Comparative Evaluation of Adaptive Price-based Admission Control Algorithms for Bandwidth Allocation

Ch. Bouras K. Stamos
Research Academic Computer Technology Institute, PO Box 1122, 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 analyze the performance characteristics of various alternatives for the admission control module of a Bandwidth Broker. In particular, we study 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 an adaptive one. Furthermore, we examine the benefits that can arise under various circumstances through usage of adaptive price-based admission control and enhancements such as the ability to handle the resubmission of previously rejected requests. We describe the results of our experimentation using a simulated environment and the operating circumstances that benefit each solution.**

I Introduction

The Bandwidth Broker [9] 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. Bandwidth Brokers have been proposed in the framework of the DiffServ architecture for Quality of Service (QoS) provision in the Internet. In order to offer better scalability than other architectures such as IntServ, the DiffServ architecture [1] only deals with individual flows at the edges of a domain, allowing the core elements of the network to only handle classes of service.

Bandwidth Brokers are an intensely studied field, and a number of architectures have been proposed for the various aspects of its operation [10]. Reference [2] proposes a novel architecture for the admission control module that aims at maximizing the resource utilization for the network provider, while keeping the computation requirements of the admission control module of the Bandwidth Broker relatively low. This model is extended in [3] in order to provide support for resubmissions of requests that were previously rejected. Although the above mentioned references contain initial evaluation of the proposed architectures, there is yet no thorough comparative investigation of the relative performance and benefits of these architectures over each other, which is the focus of our work in this paper in order to identify which admission control architecture is more suitable for various circumstances.

The rest of the paper is organized as follows: Section II presents the functionalities of the admission control module of a Bandwidth Broker and the various algorithms that have been proposed for its operation, while section III presents the algorithms that were evaluated in this paper. The setup of the simulations is described in section IV and section V gives the detailed analysis of the results of the simulation experiments that were carried out in order to comparatively evaluate the admission control algorithms. Finally section VI presents the conclusions and our future work in this area.

II Admission Control Module

Admission control is the task that a Bandwidth Broker has to perform in order to decide whether an incoming resource reservation request will be accepted or not. Once the request has been accepted, the Bandwidth Broker has to make sure that it will be met by the network. Admission control is a very important part of the Bandwidth Broker operation, because it determines the fairness between the requests and the degree of network utilization that the Bandwidth Broker will achieve for the managed domain. An improperly designed admission control module can lead to low network utilization, unfairness and therefore frustration to the users that request resources or it can also impose an unacceptable overhead to the Bandwidth Broker's operation.

In general, we can separate the types of reservation requests depending on the actual time period for which they request resources.

Immediate requests: When an immediate request is accepted, it is immediately effective, which means that the requested resources are reserved right away. This type of request leaves little room to the Bandwidth Broker for implementing a strategy that maximizes the 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 [11]. In general, book-ahead requests allow for better solutions to the admission control problem, and there are a lot of actual cases in the real world where a book-ahead request meets the requirements

of an application, like for example pre-arranged video conferences.

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).

Admission control can be done either on a hop-by-hop basis [4] or on a per-flow basis [12]. The former case can be implemented by first calculating the path for an end-to-end reservation through a routing protocol like RIP or OSPF, and then run the admission control algorithm for each link in the calculated path. This however presents the problem of implementing an efficient solution of merging the admission decisions of all links on a requested path. Reference [6] proposes a flexible model that can significantly simplify the management of resources in a virtual private network. We evaluate the algorithms using this same model, where edge routers have predefined limits on the resources that they are able to use, thus freeing the Bandwidth Broker to make a single admission decision for the ingress point of the request (and an additional decision for the egress point).

Most related work for Bandwidth Brokers examines a request as soon as it arrives and accepts it if the reservation does not exceed the unreserved link capacity [8]. Theoretically, this approach has benefits in terms of speed and efficiency, but it can lead to low network utilization. In [7], the authors show how the general admission control problem can be formulated as an optimization problem, with the goal of maximizing the net revenue. The network utilization can improve drastically if we allow the Bandwidth Broker's admission control to gather a number of requests and compute a better allocation of resources. Also [5] deals with price-based admission control, studying both online (when answers to requests have to be issued immediately) and offline (when requests can be gathered and evaluated) versions of the problem are discussed. [2] combines the above approaches with an adaptive scheme that attempts to achieve a preferable balance between optimal utilization of the network and minimal overhead for the Bandwidth Broker operation. A final addition to the admission control algorithm can be the support for resubmitted requests, which means that requests that have been rejected are notified of a later time when they will have better success chances [3]. Moreover, they are given a chance of reducing their reservation requests, in hope that a satisfactory compromise can be found that will cover the user's needs. This can be achieved by keeping a tentative list of the total bandwidth requested at any time, for both admitted and pending requests.

III Algorithms To Be Evaluated

Below is a short summary of the algorithms that were evaluated through our simulated experiments. More details on the operation of each algorithm and its overall architecture can be found at the corresponding references.

- Simple admission control: This is the simple type of admission control, where each incoming request is examined by itself, and is accepted if there is still available bandwidth for the service (that is, the total bandwidth available for the service minus the already reserved bandwidth). Therefore, this algorithm displays identical behaviour each time it is presented with the same set (and with the same temporal succession) of incoming requests. This algorithm has the advantage of simple implantation 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.
- Price-based admission control without adaptation: This type of admission control is similar to the offline version presented in [5]. The algorithm makes a decision on which requests will be accepted trying to optimize the network utilization by gathering and evaluating a group of requests. In order to solve the NP-complete problem that arises, an approximation algorithm is used which can approximate the optimal solution within a specified range. For convenience, this algorithm is labeled from now on as PBAC.
- Adaptive admission control as described in [2]: The algorithm tries to gather multiple requests and evaluate 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 always exactly the same (i.e. the same requests were accepted). Also, the algorithm includes a couple of 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 for the algorithm). In particular, the size R_{size} of the instance of the scheduling problem at a specific time is according to the algorithm:

$$R_{size}(t) = R_{size}(t\text{-}1) + (W_q - R_{size}(t\text{-}1))*a$$

where W_q is the queue with all the pending requests that have to be examined, and R_{size} is the size of the queue of requests that will actually be examined in the next instance of the scheduling problem execution. This can be written as:

$$R_{size}(t) = (1\text{-}a)^{t-1}R_{size}init + (1+(1\text{-}a)\frac{1-(1-a)^{t-2}}{a})W_q*a$$

which demonstrates that R_{size} converges to the size of W_q as quickly as $(1\text{-}a)^t$ converges to near-zero values. 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 of this paper we have generally left these parameters constant. This means that in theory more tweaking of these parameters can probably result to improved performance for this algorithm. For convenience, this algorithm is labeled from now on as AAC.

- Adaptive admission control with resubmissions as described in [3]: The previous algorithm is enhanced with the capability to recognize previously rejected requests and increase their priority. Other than that, this algorithm is very similar to AAC. For convenience, this algorithm is labeled from now on as AACR.

We have selected the above algorithms for evaluation, because their comparison can provide a good insight in the characteristics we are interested in studying. SAC is the simplest algorithm and therefore a good benchmark for the more complicated solutions. PBAC lacks adaptive capabilities, allowing us to identify the effect they have on the simulated environment, and finally AAC and AACR differ only regarding the support of resubmissions, and therefore their comparative evaluation can reveal the effect of resubmission on the overall system performance.

IV Simulation Setup

In order to evaluate the proposed mechanism, we used a simulated system developed for this purpose, which accepted random requests (requests that did not follow a specific pattern in terms of their arrival time or reservation requests) on an Intel-based PC with 512MB of memory running Windows 2000. The simulated system was first used for the evaluations in [2], and has since been expanded in order to implement the various changes and enhancements to the algorithm, and also in order to be able to support evaluations for the rest of the algorithms mentioned in this paper. The simulations examined a high-level view of the network, without taking into account details at the packet level since our main focus is on examining the relative performances of the algorithms and isolating these from impact by external or low-level parameters.

The parameters for each request were randomly produced [13] 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). Listings of the random requests generated, as well as the source code for replicating our results can be found in [15]. The simulated 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.

The main metrics that we are interested in, in order to compare the performance of the algorithms and evaluate the relative advantages and weaknesses of each, are the acceptance rate and the generated profit for the provider. The acceptance rate 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 corresponds to the network provider's revenue. The generated profit for the provider is calculated as the product of the bandwidth consumption of each reservation times its duration. This is a convention since the pricing model can vary depending on the specific circumstances. We believe though that such a metric is one of the most representative ones, since it can be understood as the amount of resources that is consumed by a reservation and the sum for all reservations shows the network utilization that each algorithm achieves.

Maximum available bandwidth for the service were set at 50 Mbps, while the duration of each simulation was set at 30 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 3 time slots.

V Evaluation Results

The first set of experiments assumed a rather short average duration of reservation request (5 time slots), in order to simulate a scenario where there is relatively less competition between requests, and more importantly where the impact of admission decisions spans a shorter time frame. Fig. 1 displays the results of the ratio of requests accepted for each algorithm, while Fig. 2 displays the results of the network utilization (which can be thought of as each provider's profit) achieved by each algorithm.

We then repeated the above experiments using longer average duration for the incoming requests (within the same timeframe of the total experiment duration), with the intention of simulating the situation where requests are more heavily competing for the available resources and where careless selection of the admitted requests can have a significant impact on the network utilization (by tying up resources that could more effectively be used by other requests). Fig. 3 displays the ratio of accepted requests for this set of experiments, while Fig. 4 displays the network utilization.

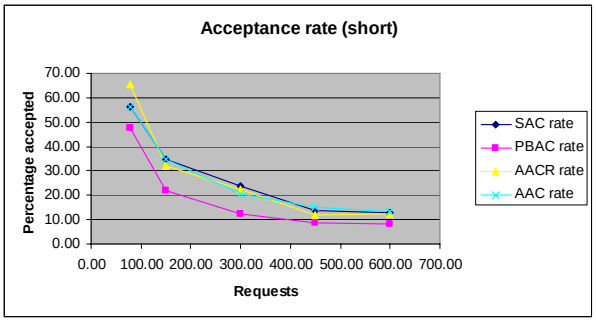

Fig. 1. Comparison of acceptance rate for shorter duration requests

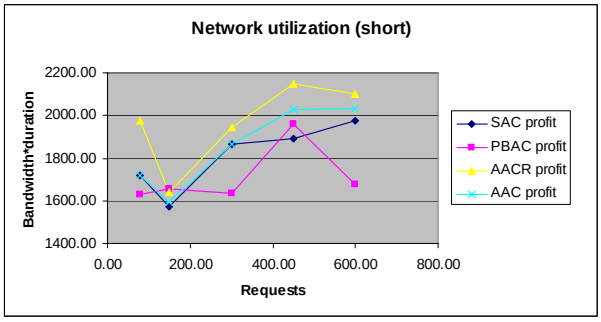

Fig. 2. Comparison of network utilization for shorter duration requests

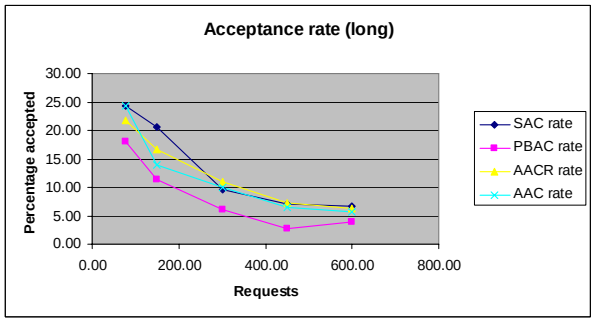

Fig. 3. Comparison of acceptance rate for longer duration requests

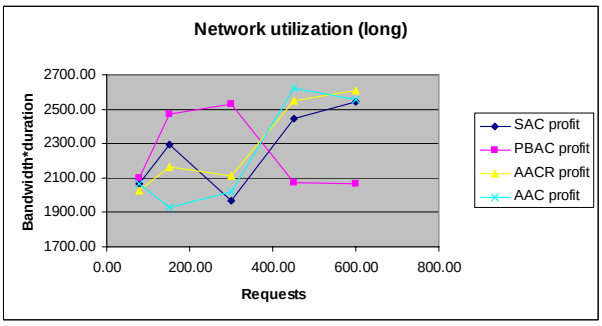

Fig. 4. Comparison of network utilization for longer duration requests

There are a lot of interesting observations that we can make based on the above results, which also assist in better understanding the algorithms and their properties.

The results show that SAC generally displays good and competitive behavior in many cases. This observation, combined with the algorithm's advantage of being the most straightforward and easiest to implement makes it a suitable choice for environments where the load is

TABLE 1 shows that for SAC algorithm the computational overhead was almost zero, since the admission decision is actually a simple comparison to the available bandwidth, and therefore the answer was always delivered within the same time slot that the request was submitted. For the rest of the algorithms there is an increase as the frequency of the request arrival increases, however for both AAC and AACR the increase is relatively mild because of the adaptation module (which avoids an exponential increase in computational overhead), while for PBAC the approximation algorithm also leads to a relatively mild increase in computational time. Overall, it is expected that none of the algorithms will pose a significant burden to a typical modern system hosting the admission module. SAC achieves this because expected not to be very high or the competition between reservation requests very intense. Furthermore, SAC proved more competitive when the requests had a longer average duration. This can be explained when we take into account the fact that longer requests have a more lasting impact and therefore the relative advantage of the other 3 algorithms (PBAC, AAC and AACR) is reduced since a request often spans multiple admission decisions (as is always the case with SAC). Its main weakness is its inferior performance in terms of network utilization, especially for requests of shorter average duration.

The PBAC algorithm displays the most varying behavior. This can be explained because of its approximation nature. It is the algorithm that is most affected from the random generation of the request parameters. Separate evaluations with different random parameters but with the same average characteristics give quite different results for PBAC compared to the rest of the algorithms. Therefore, while the admission control approach of PBAC is interesting as a theoretical study, it is less useful for practical purposes than other algorithms.

The AAC and AACR algorithms generally display the best behavior regarding the utilization of the network, which is their primary objective (Fig. 2 and Fig. 4). Especially for relatively shorter requests, AACR maintains a steadily higher rate of network revenue because of the utilization of the capability for resubmissions. For longer duration requests this effect seems to have less impact, something that is not surprising, since in that case a resubmitted request will probably face a similar situation to the one when it was first rejected (most of its formerly competing requests will probably be still reserving the network resources).

It is also important to evaluate the algorithms with respect to computational overhead. Since the measurements result from simulations at a relatively high level, they mainly focus on measuring a high level complexity of the algorithm. This approach has the benefit that it can to a certain extent isolate the algorithm from the specific low level environment it will ultimately be implemented in, and therefore provide us with more generic results. The drawback is that it does not provide us with details such as network overhead or memory consumption. The metric that we present below is the average time that a request had to wait before an answer by the admission control module arrived, because this is the one that will probably be less affected by low level implementation issues.

of its inherent simplicity, PBAC because of its approximation approach to the admission decision while AAC and AACR because of their adaptive capabilities. It has to be noted here that different selections of the parameters determining the behavior of these two algorithms directly affect the metric displayed here. It is expected that they will be set to a suitable level for the specific environment they are implemented in, which can be done following the guidelines and relevant discussion in [2].

TABLE 1
COMPUTATIONAL OVERHEAD FOR EACH ALGORITHM

Algorithm	Average time to answer request (time slots)			
	Low request frequency	Medium request frequency	High request frequency	Higher request frequency
SAC	0	0	0	0
PBAC	2.89	5.00	7.32	8.04
AAC	2.36	3.18	5.06	6.87
AACR	2.23	2.95	4.03	5.17

VI CONCLUSIONS - FUTURE WORK

In general, all algorithms taking advantage of book-ahead requests fared better than the simple admission control algorithm in situations where the network load approached congestion of allocated resources. However, for a number of situations where the competing requests do not heavily collide with each other, the simple admission control algorithm actually displays good performance and obviously has the advantage of simpler implementation and less overhead. However, if the request frequency rate increases beyond a certain limit that each time depends on the specific parameters of the situation, it is expected that the additional complexity of such algorithms is outweighed by the benefits in terms of provider's profits and users' perceived acceptance rate. Our algorithm bridges this gap and through its adaptive capabilities, it intends to be a solution that suits most situations in a reasonable way.

Our plans for future work include the extension of the comparative evaluation using a network based simulator such as NS-2 [14], which will be able to monitor each algorithm's operation at the packet level and allow us to offer a more complete assessment of the performance of each algorithm within the general Bandwidth Broker framework. Furthermore, we plan to examine and comparatively evaluate the advantages of distributed designs, as well as their impact and overhead for the network.

REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "RFC 2475: An Architecture for Differentiated Services", December 1998

[2] C. Bouras, K. Stamos, "An Adaptive Admission Control Algorithm for Bandwidth Brokers", 3rd IEEE International Symposium on Network Computing and Applications (NCA04), Cambridge, MA, USA, August 30 - September 1 2004, pp. 243-250

[3] C. Bouras, K. Stamos, "Resubmissions and Partly Defined Requests in an Adaptive Admission Control Algorithm for Bandwidth Brokers", 5th International Conference on Networking ICN06, Mauritius, 23 - 27 April 2006, in press

[4] L. Burchard, H. Heiss, "Performance Evaluation of Data Structures for Admission Control in Bandwidth Brokers", April 2002

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

[6] N. Duffield, P. Goyal, A. Greenberg, "A flexible model for resource management in virtual private networks", ACM SIGCOMM 1999"

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

[8] S. Machiraju, M. Seshadri, I. Stoica, "A Scalable and Robust Solution for Bandwidth Allocation", 2002

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

[10] 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

[11] L. Wolf, R. Steinmetz, "Concepts for Resource Reservation in Advance", The Journal of Multimedia Tools and Applications, May 1997

[12] Z. Zhang, Z. Duan, Y. Hou, L. Gao, "Decoupling QoS Control from Core Routers: A Novel Bandwidth Broker Architecture for Scalable Support of Guaranteed Services", SIGCOMM 2000

[13] www.random.org (Accessed February 2006)

[14] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/ (Accessed October 2005)

[15] http://ru6.cti.gr/diffserv-ns/default.htm (Accessed February 2006)