Local Resource Allocation for Providing End to End Delay Guarantees in ATM Networks Using PGPS Scheduling

Amr S. Ayad
asaad@ie-eg.com

Khaled M. F. Elsayed
khaled@ieee.org

Mahmoud T. El-hadidi
hadidi@frcu.eun.eg

Department of Electronics and Communications Engineering
Faculty of Engineering, Cairo University
Giza, Egypt 12613
Fax: +202 5723486

Abstract

The paper addresses the issue of reserving resources at ATM switches along the path of flows that require a hard (deterministic) bound on end-to-end delay. The switches are assumed to be using Packet-by-packet Generalized Processor Sharing (PGPS) schedulers for each outgoing link. An algorithm for call admission control of these flows is proposed. Different policies for mapping the end-to-end delay requirement into service rates to be assigned at each scheduler are suggested and analyzed.

1. Introduction

One of the main promises of ATM networks is to provide Quality-of-Service (QoS) guarantees, such as Cell Transfer Delay (CTD) and Cell Loss Ratio (CLR), to flows requiring different classes of services. Handling the variety of QoS requirements of many applications requires network switches to have a mechanism for serving flows according to their required QoS. Many scheduling disciplines have been proposed in the literature to implement such a mechanism. Examples include Packet-by-packet Generalized Processor Sharing (PGPS) [1], Earliest Deadline First (EDF) [6], and Stop & Go [7]. Each scheduling discipline requires algorithms for performing call admission control (CAC) and resource reservation.

In this paper, we propose such algorithms for the case of PGPS scheduling and of flows requiring a hard (deterministic) bound on end-to-end delay. In PGPS scheduling, the resource to be reserved at each scheduler is a guaranteed minimum service rate.

Our approach is based on the mapping of end-to-end QoS requirement into local resources to be reserved at each scheduler. Towards this end, the paper addresses the following questions:

1- How to map the end-to-end delay requirement of a flow into a local resource requirement to be reserved at each scheduler along the flow's path?

2- How to divide the resource requirement among the schedulers on the flow's path? That is, a simple division policy would be to reserve an equal amount of resources at all schedulers. However, it may be more efficient to use a policy that reserves resources in a way that takes schedulers capacities and/or loading into account.

3- How much gain (if any) would be obtained from applying more complex division policies rather than the policy of equal resource assignment? What are the factors controlling the value of this gain?

In this paper, routing of connections is not addressed. We assume that the routing decision has already been made. This allows us to focus on the problem of resource allocation without having to address the combined problem of route selection and resource allocation.

2. Delay formulas of PGPS in ATM networks

The PGPS scheduling discipline is a non-preemptive version of the Generalized Processor Sharing (GPS) scheduling discipline [1]. GPS operation is defined as follows: a GPS scheduler (*j*) that serves N flows is characterized by N positive real numbers $\phi_1^j, \phi_2^j, \ldots\ldots \phi_N^j$. All flows are served simultaneously such that a flow (*f*) is guaranteed (when it has traffic to send) a minimum service rate of

$$g_f^j = \frac{\phi_f^j}{\sum_{k=1}^{N} \phi_k^j} C^j \qquad (1)$$

Where C^j is the data rate of the link following scheduler (*j*) and will be denoted as the scheduler capacity.

GPS is not practically realizable since it assumes the simultaneous transmission of packets from all flows. PGPS closely approximates GPS by serving packets in the ascending order of service tags assigned to them. The service tags are computed as the finish times of those packets had a GPS scheduler having the same capacity and the same input traffic served them.

It has been shown in [1] and [2] that for a flow (f) that traverses a path of K_f schedulers, and has a source traffic characteristics that conforms to a linear bounded arrival process (LPAB) (with a maximum burst size of (σ_f) bits, long term average rate of (ρ_f) bps), an upper bound on the end-to-end delay (D_f^i) encountered by i[th] packet of flow (f) can be guaranteed by reserving a certain service rate at each scheduler along the flow path from source to destination. For PGPS scheduling, the value of this upper bound in seconds (from [2], Equations. 26-27) is:

$$D_f^i = \frac{\sigma_f}{g_f(i)} + \left(\sum_{j=1}^{j=K_f} \alpha^j\right) + \left(\sum_{j=1}^{j=K_f-1} \max_{n\in[1,i]}\left(\frac{l^n}{g_f^{n,j}}\right)\right) + \left(\frac{l^i}{g_f(i)} - \frac{l^i}{g_f^{i,K_f}}\right) \quad (2)$$

Where:

$g_f^{n,j}$ = Service rate of the n[th] serviced packet at scheduler (j) along the flow's path.

$g_f(i) = \min_{j\in[1,K_f]} g_f^{i,j}$ = Minimum service rate received by the i[th] packet along the flow's path.

l^i = The length of the i[th] packet in bits

$\alpha^j = \beta^j + \tau^{j,j+1}$, $\beta^j = l(j)_{\max}/C^j$

$l(j)_{\max}$ =Maximum packet length served by scheduler (j)

$\tau^{j,j+1}$ = Propagation delay from scheduler (j) to scheduler (j+1)

Note that [2] deals with the more general case in which each packet belonging to the same flow may be assigned a different rate from other packets of the same flow even at the same scheduler. We will not make use of this case, and will consider the more practical case in which all packets from the same flow are served at the same rate at a given scheduler. Taking this into account, a simpler form of (2) is:

$$D_f^i \le \frac{\sigma_f - l^i}{g_f} + \left(\sum_{j=1}^{j=K_f} \alpha^j\right) + \left(\left(\sum_{j=1}^{j=K_f-1} \frac{\max_{n\in[1,i]} l^n}{g_f^j}\right) + \frac{l^i}{g_f^{K_f}}\right) \quad (3)$$

Where:

g_f^i =Service rate of packets from flow (f) at scheduler (j)

$g_f = \min_{j\in[1,K_f]} g_f^j$

For ATM networks with fixed packet length (i.e. ATM cells), (3) becomes:

$$D_f^i \le \frac{\sigma_f - L}{g_f} + \left(\sum_{j=1}^{j=K_f} \alpha^j\right) + \left(\sum_{j=1}^{j=K_f} \frac{L}{g_f^j}\right) \quad (4)$$

Where L is the length of the ATM cell (424 bits).

For a scheduler with N accepted flows, The above formulas are valid under the following conditions:

1-The sum of reserved rates of accepted flows is no greater than the scheduler capacity, that is

$$\sum_{j=1}^{N} g_f^j \le C^j \quad (5)$$

2-The sum of average rates of accepted flows is no greater than the scheduler capacity (stability condition), that is

$$\sum_{f=1}^{N} \rho_f \le C^j \quad (6)$$

3. An algorithm for call admission control

The proposed CAC algorithm (see Figure 1) uses (4) to determine whether to accept or reject a new flow. It operates as follows:

The first test compares the value of the end-to-end delay requested by the incoming flow (D_f) with the value of the total transmission and propagation delay along the flow's path. If the value of the required end-to-end delay is smaller, the flow is rejected.

The next test is to compare the value of the required end-to-end delay of the new flow with the minimum end-to-end delay bound (D_f^*) that the network can guarantee to the incoming flow. If the value of the required end-to-end delay is smaller, the flow is rejected.

The minimum end-to-end delay bound (D_f^*) is obtained from (4) with each scheduler on the flow's path reserving a service rate equal to its remaining capacity. The remaining capacity of a PGPS scheduler (j) prior to the acceptance of a flow (f) is defined as:

$$R_f^j = C^j - \sum_{i=1}^{N} g_i^j \quad (7)$$

R_f denotes the minimum remaining capacity along the path of flow (f).

On passing the previous tests successfully, a division policy is used to map the required end-to-end delay into a local resource reservation at each scheduler. Different division policies are discussed in section 5.

Finally a test is made to verify that the conditions in (5) and (6) hold true if the new flow is accepted. If the conditions are met, the flow is accepted. The algorithm is shown in Figure 1. A mapping of the CAC algorithm to ATM call setup procedures is given in section 7.

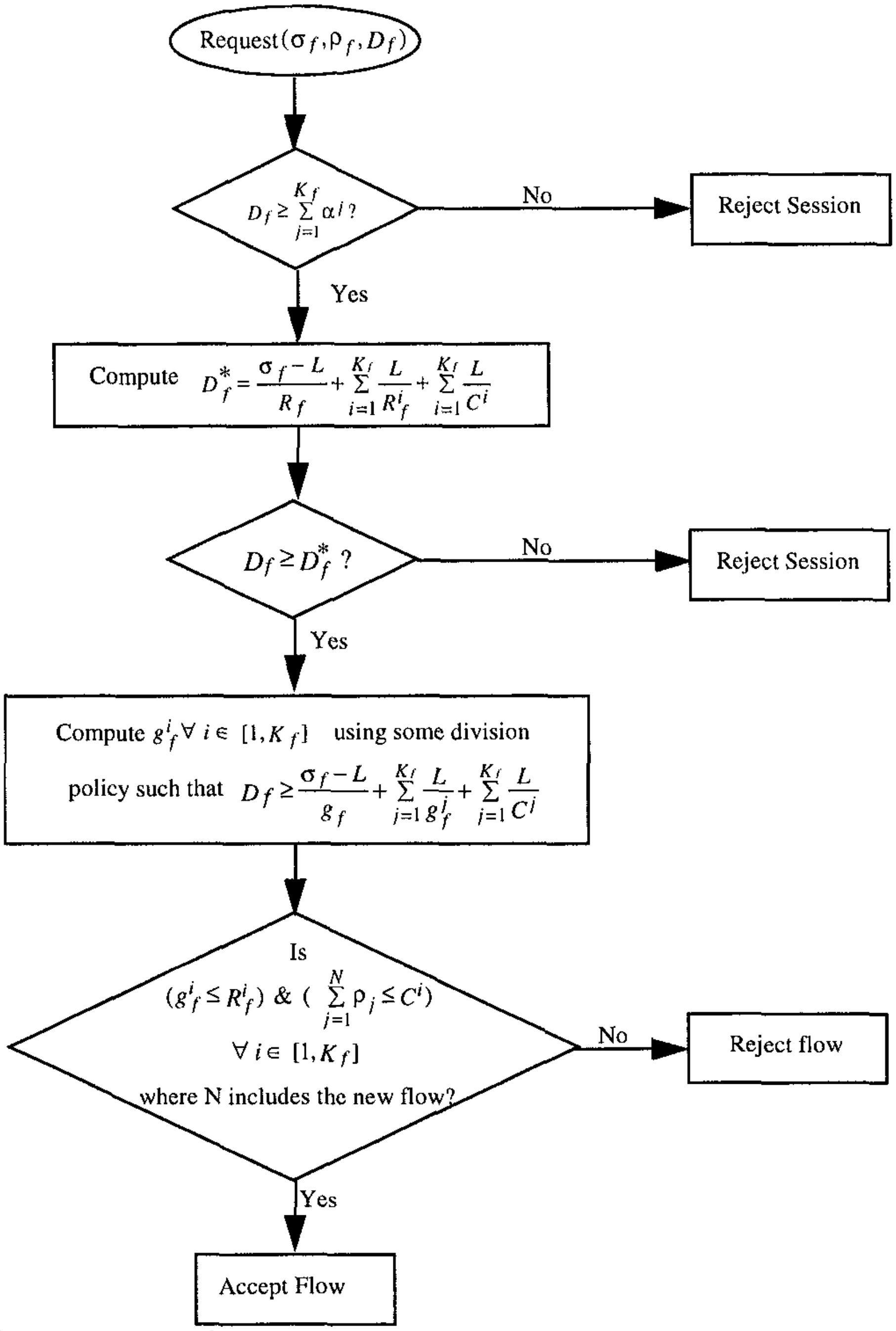

Figure 1. CAC algorithm for a network of PGPS schedulers

4. Local resource reservation

There are two approaches to compute the rate to be reserved at each scheduler along the flow's path. One approach is to map the required end-to-end delay bound D_f into a local delay bound (d_f^j) that must be guaranteed at each scheduler, and then compute the rate to be reserved to guarantee the local delay bound. The other approach is to directly use the end-to-end delay expression (4) to compute the rate to be reserved at each scheduler.

The first approach depends on noting that inequality (4) gives an upper bound on the end-to-end delay experienced by an ATM cell traveling through K_f schedulers. Rewriting the inequality for K_f=1 would give the delay bound due to one scheduler, i.e. the delay bound due to the first scheduler on the path from the source to the destination is:

$$d_f^1 \le (\sigma_f / g_f^1) + \alpha^1 \tag{8}$$

Which is the same expression derived in [1] for a single PGPS scheduler with a leaky-bucket constrained input traffic.

This expression is true for the first scheduler on the flow's path from the source to the destination. The same inequality holds true for any other scheduler (j) along the flow's path but with the following parameters changed:

- Replacing σ_f by σ_f^j which is the maximum burst size of the flow traffic on arriving to scheduler (j), σ_f^j is computed from the relation ([5], p.117, Theorem 2.1) as:

$$\sigma_f^j = \sigma_f^{j-1} + \rho_f d_f^{j-1} \tag{9}$$

- Replacing g_f in by the service rate reserved for flow (f) at scheduler (j), from (8), (9):

$$d_f^j \le (\sigma_f^j / g_f^j) + \alpha^j \tag{10}$$

Equation (9) implies that the upper bound on delay at each scheduler (d_f^j) will grow as we move towards the destination. Therefore, the sum of these upper bounds (which is the upper bound on end-to-end delay) will be much looser than the bound given in (4) and will thus result in over-reservation of resources.

The other approach is to directly compute the set of rates { g_f^i } that satisfies (4) according to a certain division policy as discussed in the next section. This approach is more efficient as it avoids the over-reservation of resources present in the first approach.

5. Division policies of resource requirement

This section presents three policies for distributing the necessary resources for meeting the end-to-end delay bound among the switches on the flow's path.

5.1 The condition of local stability

A flow (f) is said to be locally stable at scheduler (j) if

$$g_f^i \ge \rho_f \tag{11}$$

The local stability condition is not required for each flow passing by a scheduler as long as all flows have a leaky-bucket constrained traffic pattern. However, if some flow is not leaky-bucket constrained, then (11) must hold true for all accepted flows. Therefore, the local stability condition is not necessary if the network operator uses leaky bucket traffic shapers at the entrance of the network. Since the algorithms used to implement different division policies depend on whether local stability is imposed or not, each case will be treated separately.

5.2 Local stability is not imposed

5.2.1 Even policy (EVEN). In this policy, the reserved rates for a certain flow are the same at all schedulers, i.e.

$$g_f^i = g_f \, \forall i \in [1, K_f] \tag{12}$$

Substituting in (4), after converting it to an equality to reserve the least amount of resources required to meet the delay bound gives:

$$g_f^i = \frac{\sigma_f + (K_f - 1)L}{D_f - \sum_{j=1}^{K_f} \alpha^j} \forall i \in [1, K_f] \tag{13}$$

The rate computed from (13) may be greater than the remaining capacity of one or more schedulers. There are two approaches to account for this case; one approach is to simply reject the flow since the reservation of the whole remaining capacity of a scheduler will prevent it anyway from accepting new flows until one of the ongoing flows finishes service.

Another approach is to reserve all of the remaining capacity (i.e. $g_f^n = R_f^n$) at schedulers with a remaining capacity less than g_f^i as computed from (13) and then redistribute the rest of the QoS requirement on other schedulers. The first approach is simpler and it is not much less efficient than the second more complicated one [9]. The proposed CAC algorithm uses the first approach

The advantage of even policy is the simplicity of implementation when the flows are rejected for the case in which one or more schedulers has $g_f^n \ge R_f^n$ for some $n \in [1, K_f]$. All the parameters required to compute (13) are static (link capacities and propagation delays) and can easily be communicated among network switches during call setup phase.

The disadvantages of the even policy are:

- The fact that (13) by itself does not guarantee that $g_f^i \le R_f^i \,\forall i \in [1, K_f]$. This restriction is removed in the remaining capacity proportional policy as shown in 5.2.3.
- It does not account for the possible imbalances in schedulers' capacities or loading.

5.2.2 Capacity proportional policy (CP). In this policy, the rate reserved at a scheduler for a flow (*f*) is proportional to the scheduler capacity, i.e.

$$g_f^i = \eta_f C^i \,\forall i \in [1, K_f] \tag{14}$$

Where η_f =Constant for the path and flow (*f*) parameters.

Substituting in (4), after converting it to an equality to reserve the least amount of resources required for meeting the delay bound and solving for η_f gives:

$$g_f^i = \frac{\frac{\sigma_f - L}{C} + \sum_{j=1}^{K_f} (L/C^j)}{D_f - \sum_{j=1}^{K_f} \alpha^j} C^i \,\forall i \in [1, K_f] \tag{15}$$

Where $C = \min_j C^j$

The same discussion for even policy applies for the case in which one or more schedulers has $g_f^n \ge R_f^n$ for some $n \in [1, K_f]$.

The advantages of the CP policy are:

- Simple implementation when flows are simply rejected for the case in which one or more schedulers has $g_f^n \ge R_f^n$ for some $n \in [1, K_f]$. All the parameters required to compute (15) are static (link capacities and propagation delays) and can easily be communicated among network switches during call setup phase.
- Accounts for possible imbalance in schedulers' capacities.

The disadvantages of the CP policy are:

- The fact that (15) by itself does not guarantee that $g_f^i \le R_f^i \,\forall i \in [1, K_f]$. This restriction is removed in the remaining capacity proportional policy as shown in 5.2.3.
- It does not account for the possible imbalances in schedulers' loading.

5.2.3 Remaining capacity proportional policy(RCP). In this policy, the rate reserved at a scheduler for a flow (*f*) is proportional to the remaining capacity at this scheduler i.e.

$$g_f^i = \eta_f R_f^i \,\forall i \in [1, K_f] \tag{16}$$

Where η_f = Constant for the path and flow (*f*) parameters as long as no other flows are accepted at any of the schedulers along the flow path's during call setup phase.

Substituting in (4) after converting it to equality to reserve the least amount of resources required for meeting the delay bound and solving for η_f gives:

$$g_f^i = \frac{\frac{\sigma_f - L}{R_f} + \sum_{j=1}^{K_f} (L/R_f^j)}{D_f - \sum_{j=1}^{K_f} \alpha^j} R_f^i \,\forall i \in [1, K_f] \tag{17}$$

Where $R_f = \min_j R_f^j$

Although equation (17) by itself does not guarantee that $g_f^i \le R_f^i \,\forall i \in [1, K_f]$, We prove below that this condition will be automatically met through the process of the proposed CAC algorithm (shown in Figure 1). Thus it avoids the disadvantage present in the two policies described above. The proof is as follows:

For any accepted flow (*f*), the CAC algorithm imposes the following condition

$$D_f \ge D_f^* \tag{18}$$

D_f^* is the minimum feasible end-to-end delay bound that can be guaranteed to flow (*f*) (before actually accepting it) by reserving all of remaining network resources along the path, thus from (4):

$$D_f^* = ((\sigma_f - L)/R_f) + \sum_{j=1}^{j=K_f} L/R_f^j + \sum_{j=1}^{j=K_f} \alpha^j \tag{19}$$

Using (19) into (18) gives:

$$\frac{((\sigma_f - L)/R_f) + \sum_{j=1}^{j=K_f} L/R_f^j}{D_f - \sum_{j=1}^{j=K_f} \alpha^j} \le 1 \tag{20}$$

Comparing (17), (20) gives:

$$g_f^i \le R_f^i \,\forall i \in [1, K_f] \tag{21}$$

It follows from (21) that using the RCP policy in conjunction with the proposed CAC algorithm guarantees that $g_f^i \le R_f^i \,\forall i \in [1, K_f]$.

The advantages of the RCP policy are:

- It accounts for possible imbalances in schedulers' loading
- It automatically guarantee that $g_f^i \le R_f^i \,\forall i \in [1, K_f]$

The disadvantage of RCP policy is that it requires network switches to exchange the time-varying remaining capacity information.

5.3 Local stability is imposed

In this case, all flows must be locally stable at each scheduler along the flow's path. That is:

$$g^i_f \geq \rho_f \; \forall i \in [1, K_f] \tag{22}$$

Therefore the minimum possible assignment of rates (irrespective of the required delay bound) is:

$$g^i_f = \rho_f \; \forall j \in [1, K_f] \tag{23}$$

The following two cases follow:

1-If the assignment in (23) satisfies (4) for an incoming flow (*f*), then this is the optimal resource assignment for flow (*f*), since it satisfies the required delay bound with the minimum amount of reserved resources.

2-If the assignment in (23) does not satisfy (4) for an incoming flow (*f*), then one of the above policies is needed to compute the amount of resources to be reserved at each scheduler.

5.3.1 Even policy. If the assignment in (23) satisfies the required delay bound, then this will be the set of rates to be reserved for the new flow. Otherwise, use (13) to determine { g^i_f }.

5.3.2 Capacity proportional policy. If the assignment in (23) satisfies the required delay bound, then this will be the set of rates to be reserved for the new flow. Otherwise, use (15) to determine { g^i_f }. The following two cases follow:

i- If $g_f \geq \rho_f$ where g_f is the minimum reserved rate (it occurs at the scheduler with the minimum capacity), then { g^i_f } is as obtained from (15).

ii- If $g_f < \rho_f$, then there are two approaches to solving this problem. One approach is to take $g_f = \rho_f$, and $g^j_f = \rho_f \frac{C^j}{C}$. This approach may involve an amount of over-reservation of resources. The amount of over-reservation will increase with the imbalance in schedulers' capacities. Another (and probably more efficient) approach is to modify the rate assignment resulting from (15) for the schedulers with $g^j_f < \rho_f$ by taking $g^j_f = \rho_f$ for such schedulers.

5.3.3 Remaining capacity proportional policy. If the assignment in (23) satisfies the required delay bound, then this will be the set of rates to be reserved for the new flow. Otherwise, use (17) to determine { g^i_f }. The following two cases follow:

i- If $g_f \geq \rho_f$ where g_f is the minimum reserved rate (it occurs at the scheduler having the minimum remaining capacity), then { g^i_f } is as obtained from (17).

ii- If $g_f < \rho_f$, then one of the two approaches discussed in the case of (CP) may be applied.

6. Performance gain of non-even division policies

The objective of this section is to investigate the factors affecting the amount of performance gain of non-even division policies. To calculate the amount of performance gain, a network consisting of a single path of schedulers (Figure 2) is used to avoid the effect of cross-traffic present in more general network topologies. Determining the factors affecting the amount of gain improvement will subsequently help in explaining simulation results made over more general network topologies.

Evaluating the performance gain of non-even division policies requires the definition of a performance metric against which different policies may be compared. A suitable performance metric would be the blocking probability of incoming calls. For the case of a single path network, the maximum number of flows (N) supportable by the path while still meeting QoS guarantees of all accepted flows is a measure of the call blocking probability. The network may be modeled as a M/M/N/N system, in which case the blocking probability can be easily evaluated from Erlang B formula with N servers.

We define the relative gain value of a certain policy with respect to even division policy as:

$$G_{policy} = \frac{N_{policy} - N_{EVEN}}{N_{EVEN}} x100\% \tag{24}$$

Where N_{policy} is equal to the maximum number of simultaneously supported calls on a single path network.

A problem that impedes the estimate of the gain value is that it depends on the following factors:

- i- Link capacities configuration.
- ii- Source traffic characteristics.
- iii- Requested delay bound.

With the large number of possible combinations of the values of the above factors, it is not easy to derive a general result for the gain value. To evaluate the above policies, the following simplifying assumptions are made:

- i- Source traffic parameters (σ_f, ρ_f) are the same for all flows. They are used as parameters against which the gain value is computed.
- ii- Only one class of QoS is assumed, i.e. all flows request the same value of end-to-end delay.

For even policy, (13) gives:

$$N_{EVEN}=\frac{C}{g_f}=\frac{D_f-\sum_{j=1}^{K_f}\alpha^j}{\sigma_f+(K_f-1)L}C \qquad (25)$$

and from (15), (17):

$$N_{CP}=N_{RCP}=\frac{C^j}{g_f^j}=\frac{D_f-\sum_{j=1}^{K_f}\alpha^j}{\frac{\sigma_f-L}{C}+\sum_{j=1}^{K_f}(L/C^j)} \qquad (26)$$

Note that the gain of CP & RCP is the same when all schedulers are initially unloaded, because the proportionality of the remaining capacities at all schedulers will always be the same as the proportionality of their total capacities (which are the initial remaining capacities for initially unloaded schedulers).
Using (24), (25), and (26) gives:

$$G_{CP}=G_{RCP}=\frac{(K_f/C)-\sum_{j=1}^{K_f}(1/C^j)}{\frac{(\sigma_f/L)-1}{C}+\sum_{j=1}^{K_f}(1/C^j)} \qquad (27)$$

We note that the improvement factor is directly proportional to K_f and inversely proportional to. σ_f/L

Figure 3 shows the effect of different parameters in (27) for the special case of two switches in tandem. The ratio of link capacities is varied to represent the effect of imbalance of network resources.

We note the gain increases with ratio of link capacities (i.e. more imbalance in network resources). We note that the gain obtained by applying CP and RCP decreases rapidly with the increase in burst size. From (27), when the capacity imbalance tends to infinity, the gain will reach a maximum of $(K_f-1)L/\sigma_f$

Figure 4 shows the effect of different parameters in (27) for the special case of (K_f) switches in tandem with the capacity of each link equal to the capacity of the previous link multiplied by a factor (m >1). Thus the factor (m) reflects the size of imbalance in network resources. The burst size is fixed to one cell. Although such topology is not realistic, it reflects the effect of both the imbalance in network resources and the number of schedulers (K_f) serving the flow on the gain value.

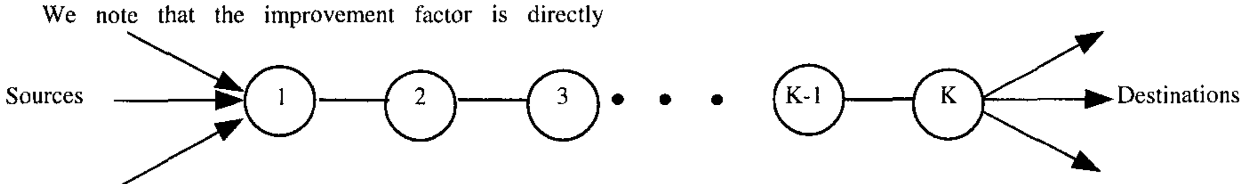

Figure 2. Single path network model

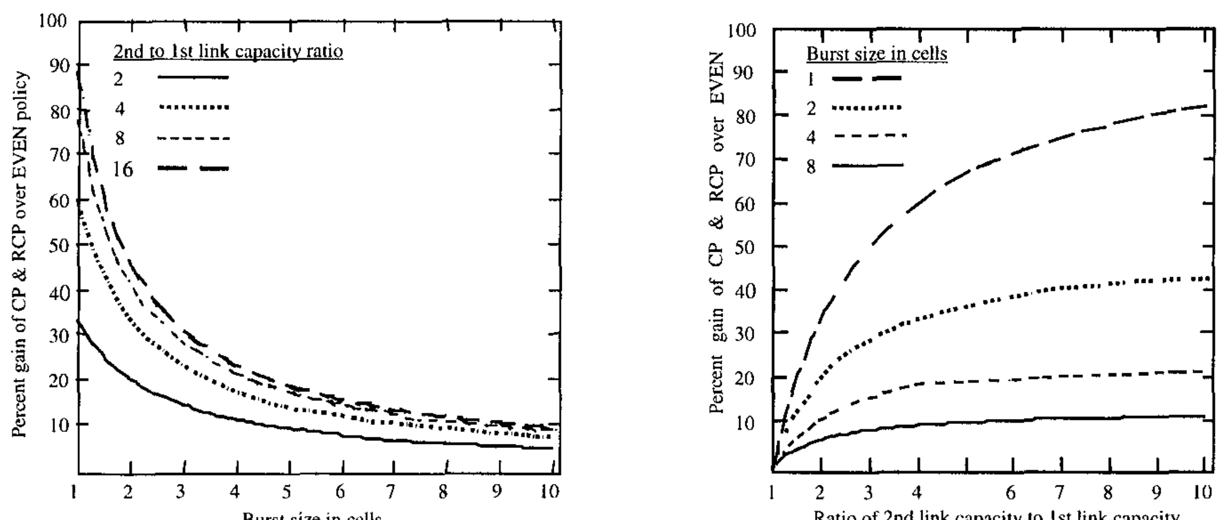

Figure 3. Two switches

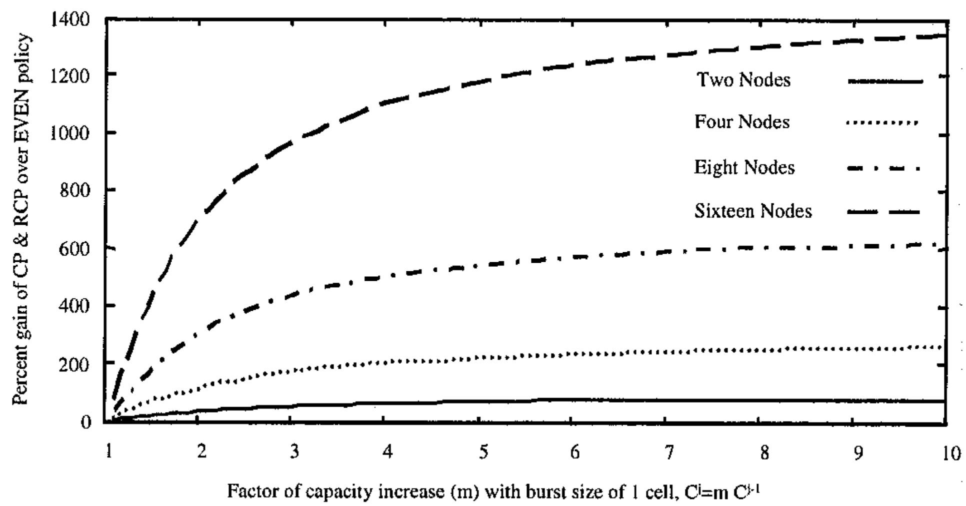

Figure 4. Link capacity increases exponentially along the path, $C^j = m\, C^{j-1}$

7. Implementation in ATM networks

This section outlines how to map the proposed CAC algorithm to call setup procedure in ATM networks (see [6], [8]). The calling entity sends a SETUP message to the destination, including the flow's traffic parameters (e.g. burst size and average rate) and the required end-to-end delay bound. At each scheduler along the flow's path, the stability condition in (6) is examined should the new call be accepted. If the condition is not met, the flow is rejected and a RELEASE message is returned. Otherwise, the scheduler appends its state information (e.g. capacity, remaining capacity) to the SETUP message and forwards it to the next scheduler on the flow's path. If the SETUP message reaches the final scheduler, it also tests the stability condition and if met, it computes the rates to be reserved at each scheduler and returns the set of reserved rates in a CONNECT message sent on the same path to the source.

8. Conclusions and future work

The remaining capacity proportional policy is the most efficient from a call blocking probability point of view in the single path network. However, the amount of gain over the simple even division policy is highly dependent on the burst size of the source traffic. For more bursty sources, the gain is very limited, and a simple even policy may be enough for such case.

Future work includes simulating the proposed policies on more general network topologies to account for the effect of cross-traffic. The simulation will also use different traffic parameters and different QoS requirements for different flows.

9. References

[1] A.K. Parekh. "A generalized processor sharing approach to flow control in integrated services networks," Ph.D. dissertation, Dep. Elec Eng. Comput. Sci., M.I.T., Feb 1992.

[2] P. Goyal and H.M. Vin. "Generalized guaranteed rate scheduling algorithms: A framework," Department of Computer Sciences, University of Texas at Austin, 1995.

[3] V. Firoiu and D. Towsley. "Call admission and resource reservation for multicast flows," Technical Report TR 95-17, Univ. of Massachusetts, Amherst, 1995.

[4] R. Nagarajan, J. F. Kurose and D. Towsley. "Local allocation of end-to-end quality of service in high-speed networks," Proc. of IFIP Workshop on the Performance Analysis of ATM Systems, (Martinique, Jan. 1993).

[5] R.L. Cruz. "A calculus for network delay, Part I: Network elements in isolation," IEEE Transactions on Information Theory, vol. 37, pp. 114-131, January 1991.

[6] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," IEEE JSAC, vol. 8, no. 4, pp. 368-379, April 1990.

[7] S. J. Golestani, "A framing strategy for congestion management," IEEE JSAC, vol. 9, no. 7, September 1991.

[8] V. Firoiu, J. Kurose, and D. Towsley. "Efficient admission control for EDF schedulers," In IEEE Infocom'97, April 7-11, 1997 in Kobe, Japan.

[9] Amr S. Ayad. "Resource reservation for deterministic end-to-end Delay services in ATM networks," Master thesis, Dept. of Electronics and Communications,. Cairo University, In preparation.