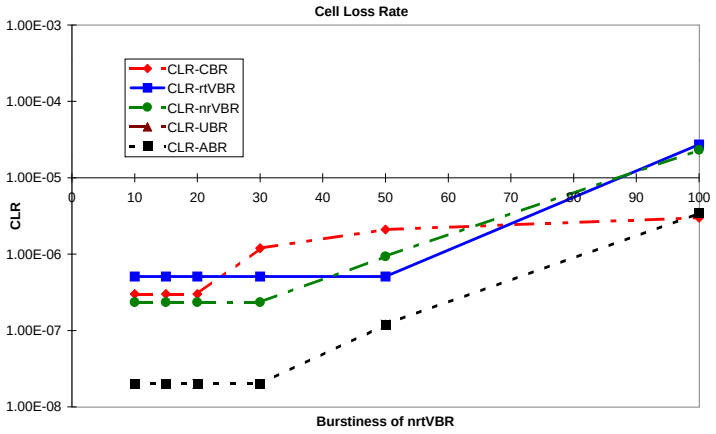

Figure 4.13. CLR vs. nrVBR burstiness (reference system)

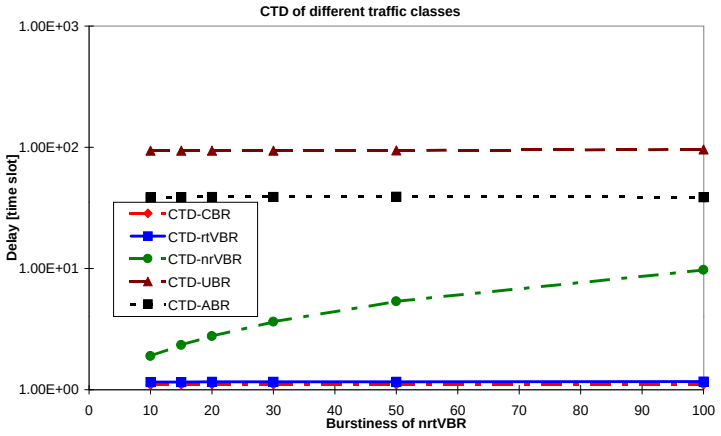

Figure 4.14. Delay vs. nrVBR burstiness (ARR)

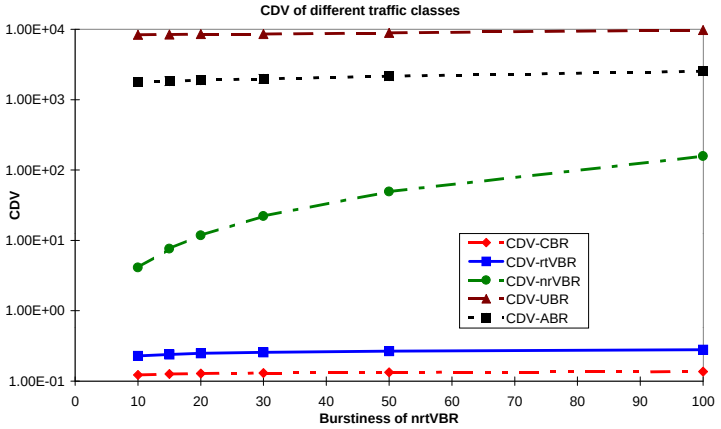

Figure 4.15. Jitter vs. nrVBR burstiness (ARR)

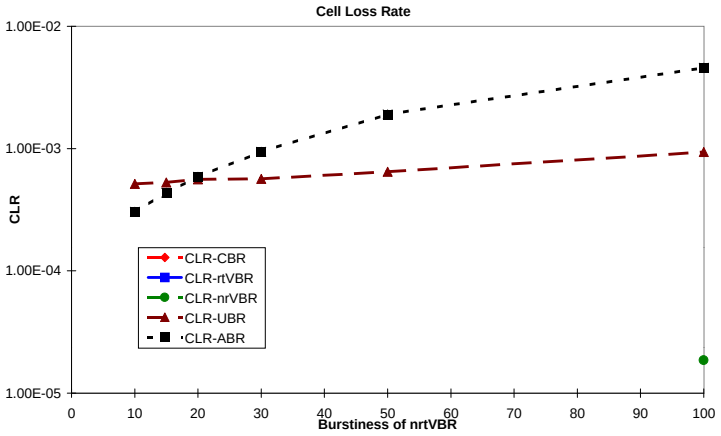

Figure 4.16. CLR vs. nrVBR burstiness (ARR)

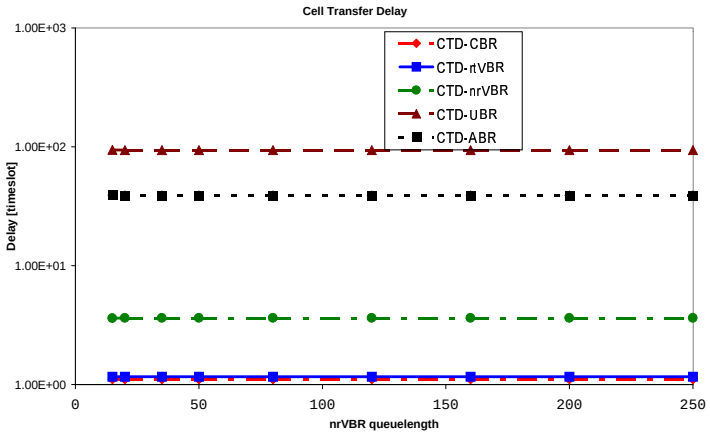

Figure 4.17. Delay vs. nrVBR queue length (ARR, bursty nrVBR)

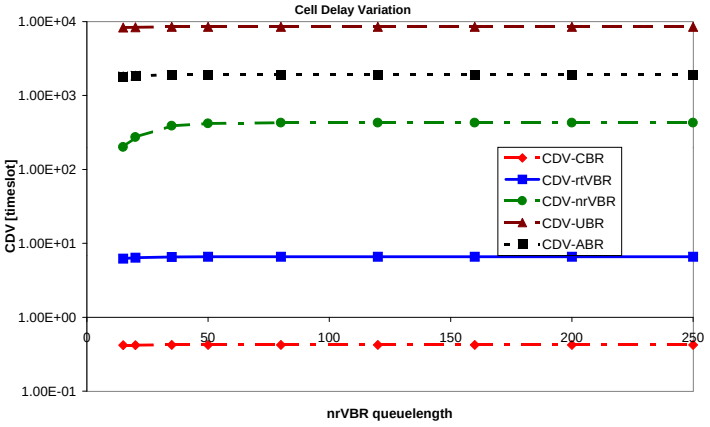

Figure 4.18. Jitter vs. nrVBR queue length (ARR, bursty nrVBR)

4.1.8 The price of simplicity

To explore the robustness of the Advanced Round Robin algorithm we made simulations with normal and more bursty nrVBR traffic using Round Robin and ARR sceduling algorithms. In the basic state the c^2 parameter of nrVBR flow was 20, while in the bursty case it was 30, which is greater than any other c^2 parameter in the basic state. In Figures 4.17-4.18 we can see that the increasing buffer space caused no significant change in the delay and jitter of flows. Only the jitter of the flow, whose queue length is extended (i.e. nrVBR) increases slightly before it become constant. The reason of this is that in the case of a small buffer the cells in the tail of the long bursts exceed the queue length and get lost. Using a larger buffer these bursts go through to the network and the last cells of them get high jitter. In Figure 4.19 the packet loss ratio is depicted as a function of the increasing buffer space. A doubled buffer space yields an order of magnitude packet loss rate decrease.

Figure 4.20 summarizes our results with the buffer space dependency of packet loss. The difference between the performances of the RR and the ARR algorithms is at least two orders of magnitude. Readers can recognize two things: i) in the case of lower burstiness the advantages of the ARR scheme are more pronounced, and ii) the slope of the curves of the ARR scheduling rule are greater, which means that the improvement of packet loss per extra buffer space is better.

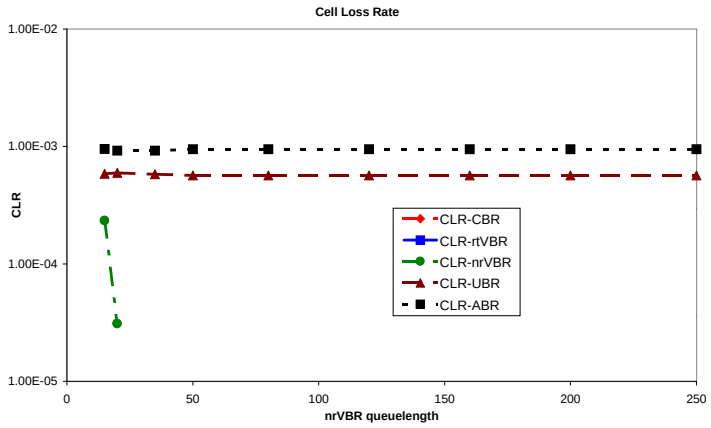

Figure 4.19. CLR vs. nrVBR queue length (ARR, bursty nrVBR)

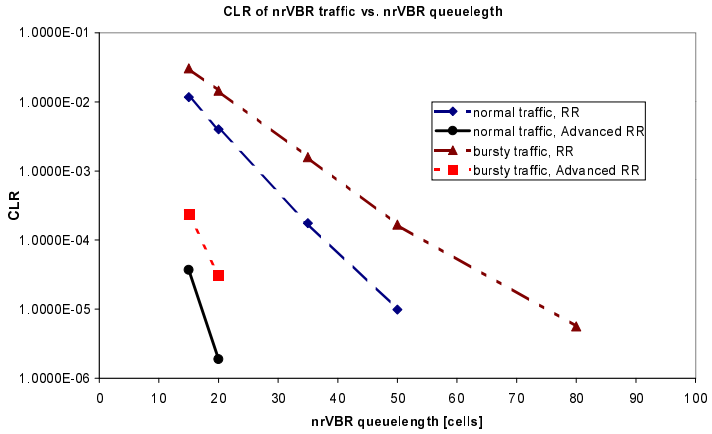

Figure 4.20. nrVBR packet loss vs. nrVBR queue length

4.1.9 Generalization of the Advanced Round Robin scheduler

Although, in the above presented analysis and simulation scenarios Advanced Round Robin was referred as a scheduler working in servers transmitting packets of constant length it is possible to use ARR in a more general environment. The generalization has two significantly different ways:

- we can take the advantage of the relationship with Generalized Processor Sharing, or
- we can attach counters to the individual flows which keep track of the number of bytes transmitted which leads to a Deficit or Surplus Round Robin type scheduler.

In the first possibility ARR is regarded to a special variant of GPS (with weights of rational numbers instead of real number). Just as GPS has a packet-by-packet version called PGPS [44] which closely[9] approximates GPS, thus a Packet-by-packet ARR should be constructed to apply it in the networks which transmit packets with variable length. This transformation keeps the quality of service guarantees and other characteristics of the Advanced Round Robin scheduler almost untouched but the computational complexity of the serving a packet increases, because in the background we we have to simulate the GPS service discipline.

The other possibility preserves the per packet work complexity of $O(1)$ during the scheduling of the flows within a service cycle (round). However, from the viewpoint of a

[9]The maximum difference is the transmission time of the packet with maximum length.

flow the counter means a limitation in the service which results the break down of the service guarantees, the latency and the fairness.

4.2 Construction of the service cycle for Advanced Round Robin scheduler

Previously we saw what service quality the ARR scheduling algorithm can provide for a certain parameter set. Here a backwards calculation should be done, the QoS requirements are given and we want to know the parameter set of the scheduler.

The results mentioned in Section 4.1 are based on the consideration that we already have an optimal organized service cycle in which the k_i service opportunities of connection j in group i are uniformly distributed for every $1 \leq j \leq N_i$ and $1 \leq i \leq G$. Obviously this can not be made for every possible combination of traffic parameters and service requirements if we want to have a work-conserving scheduler[10]. However, we can build suboptimal service cycles and can estimate the difference between the optimal and the suboptimal solutions. Suboptimal means here that we should balance between the best achievable arrangement and the effectiveness of the call acceptance control procedure presented in [**C5**] which is responsible for the building of the service cycle at the end. Note, that even in suboptimal solutions all of the delay and loss requirements are met and the server is a work-conserving one. However, it is possible, that in some scenarios a shorter service cycle could be established using an other method for the construction. If there is any method which finds the shortest service cycle in more case than ARR does, its complexity must be higher.

As a starting point of building service cycles we should have the order of each flow. The order of a connection can be obtained from the Call Admission Control algorithm (CAC) belonging to the ARR scheduler. The CAC function ([**C5**]) accepts a newcomer flow only if the appropriate order for the new and all the former connections can be calculated.

The sum of the order of the flows is the length of the service cycle. We enumerate

[10]The exact necessary condition for the possibility of constructing optimal service cycle is that the length of service cycle (L) should be divisible by k_i for all $1 \leq i \leq G$. On the other hand, L will be changed by accepting a new connection or finishing an old one. Let we suppose that there are 3 connections in the system with orders 3, 1, and 1, respectively. In this case the above condition does not met and the service cycle cannot be optimal. But if a new connection with the order of 1 demands for service we will have 6 slots in the cycle and an optimal arrangement can be e.g., that we allocate the even slots for the connection with order 3 and each remaining flow will get one of the odd slots . However, the condition is not sufficient which is easy to see if we consider 3 connections in the systems with orders 3, 2, and 1. Although, the length of the service cycle will be 6 and all orders are divisor of 6, the construction of optimal service cycle is impossible.

the flows by decreasing order, so the first flow will have the largest order and last will have the smallest. The first flow with the order of k_{max} means also that this flow should have k_{max} service opportunities during a service cycle. While these service opportunities are uniformly distributed in the service cycle there should be $(L/k_{max}) - 1$ free time slots between two of them. In practice, we reserve the $\lceil L(k_{max} - i + 1)/k_{max} \rceil^{\text{th}}$ time slot for the i^{th} service opportunity of the first flow. This means that the first reserved service opportunity of the first flow with the highest order will be the last time slot of the planed service cycle. For the second flow we begin the reservation with next-to-the-last time slot. In the following, if we found a time slot already reserved, we go further until the next free slot and continue the procedure from that (see Figure 4.21).

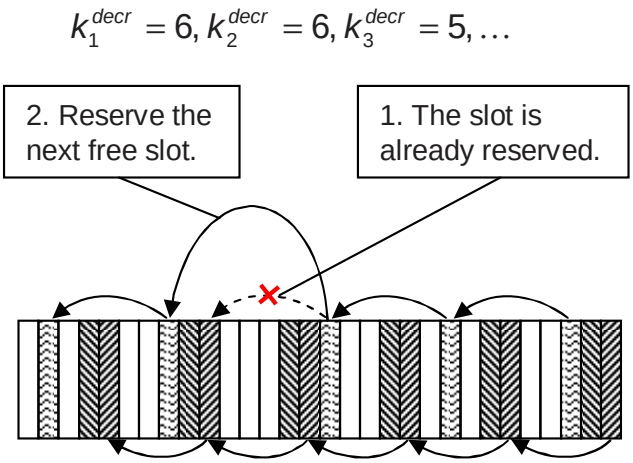

Figure 4.21. The building of service cycle for Advanced Round Robin server ($L = 30$)

A more precise algorithm can be found in [**J3**] and also in Appendix A.1.

Since there is a decision in the algorithm which in some cases leads back to the call admission control it seems to be inefficient at first sight. However, we can observe that the most delay and jitter sensitive connections will have the most higher order, so we put them among the first flows into the service cycle, when the cycle is mostly empty. Furthermore, in the course of simulations I experienced, that only a limited group of connections will be evolved.

The building of the service cycle is a recursive procedure. In some cases, when the server capacity is nearly exhausted it will take unacceptable much time to evaluate the appropriate order for each connection. This means that over 95% utilization and/or availability the convergence is too slow, we should rather refuse the newcomer connection.

4.2.1 Computation of the set of the orders

I have proposed an algorithm for the computation of set of the orders k_i for the flows admitted to the server.

The algorithm presented below describes how a new applicant can be accepted to the system and how groups should be reorganized if the new connection is accepted such that all connections meet their QoS requirements.

There are G groups with N_i $(1 \geq i \geq G)$ flows in each group. Group i has the order k_i. Flow j in the ith group $(1 \geq j \geq N_i)$ has an arrival intensity $\lambda_{i,j}$, a maximum acceptable delay $D_{i,j\,req}$ and a buffer length $B_{i,j}$. The server capacity is $C\,bps$ and the packet length is $l\,bits$.

The "newcomer" connection has an intensity λ_0 and should have no more delay and packet loss rate than $D_{0\,req}$ and $R_{0\,req}$, respectively.

Step 1 *Using per connection utilization value (ρ_0) calculate the buffer size (B_0) to satisfy the packet loss requirement. Each queue is approximated by the M/D/1 queueing approximation [49].*

$$B_0 = \left\lceil \frac{2\lambda_0 D_{0\,req} + \sqrt{4\lambda_0^2 D_{0\,req}^2 - 8\lambda_0 D_{0\,req} \ln R_{0\,req}}}{4} \right\rceil,$$

where $\lceil \ldots \rceil$ is the ceiling function. The proof of the above buffer calculation can be found in Appendix A.2.

Step 2 *Calculate an order k_0 for the new connection:*

$$k_0 = \left\lceil \frac{L\,l}{C} \frac{B_0}{D_{0\,req}} \right\rceil.$$

Step 3 *Calculate the new length of the service cycle:*

$$L' = k_0 + \sum_{i=1}^{G} N_i k_i.$$

Step 4 *Using (4.4) calculate the new delay guarantees ($D_{i,j}$) for every legal i, j pairs including the new flow. Compare these to the requirements ($D_{i,j\,req}$). If $D_{i,j} \geq D_{i,j\,req}$ add i, j pair to a list.*

Step 5 *If the list is empty then* **STOP**, *otherwise remove the first flow from the list and calculate a new order for it:* **GO TO** Step 2.

4.2.2 Conditions of acceptance of a new connection

I have given the necessary and sufficient conditions of acceptance of a new connections.

Based on the operation of the service cycle building algorithm to the ARR scheduler we can formulate the conditions of the acceptance of a new connection. With this theorem we can decide whether it is possible to accept the new connection in the appropriate group knowing its traffic parameters and QoS requirements without hurting the service quality of other flows.

Theorem 1 *The new flow applying for service can be accepted if and only if the following conditions fulfilled:*

$$\begin{aligned} M &\geq B_0 + \sum_{i=1}^{G}\sum_{j=1}^{N_i} B_{i,j} \\ \frac{C}{l} &\geq \lambda_0 + \sum_{i=1}^{G}\sum_{j=1}^{N_i} \lambda_{i,j} \\ C &\geq l\Big(\frac{B_0}{D_{0\,req}} + \sum_{i=1}^{G}\sum_{j=1}^{N_i} \frac{B_{i,j}}{D_{i,j\,req}}\Big), \end{aligned}$$

where M is the memory size available in the switch. □

The proof of the Theorem 1 is as follows:

PROOF The first two conditions are associated with the packet loss. They are straightforward: the admission function should reject the new call if there is not enough buffer space to achieve the required packet loss ratio or there is not enough server capacity to serve it. The proof of the third condition is the following. From (4.4) we can write the relationship between contracted maximum delay and the order of the flow:

$$D_{i,j\,req} \geq \frac{LB_{i,j}}{k_i}\frac{l}{C}$$

Rearranging this we get:

$$k_i \geq \frac{L\,l}{C}\frac{B_{i,j}}{D_{i,j\,req}}$$

Summing this for all j $(1 \geq j \geq N_i)$ then for all i $(0 \geq i \geq G)$ and taking into account that the length of the virtual service cycle is the sum of k_i's the evaluated inequality is a rearrangement of the third condition. ■

According to the definitions of utilization and availability the second and the third conditions of Theorem 1 are stability criteria.

4.2.3 Performance analysis of the ARR service cycle construction procedure

The algorithm was modeled in Wolfram Research's Mathematica. Several types of flows applied for service in two different scenarios. In the first case customers can generate flows which use by themselves a considerable amount of link capacity (e.g. 5%), while in the second case only "narrowband" services are available. The establishment of traffic parameters of flows was based on [13] and the Quality of Service requirements were determined using the ITU-T Recommendation I.356 [32]. Table 4.3 summarizes parameters and requirements. The link capacity was 620 Mbps.

Table 4.3. Traffic parameters and QoS requirements

Name	λ	D_{req}	R_{req}
video on demand	88542	$6 \cdot 10^{-4}$	10^{-7}
videophone	5334	$6 \cdot 10^{-4}$	10^{-7}
phone	167	$6 \cdot 10^{-4}$	10^{-7}
data transfer	500	$5 \cdot 10^{-2}$	10^{-6}
CD - music on demand	3675	$5 \cdot 10^{-2}$	10^{-6}
network game	50	$2 \cdot 10^{-2}$	10^{-5}
ftp	5334	10^{-2}	10^{-5}

In the "broadband" scenario the ratio of the services is 1/6 except videophone (2/15) and video on demand (1/30). In the "narrowband" scenario video on demand was not allowed, and the ratio of other connections was 1/6. The results of the performance evaluation can be seen in Table 4.4.

The maximum orders are dedicated to the maximum requirement set, i.e. the video on demand and the videophone services in the broadband and in the narrowband scenarios, respectively. Maximum buffer size is associated in both cases with the CD-quality music on demand services.

The conditions of Theorem 1 describe the borderline case of the CAC method. Obviously, the link capacity (C) cannot be fully exhausted because the bursty character of traffic may cause the accumulation of packets in the switch memory. In the same way, if the *availability* mentioned in (4.2) is very close to 1, it not only endangers the service quality but also increases the time needed by the call acceptance procedure. On

Table 4.4. Results of the analysis of ARR CAC algorithm

Scenario	Broadband case	Narrowband case
Number of flows	265	450
Utilization	0.929788	0.360321
Availability	0.780826	0.984603
Length of virtual service cycle	617	3288
Maximum orders	34	15
Number of flows with maximum orders	8	55
Maximum buffer size	408	146

Figures 4.22 and 4.23 we can see the availability and utilization of the ARR server and the construction time of the service cycle in the narrowband case and broadband case, respectively. Because the exact value of the construction time depends on the hardware of the switch, we have to deal with the trend of it. We can observe that availability and utilization have considerable steps in the broadband case. These steps are caused by the video on demand traffic with high bandwidth requirement. In the range of higher utilization and availability these steps induce high values of the service cycle establishment time, which means that the ARR algorithm cannot be used any longer. Although, there are no such steps of the availability and utilization in the narrowband case, the function of the construction time has a significant change in the range of $0.8 - 0.9$ of availability. As a rule of thumb, we find that an availability and/or utilization over 0.85 slows down the working of the CAC algorithm so much that it is better to refuse the new connection. Using the ARR algorithm the limiting condition is the *availability* if the required bandwidth of the applying connections is small[11] compared to the link capacity. Utilization of the ARR method can be increased by using more information from the traffic description (for example see the worst case delay of CBR flows).

I have carried out the performance analysis of the previously presented call acceptance control function simulating the working of the algorithm with mixtures of realistic traffic patterns. Based on the simulation results I have given the practical limits of the call acceptance control algorithm. Although the above presented CAC method can handle hundreds of connections with different QoS requirements, it works better,

- if the smallest required delay is at least hundred times greater then the service time

[11]The capacity should be at least one hundred times the bandwidth.

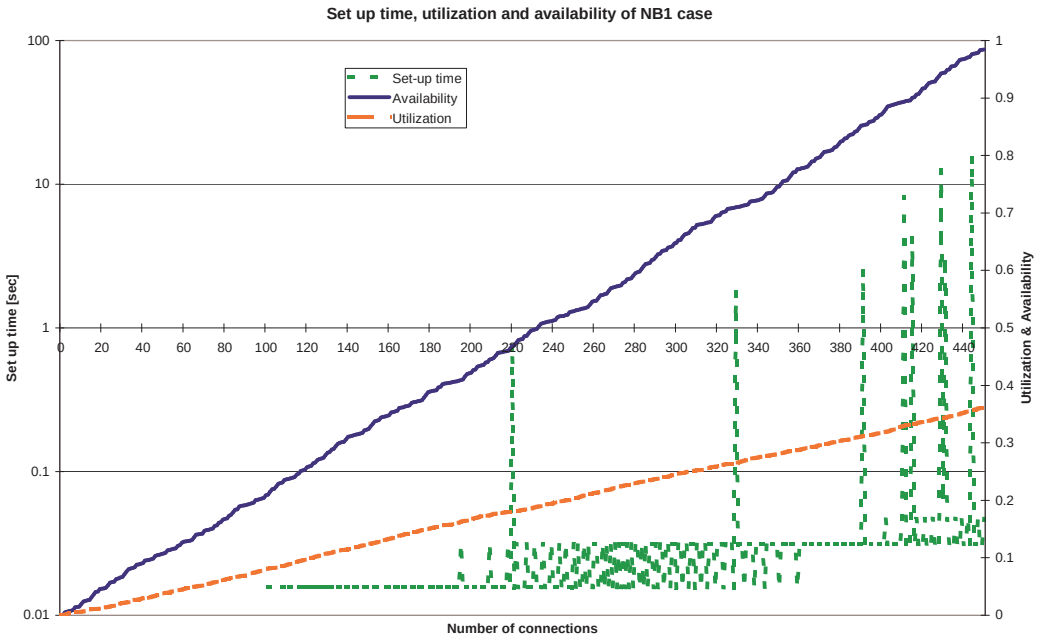

Figure 4.22. Time of the service cycle establishment in narrowband case

of one packet (l/C),

- if there are delay requirements assigned to not guaranteed services too, and
- if there are similar characteristics and requirements, i.e. natural service classes can be established.

4.3 Conclusion

In this chapter a new packet scheduler called Advanced Round Robin was presented and analyzed. This server is a special variant of the well-known Round Robin method, in which a flow can receive more service opportunities in a service cycle (round) and these opportunities are evenly distributed in the cycle. The architecture of the scheduler was outlined, and on this basis the deterministic and statistical delay and jitter bounds were evaluated. The Advanced Round Robin server was characterized as a Latency Rate server and the latency parameter was calculated as well as the relative fairness index of ARR was determined.

Because of the evenly distributed service opportunities it is obvious that the ARR may

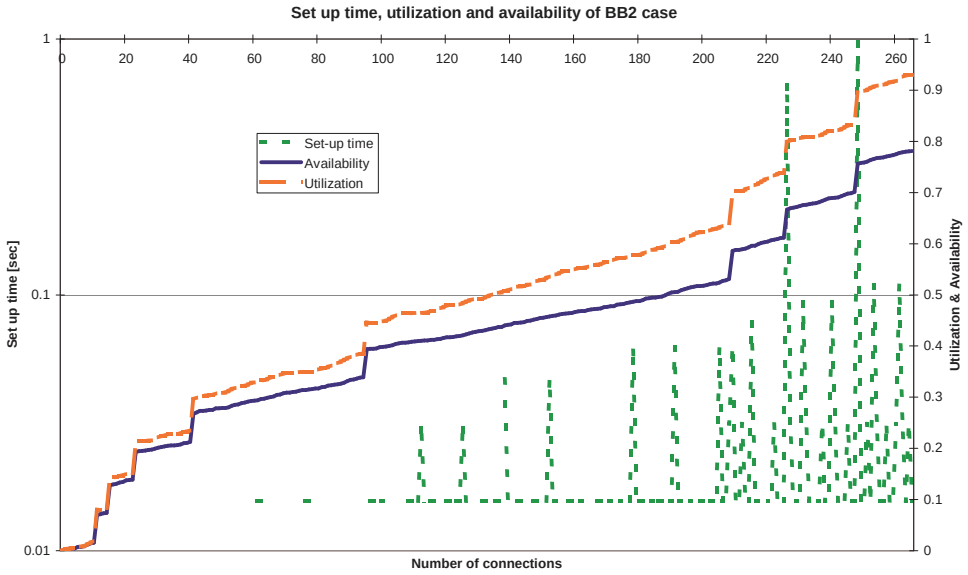

Figure 4.23. Time of the service cycle establishment in narrowband case

be potential candidate to emulate the ideal Generalized Processor Sharing in a simple and computationally feasible way. The limitations of this "substitution" was investigated, too.

The premise of the performance analysis of ARR is that the planned service cycle is an ideal one. Although, this condition cannot be always ensured, a new connection will be accepted only if it will not violate the quality of service of the calls already in the system. There are two cascading algorithmic procedures proposed for the calculation of the number of service opportunities allowed to the individual flows in one service cycle (orders) based on the quality of service requirements of the flows and for the construction of the service cycle based on the previously computed orders. From these two algorithms related to the preparation of service cycle of the Advanced Round Robin scheduler the necessary and sufficient conditions of the admission of a new flow were followed.

The service cycle construction was verified using simulation. In these scenarios flows have been arriving to the Advanced Round Robin server. The individual flows characterized by the randomly chosen pair of a traffic descriptor and required service. According to this characterization the server should maintain the service cycle in a reasonable time, which can be realized if the availability and/or the utilization is under 0.85. These results induced the formulation of practical limits about the usage of this method.

Chapter 5

Application of new results

The results of this dissertation are about the traffic control methods of constant packet length packet switched networks. A typical example is the ATM. Although the new protocols of IP-based networks seem to satisfy the demand on a general bearer service for the diverse applications of multimedia networks, ATM is still applied. As the protocol of backbone networks such as private networks there are many fields where it can be used. Furthermore, the data link layer of the xDSL access network is still ATM. in which we have a multiplexing possibility, called VC multiplexing described in [27]. This possibility is often unused yet, but recent 3Play services provided on twisted copper pairs may require the use of this technology.

However, the schedulers presented and analyzed in Section 3.1 and 4.1 can be used also in networks with variable packet length with some minor modifications, which were described in Section 3.3 and Section 4.1.9, respectively. Moreover, as it was presented Advanced Round Robin scheduler is a member of the Latency Rate class which enables to use it in the network of LR-servers in addition to achieve tight end-to-end delay.

The modification of the Advanced Round Robin service cycle construction procedure to make it capable to handle variable packet length leads to a GPS-like traffic control.

Chapter 6

Summary

In Chapter 2 of this dissertation I gave an overview on the packet scheduler methods which are considered to provide deterministic (and preferably statistic) quality-of-service guarantees. I also presented characteristic properties which are specific to each schedulers and allows us to classify schedulers according to different points of views.

I introduced a scheduling method for fixed packet length networks (e.g. ATM) in Chapter 3. This scheduler evaluates weighting functions for each flow competing to transmit a cell in a time slot. The proposed weighting functions take into account the service class to which the identical flows belong, the requirements laid down in the traffic contract at call acceptance, and the quality-of-service received by the flow till the evaluation time. This structure provides statistical quality of service guarantees - or can provide deterministic guarantees according to the parametrization of the weighting functions -, but may allow us to avoid the starving of the any flows - even the flows with the service type of Unspecified Bit Rate. The system of using dynamically changing priorities gives us the possibility that customers receive the service which they ordered and purchased but not more. Service providers need not over-prioritize premium traffic.

Chapter 4 is devoted to the combined packet scheduler and call acceptance function called Advanced Round Robin. The aim of this proposal was to present a system, which is easy to analyze, robust and provide statistical and deterministic quality of service guarantees for the accepted calls, and besides it is computationally feasible. To achieve these goals I introduced a scheduling architecture based on the service cycle established - and updated - by the call acceptance functionality and I carried out the detailed analysis of the ARR scheduler. The premise of the analysis is that the service cycle is an ideal one. Although, this condition cannot be not always ensured, a newcomer call will be accepted only if it does not violate the quality of service of the calls already under service.

I proposed an algorithm which can be used to maintain the service cycle whenever the orders (number of the service opportunities of a the individual flows in one service cycle) are given. The order of a flow depends principally on the quality of service required by the flow, and then on the parameters of the other flows under service. I proposed a method for the calculation of the orders of the flows. From these two algorithms associated with the preparation of service cycle of the Advanced Round Robin scheduler the necessary and sufficient conditions of the acceptance of a new call were derived.

When presenting a new scheduler it is important to compare it with other well-known solutions to accurately appraise it. Therefore I characterized Advanced Round Robin scheduler as a Latency Rate server and as a Guaranteed Rate server and derived the latency and error term, respectively. By the help of these results we can calculate quality of service guarantees (actually end-to-end delay bound, internal burstiness, and buffer requirements) for individual flows in the case of network of this type of servers. I also evaluated the fairness index of the ARR in order to compare it with other schedulers.

Finally, in Chapter 5 I outlined the possible applications of the presented methods in the field of traffic control.

Appendix

A.1 An algorithm to establish service cycle for ARR scheduler

The service cycle arrangement algorithm is the next:

Step 1 *Receive the order k_i attached to each flow j in group i from CAC.*

Step 2 *Enumerate the flows by decreasing order. Let denote k_x^{decr} the x^{th} element of this series. Note, that $x = 1 \dots \sum_{i=1}^{G} N_i$. Set $x = 1$.*

Step 3 *Reserve the $\lceil L(k_x^{decr} - i + 1)/k_x^{decr} \rceil^{th}$ time slot for the i^{th} service opportunity of the x^{th} flow. If we found a time slot already reserved, we go further until the next free and continue the procedure from that.*

Step 4 *If $x < \sum_{i=1}^{G} N_i$ increment x and go back to Step 3.*

Step 5 *Calculate worst case delay and jitter of all connections based on the ideal service cycle. If there is just one requirement violated, calculate new order for that connection and go back to Step 1. Otherwise STOP.*

The algorithm is very simple. Because of the margin in resources due to limitations according to throughput and availability in Step 5 the backward direction is almost in all cases omitted.

We have already mentioned that even if the necessary condition for the possibility of optimal arrangement is met we can find traffic scenarios, in cases of which our algorithm cannot establish the optimal service cycle. Sometimes because it is impossible (see Section 4.2), but we can present traffic scenarios too where with the use of heuristics we can achieve the optimal service cycle. As an example, let we consider 6 flows with the orders of 3, 3, 2, 2, 1, and 1, and represent them by a, b, c, d, e, and f, respectively. Our algorithm

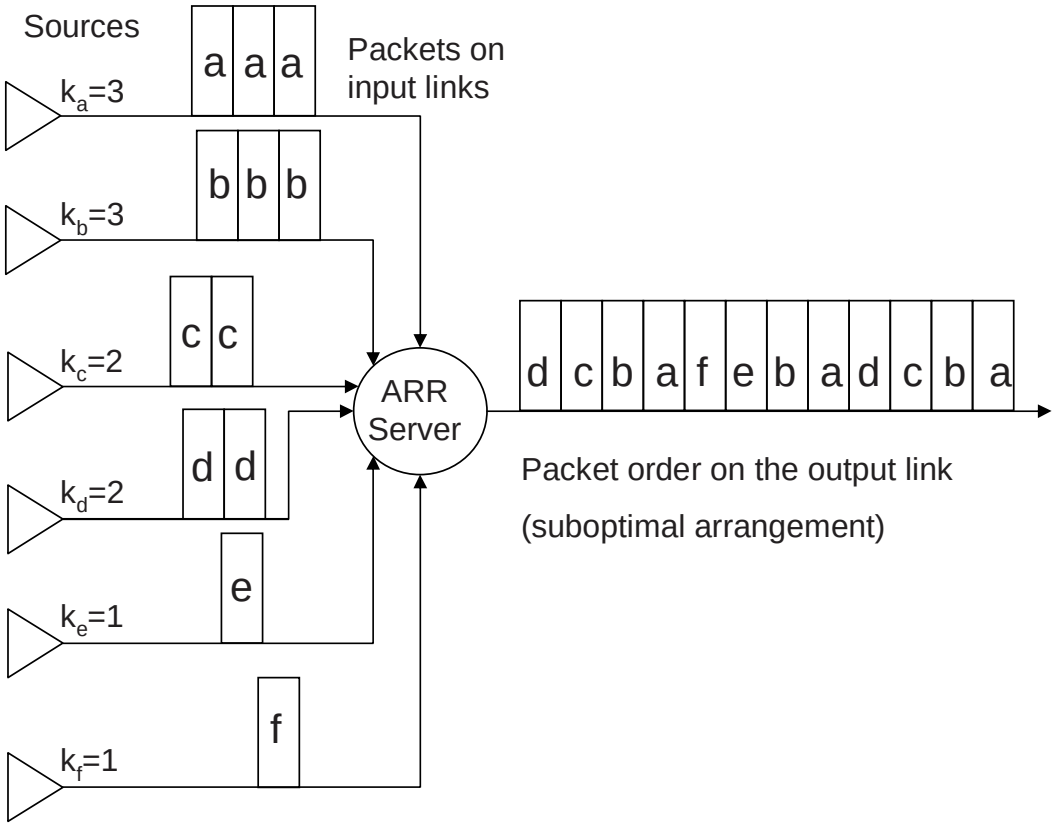

Figure A.1. The operation of the Advanced Round Robin server

results a service cycle of $[d|c|b|a|f|e|b|a|d|c|b|a]$ which is suboptimal, but we can construct an optimal one as it can be seen in Fig. A.2: $[f|b|d|a|c|b|e|a|d|b|c|a]$ - by the use of some heuristic considerations.

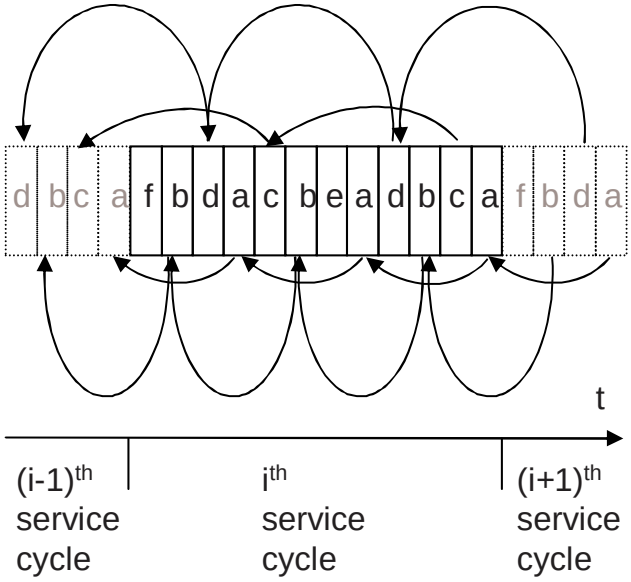

Figure A.2. Optimal service cycle constructed heuristically ($L = 12$)

A.2 The proof of the formulation of the required buffer space

The packet loss is approximated by the approximate asymptotic complementary distribution function of the queue length in the case of M/D/1 queue (using the notations of our paper) [49] by

$$R_{i,j} \approx e^{-2\frac{1-\rho_{i,j}}{\rho_{i,j}}B_{i,j}}.$$

Based on the packet loss rate ($R_{i,j}$) requirement we calculate the length of the buffer:

$$B_{i,j} = \left\lceil \frac{\rho_{i,j}}{2(1-\rho i,j)} \ln R_{i,j} \right\rceil.$$

The utilization caused by connection j in group i is the ratio of its intensity and the service rate guaranteed by the ARR scheduler:

$$\rho_{i,j} = \lambda_{i,j} \frac{L\,l}{k_i\,C}$$

In the above equation we use k_i, which can be calculated with the full knowledge of $B_{i,j}$ (see *Step 2* in Section 4.2.1). Substituting k_i we get

$$\rho_{i,j} = \lambda_{i,j} \frac{L\,l}{C} \frac{C\,D_{i,j\,req}}{L\,l\,B_{i,j}} = \lambda_{i,j} \frac{D_{i,j\,req}}{B_{i,j}}.$$

Using this in the expression of the buffer length we get a quadratic equation which has only one positive root. This is used to calculate the necessary buffer space in *Step 1* in Section 4.2.1.

Bibliography

[1] H. Adiseshu, G. Parulkar, and G. Varghese, "A reliable and scalable striping protocol", in *SIGCOMM Comput. Commun. Rev.*, Vol. 26, No. 4, pp. 131-141, October 1996.

[2] T. Al-Khasib, H. Alnuweiri, H. Fattah, and V. C. M. Leung, "Mini Round Robin: An Enhanced Frame-based Scheduling Algorithm for Multimedia Networks, in *Proceedings of IEEE International Conference on Communication (ICC 2005)*, Vol. 1, pp. 363-368. Seoul, Korea, 16-20 May 2005.

[3] ATM Forum, *ATM User Network Interface Specification Version 3.1*, September 1994.

[4] ATM Forum, *Traffic Management Specification Version 4.0*, April 1996.

[5] ATM Forum, *B-ISDN Inter Carrier Interface (B-ICI) Specification Version 1.0*, August 1993.

[6] J.C.R. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queueing", *in the procedings of the IEEE INFOCOM'96*, pp. 120-128, San Francisco, CA, USA, March 1996.

[7] Dimitri P. Bertsekas, "Dynamic Programming, Deterministic and Stochastic Models", *Prentice-Hall Inc., Englewood Cliffs, N.J. 07632*, 1987.

[8] S. Blaabjerg and S. Molnár, "Methods for UPC Dimensioning of a CDV Perturbated Cell Stream", RACE BRAVE Workshop, Milano, Italy, June 14-15, 1995.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "RFC2475: An Architecture for Differentiated Service", http://tools.ietf.org/html/rfc2475, December 1998.

[10] R. Braden, D. Clark, and S. Shenker, "RFC1633: Integrated Services in the Internet Architecture: an Overview", http://tools.ietf.org/html/rfc1633, June 1994.

[11] F. Bonomi, K. W. Fendick, "The Rate-Based Flow Control Framework for the Available Bit Rate ATM Service", *IEEE Network*, pp. 25-39, March/April 1995.

[12] I. Cidon, L. Georgiadis, R. Guérin, A. Khamisy, "Optimal Buffer Sharing", *IEEE J-SAC*, Vol. 13, No. 7, September 1995.

[13] J. P. Cosmas, "Stochastic Source Models and Applications to ATM", *Performance Evaluation and Applications of ATM Networks* (edited by Demetres Kouvatsos), Kluwer Academic Publishers, U.S.A., September 1999.

[14] R. L. Cruz, "SCED+: Efficient Management of Quality of Service Guarantees", in *Proceedings of the Conference on Computer Communications (IEEE Infocom'98)*, pp. 625, San Francisco, California, March/April 1998.

[15] A. Demers, S. Keshav, and S. Shenkar, "Analysis and simulation of a fair queueing algorithm", in *Proceedings of SIGCOMM'89*, pp. 1-12, Austin, Texas, USA, September 1989.

[16] M. De Prycker, "Asynchronous Transfer Mode. Solutions for Broadband ISDN", Prentice Hall, 1993.

[17] A. Elwalid and D. Mitra, "Design of Generalized Processor Sharing Schedulers which Statistically Multiplex Heterogeneous QoS Classes", in *Proceedings of IEEE INFOCOM'99*, pp. 1220-1230, New York, NY, USA, 21-25 March 1999.

[18] S. Floyd, "Notes on CBQ and Guaranteed Service", Draft document, at *http://www.icir.org/floyd/cbq.html*, July 1995.

[19] S. Floyd and V. Jacobson, "Link-sharing and Resource Management Models for Packet Networks", in *IEEE/ACM TRANSACTIONS ON NETWORKING*, Vol. 3, No. 4, pp. 365-386, 1995.

[20] P. A. Ganos, M. N. Koukias, G. K. Kokkinakis, S. A. Kotsopulos, "A Novel Dynamic Priority Scheduling Method for Multiple Classes of ATM Traffic in an ATM Statistical Multiplexer", in *Performance Modelling and Evaluation of ATM Networks*, Vol. 1, Ed. Demetres Kouvatsos, IFIP, Chapman and Hall, 1995.

[21] R. Garg and X. Chen, "RRR: Recursive round robin scheduler", *Computer Networks*, Vol. 31, pp. 1951-1966, 1999.

[22] Erol Gelenbe, Vilay Srinivasan and Sridhar Seshadri, "Single Node and End-to-End Buffer Control in Real Time", in *Performance Modelling and Evaluation of ATM Networks*, Vol. 1, Ed. Demetres Kouvatsos, IFIP, Chapman and Hall, 1995.

[23] S.J. Golestani, "A self-clocked fair queueing scheme for broadband applications", *in the proceedings of the INFOCOM'94*, pp. 636-646, Toronto, Canada, 12-16 June, 1994.

[24] P. Goyal, S. S. Lam, and H. M. Vin, "Determining End-to-End Delay Bounds In Heterogeneous Networks", *Multimedia Systems*, Vol. 5, No. 3, pp. 157-163, 1997.

[25] P. Goyal, and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: a framework", *IEEE/ACM Trans. Netw.*, Vol. 5, No. 4, pp. 561-571, 1997.

[26] P. Goyal, H. M. Vin, and H. Cheng, "Start-time Fair Queuing: A Scheduling Algorithm for Integrated Services Packet Switching Networks", *in the proceedings of ACM SIGCOMM'96*, pp. 157-168, 1996.

[27] D. Grossman, and J. Heinanen, "RFC2684: Multiprotocol Encapsulation over ATM Adaptation Layer 5", http://tools.ietf.org/html/rfc2684, September 1999.

[28] R. Guérin and V. Peris, "Quality-of-Service in Packet Networks: Basic Mechanisms and Directions", Invited Paper. *Computer Networks*, Vol. 31, No. 3, pp. 169-179, February 1999.

[29] C. Guo, "G-3: An O(1) Time Complexity Packet Scheduler That Provides Bounded End-to-End Delay", in *Proceedings of IEEE INFOCOM 2007*, pp. 1109-1117, Anchorage, AK, USA, 6-12 May 2007.

[30] C. Guo, "SRR: An O(1) Time Complexity Packet Scheduler for Flows in Multi-service Packet Networks", *IEEE/ACM Transactions on Networking*, Vol. 12, No. 6, pp. 1144-1155, Dec. 2004.

[31] C. Guo, "Improved Smoothed Round Robin Schedulers for High-Speed Packet Networks", in *Proceedings of IEEE INFOCOM 2008*, pp. 906-914, Phoenix, AZ, USA, 13-18 April 2008.

[32] ITU-T Recommendations I.356 *B-ISDN ATM Layer Cell Transfer Performance*, October 1996.

[33] ITU-T Recommendations I.371 *Traffic Control and Congestion Control in B-ISDN*, August 1996.

[34] Yuming Jiang, "Relationship between guaranteed rate server and latency rate server", *Computer Networks: The International Journal of Computer and Telecommunications Networking*, Vol. 43, pp. 307-315, October 2003.

[35] S. S. Kanhere, and H. Sethu, "On the latency bound of deficit round robin" *in the proceedings of 11th International Conference on Computer Communications and Networks*, pp. 548-553, Miami, Florida, USA, 14-16 October, 2002.

[36] S. S. Kanhere, H. Sethu, and A. B. Parekh, "Fair and Efficient Packet Scheduling Using Elastic Round Robin", *IEEE Trans. Parallel Distrib. Syst.*, Vol. 13, No. 3, pp. 324-336, 2002.

[37] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip", *IEEE J-SAC*, Vol. 9, No. 8, pp. 1265-1279, October 1991.

[38] L. Kleinrock, "Queueing Systems", John Wiley and Sons, 1975.

[39] Harold J. Kusnher, "Analysis of Controlled Multiplexing Systems via Numerical Stochastic Control Methods", *IEEE J-SAC*, Vol. 13, No. 7, September 1995.

[40] A. Y.-M. Lin, J. A. Sylvester, "Priority Queueing Strategies and Buffer Allocation Protocols for Traffic Control at an ATM Integrated Broadband Switching System", *IEEE J-SAC*, Vol. 9, No. 9, pp. 1524-1536, December 1991.

[41] J. W. Mark, Jing-Fei Ren, "A Traffic Management Framework for ATM Networks", *IFIP Performance Modelling and Evaluation of ATM Networks*, Chapman and Hall, 1996

[42] D. Nikolova and C. Blondia, "Evaluation of Surplus Round Robin Scheduling Algorithm", in *Proceedings of the 2006 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 06)*, Calgary, Alberta, Canada, 31 July - 2 August 2006.

[43] C. Oottamakorn, S. Mao, and S. S. Panwar, "On Generalized Processor Sharing With Regulated Multimedia Traffic Flows", *IEEE Transactions on Multimedia*, Vol. 8, No. 6, pp. 1209-1218, December 2006.

[44] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 3, pp. 344-357, 1993.

[45] A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple Node Case", *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, pp. 137-150, April 1994.

[46] H. G. Perros, "Connection-oriented networks: SONET/SDH, ATM, MPLS and optical networks", John Wiley and Sons, 2005.

[47] S. Rajagopal, V. G. Kulkarni and S. Stidham, Jr., "Optimal Flow Control of a Stochastic Fluid-Flow System", *IEEE J-SAC*, Vol. 13, No. 7, September 1995.

[48] J.-F. Ren, J. W. Mark, J. W. Wong, "A Dynamic Priority Queueing Approach to Traffic Regulation and Scheduling in B-ISDN",*Proceeding of the IEEE Globecom*, pp. 612-618, 1994.

[49] J. Roberts (ed.), "Performance evaluation and design of multiservice networks", Final Report of COST 224, 1991.

[50] E. Rosen, A. Viswanathan, and R. Callon, "RFC3031: Multiprotocol Label Switching Architecture", http://tools.ietf.org/html/rfc3031, January 2001.

[51] Chen ShanZi and Yan Liemin, "A New Priority Control of ATM Output Buffer", *Telecommunication System Journal*, 4, pp. 61-69, 1995.

[52] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin", *in the Proceedings of ACM-SIGCOMM '95: Conference on Applications, Technologies, Architectures, and Protocols For Computer Communication*, pp. 231-242,Cambridge, Massachusetts, United States, August 28 - September 01, 1995.

[53] D. Stiliadis and A. Varma, "Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms", *IEEE/ACM Transactions on Networking*, October 1998.

[54] D. Stiliadis and A. Varma, "Rate-Propotional Servers: A Design Methodology for Fair Queueing Algorithms", *IEEE/ACM Transactions on Networking*, April 1998.

[55] R. Szabó, P. Barta, J. Bíró and F. Németh, "Non-Rate Proportional Weighting of Generalized Processor Sharing Schedulers", *in the Proceedings of GLOBECOM'99*, Rio de Janeiro, Brasil, December 1999.

[56] S. Valaee, M. A. Kaplan, "Congestion Control in a Constraint-Worst-case Framework",*IFIP Performance Modelling and Evaluation of ATM Networks*, Chapman and Hall, 1996

[57] P. Valente, "Exact GPS Simulation and Optimal Fair Scheduling with Logarithmic Complexity", *IEEE/ACM Transactions on Networking*, Vol. 15, No. 6, pp. 1454-1466, December 2007.

[58] J. Xu and R. J. Lipton, "On fundamental tradeoffs between delay bounds and computational complexity in packet scheduling algorithms", *IEEE/ACM Transactions on Networking*, Vol. 13, No. 1, pp. 15-28, February 2005.

[59] J. Yao, J. Guo, and L. N. Bhuyan, "Ordered Round-Robin: An Efficient Sequence Preserving Packet Scheduler",*IEEE Transactions on Computers*, Vol. 57, No. 12, pp. 1690-1703, December, 2008.

[60] H. Zhang, "Service Disciplines for Guaranted Performance Service in Packet-Switching Networks", *Proceedings of the IEEE*, Vol. 83, No. 10, October 1995.

[61] H. Zhang, "Providing end-to-end performance guarantees using non-work-conserving disciplines", *Computer Communications: Special Issue on System Support for Multimedia Computing*, Vol. 18, pp. 769-781, 1995.

[62] L. Zhang, "VirtualClock: a new traffic control algorithm for packet-switched networks", *ACM Transactions on Computer Systems*, Vol. 9, No. 2, pp. 101-124, May, 1991.

Publications of new results

[J] **JOURNALS**

[J1] Gábor Fodor, Tamás Henk, **Tamás Marosits**, Róbert Szabó, Lars Westberg, "Simulative Analysis of Optimal Routing and Link Allocation Strategies in B-ISDN Networks", *PERIODICA POLYTECHNICA Ser. El. Eng.*, Vol. 42, No. 3. pp 275-297, 1998.

[J2] **Tamás Marosits**, Sándor Molnár, Gábor Fodor, "Supporting All Service Classes in ATM: A Novel Traffic Control Framework", *INFORMATICA JOURNAL: Design Issues of Gigabit Networking*, Vol. 23, No. 3, pp 305-315, September, 1999.

[J3] **Tamás Marosits**, Sándor Molnár, "Characterization of Advanced Round Robin Scheduler", *WSEAS TRANSACTIONS on COMMUNICATIONS*, Vol. 8, No. 12, pp 1233-1242, December, 2009.

[C] **CONFERENCES**

[C1] Gábor Fodor, Tamás Henk, **Tamás Marosits**, Róbert Szabó, Lars Westberg, (1995)" On the Call and Cell Level Resource Allocation in ATM Networks", *in the proceedings of the European Simulation Symposium 1995*, Erlagen, Germany, pp 475-484, 26-28 October, 1995.

[C2] Gábor Fodor, **Tamás Marosits**, Róbert Szabó, "Comparison of Null-Message Reduction Techniques in the Parallel Simulation of Multistage Interconnection Networks", *in the proceedings of the ESM'96*, Budapest, Hungary, pp 513-517, 2-6 June, 1996.

[C3] Gábor Fodor, Sándor Molnár, **Tamás Marosits**, "A General Traffic Control Framework in ATM Networks", *in the proceedings of the IEEE Singapore ICCS'96*, Singapore, pp 160-164, 25-29 November, 1996.

[C4] **Tamás Marosits**, Sándor Molnár, Gábor Fodor, "Performance Evaluation of a General Traffic Control Framework in ATM Networks", *in the proceedings of the IEEE IPCCC'99*, Phoenix, Arizona, USA, pp 240-249, 10-12 February, 1999.

[C5] **Tamás Marosits**, Sándor Molnár, János Sztrik, "CAC Algorithm Based on Advanced Round Robin Method for QoS Networks", *in the proceedings of The 6th IEEE Symposium on Computers and Communications*, pp 266-274, Hammamet, Tunisia, 3-5 July, 2001.

[C6] **Tamás Marosits**, Sándor Molnár, "Traffic Control Methods for High Speed Packet Switched Networks", *in the proceedings of 2002 International Polish-Czech-Hungarian Workshop*, pp 58-64, Warsaw, Poland, 12-15 September, 2002.

[C7] **Tamás Marosits**, Sándor Molnár, "Flexible Scheduling Discipline for Fixed-Size-Packet Switched Networks", *in the proceedings of International Conference Probability and Statistics with Applications*, pp 40-41, Debrecen, Hungary, 8-12 June, 2009.