TRAFFIC CONTROL METHODS FOR HIGH SPEED PACKET SWITCHED NETWORKS

Tamás Marosits, Sándor Molnár
Department of Telecommunications and Telematics
Budapest University of Technology and Economics
H-1117, Magyar Tudósok krt. 2. Budapest, Hungary
E-mail: marosits@ttt.bme.hu

Abstract

In this study 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 and the relationship between ARR and the well known Generalized Processor Sharing (GPS) is investigated.

A comparative performance analysis of ARR using simulation can be found in a previous work of the authors.

Keywords: Call Admission Control, scheduling methods, QoS, GPS

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.
Robustness and granularity become the main questions.
Using our previous work ([2]) as a starting point in this paper we propose a simple and robust scheduling algorithm. This scheduling method called Advanced Round Robin is a modified version of the well-known Round Robin 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 Advanced Round Robin (ARR) method has the following advantages:

- it is a call admission control and an appropriate scheduling algorithm together, which can support connections with quite different QoS requirements,
- both worst case and statistical bounds can be guaranteed,
- packet loss and delay guarantees are decoupled, and
- 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 analytical investigations are made and the CAC function is formulated. In Section 4 we will show that ARR can be approximated by GPS and result regarding GPS can be used to analyze ARR traffic control. 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 [2]. This method provides outstanding performance whilst all of the guaranteed service classes meet their quality-of-service[1] requirements. 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 points. If all of the points are inside the region bordered by the corresponding surface then QoS requirements are met. 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.

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.

The scheduling function of the reference system was described in detail previously in [2]. We call this a reference system, because it works properly allowing high utilization.

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. Of course, in the network dimensioning phase operators have information about QoS requirements and traffic characteristics.

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 Round Robin-type algorithm [3]. 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 we modified the Round Robin scheme, such that some flows could have access to a more frequent service.

To increase the service frequency we have constructed 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 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. The planned service sequence enumerates flows in that order in which they can transmit packets.

The scheduler is a work conserving one [7], 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.

[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.

3 Admission control based on the Advanced Round Robin algorithm

In this section we propose a novel CAC algorithm based on the Advanced Round Robin 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 connections. In this paper we deal with delay analysis. 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.

First we define some notions about the Advanced Round Robin algorithm. The definitions are mostly straightforward and some of them are written above, but the summary of our notations is also given here.

Definition: The ***service cycle*** is the shortest time interval in which all flows get at least once the opportunity to transmit a packet.

Definition: 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.

Table 1 contains the notations about the Advanced Round Robin 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.

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*
$B_{i,j}$	buffer length of the *j*th flow in group *i* in packets
$Q_{i,j}$	number of packets in the buffer of *j*th flow in group *i*
$\overline{Q}_{i,j}$	average number of packets in the buffer of *j*th flow in group *i*
$\lambda_{i,j}$	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]

Table 1 Notations of the Advanced Round Robin scheme

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 packet loss and delay requirements of connection *j* in group *i* ($D_{i,j,req}$):

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

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 *i*th 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:

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

$$D_{i,j} = \frac{LB_{i,j}}{k_i}\frac{l}{C} \qquad (2)$$

$$J_{i,j} = \frac{L}{k_i}\frac{l}{C} \qquad (3)$$

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.

$$\lambda_{i,j} l \le C\frac{k_i}{L} \qquad (4)$$

The consequences of this criterion:

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

where ρ_{ij} and $\hat{\rho}_{ij}$ are the utilization and availability, respectively, which are caused by connection *j* in group *i*. Note that (4) should hold only for services with guaranteed delay and jitter. For this case the maximum delay is given by

$$D_{i,j} = \frac{L}{k_i}\frac{l}{C} \qquad (5)$$

where flow *j* in the *i*th group is a CBR flow.
Replacing $B_{i,j}$ in (2) by one packet we also obtain (5), 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 starting point. Two factors should be considered to capture average characteristics:

- 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
- 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:

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

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

$$\bar{J}_{i,j} = \frac{L\rho}{k_i}\frac{l}{C}$$

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

$$\bar{Q}_{i,j} = \frac{\rho_{i,j}}{2(1-\rho_{i,j})}$$

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 ($1 \le i \le G$) flows in each group. Group i has the order k_i. Flow j in the ith group ($1 \le j \le N_i$) 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 [5].

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

where $\lceil \ldots \rceil$ is the ceiling function. The proof of the above buffer calculation can be found in [5] and in [3].

Step 2 Calculate an order k_0 for the new connection:

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

Step 3 Calculate the new length of the service cycle:

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

Step 4 Using (2) 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 $D_{i,j} \ge D_{i,j,req}$ 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.

3.3.1 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:*

$$M \geq B_0 + \sum_{i=1}^{G} \sum_{j=1}^{N_i} B_{i,j}$$

$$\frac{C}{l} \geq \lambda_0 + \sum_{i=1}^{G} \sum_{j=1}^{N_i} \lambda_{i,j}$$

$$C \geq l \left(\frac{B_0}{D_{0,req}} + \sum_{i=1}^{G} \sum_{j=1}^{N_i} \frac{B_{i,j}}{D_{i,j,req}} \right)$$

where M is the memory size available in the switch.

The proof of the theorem is presented in our previous work [3].

4 Towards GPS

The description of GPS could be found in many papers, in the following we will use Parekh's notations [4].

A **GPS server** serving N sessions is characterized by N positive real numbers, $\varphi_1, \varphi_2 \ldots \varphi_N$. The server operates at a *fixed rate r* and is work-conserving. A server is work-conserving if it is never idle whenever there are packets to send. Let $W_i(t_1,t_2)$ be the amount of session i traffic served in the interval $[t_1,t_2]$, then a GPS server is defined as one for which

$$\frac{W_i(t_1,t_2)}{W_j(t_1,t_2)} \geq \frac{\varphi_i}{\varphi_j} \quad j = 1, 2 \ldots N \qquad (6)$$

for any session i that is continuously backlogged[3] in the interval $[t_1,t_2]$. The immediate consequence of this definition is that every session has a minimum *guaranteed service rate* that is

$$g_i = \frac{\varphi_i}{\sum_{j=1}^{N} \varphi_j} r .$$

In GPS each input traffic can be shaped by a *leaky bucket* that is characterized 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 incoming traffic arriving from a leaky bucket in the interval $[t_1,t_2]$ from the active source i assuming infinite capacity links can be characterized by the function $A_i(t_1,t_2)$. If $A_i(t)= A_i(t,0)= \sigma_i + \rho_i t$, then by definition session i starts *greedy*, i.e., it starts with its maximal burst at time zero and continues to transmit with its maximal rate ρ_i. If all sessions start greedy one gets a *greedy GPS system*.

Suppose that $C_j > r$ for every session j, where C_j is the internal link capacity between the session j leaky bucket and the session j queue, and r is the GPS server capacity. 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[4]. Furthermore, assuming that for each session i $g_i \geq \rho_i$, then

$$Q_i^* \leq \sigma_i \quad \text{and} \quad D_i^* \leq \frac{\sigma_i}{g_i} \qquad (7)$$

The GPS service discipline 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

[3] The session buffer is not empty.

[4] An interval in which the server is continuously working.

capacity. 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.

4.1 Parameter conversion

Using ARR the rate of the integer k_i and the sum of k_js (L) express the bandwidth guaranteed for connection j in group i. Rewrite (6) we get

$$\frac{W_{ij}(t_1,t_2)}{W_{xy}(t_1,t_2)} \geq \frac{\varphi_i}{\varphi_x} \quad x=1,2\ldots G \quad j=1,2\ldots N_i \quad y=1,2\ldots N_y$$

for any session *(i,j)* that is continuously backlogged in the interval $[t_1,t_2]$. The minimum *guaranteed service rate* for any connections in group i is

$$g_i = \frac{k_i}{\sum_{j=1}^{G} N_j k_j} C = \frac{k_i}{L} C \qquad (8).$$

If we want our ARR to provide the same worst case guarantees as the GPS then from the comparing of (1) and (7) we get

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

Substituting g_i from (8) we get $\sigma_i = B_{ij} l$.

By these simple evaluations we have shown that ARR could be approximated by GPS. Obviously, all of the refinements of the worst case guarantees of GPS (e.g. [6]) can be easily applied in the analysis of ARR.

5 Conclusions

In this study a new scheduling method called Advanced Round Robin has been proposed. Using the results of this algorithm a new Call Admission Control 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 relationship between the proposed ARR algorithm and GPS was investigated.

References

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

[2] T. Marosits, S. Molnár, 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.

[3] T. Marosits, S. Molnár, J. 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

[4] A. K. Parekh, R. G. Gallager: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case, *IEEE/ACM Transactions on Networking*, 1, p 344-357, 1993

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

[6] 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.

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