Characterization of Advanced Round Robin Scheduler

TAMÁS MAROSITS and SÁNDOR MOLNÁR
Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Magyar tudósok krt. 2, H-1117, Budapest
HUNGARY
marosits@tmit.bme.hu http://www.tmit.bme.hu

Abstract: - In this paper we show that Advanced Round Robin (ARR) scheduler is in the class of Latency Rate servers and it also belongs to the class of Guaranteed Rate servers. We present the latency parameter and the error term of ARR. Next, we evaluate the fairness characteristic. Finally, we describe the relationship between ARR and Generalized Processor Sharing. All of these results can make the comparison of ARR easily to other known schedulers and help designers to apply ARR.

Key-Words: - Round Robin scheduling, Latency Rate servers, GPS, fairness

1 Introduction

How to provide Quality of Service is one of the main questions in the recent and future internet. The wide variety of services demand granularity while the huge amount of traffic require robustness and simplicity in traffic control (e.g., call acceptance, routing, scheduling).

In our previous work [12] the Advanced Round Robin (ARR) scheduling algorithm for fix packet length packet switching networks was introduced. In this paper we present some further important features of ARR. These features make it possible to locate ARR in the broad list of known schedulers. We show that ARR is in the class of Latency Rate servers, therefore all results of $\mathcal{LR}$-servers can be applied for ARR. We also show the connection between ARR and Generalized Processor Sharing (GPS). This connection can provide us to apply GPS results for ARR. Moreover, we also present the fairness-index of ARR which is one of the most important features of schedulers. These results can provide a good framework for designers to apply ARR and calculate its parameters and performance.

In the literature there are a number of proposals for Round Robin-type (RR) algorithms. Katevenis *et al.* in [11] 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 find in [4] 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. Mini Round Robin [1] and Surplus Round Robin [13] seem to be inadequate also, because the flows having most service opportunities are scheduled mostly at the end of the round which yields bursty arrival of this flows at the next hop in a multi-nodal scenario. In [8] also a RR-type scheduler called G3 for fixed size packet networks was presented. G3 is an $O(1)$ time complexity packet server built up on the Recursive Round Robin described in [4] and Smoothed Round Robin [9].

A wide range of schedulers can be described as a Latency Rate server [18]. 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 $\mathcal{LR}$ -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 $\mathcal{LR}$-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 of $\mathcal{LR}$-servers. We can now easily calculate tight delay bounds for a series of ARR servers or use ARR-server with other $\mathcal{LR}$-schedulers in a network.

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*. Following the definition given by Golestani in [5] fairness is defined as the maximum difference between the normalized service received by two backlogged flows over an interval of time in which both are continuously backlogged. Calculating the fairness of ARR it becomes comparable to the known schedulers and the applicability of the method will be revealed.

The Generalized Processor Sharing service discipline [15] 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. In the previous decade there have been a vast work on developing and analyzing GPS schedulers, see e.g. [3], [14], [20]. 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. 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.

Presenting the relationship between ARR and GPS 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 ARR-scenario can be easily analyzed taking into account the results [19] and [16], respectively.

This paper is organized as follows. In Section 2 the ARR scheduler is introduced. In Section 3 it is shown that ARR belongs to the Latency Rate servers and new delay and jitter bounds are formulated. We derive the characterization of ARR as a Guaranteed Rate server as well. In Section 4 the fairness of ARR is evaluated. Section 5 the relationship between the ARR scheduler and the GPS will be described. At last we conclude our work in Section 6.

2 Background

As its name suggests Advanced Round Robin algorithm [12] is a round robin-type scheduling method in which every connection has its own buffer and there is a known time limit between the service opportunities of these buffers. The main difference is that queues in ARR may be served more than once in a cycle.

2.1 Working of Advanced Round Robin Scheduler

To increase the service frequency the following procedure was constructed: 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 a 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.

For the notations used in the rest of this paper see Table 1.

Table 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 i^{th} group
k_i	the service preference order of group i
g_i	the service rate allocated to a flow in group i
$B_{i,j}$	buffer length of the j^{th} flow in group i in packets
$Q_{i,j}$	number of packets in the buffer of the j^{th} flow in group i
$\overline{Q}_{i,j}$	average number of packets in the buffer of the j^{th} flow in group i
$l_{i,j}$	arrival intensity of the j^{th} 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 j^{th} flow in group i [sec]
$\overline{D}_{i,j}$	average delay of the j^{th} flow in group i [sec]
$J_{i,j}$	maximum difference between successive packet departures of the j^{th} flow in group i [sec]
$\overline{J}_{i,j}$	average difference between successive packet departures of the j^{th} flow in group i [sec]

The planned service sequence enumerates flows in that order in which they can transmit packets. The scheduler is a work conserving one, the length of a cycle in the planned sequence is the upper bound of the length of a cycle in the realized sequence. This causes that the service periods of a service group inside a service cycle can also differ.

2.2 Service Guarantees of the Advanced Round Robin Scheduler

Considering the above presented architecture we can easily express the rate g_i guaranteed to the flows of group i and formulate worst case delay $D_{i,j}$ and the maximum difference between successive packet departures $J_{i,j}$ [1] of the j^{th} flow in group i in seconds.

$$g_i = \frac{L}{Ck_i} \tag{1}$$

$$D_{i,j} = \left\lceil \frac{LB_{i,j}}{k_i} \right\rceil \frac{l}{C} \tag{2}$$

$$J_{i,j} = \left\lceil \frac{L}{k_i} \right\rceil \frac{l}{C} \tag{3}$$

Moreover, if we know the arrival process of the connections even average delay ($\overline{D}_{i,j}$) and average difference between successive packet departures ($\overline{J}_{i,j}$) can be given as follows:

$$\overline{D}_{i,j} = \left\lceil \frac{Lr\overline{Q}_{i,j}}{k_i} \right\rceil \frac{l}{C} \tag{4}$$

$$\overline{J}_{i,j} = \left\lceil \frac{Lr}{k_i} \right\rceil \frac{l}{C}, \tag{5}$$

where r is the *utilization* of the scheduler defined by (6).

$$r = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i} l_{i,j}}{C/l} \tag{6}$$

While the above definition refers to the traditional meaning of utilization, we should involve another quantity called *availability* which pointed to the maximum permissible load of the scheduler taking into account the requested delay of connection j in group i ($D_{i,j,req}$):

$$\hat{r} = \frac{\sum_{i=1}^{G}\sum_{j=1}^{N_i} B_{i,j}/D_{i,j,req}}{C/l}. \tag{7}$$

For more detailed description see [12].

2.3 Construction of Service Cycle for Advanced Round Robin Scheduler

The above mentioned results 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 $1 \le j \le N_i$ and $1 \le i \le G$. 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[2]. However, we can build

[1] This is a jitter-like quantity, more precisely the deviation of $J_{i,j}$ is the jitter.

[2] The exact 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 $1 \le i \le G$. 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 is 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 an optimal service cycle is impossible.

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 [12] 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 a different 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 every flow. The order of a connection can be obtained from the Call Admission Control algorithm (CAC) belonging to the ARR scheduler. The CAC function (which was described in [12]) 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 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 Fig. 1). For the precise algorithm see the Appendix.

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 we experience, that only a limited group of connections will be evolved.

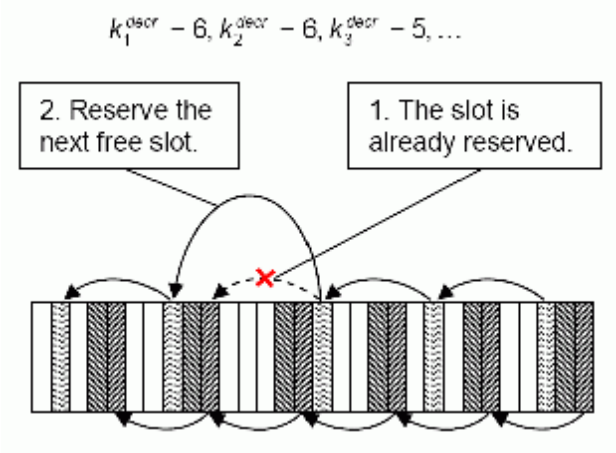

Fig. 1 The building of service cycle for Advanced Round Robin server (*L=30*)

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 to slow, we should rather refuse the newcomer connection.

3 Classification of the ARR Scheduler

In this section we will show that the ARR scheduler belongs to the class of Latency-Rate servers and thereupon it belongs to the class of Guaranteed Rate servers too.

As several well known scheduling algorithms such as Weighted Fair Queueing (WFQ) [2], Virtual Clock (VC) [21], Self-Clocked Fair Queueing (SCFQ) [5], Weighted Round Robin (WRR) [11] and Deficit Round Robin (DRR) [17], do belong to the class of Latency Rate servers (described in [18]) also does Advanced Round Robin.

According to the definition a server belongs to the $\mathcal{LR}$ class if and only if for all times t after t that j^{th} busy period started and until the packets that arrived during this period are serviced

$$W_{i,j}(t,t) \geq \max\left[0, \overline{g_i}(t-t-\Theta_i)\right]. \quad (8)$$

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 (8) defines an envelop to bound the minimum service offered to any backlogged session in group *i* in the j^{th} busy period. However, using ARR we can also give such an envelop for minimum service.

The average service rate $\overline{g_i}$ can be obviously evaluated dividing the guaranteed service rate with utilization r . The latency Θ_i is determined by the architecture of ARR. We should answer the question: how become empty a queue after a greedy system start[3]. For the beginning of serving the queue of connection *j* in group *i* we should wait at most $(L/k_i)(l/C)$. At the end, the last packet is served when its last bit is served, which means l/C . For the latency we have

$$\Theta_i = \frac{Ll}{k_i C} + \frac{l}{C} = \frac{l}{C}\left(\frac{L}{k_i} + 1\right) \tag{9}$$

According to the characterization of $\mathcal{LR}$ servers new bounds can be given for the ARR, however, these bounds might be looser then those which were evaluated based on the architecture of ARR.

$$Q_{i,j}(t) \le s_{i,j} + g_i\Theta_i = l\left(B_{i,j} + 1 + \frac{k_i}{L}\right) \tag{10}$$

$$D_{i,j}(t) \le \frac{s_{i,j}}{g_i} + \Theta_i = \frac{lL}{k_i C}\left(B_{i,j} + 1\right) + 1 \tag{11}$$

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

In [10] Jiang proves that if a server belongs to the Latency Rate class it also belongs to the Guaranteed Rate class (defined in [6] and [7]) and vice versa. According to definition a scheduler is a $\mathcal{GR}$ server for a flow with error term b if

$$f^j \le GRC^j + b \tag{12}$$

where $GRC^j = \max\left(a^j, GRC^{j-1}\right) + \frac{l^j}{g_j}$ is the guaranteed rate clock of the j^{th} packet of the flow, $GRC^0 = 0$, a^j is the time the j^{th} packet of the flow arrives to the scheduler and f^j is the time the j^{th} packet finishes service from the scheduler. According to the conversion rules proven in [10] ARR is a member of class of Guaranteed Rate servers with guaranteed rate g_i and error term

$$b = \Theta - \frac{L^{\min}}{\overline{g_i}} = \frac{l}{C} \tag{13}$$

where $L^{\min}$ is the minimum packet size, which is *l* in the case of ARR. One can easily observe that the value of error term is equal to the transmission time of a packet.

4 Fairness of the Advanced Round Robin Scheduler

According to the definition of Golestani [5] 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.

$$\left|\frac{W_i(t_1,t_2)}{k_i} - \frac{W_k(t_1,t_2)}{k_k}\right| \le F^S \tag{14}$$

The interval (t_1, t_2) can be divided to two parts. The first part consists of many complete realized service cycles[4] in which the normalized service received by two continuously backlogged connections is the same. The second part is the interval $(t, t_2) \le \frac{Ll}{C}$ (see Fig. 2).

[3] All queues are continuously backlogged and $r = 1$. Each queue has only its minimum guaranteed service rate.

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

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 service cycle fraction described by (15) and (16), 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.

$$\min W_{i,j}(t,t_2) = l\left\lfloor \frac{(t_2 - t)Ck_i}{Ll} \right\rfloor \tag{15}$$

$$\max W_{i,j}(t,t_2) = l\left\lceil \frac{(t_2 - t)Ck_i}{Ll} \right\rceil \tag{16}$$

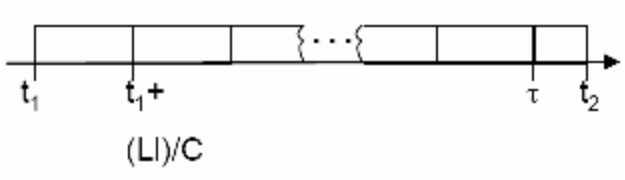

Fig. 2 The calculation of fairness of Advanced Round Robin scheduler

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

$$\max\left\lfloor (s-1,0)\frac{k_x}{k_i} \right\rfloor \le u \le \min\left\lceil (s+1,k_i)\frac{k_x}{k_i} \right\rceil. \tag{17}$$

Moreover, in (17) 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:

$$F^S = \frac{k_x + k_i}{k_x k_i}\frac{lL}{C} = \frac{l}{g_i} + \frac{l}{g_x}. \tag{18}$$

5 Parameter Conversion between ARR and GPS

In this Section we will describe the relationship between the Advanced Round Robin and the Generalized Processor Sharing scheduling discipline.

Thinking about the definition of GPS presented in [15] one can remark, that the orders k_i of ARR have the same role as j_is in GPS. The main difference is that $k_i \in Z$ and $j_i \in R$. However, one can easily observe, that any real number j_i can be approximated with arbitrary accuracy by a rational number. Because in the definition of GPS only the ratio of weights is used, the rational approximation of the weights can multiplied by the smallest common denominator resulting a set of integer weights.

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

$$\frac{W_{i,j}(t_1,t_2)}{W_{x,y}(t_1,t_2)} \ge \frac{k_i}{k_x} \tag{19}$$

where $x = 1,2,\mathbf{K},G$, $j = 1,2,\mathbf{K},N_i$, $y = 1,2,\mathbf{K},N_x$.

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

$$g_i = \frac{k_i}{\sum_{j=1}^{G} N_j k_j} C = \frac{k_i}{L} C \tag{20}$$

Assuming the same considerations as in GPS the upper bound of maximum backlog $s_i = 1,\mathbf{K},B_{i,j}$ and from this the upper bound of the maximum delay can be expressed as

$$D_{i,j}(t) \le \frac{s_i}{g_i} = \frac{LB_{i,j}}{k_i}\frac{l}{C} \tag{21}$$

which is not greater then the worst case delay evaluated in (2) from the architecture of ARR.

6 Conclusion

In this paper we have shown that the formerly presented Advanced Round Robin (ARR) scheduling method belongs to the Latency Rate servers. The ARR can also be characterized as a Guaranteed Rate server and we have given the error term to this characterization. Next, we have calculated the fairness index of ARR. In the last part of this paper we have shown that ARR has a very close relationship with the Generalized Processor Sharing (GPS) discipline.

A detailed analysis of several well known work-conserving schedulers can be found in [18]. In order to compare the latency and fairness of these servers with Advanced Round Robin we listed them in Table 2 and Table 3, respectively, *as they have been serving fixed-size packets*.

Table 2 Latency of several work-conserving servers

Server	**Latency**
GPS	0
PGPS	$\frac{l}{g_i}+\frac{l}{C}$
SCFQ	$\frac{l}{g_i}+\frac{l}{C}(V-1)$
Virtual Clock	$\frac{l}{g_i}+\frac{l}{C}(V-1)$
DRR	$\frac{3Ll-2lk_i}{C}$
WRR	$\frac{Ll-lk_i+1}{C}$
FFQ	$\frac{l}{g_i}+\frac{l}{C}$
ARR	$\frac{Ll}{Ck_i}+\frac{l}{C}=\frac{l}{g_i}+\frac{l}{C}$

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. 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*.

Based on these result we can find that the latency of Advanced Round Robin is not worse that 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, which was referred in Section 1 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.

Table 3 Fairness of several work-conserving servers

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

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*.

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 other packet schedulers but Weighted Round Robin. According to WRR in the case of both k_i and k_j are greater than 2 the fairness of ARR will be better.

These results can provide a good guideline framework for designers to apply ARR and calculate its parameters and performance.

7 Appendix

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 $x = 1\mathbf{K}\sum_{i=1}^{G} N_i$. 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 reserve in resources due to limitations according to throughput and availability the backward direction in Step 5 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, when our algorithm cannot establish it. Sometimes because it is impossible (see Section 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 mark their sources and packets by *a*, *b*, *c*, *d*, *e*, and *f*, respectively.

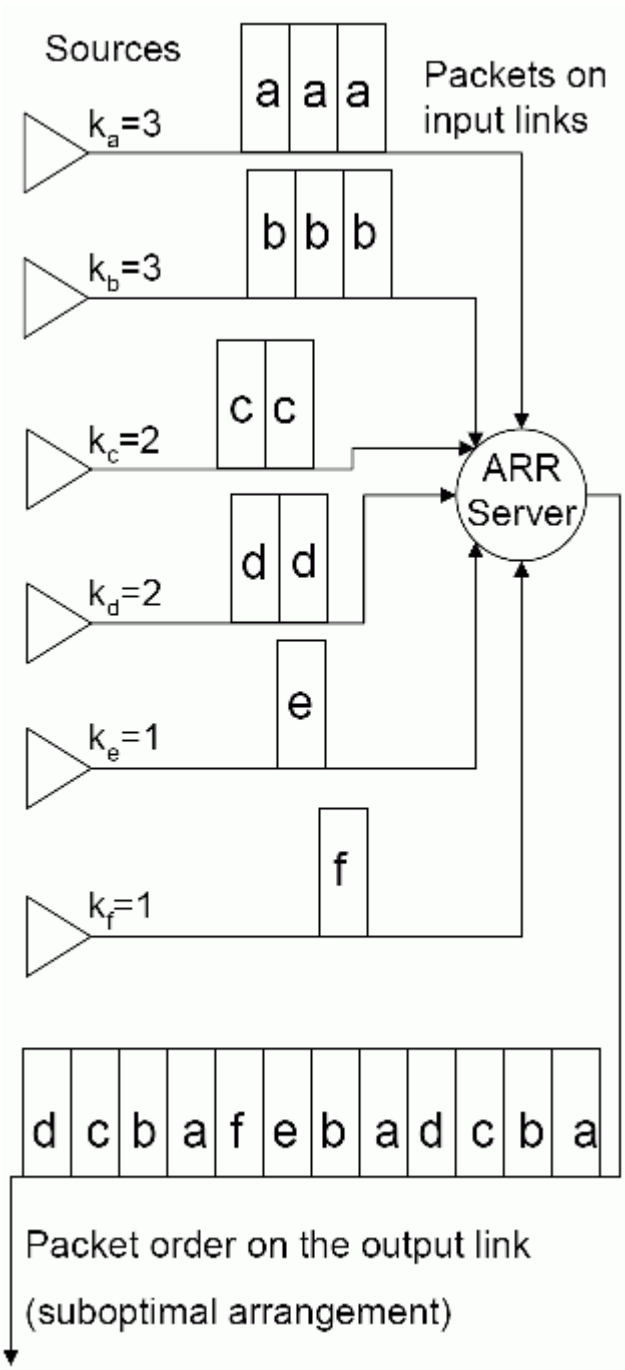

Fig. 3 The working of the Advanced Round Robin Server

Our algorithm results a service cycle of [*d*|*c*|*b*|*a*|*f*|*e*|*b*|*a*|*d*|*c*|*b*|*a*] which is suboptimal (see Fig. 3), but we can construct an optimal one using heuristics as it can be seen on Fig. 4.

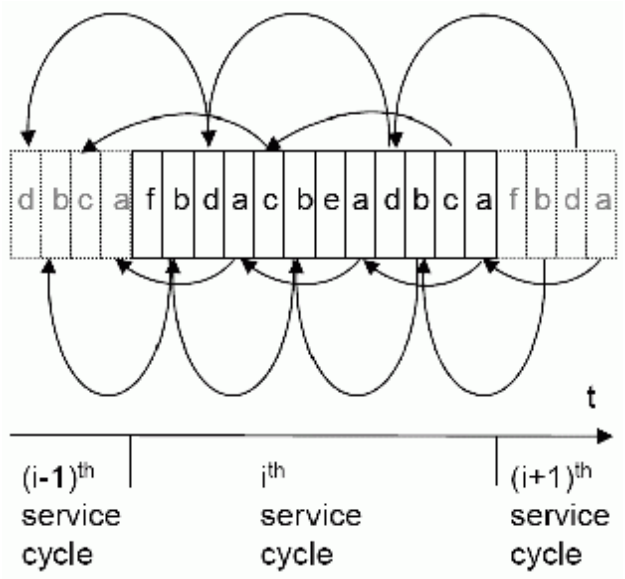

Fig. 4 Optimal service cycle constructed heuristically (L=12)

References:

[1] T. Al-Khasib, H. Alnuweiri, H. Fattah, and V. C. M. Leung, Mini round robin: an enhanced frame-based scheduling algorithm for multimedia networks, *Proceedings of IEEE International Conference on Communication (ICC 2005)*, Vol.1, pp. 363-368. Seoul, Korea, 16-20 May 2005.

[2] A. Demers, S. Keshav, and S. Shenker, Analysis and simulation of a fair queueing algorithm, *Internetworking: Research and Experience*, Vol.1, No.1, pp. 3-26, 2005.

[3] A. Elwalid and D. Mitra, Design of generalized processor sharing schedulers which statistically multiplex heterogeneous QoS classes, *Proceedings of IEEE INFOCOM'99*, pp. 1220-1230, New York, NY, USA, 21-25 March 1999.

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

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

[6] 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, May 1997.

[7] P. Goyal and H. M. Vin, Generalized guaranteed rate scheduling algorithms: a framework, *IEEE/ACM Transactions on Networking*, Vol.5, No.4, pp. 561-571, August 1997.

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

[9] 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.

[10] Y. 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.

[11] 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.

[12] T. Marosits, S. Molnár, and J. Sztrik, CAC Algorithm Based on Advanced Round Robin Method for QoS Networks, *Proceedings of The 6th IEEE Symposium on Computers and Communications*, pp. 266-274, Hammamet, Tunisia, 3-5 July 2001.

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

[14] 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.

[15] 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, June 1993.

[16] 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.

[17] M. Shreedhar and G. Varghese, Efficient fair queueing using deficit round-robin, *IEEE/ACM Transaction on Networking*, Vol.4, No.3, pp. 375-385, June 1996.

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

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

[20] 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.

[21] 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.