CAC Algorithm Based on Advanced Round Robin Method for QoS Networks

Tamás Marosits, Sándor Molnár
Department of Telecommunications and Telematics
Budapest University of Technology and Economics
Pázmány Péter s. 1/D., 1117 Budapest – Hungary
E-mail: {marosits, molnar}@ttt-atm.ttt.bme.hu

János Sztrik
Department of Information Technology
Lajos Kossuth University
Debrecen, P.O. Box 10, 4010 – Hungary
E-mail: jsztrik@math.klte.hu

Abstract

In this paper a new traffic control method called Advanced Round Robin (ARR) is presented. It is shown that ARR can provide both worst case and statistical Quality of Service guarantees without coupling cell loss and delay requirements. A related Connection Admission Control algorithm with performance study is also presented.

A comparative performance analysis of ARR using simulation was performed in different real scenarios. Results are compared to a reference system and to the simple Round Robin method.

Keywords: Call Admission Control, Scheduling methods, QoS, guaranteed services

1. Introduction

The continuous and rapid evolution of the telecommunication networks was experienced in the last decade. A number of new applications with diverse requirements have been arising. Recent communication networks are intended to provide Quality of Service guarantees for their users. In the Internet world IP packets used to have TOS bits in their header for QoS differentiation but the real pioneer in the field of QoS support was the ATM. The recent development of Internet technology yielded a number of possibilities for QoS networks, e.g. Integrated or Differentiated services with or without MPLS 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.

To support QoS we have two classical opportunities: buffer management and scheduling [7]. The first one is associated mainly with packet loss and the delay characteristics of the flow mostly controlled by the latter, but the limits 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, the short buffer bounded the delay of flows which use it). Both the buffer allocation and the scheduling policies influence the total amount of buffer space required in 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 GPS schedulers fed by leaky bucket shaped sources by Szabó et al. e.g. in [14]. However, these solutions yield overdimensioning of resources in practice.

In [5] Cruz presented a 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.

Using our previous work ([12]) as a starting point in this paper we propose a simple and robust scheduling algorithm. This scheduling method called Advanced Round Robin (ARR) is a modified version of the well-known Round Robin (RR) scheme and it can provide quality of service for guaranteed traffic classes. We make the performance analysis of the presented algorithm and formulate a CAC function. The ARR method has the following advantages: i) it is a CAC and an appropriate scheduling algorithm together, which can support connections with quite different QoS requirements, ii) both worst case and statistical bounds can be guaranteed, iii) packet loss and delay guarantees are decoupled, and iv) it is computationally feasible.

The paper is organized as follows. In Section 2 we present our model. Some basic assumptions are made here to highlight the main results achieved previously. In Section 3 our analytical investigations and the CAC function are described. In Section 4 we overview the network and traffic parameters and simulation results are presented with comparison to other results. Finally we conclude our work in Section 5.

2. Scheduling methods

In our previous work we proposed and analyzed a traffic control algorithm which was able to support all the service classes defined for ATM networks [12]. This method provides outstanding performance whilst all of the guaranteed service classes meet their quality-of-service[1] requirements.

The method is very flexible and adaptive. It can be used with a wide-ranging scale of buffer management algorithms, but in some cases it needs too much computational time and because of the feedback of ABR flows this algorithm cannot be easily analyzed.

It is an obvious thought to find other algorithms by which the performance of our proposed method could be bounded or estimated and which are easy to analyze and implement. The simplest methods are static: the rules of scheduling are not changed during the service. In other words: the server knows immediately nothing about the QoS of the connections and description of traffic flows. The scheduling function of the reference system was described in detail previously in [12]. We call this a reference system, because it works properly allowing high utilization.

To approximate the performance of our reference system, and on the other hand to get a simple and practically implementable scheme we have developed a RR-type algorithm. RR 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 we modified the Round Robin scheme, such that some flows could have access to a more frequent service.

In the literature there is a number of proposals for RR-type algorithms. Katevenis et all. in [10] proposed a RR-type algorithm for the scheduler of an ATM switch. Although they also implemented it, the delay and jitter guarantees were weak because of the non-uniform distribution of service of the individual flows. The reader can also find in [6] a more ingenious algorithm which provides delay and jitter bounds, but the building of the "scheduling tree" is based only on the rate of the sources and disregards the delay requirements.

To increase the service frequency we have constructured the following Advanced Round Robin algorithm: 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. The *service cycle* is the shortest time interval in which all flows get at least once the opportunity to transmit a packet. Then the flows of the group should be scheduled in a service cycle according to the service preference order. The *service preference order* (referred to as "order" in the following) of flow is the number of the opportunities that the flow could gain service. The order of a group is the same as the order of any flow in that group. The access periods of a group are uniformly distributed during the service cycle.

The planned service sequence enumerates flows in that order in which they can transmit packets. The scheduler is a work conserving one [15], the length of a cycle in the planned sequence is the upper bound of the length of a cycle in the realized sequence (for notations see Table 1 in Section 3). Obviously the consecutive service cycles are not the same. The flows are serviced only if they have at least one packet in their buffer. This causes that the service periods of a service group inside a service cycle can also differ. A detailed analysis can be found in Section 3.

Buffer management is an essential part of the traffic control task. In our previous work, where the performance analysis of the reference system was presented [12] we dealt with per-connection buffering. Complete buffer partitioning with a dedicated buffer for each connection is the most secure way to provide quality of service and to protect the services from each other. To control the cell loss it is obvious that we should keep the separate queues with different lengths for the different classes. Moreover the delay guarantees which can be given using shared buffers are much looser than using per-connection buffering.

3. Admission control based on the Advanced Round Robin algorithm

In this section we propose a novel CAC algorithm based on the ARR algorithm (see Section 2). Prior to the operation of buffer management, scheduling, and traffic shaping the CAC functionality must be performed to provide QoS. We should take the first step towards security of other connections and congestion avoidance during the setup of the new connection. In this paper we deal with delay analysis. Buffer space required by the contracted packet loss ratio can be determined from measurements and analytical and simulation results (see Section 4.3). We give worst case delay bounds and average delay. Worst case bounds are used for guaranteed services while average values are typically considered at transmission of best effort services. Based on the delay bounds a Call Admission Control function is formulated.

Table 1 contains the notations about the ARR algorithm. For simplicity 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

[1] In the following part of this paper the most commonly used QoS parameters: cell/packet loss rate, average cell/packet delay, and cell/packet delay variation are referred to as QoS.

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
$D_{i,j}$	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]

Table 1. Notations of the Advanced Round Robin scheme

refers to the maximum permissible load of the scheduler taking into account the requested delay of connection j in group i ($D_{i,j\,req}$):

$$\rho = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i}\lambda_{i,j}}{C/l} \quad (1)$$

$$\hat{\rho} = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i}B_{i,j}/D_{i,j\,req}}{C/l} \quad (2)$$

3.1. Worst case guarantees

For calculating the worst case delay, 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 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:

$$L = \sum_{i=1}^{G} N_i k_i \quad (3)$$

$$D_{i,j} = \frac{LB_{i,j}}{k_i}\frac{l}{C} \quad (4)$$

$$J_{i,j} = \frac{L}{k_i}\frac{l}{C}\,. \quad (5)$$

However, this bound may be very loose in some cases, e.g. if we had CBR flows. If buffering delay is not allowed[2] for regular traffic (e.g. CBR) the bandwidth guaranteed by the system is always greater then the arrival rate.

[2]This means to allocate an amount of one packet for buffering.

$$\lambda_{i,j}\, l \leq C\frac{k_i}{L} \quad (6)$$

The consequences of this criterion:

$$\lambda_{i,j} < \frac{B_{i,j}}{D_{i,j}} \qquad \rho_{i,j} < \hat{\rho}_{i,j}, \quad (7)$$

where $\rho_{i,j}$ and $\hat{\rho}_{i,j}$ are the utilization and availability, respectively, which are caused by connection j in group i. Note that (6) should hold only for services with guaranteed delay and jitter. For this case the maximum delay is given by

$$D_{i,j} = \frac{L}{k_i}\frac{l}{C}, \quad (8)$$

where flow j in the ith group is a CBR flow, which means that it is enough to allocate an amount of memory of one packet for buffering this type of traffic. A tighter delay bound for other types of traffic can also be given, but this is not discussed in this paper.

3.2. Average delay guarantees

Estimating average quantities we take the worst case guarantees as a starting point. Two factors should be considered to capture average characteristics: i) the buffer of a flow is not always full in general, which means that $B_{i,j}$ can be substituted by $\overline{Q}_{i,j}$, and ii) 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:

$$\overline{D}_{i,j} = \frac{L\,\rho\,\overline{Q}_{i,j}}{k_i}\frac{l}{C} \quad (9)$$

$$\overline{J}_{i,j} = \frac{L\,\rho}{k_i}\frac{l}{C}. \quad (10)$$

$\overline{Q}_{i,j}$ can be approximated from the M/D/1 queue (see e.g. [11]), i.e.:

$$\overline{Q}_{i,j} = \frac{\rho_{i,j}}{2(1-\rho_{i,j})}\,. \quad (11)$$

3.3. Call Admission Control algorithm

Previously we saw what service quality the ARR scheduling algorithm could provide with a certain parameter set. Here a backward calculation should be done, the

QoS requirements are given and we want to know the parameter set of the scheduler.

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 [13].*

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

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

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) 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.

3.4. Conditions of acceptance

During the steps of the ARR CAC algorithm we can formulate the conditions of the acceptance of a new connection based on the operations. 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:*

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

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

PROOF The first two conditions are associated with the packet loss. 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) 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.

3.5. Performance evaluation of the CAC algorithm

The algorithm was modeled in Wolfram Research's Mathematica. Several types of applications 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 [4] and the Quality of Service requirements were determined using the ITU-T Recommendation I.356 [8]. Table 2 summarizes parameters and requirements. The link capacity was 620 Mbps.

In the "broadband" scenario the ratio of the services is $1/6$ except videophone ($2/15$) and video on demand ($1/30$).

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}

Table 2. Traffic parameters and QoS requirements

Scenario	Broadband case	Narrowband case
Number of flows	250	400
Utilization	0.793586	0.724162
Availability	0.8979	0.988581
Length of virtual service cycle	1229	7877
Maximum orders	66	63
Number of flows with maximum orders	6	66
Maximum buffer size	145	145

Table 3. Results of the analysis of ARR CAC algorithm

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 3.

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 (2) is very close to 1, it not only endangers the service quality but also increases the complexity of the algorithm. As a rule of thumb, we find that an availability over 0.95 slows down the usage of the CAC algorithm so much that it is better to refuse the new connection. Using the ARR algorithm the limiting condition in most cases is the *availability*. Although the above presented CAC method can handle hundreds of connections with different QoS requirements, it works better, if i) the smallest required delay is at least a hundred times greater then the service time of one packet (l/C), ii) there are delay requirements assigned to not guaranteed services too, and iii) there are similar characteristics and requirements, i.e. natural service classes can be established.

4. Performance evaluation of the scheduling method

In this section the ARR scheduling has been analyzed in real scenarios by simulations. The performance of the ARR is compared to the performance of our reference system (see Section 2 and [12]) and to the performance of the RR. Note that the simulation results of this section are related to a one switch scenario.

4.1. Simulation scenarios

The traffic control task is based on relationships between the three sets of parameters: description of traffic, network resources and quality of service requirements. From CAC to scheduling every network function should take into account these groups. For our traffic control framework we have chosen the parameters in agreement with the standardisation work of ATM Forum [1] [2] and ITU-T [8] [9].

According to the goals of this paper the simulation results can be divided into two groups: i) first we show that QoS can be provided with our ARR algorithm for traffic flows which have applied for guaranteed service, and ii) then we examine how much throughput degradation should network endure or how should we increase the network capacities to give the flows the same service guarantees which they had in the reference system presented in [12].

There are five service classes supported by ATM networks, which means that an exhaustive performance study should have examined the dependence of the Quality of Service on the load and burstiness changing of five traffic 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 [3], but this feature makes them resistant to the danger of 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.

The traffic parameters of the basic state can be seen in Table 4. The rates are given in Mbps. We gave the burstiness parameters of all services measured by the squared coefficient of variation of the interarrival time (i.e. the c^2 parameter). Link capacity was considered to be 45 Mbps, and the multiplexer had 5 input ports corresponding to the 5 service classes.

	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	-

Table 4. Basic input traffic characteristics

Note that with the above 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 5-6 display the QoS requirements of different services and the buffer sizes available for different service classes. 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. CTD and CDV requirements are given in time unit.

	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 5. The QoS requirements

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

Table 6. Buffer sizes in cells

4.2. Performance results of different schedulers

In this subsection we show that the ARR is able to provide QoS. In our performance study we examined the impact of the traffic load and traffic burstiness increase. Due to the lack of space only the scenarios with rtVBR burstiness increase are presented here. We consider only one permanent connection from each traffic class in our simulation. The traffic and network parameters of the basic state are given above. In the following we will describe the differences in each situation.

In Fig. 1-9 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 Fig. 1-3 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 6. 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.

Fig. 4-6 show the performance of the RR algorithm. 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 make 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 Fig. 7-9). In the this simulation study CBR and VBR queues are scheduled twice in a service cycle. The 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.

The impacts of increasing burstiness of non real time VBR traffic can be demonstrated in similar way. The authors examined also the impacts of increasing load of any traffic type. These results can be not detailed here due to the lack of space, but we should note that the ARR worked according to the expectations. The performance was in some cases even better than with the 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 much 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 .

4.3. The price of simplicity

In this subsection we examine what is the degradation of utilization using our simplified ARR algorithm and how much extra buffer space is needed to achieve the quality of service of the reference system.

Table 3 contains utilization information, however reader can recognize, that using the ARR algorithm the limiting condition in presented scenarios is the *availability*. Maximum 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 in Section 3.1).

To explore the robustness of the modified RR algorithm we made simulations with normal and more bursty nrVBR traffic using RR 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 then any other c^2 parameter in the basic state. 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 this bursts go through to the network and the last cells of them get high jitter. In Fig. 10 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.

Fig. 11 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.

5. Conclusions

In this paper a new scheduling method called Advanced Round Robin has been proposed. Using the results of this algorithm a new CAC function has been formulated and the necessary conditions of the acceptance of a new connection were given. Both worst case bounds and average values were analytically derived for delay and jitter. The proposed CAC was evaluated by numerical studies. The performance evaluation of the ARR was carried out by simulations. Results were compared to a reference system and to the simple RR.

Acknowledgement

Authors would like to thank István Maricza for his comments on the paper.

Appendix

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) [13] 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 3.3). 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 3.3.

References

[1] ATM Forum. ATM User Network Interface Specification Version 3.1. September 1994.

[2] ATM Forum. Traffic Management Specification Version 4.0. April 1996.

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

[4] 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.

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

[6] R. Garg and X. Chen. RRR: Recursive Round Robin Scheduler. *Computer Networks*, 31, pp. 1951-1966, 1999.

[7] 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.

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

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

[10] 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, p 1265-1279, October 1991.

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

[12] T. Marosits, S. Molnár and G. Fodor. Supporting All Service Classes in ATM: A Novel Traffic Control Framework. *in a special issue of Informatica Journal on Design Issues of Gigabit Networking*, vol 23, no 3, pp 305-315, September 1999.

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

[14] 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.

[15] H. Zhang. Service Disciplines for Guaranted Performance Service in Packet-Switching Networks. *Proceedings of the IEEE*, 83(10), October 1995.

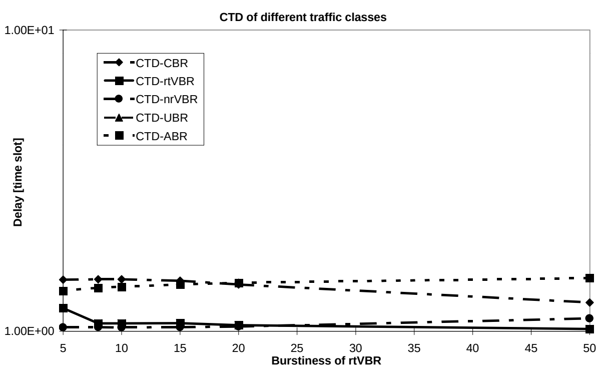

Figure 1. Delay vs. rtVBR burstiness (reference system)

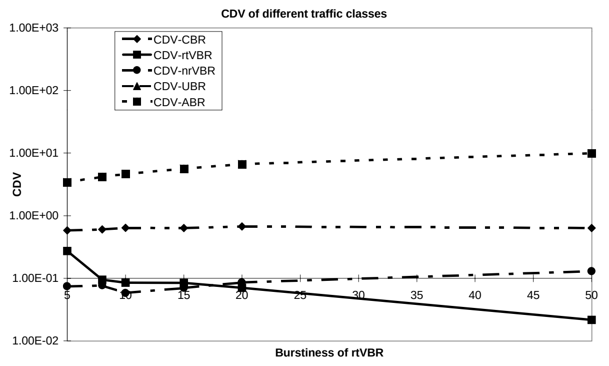

Figure 2. Jitter vs. rtVBR burstiness (reference system)

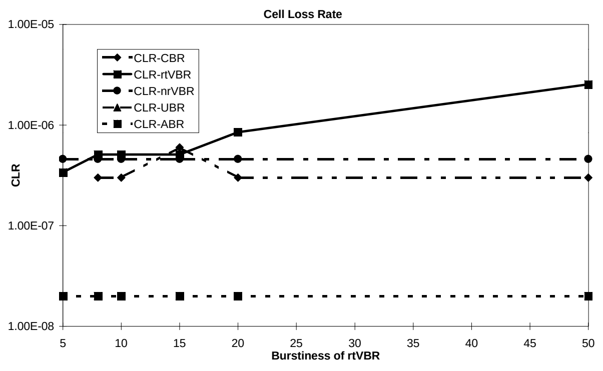

Figure 3. CLR vs. rtVBR burstiness (reference system)

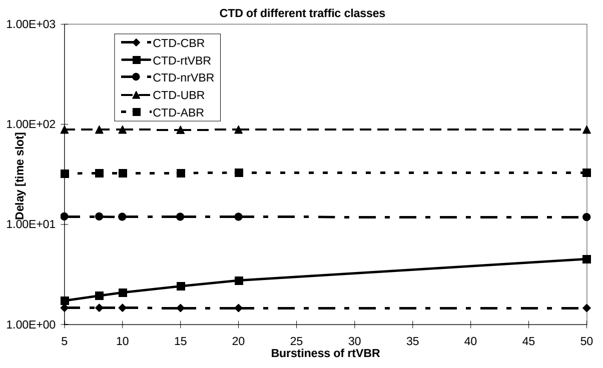

Figure 4. Delay vs. rtVBR burstiness (RR)

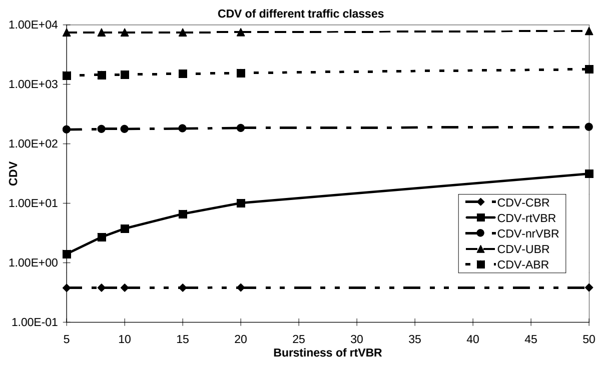

Figure 5. Jitter vs. rtVBR burstiness (RR)

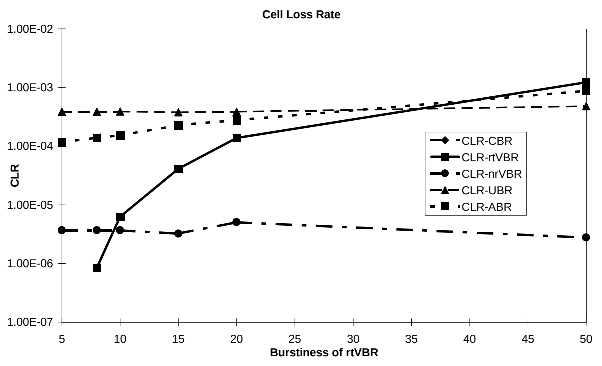

Figure 6. CLR vs. rtVBR burstiness (RR)

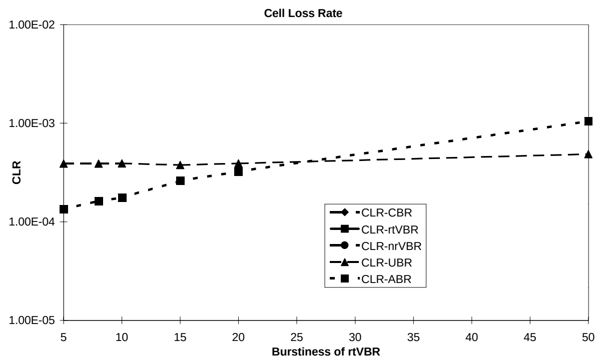

Figure 9. CLR vs. rtVBR burstiness (ARR)

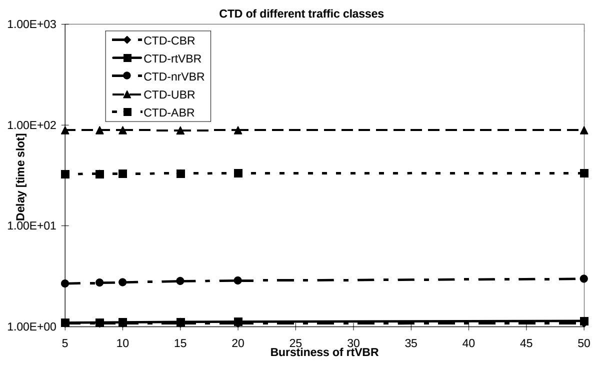

Figure 7. Delay vs. rtVBR burstiness (ARR)

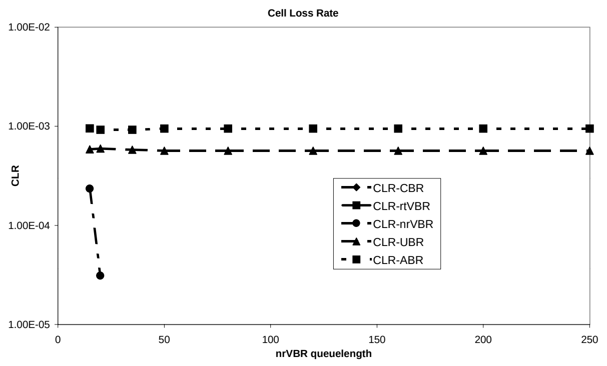

Figure 10. CLR vs. nrVBR queue length (ARR, bursty nrVBR)

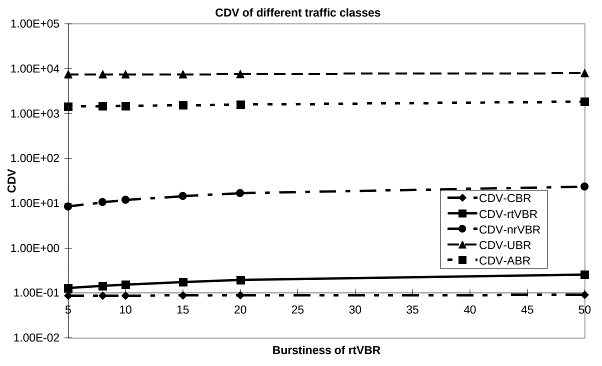

Figure 8. Jitter vs. rtVBR burstiness (ARR)

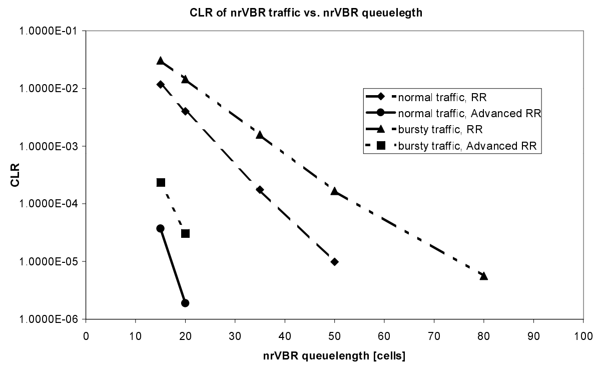

Figure 11. nrVBR packet loss vs. nrVBR queue length