Evaluating Admission Control Modules for Bandwidth Brokers in DiffServ Networks Using ns-2

Ch. Bouras K. Stamos
Research Academic Computer Technology Institute, PO Box 1382, Patras, Greece and
Computer Engineering and Informatics Dept., Univ. of Patras, GR-26500 Patras, Greece
Tel:+30-2610-{960375, 990316}
Fax:+30-2610-{969016, 960358}
e-mail: {bouras, stamos}@cti.gr

Abstract **- In this paper we use the well known network simulator ns-2 in order to study the performance characteristics of various alternatives for the admission control module of a Bandwidth Broker. In particular, we present the modifications that have to be made at the ns-2 architecture in order to simulate these architectures, the metrics such as acceptance rate, fairness towards requests and computation overhead and how they are affected when comparing a simple admission control module and more sophisticated ones. Furthermore, we examine the benefits that can arise under various circumstances through usage of enhancements such as the ability to handle the resubmission of previously rejected requests.**

I. INTRODUCTION

The growing trend towards provisioning of Quality of Service (QoS) in the Internet has led to the development of several frameworks, among which DiffServ (Differentiated Services) is probably the most widely deployed one. The Bandwidth Broker [1] is an entity that manages the resources within a specific DiffServ domain by controlling the network load and by accepting or rejecting bandwidth requests. Every user (service operator) who is willing to use an amount of the network resources, between its node and a destination, sends a request to the Bandwidth Broker. For requests that span multiple domains (inter-domain requests), the Bandwidth Broker will have to communicate with Bandwidth Brokers in the adjacent domains that are traversed by the requested flow. The Bandwidth Broker architecture makes it possible to keep state on an administrative domain basis, rather than at every router and the DiffServ architecture makes it possible to confine per flow state to just the leaf routers. Bandwidth Brokers are an intensely studied field and a number of architectures have been proposed for the various aspects of its operation ([2], [3], [4], [5]).

Despite the large amount of related work that contains initial evaluations of the proposed architectures, there is a lack of thorough comparative investigations of the relative performance and benefits of adaptive admission control architectures, in a simulated environment with low-level networking details. This is the focus of our work in this paper. The simulator within which we implemented a Bandwidth Broker is the Network Simulator (ns-2) [6], which is a free and widely used open-source simulator.

The rest of the paper is organized as follows: Section II describes the problem related to the admission control module of the Bandwidth Broker. Section III provides a brief description of the algorithms evaluated in this paper, while section IV presents the simulation setup and the topologies that were used for the ns-2 simulator. Section V contains the results of the evaluations, and section VI presents our conclusions from the evaluations and the future work that we intend to do in this area.

II. PROBLEM DESCRIPTION

In order for the Bandwidth Broker to decide whether an incoming resource reservation request will be accepted or not it has to perform some sort of admission control. This function can be performed using straightforward algorithms or using more sophisticated architectures. Generally, the problem of deciding which requests should be satisfied in the context of limited resources has been widely investigated by researchers and similar ideas can be adapted and applied to the DiffServ environment and the Bandwidth Broker entity. Once a request has been accepted, the Bandwidth Broker has to make sure that its requirements will be met by the network. Admission control is very important because it determines the fairness between the requests and the degree of network utilization that will be achieved. An improperly designed admission control module can lead to low network utilization, unfairness and frustration to the users or it can also impose an unacceptable overhead to the system operation.

In general, we can categorize the reservation requests as follows:

Immediate requests: When an immediate request is accepted, it is immediately effective and the requested resources are reserved right away. This type of request leaves little room for implementing a strategy maximizing network utilization.

Book-ahead (or advance) requests: A book-ahead request specifies the resources that will be needed at some later point in time, which has to be specifically defined. A thorough presentation of the concept of book-ahead reservations can be found in [7]. Book-ahead requests allow for better solutions to the admission control problem, and there are a lot of real world cases where a book-ahead type of request is suitable, like for example pre-arranged video conferences.

For both immediate and book-ahead requests, it may be possible to either specifically declare the ending time of the reservation, or not. An intermediate case is when a reservation request has to provide both its starting and ending times, but can make new additional requests that extend its initial reservation period.

In some cases, a book-ahead request may have a flexibility of allowing the Bandwidth Broker to answer the request by either accepting it or rejecting it not immediately, but after a period of time (which can be specified).

For the purposes of our evaluations, we have made the assumption that time is comprised of distinct slots and reservations are processed using a predefined granularity. Also, we have assumed that the network has provisioned in such a way that the hose model [8] can be applied in order to determine the accepted flows at an ingress point of the network, as described in section IV (simulation setup).

III. Description of Algorithms

In this paragraph we briefly introduce the algorithms that were evaluated through our experiments. The algorithms are explained in detail in the corresponding references.

The first algorithm is Simple admission control and has easy implementation and low complexity. Each incoming request is examined alone, and is accepted if there is still available (non reserved) bandwidth for the service. Therefore, this algorithm displays identical behaviour each time it is presented with the same reservation request instance set. This algorithm has the advantage of simple implementation and management, since only the most basic constraints (such as the available resources for the premium service) need to be configured by some administrative entity. For convenience, this algorithm is labeled from now on as SAC.

Another approach to the admission control issue is taken by the Price-based admission control without adaptation. This type of admission control is similar to the offline version of the algorithm presented in [9]. The algorithm tries to optimize the network utilization by gathering and evaluating multiple requests together. In order to solve the resulting NP-complete problem, an approximation algorithm is used which can approximate the optimal solution within a specified range. This algorithm is labeled from now on as PBAC.

The Adaptive admission control [10] is an algorithm that gathers multiple requests and evaluates them together for purposes of increasing the resource utilization, but also uses an adaptation module in order to keep processing requirements low. The adaptation module is responsible for interrupting the process of solving the scheduling problem and for adjusting the size of subsequent instances of the scheduling problem based on constant monitoring of computation time. Because the adaptation module takes into account the computational overhead of the Bandwidth Broker, the output of the algorithm may in theory vary slightly if an experiment is repeated with exactly the same set of requests. In practice however, each time we repeated an experiment we describe in this paper, the output of the algorithm was identical (i.e. the same requests were accepted). Also, the algorithm defines parameters that can influence its behaviour. The first parameter is the adaptation parameter *a*, which takes values in the range from 0 to 1 and determines the aggressiveness of the adaptation (values closer to 1 define more aggressive adaptive behaviour). The second parameter is the threshold, which roughly determines the limit of the computational overhead that the algorithm incurs to the system. These parameters can be defined by the architecture's administrative entity, and in the experiments described in this paper we have used predefined values for these parameters [10], in order not to assume an additional burden for the administrator. This algorithm is labeled from now on as AAC.

Adaptive admission control with resubmissions [11] is a variation of the AAC algorithm. It is enhanced with the capability to recognize previously rejected requests and increase their priority. Other than that, this algorithm is very similar to AAC. The basic idea is that the client will resubmit a rejected request only if the Bandwidth Broker has indicated that the request should indeed be resubmitted, and in addition if the user is willing to compromise for a delayed reservation. In order for the Bandwidth Broker to utilize resubmitted requests, it keeps a list of the standby requests. It will actively prioritize such requests in expense of newly received requests, and the prioritization will depend on the duration that a specific user has been waiting and resubmitting a request. This algorithm is labeled from now on as AACR.

IV. Simulation Setup and Topologies

The simulator was running on an Intel-based Linux PC with 288 MB of main RAM memory available and a Pentium III Coppermine with 256 KB cache memory on the processor chip, which operated at the frequency of 700MHz. The ns-2 software was enhanced in order to simulate QoS-configured networks and specifically to simulate the Bandwidth Broker entities. The patches and instructions for enhancing the standard ns-2 simulator with the required functionality and replicating our results can be found in [12].

For all algorithms compared in this paper we have assumed that the hose model [8], which has been proposed for VPN provisioning, is being used by the domain in order to provision the network resources. Its basic idea is that bandwidth management is simplified by assigning a limit at the bandwidth that each edge router is allowed to accept in the domain. Its operation assumes that proper dimensioning of the network has taken place and that part of the available bandwidth for the links has been assigned to the management of the Bandwidth Broker for the DiffServ service.

The parameters for each request were randomly produced within suitable boundaries (regarding the total duration of

each simulation, the total available bandwidth, the minimum and maximum reservation requests) for each situation that we wanted to simulate, and each set of requests designated a specific ingress point at the network (so all requests competed for the same resource limit at the ingress point of the simulated network). We simulated a scenario where every request had to specify a steady amount of bandwidth for a specific duration with specific time bounds (there was no possibility for a request to specify a variable bandwidth rate). Randomness was obtained by using the ns-2 RNG class. This class contains an implementation of the combined multiple recursive generator MRG32k3a [13]. The period of the generator is 3.1×10^{57}, thus more than adequate for generating randomness for our purposes.

The main metrics that we are interested in, in order to evaluate the algorithms are:

- The acceptance rate, which shows the percentage of requests accepted out of the total number of submitted requests. In case that a flat pricing model is followed (where there is a standard profit per reservation) this metric also shows the network provider's revenue.
- The generated profit for the provider, which is calculated as the product of the bandwidth consumption of each reservation times its duration. Although the pricing model can vary, we believe though that such a metric is representative, since it can be understood as the amount of resources that is consumed by a reservation and the sum for all reservations shows the achieved network utilization. We are also interested in the average profit achieved per request, which can be in several environments an additional indicator of the effectiveness of the algorithm. For example, when there is an additional overhead to the provider for signaling and allocating a new reservation, it would be beneficial to achieve better network utilization per individual request.
- The delay of being able to deliver either positive or negative answers to the submitted requests.
- The average size of the set of requests examined together, which is a measure of the complexity of the optimization problem solved, and therefore of the overhead to the system.

Maximum available bandwidth for the service was set at 100 Mbps, while the duration of each simulation was set at 50 time slots. For algorithms AAC and AACR, the results were obtained setting the adaptation parameter *a* at a value of 0.5 (moderate adaptation) and a computational threshold of 5 time slots for AACR and both 5 and 10 time slots for AAC. The used topology was a simple star network, with the Bandwidth Broker module being located in the centre and requests originating from one leaf node towards another leaf node of the network.

V. Evaluation Results

For each experiment we have measured the percentage of accepted requests, the delay that was required before the Bandwidth Broker would reply to a request and the percentage of network utilization achieved by each algorithm. These results are summarized in Table 1.

Table 1. Summary of results

Algorithm (averages)	Acceptance rate	Delay (time slots)	Network utilization (bytes x time slots)
SAC	29.60%	0	3920014
PBAC	21.79%	7.08	5243307
AAC thr=5	25.72%	5.44	4532672
AAC thr=10	24.77%	5.48	4780385
AACR	42.56%	5.58	5594577

As Figure 1 demonstrates, the acceptance ratio of all algorithms except AACR remains fairly similar throughout the experiments. SAC is the algorithm that slightly achieves the highest acceptance rate, while PBAC is the one with the lowest, with AAC variations covering the middle. This is not a surprising result, since SAC will always accept a request if there are enough resources available, while PBAC is more oriented towards generating the maximum amount of resource utilization, rather than treating all requests alike. Because of the resubmission capability, AACR displays clearly better performance with regard to this metric. This result leads us to the conclusion that in environments where the most significant factor is the satisfaction of the maximum amount of users regardless of their relative weight, the good performance of the SAC algorithm combined with its simplicity make it the most suitable choice. If resubmissions are desirable and supported, AACR can be preferred.

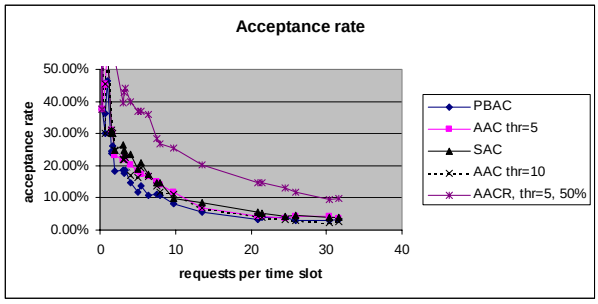

Figure 1. Acceptance rate

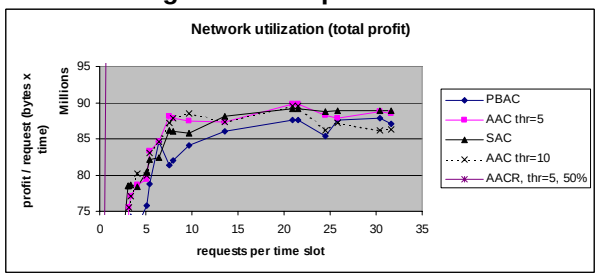

Figure 2. Network utilization

In most cases however, all users will not generate the same revenue for the network provider and a cost scheme will most probably have to take into account both the relative weight of each request, and the effort to maximize the efficiency and utilization of currently available resources. We have tried to cover this aspect with Figure 2 and Figure 3, which display the total absolute profit generated for each experiment and the profit per request respectively. We have chosen to measure the provider's profit by calculating the product of a request's

duration (in time slots used by the ns-2 simulator) times the resource allocation that a reservation requires.

We have to mention that in Figure 2 AACR results are not displayed because they are far larger than all other results, in order to have better distinguishing capability for the rest of the algorithms. These results demonstrate the relative strengths of the price-based approaches, since PBAC is the most efficient algorithm in this regard, followed by AAC, with SAC displaying the worst performance. AAC even surpasses the PBAC performance in several cases when the request arrival ratio increases. The most plausible explanation for this result is that the increased arrival rate of new requests makes the larger size of the set examined by the PBAC algorithm unnecessary. Increasing the threshold for the AAC algorithm seems to have a positive effect on its performance, but comparison with PBAC shows that a restrained increase in the threshold value is enough for obtaining equal or superior results. Therefore, the recommendation for fine-tuning the AAC algorithm is that it is beneficial to increase the threshold value as soon as the arrival rate of request increases. As expected, AACR again displays the best overall performance, which on the case of total profit exceeds several times the results of other algorithms.

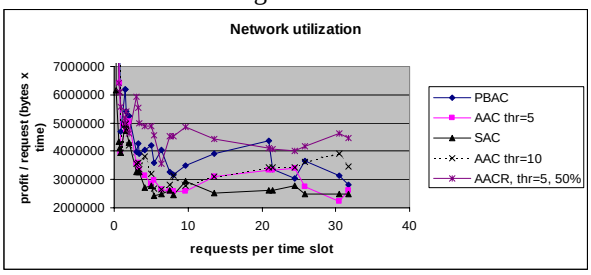

Figure 3. Network utilization per request

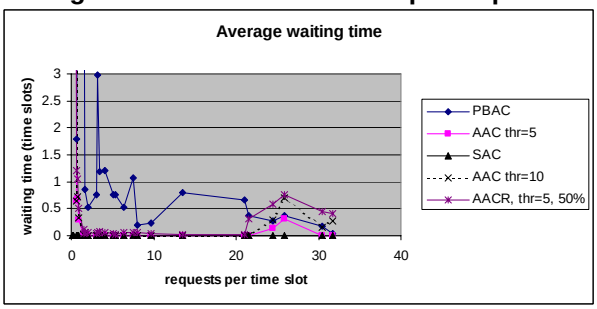

Figure 4. Average waiting time

In most real environments it is expected that a relatively quick response to a request will be essential. As Figure 4 demonstrates, SAC is extremely responsive. This suggests that there is room for a trade-off to improve performance in other areas such as the utilization of the network resources. PBAC is not efficient in that regard, as it demands the most time in order to respond to the reservation requests, a situation that in many real-world scenarios is unattainable. The adaptive variations prove to be attractive trade-offs, since for most of the experiments the additional delay they incur is minimal, while at the same time they manage to improve the utilization of the provider's resources, as demonstrated above.

VI. Conclusions – Future work

We believe that the results presented in this paper offer a strong case for the adaptive algorithms (AAC and AACR) in cases where more efficiency in the utilization of the network resources is required, since their adaptive nature incurs minimal overhead and very small delays to the request responses. In that sense, they offer useful alternatives for real world situations by combining the benefits of the simple SAC and the more complicated PBAC algorithm, without introducing any significant drawback of their own.

However, the nature of the studied environments and the multiplicity of factors that affect the outcome of the experiments encourages us to investigate more scenarios that simulate several discrete actual cases and circumstances. Our plans for future work also include the examination and comparative evaluation of the advantages of distributed designs, as well as their impact and overhead for the network.

References

[1] RFC 2638 "A Two-bit Differentiated Services Architecture for the Internet", K. Nichols, V. Jackobson, L. Zhang, July 1999

[2] S. Sohail, S. Jha, "The Survey of Bandwidth Broker", Technical Report UNSW CSE TR 0206, School of Computer Science and Engineering, University of New South Wales, Sydney 2052, Australia, May 2002

[3] J. Ogawa, A. Terzis, S. Tsui, L. Wang, L. Zhang. "A Prototype Implementation of the Two-Tier Architecture for Differentiated Services", RTAS99 Vancouver, Canada

[4] C. Brandauer, W. Burakowski, M. Dabrowski, B, Koch, H. Tarasiuk, "AC algorithms in Aquila QoS IP network", 2nd Polish-German Teletraffic Symposium PGTS 2002, Gdansk, Poland, September 2002

[5] C. P. W. Kulatunga , J. Kielthy, P. Malone, M. Ófoghlú, "Implementation of a simple Bandwidth Broker for DiffServ networks", IPS 2004, Budapest, Hungary, March 2004

[6] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/

[7] A. Greenberg, R. Srikant, W. Whitt, "Resource Sharing for Book-Ahead and Instantaneous-Request Calls", IEEE/ACM Transactions on Networking, February 1999

[8] N. Duffield and P. Goyal and A. Greenberg and P. P. Mishra and K. K. Ramakrishnan and J. E. van der Merive, "Flexible Model for Resource Management in Virtual Private Networks", SIGCOMM 1999, pp. 95-108

[9] C. Chhabra, T. Erlebach, B. Stiller, D. Vukadinovic "Price-based Call Admission Control in a Single DiffServ Domain", TIK-Report Nr. 135, May 2002

[10] C. Bouras, K. Stamos, "An Adaptive Admission Control Algorithm for Bandwidth Brokers", IEEE NCA 2004, Cambridge, MA, USA, pp. 243-250

[11] C. Bouras, K. Stamos, "Resubmissions and Partly Defined Requests in an Adaptive Admission Control Algorithm for Bandwidth Brokers", 5th International Conference on Networking (ICN 2006), 23-26 April (to appear)

[12] http://ouranos.ceid.upatras.gr/diffserv-ns/intro.htm

[13] Pierre L'Ecuyer. Good parameters and implementations for combined multiple recursive random number generators. Operations Research, 47(1):159–164, 1999.