INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2008; **18**: 27–46
Published online 26 February 2007 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/nem.650

An efficient architecture for Bandwidth Brokers in DiffServ networks

Ch. Bouras*,† and K. Stamos

Research Academic Computer Technology Institute, Patras, Greece; and Computer Engineering and Informatics Department, University of Patras, GR-26500 Patras, Greece

SUMMARY

In this article we examine the architecture of an entity used for automatic management and provisioning of resources for DiffServ networks. We examine the existing literature and implementations in this area, focusing on the design choices made, and we propose an architecture for the design of Bandwidth Brokers that combines an adaptive admission control algorithm for increased utilization of network resources and a mechanism for reducing the complexity overhead that intends to be both simple and effective. Specifically, we present a novel architecture for the admission control module that aims at achieving a satisfactory balance between maximizing the resource utilization for the network provider and minimizing the overhead of the module. We complement our theoretical discussion with extensive experimental simulations for the proposed Bandwidth Broker components and analysis of the results. The simulations study the possible configurations of the proposed algorithm and also compare it with alternative admission control policies. Copyright © 2007 John Wiley & Sons, Ltd.

1. INTRODUCTION

The Differentiated Services (DiffServ) framework is one of the basic architectures that have been proposed for QoS provisioning in the Internet and is widely supported by network equipment vendors. Because the Internet consists of numerous network domains with each one acting as an autonomous system, just using the current DiffServ framework does not solve the problem of providing end-to-end QoS, since each domain may be incompatibly configured. One entity that has been proposed in order to overcome this problem and provide end-to-end QoS across network domains is the Bandwidth Broker.

A Bandwidth Broker [1] entity is responsible for providing QoS within a network domain. It manages the resources within the specific domain by controlling the network load and by accepting or rejecting bandwidth requests. A user within the domain that is willing to use an amount of the network resources between two nodes has to send a request to the Bandwidth Broker. The Bandwidth Broker chooses either to accept or reject a request based on the network load, its admission control policy and the Service-Level Agreement (SLA). The SLA [2] is the service contract between the service provider and every customer that describes the service that both the costumer and the service provider have agreed upon. The decision to accept or reject a request is made by the admission control module. In the case that the requested resource is managed by multiple domains, the Bandwidth Broker is also responsible for the inter-domain communication with Bandwidth Brokers of adjacent domains. This procedure requires communication between adjacent Bandwidth Brokers and also a special agreement between the domains.

*Correspondence to: Ch. Bouras, Research Academic Computer Technology Institute, N. Kazantzaki, University of Patras Campus, GR-26500 Rion, Patras, Greece.
†E-mail: bouras@cti.gr

Received 20 June 2006
Revised 8 November 2006
Accepted 2 January 2007

In this paper we primarily focus on the admission control part of the Bandwidth Broker's operations. We propose an adaptive algorithm for resource reservation requests that arrive ahead of the actual time for which they request the resource reservation. This allows our algorithm to gather a set of multiple requests and examine them in order to make better utilization of the network, making use of the existing literature on scheduling problems. The algorithm then attempts to balance the calculation of the optimal solution with a limitation on the computational overhead that the algorithm itself incurs at the Bandwidth Broker operation. The importance of monitoring and adapting the computational overhead for the Bandwidth Broker can be highlighted when there is a high arrival rate of requests, while the requested bandwidth for each reservation is small. Such an example is the case of multiple VoIP requests at a high bandwidth link. We have designed the architecture so that it can be configured according to the specific parameters of each deployment (processing power of the Bandwidth Broker module, acceptable delays for notification from the Bandwidth Broker), so that the proposed solution is suitable for a variety of real-world cases. There are emerging types of networks, such as wireless networks, where the demand for quality of service is strong, and where sophisticated solutions are applicable because of the nature of the wireless environment: smaller-capacity links compared to wired networks, frequent and more unpredictable entry of new users for the existing resources. Other research teams have already taken over the work presented in this paper in order to apply it to wireless environments.

For network providers, utilization of the network resources and maximization of revenue is one of the most important aspects of the Bandwidth Broker's operation. Our approach focuses on the maximization of the used bandwidth, on the assumption that the network provider's revenue will depend on the product of the bandwidth requested by a reservation, times its duration, which corresponds to the network utilization and more closely represents the actual cost that the serving of a request incurs to the network. Other models can also be used, such as the maximization of the number of admitted requests, in case there is a flat pricing model. Our performance evaluations examine this aspect of the algorithm's performance as well.

The rest of the paper is organized as follows: Section 2 discusses related work in the areas of architectural design, the admission control module of a Bandwidth Broker and the possible distribution of its functionalities. In this section we examine and compare the characteristics of some of the most important already existing architectures. Section 3 describes our proposed algorithm for a centralized admission control module, which is evaluated through simulation in Sections 5.1 and 5.2. Section 4 discusses the operation of the architecture at an inter-domain level, while Section 5 contains the simulations that were conducted in order to evaluate our proposals. Section 6 describes our final conclusions and the future work that we intend to do in this area.

2. RELATED WORK

2.1. Admission control module

Admission control means that the Bandwidth Broker has to define whether the incoming resource reservation requests will be accepted or not. Once a 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. However, since the operating circumstances can vary widely from one case to another, it is improbable that a single solution will fit all operating requirements. Our approach tries to tackle this problem by incorporating an adaptive mechanism with the intent to converge to a suitable level for each deployment scenario.

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

DOI: 10.1002/nem

- *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. Wolf and Steinmetz [3] give a thorough presentation of the concept of book-ahead reservations, while a detailed discussion on the benefits and potential problems with book-ahead requests can be found in Gupta [4]. In general, book-ahead requests provide a richer functionality to the service and allow for better solutions to the admission control problem. There are a lot of actual cases in the real world where a book-ahead request meets the requirements of an application: for example, pre-arranged video conferences.

Researchers have dealt with both types of admission requests. An approach that deals with both types of incoming requests is the resource partitioning proposed in Ferrari *et al.* [5], which separates the admission decision for immediate and book-ahead requests. Immediate requests that were rejected can be reconsidered for a book-ahead reservation. In order to avoid wasting resources because of fragmentation, the authors propose a moving boundary between the two partitions. The most common mechanism for admitting book-ahead requests is to divide time in intervals (slots) of equal size, and calculate the resources requested by a new reservation for the time slots that it overlaps [5,6].

For both immediate and book-ahead requests, it may be possible either to 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. Hwang *et al.* [7] present a Bandwidth Management Point (BMP), which uses centralized network state maintenance and pipe-based intra-domain resource management schemes in order to reduce the admission control time and the scalability problems. Another approach is used by Wong and Law [8], where SLA requests are converted by the border routers in messages of the COPS protocol [9] and refined before they are sent to the Bandwidth Broker, which makes simple decisions based on the refined requests.

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). Our algorithm takes advantage of this flexibility, in order to calculate the most efficient admission decisions that maximize the network utilization and subsequently the network provider's profit [10]. The provider may choose to allow requests that demand an immediate answer, balancing perhaps this capability with additional cost.

Admission control can be done either on a hop-by-hop basis [11,12] or on a per-flow basis [13]. 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 running the admission control algorithm for each link in the calculated path.

A number of data structures have been proposed for efficiently implementing an admission control algorithm. The most common are the simple implementation using an array, and various variations using trees like segment trees and binary search trees [11].

In the more general case, time is considered continuous and therefore reservations can start and end at any time. It is also very common, however, to use slotted time, so that reservations are considered using a predefined granularity. Algorithms may either benefit from this granularity or completely depend on it for their operation [11].

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 [14]. This approach has benefits in terms of speed and efficiency, but it can lead to low network utilization. Greenberg *et al.* [15] 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. This is

DOI: 10.1002/nem

the approach we have taken in this paper. Also Chhabra *et al.* [16] deal with price-based admission control, in which 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. Our work 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. An adaptive scheme designed for the scheduling of popular video on demand that also uses delayed notification is presented in Bouras *et al.* [17]. In Mykoniati *et al.* [18], the admission control approach taken by the TEQUILA project is presented, which is based on a feedback model that can be tuned by operational policies and strategies. Its main characteristic is that it is based on the ability of the network to provide QoS using functions that dimension the network on the basis of anticipated demand, and on the actual status of the network using measurements.

2.2 Centralized and distributed architectures: comparison description and discussion

In this section we present proposed bandwidth architectures from the relevant literature on the design of Bandwidth Brokers [19–22]. Our intention is to outline the advantages and disadvantages of the different strategies for the Bandwidth Broker design. Many researchers have worked on this area, producing various solutions, each of which is potentially suitable for several situations and weak on others. A survey on existing Bandwidth Broker implementations can be found in Sohail and Jha [23].

Some of the architectures presented in the comparison table (Table 1) have been implemented, like the University of Kansas and the UCLA architectures, while others are theoretical or at the design phase. The table offers a view of the wide variation between the characteristics of the various proposed and

	University of Kansas	MIPTel	QBone	UCLA	CTI NS-2 implementation
Architecture	Centralized	Centralized	Centralized, can be extended for distributed	Distributed	Distributed
Admission control	Maintains and consults policy database	Pre-configured bandwidth threshold	Traffic demand matrix	Accepted if SLA is not violated	Restricted by available bandwidth
Routing table	No	Yes	Yes	Yes	No
Security	Not defined	Not defined	Not defined	Not defined	Not defined
Robustness/ failure recovery	Not defined	Not defined	Recovery actions from most common failures	Recovery actions from most common failures	Not defined
Inter-domain interface	TCP sockets for Linux routers/ Telnet automated script for Cisco routers	Custom protocol	Custom protocol	Custom protocol	TCP
Intra-domain interface	TCP sockets for Linux routers/ automated script for Cisco routers	Custom protocol	COPS/SNMP/ Telnet	COPS	TCP
User interface	Web-based GUI	Message exchange	GUI, Host/User, Server/Gateway Interface	Web-based GUI (PHP)	—

Table 1. Comparison table of various Bandwidth Broker architectures

DOI: 10.1002/nem

implemented Bandwidth Broker architectures that one can find in the current literature. Most of the existing literature proposes Bandwidth Brokers that consist of a central module that deals with functionalities such as admission control, inter-domain communication, maintaining a routing table interface, connecting to the network routers and sending the proper configuration parameters.

In order to overcome the scalability issues that are associated with a centralized Bandwidth Broker model and to avoid having the Bandwidth Broker as the bottleneck while the network itself is underutilized, Zhang *et al.* [13] propose a distributed Bandwidth Broker architecture. Their design consists of one central Bandwidth Broker (cBB) and a number of edge Bandwidth Brokers (eBBs) in the domain. There are two levels that represent the QoS states: link level and path level. The idea is to maintain databases with information regarding both the reservations on the link and on the path level. The path-level database information is extracted from the link QoS state database. While the link state database is only maintained by the cBB, however, each of the eBBs maintains a mutually exclusive subset of the path state database, and can therefore handle the admission control load for the relevant paths. The authors then propose a number of variations on how the admission requests can be handled, depending on whether they will be accepted or not. Also Cortese *et al.* [24] implements a hierarchical scheme of Bandwidth Brokers by dividing the domain into small areas, each with its own Bandwidth Broker, and a parent Bandwidth Broker at the domain level.

Comparing the distributed architectures with the centralized model, we note that while the distributed architectures offer scalability advantages over the centralized model, they offer inferior resource management and introduce bandwidth wastage along the paths. Furthermore, if the link database needs to be frequently accessed, the processing overhead can increase and become counter-productive.

Another important comparison point is the robustness of each solution and its behavior in unexpected or undesirable circumstances. Failures at a network component can be categorized as follows:

- The component does not operate at all.
- The component operates erroneously and sends unexpected and unknown messages to its communicating nodes.
- The component is overrun by a malicious entity (e.g., a virus) and appears to be sending valid messages but it does not obey the proper behavior and tries to downgrade, corrupt or completely halt the proper operation of the architecture.

The last two categories are also called Byzantine behaviors and can be summarized as states where the component operates in an arbitrary fashion, not according to the algorithm it was designed to follow.

We also mention another trade-off between the centralized and the distributed classes of architectures: a distributed architecture can be designed in such a way that greater robustness and tolerance for some failed entities of the Bandwidth Broker architecture can be achieved, while a centralized architecture has a single point of failure. On the other hand, it is much easier to secure and closely protect a single Bandwidth Broker entity than it is to safeguard and scrutinize the behavior of a multitude of components that comprise a distributed Bandwidth Broker architecture.

Furthermore, a distributed architecture also has to take into account the issue of consistency between the components that comprise the Bandwidth Broker. This is also true in some cases of a single Bandwidth Broker entity, where in order to increase robustness additional backup components are introduced (like a duplicate Bandwidth Broker).

It also has to be taken into account that the relative pros and cons of the architectures are affected by the deployment environment. Since the first category of failures (complete shutdown of the failed component) is usually much more easily discovered and more desirable than the rest of the failure categories, if the Bandwidth Broker architecture is implemented and deployed in an environment where such failures can be ruled out, the relevant types of consideration can naturally be discarded. Our proposal presented in the sections below is based on a centralized module, while an attempt to examine possible benefits from a distributed architecture can be found in Bouras and Stamos [25].

DOI: 10.1002/nem

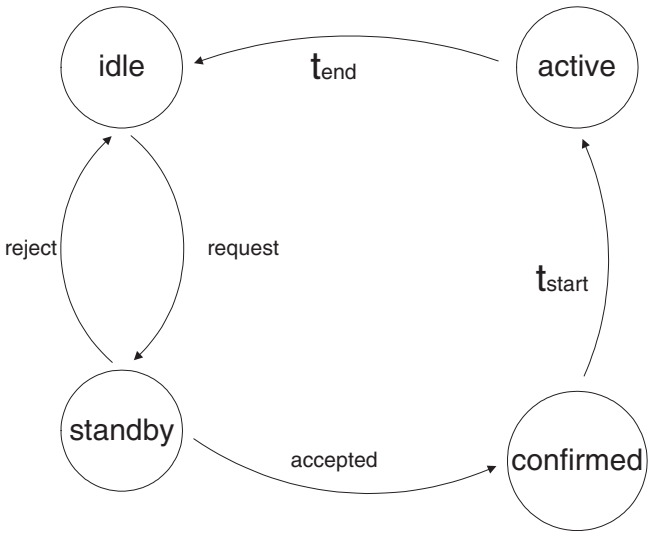

Figure 1. Request states

3. DESCRIPTION OF ADMISSION CONTROL ALGORITHM

We define standby requests as requests that have not yet received an answer (either confirmation or rejection). Confirmed is a book-ahead request that has received an affirmative answer but waits to be activated. These states are shown in Figure 1.

For a DiffServ domain, it is not efficient to keep per-flow status in the core routing devices, and therefore an aggregated approach is used at the core routers. Admission control is performed by the Bandwidth Broker at the domain scale, using the hose model [26] that has been proposed for VPN provisioning. 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. We have to note here that this does not mean that all or most of the reservation requests will necessarily be accepted, but that the network design will guarantee that there will be enough resources in order to service the reservation requests that have been accepted. The hose model has the benefit of offering the flexibility to the edge device of sending traffic to a set of endpoints without having to specify the detailed traffic matrix and can also reduce the required size of access links through multiplexing gains because of the aggregation of flows between endpoints. Various methods of implementing the hose service model are given in Duffield *et al.* [26]. Our approach decouples the admission control decision from the routing issues. This means that core routers are not involved in the process of admission control and signaling. Furthermore, while combining routing decisions with admission control can be beneficial, it is not always possible, or desirable for scalability reasons. Packets of a specific flow can be routed using different paths without affecting the admission control process.

An important problem for admission control and dimensioning of network resources is how to take into account the burstiness and self-similar behavior of network traffic. The problem can be simplified by defining a unifying parameter for the characteristics of network resources that accurately captures their effect on resource usage. Various researchers [27–29] have dealt with the concept of effective bandwidth, which can be computed based on traffic parameters such as mean rate or maximum burst size. It is an additive amount of bandwidth that is large enough to guarantee that the QoS requirements of all flows are met.

DOI: 10.1002/nem

We suppose a new request has the form of

$$r(t_{\text{start}}, t_{\text{end}}, b, w) \tag{1}$$

where t_{start} and t_{end} are the starting and finishing times for the reservation, b is the requested bandwidth, and w is the period for which the request can wait until it receives either a confirmation or a rejection of the reservation. Our model is based on the peak rate allocation, so that each request specifies its maximum transmission rate and the admission control module has to make sure that the sum of all peak rate does not exceed the allocated service capacity. Bandwidth is defined at Layer 3, as the maximum amount of bits/second of IP traffic, including the IP header that an endpoint may insert to the network. The Bandwidth Broker's admission control module keeps a list of unanswered requests, which we call waiting queue, W_q, sorted by their starting time t_{start}. As soon as any item in W_q, say r_1 (with the closest w to the current time) is about to expire, the admission control module calculates the answers that it will provide to this and a number of other requests, essentially by solving an offline scheduling problem.

Suppose n is the cardinality of W_q. We define

$$R = \{r_1, r_2, \ldots, r_m\} \subseteq W_q \tag{2}$$

and we want to find a subset $R_c \subseteq R$ such that $\Sigma_{ri \in Rc} b_i \leq B$ at any time point where B is the total available bandwidth for the service (defined by the dimensioning for the DiffServ service according to the hose model) and try to maximize $\Sigma_{ri \in Rc} b_i$ throughout the period from the earliest t_{start} to the latest t_{end} in the R set. R_c is the set of requests that will be accepted by the algorithm, while requests in the set $R–R_c$ will be rejected. This problem is NP-complete [30] and therefore it is proposed to use an approximation algorithm to solve the following linear programming relaxation in polynomial time [16] and then make the solution discrete regarding the variables x_i, which represent whether r_i is accepted or not:

$$\max \ \Sigma_{i \in R} b_i (t_{\text{end}} - t_{\text{start}}) x_i$$
$$\Sigma_{i \in R(t)} b_i x_i \leq B, \text{ for all } t \in (\text{earliest } t_{\text{start}}, \text{ latest } t_{\text{end}}) \text{ in } R$$
$$0 \leq x_i \leq 1, \ i \in R$$

In order to avoid approximation of the solution, this problem can be solved using integer linear programming, which, however, becomes very costly computationally as the problem instance increases (which happens for a high rate of incoming requests). A mechanism monitoring the instance size and carefully adapting it is needed in this case. The important point is how to select the R set. A simple approach would be simply to set $R = W_q$. This solution, however, can become computationally costly, and it can furthermore lead to low network utilization, because requests that have been made very far in advance will probably have little competition, and will therefore be most likely accepted. In the example in Figure 2, this can be demonstrated by request r_6. If r_6 is included in the R set as soon as r_1, it will certainly be accepted, since there it has no other competition. It would be better though to delay the decision for r_6, since by that time other requests (more profitable than r_6) could have arrived.

Including in R only requests that overlap with t_{end} for r_1 may also not be an attractive solution, because it might require the algorithm to be invoked too frequently, and that could introduce an unacceptable overhead. In Figure 2, the algorithm will have to be invoked separately for r_1, r_3, r_5 and r_6. As our algorithm does not provide for overbookings, we have also drawn a line that shows the maximum bandwidth that can be allocated to the requests.

In order to combine the benefits of both these extremes and reduce their shortcomings, our solution is to have an adaptive parameter for the size of R, which will increase if the number of requests in W_q increases or if the algorithm was very time consuming, and decrease otherwise, according to

```
if (C_k-1 - T > T)
  R_size(k) = 1
else if (C_k-1 - T > 0)
  R_size(k) = (1 - (C_k-1 - T)/T) R_size(k - 1)
else
  R_size(k) = R_size(k - 1) + (W_q - R_size(k - 1))*a
```


DOI: 10.1002/nem

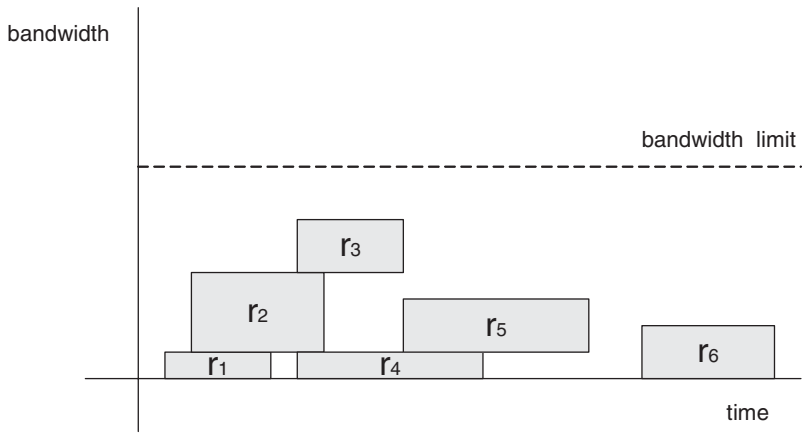

Figure 2. Reservation requests

C_{k-1} is the duration of the previous computation of the requests to be accepted, a is a parameter that determines the increase rate of R_{size} and T is a threshold value of the maximum allowable time for a computation. Configuring parameter a determines how close to W_q we want the size of the R set to become after an increase, so a is essentially the adaptation factor of the algorithm's operation. The above pseudocode guarantees that if the algorithm lasts more than an accepted threshold, it will be simplified to admission control for a single request, which is the simplest computation possible and we can reasonably expect to reduce the computational overhead to very low levels, from which the algorithm can slowly progress to more complex computations according to the adaptation parameter a. In general, an adaptive scheme can have problems of oscillation between extreme values, so we have considered the above scheme that does not sharply increase or decrease the size of the R set, except for the case that for some reason the computation time far exceeds the acceptable threshold.

As an example, if the last computation lasted more than twice the predefined threshold, then the size of the subset R that will be examined for the current computation is reduced to 1, and therefore the computation is simplified to examine whether accepting the next request violates the resource restrictions or not. The assumption is that in that case the computation of an optimal solution is adding a large overhead to the Bandwidth Broker's operation, and we therefore simplify the computation as much as possible. In the case that the computation time exceeds the threshold but not twice its value, we assume that the overhead is significant and must be reduced (but is not unacceptable as in the previous case). We reduce it by a factor of $1 - (C_{k-1} - T)/T$, so that the reduction becomes more aggressive as C_{k-1} becomes larger. Finally, if the computation time is still below the threshold, we assume that there is space for increasing the computation overhead by increasing the size of the subset R, and this is done using a factor $a \in (0, 1)$. The closest to 1 this factor is chosen, the more aggressive the increase is, with the obvious limit of the size of the whole W_q.

In general, finding an optimal solution to the problem of optimally scheduling the requests is NP-complete, since the Knapsack problem, which is known to be NP-complete [30], is equivalent to a simpler version of our scheduling problem where all t_{start} and t_{end} times are equal. However, because it is not critical to have the optimal solution, we can either use an approximation scheme that runs in polynomial time and approximates the optimal solution with the desired accuracy, or try to limit the instance size of the problem in order to be within the computational capabilities of the underlying equipment.

Following is a summary of the algorithm for calculating the accepted requests at an ingress point of the domain:

DOI: 10.1002/nem

```
while (Wq not empty)
  while (next request has not expired)
    forall i where (bi > B)
      xi = 0 // reject overbooking requests
    Solve LP maximization:
    max Σi∈R bi(tend - tstart)xi
    Σi∈R(t) bixi ≤ B, forall t ∈ (earliest tstart, latest tend) in R
    xi ∈ {0,1}, i ∈ R
    exit loop
  end while
  if ((C - T) >= T)
    Rsize = 1
  else if ((C - T) > 0)
    Rsize = (1 - (C - T)/T)* Rsize
  else
    Rsize = Rsize + (Wq - Rsize)*a
end while
```

If the current computation takes too long and w_1 is about to expire, the computation is ignored and a simpler fall-back mechanism is used which will individually examine r_1 and then restart the computation for $R - \{r_1\}$.

Because of the way the algorithm is constructed, it is not generally optimal on network utilization. As we have mentioned, we make this trade-off in order to reduce the computation overhead for the Bandwidth Broker module. A very fast processing module (or, conversely, a low rate of admission requests) leads the algorithm to quickly converge to a good approximation of the optimal solution.

Thus, assuming that the computation time does not reach or exceed the threshold, we have that

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

Solving the recursive function gives

$$R_{\text{size}}(t) = (1-a)^{t-1} R_{\text{size init}} + \left(1 + (1-a) + \ldots + (1-a)^{t-2}\right) W_q \times a$$

and therefore

$$R_{\text{size}}(t) = (1-a)^{t-1} R_{\text{size init}} + \left(1 + (1-a)\frac{1-(1-a)^{t-2}}{a}\right) W_q \times a \tag{3}$$

where $R_{\text{size init}}$ is the initial size of R.

Therefore, R_{size} converges to the size of W_q as quickly as $(1 - a)^t$ converges to near-zero values, which happens quite rapidly, especially if a has been chosen close to 1, which means a very high adaptation capability.

The algorithm is also not fair with respect to maintaining a first-come first-served order (since an earlier request might be rejected in favor of a later request that will better utilize the network), but it guarantees efficient operation with respect to the response to requests (it assures that all requests will be answered on time, either positively or negatively) and it respects each request's maximum waiting time w. Compared to a simpler admission control algorithm that examines the requests as they arrive, our proposal also is more cautious regarding 'big' requests that consume a lot of bandwidth for a long duration. The reason is that whereas a simple admission algorithm would accept a big request and then 'starve' the rest of the requests until the 'big' request was satisfied, our proposal examines requests over a longer time period and is therefore capable of promoting a large number of 'small' requests (which

DOI: 10.1002/nem

make better utilization of the network because they are more capable of filling availability 'holes') than a single 'big' request that blocks most of the subsequent requests. This theoretical assumption is also experimentally examined in the evaluation section of our work.

4. INTER-DOMAIN RESERVATIONS

Dealing with inter-domain reservations is a very important but also difficult and relatively untouched issue in practical environments. Some efforts (Geant2 JRA3 [31], DRAGON [32]) have focused on the automated inter-domain provisioning of Layer 2 services, while Geant2 SA3 [33] have been working on the automated multi-domain provisioning of Differentiated Services. Other proposals include the one from the TISPAN standardization body of the European Telecommunications Standards Institute (ETSI) [34], which has produced the Resource and Admission Control Subsystem (RACS) specification identified in the overall TISPAN Next Generation Networking (NGN) architecture. RACS [35] is the TISPAN NGN subsystem, responsible for elements of policing control including resource reservation and admission control in the access and aggregation networks. In the specific area of providing end-to-end QoS requirements for a PSTN-grade voice and multimedia service, the MultiService Forum (MSF) has proposed a QoS solution [36] and considered how it might best be supported over a packet network infrastructure. This solution has a number of components, among which is the Bandwidth Manager [37], and has chosen to adopt the DiffServ PIB (Policy Information Base) for the interface between the Bandwidth Manager and the edge routers of a domain. It covers hierarchical organization of Bandwidth Manager components, and reservations between managers in adjacent domains. Additionally, an end-to-end QoS solution has also been defined by 3GPP [38], and it incorporates a Policy Decision Function that approximates part of the Bandwidth Broker functionality discussed here. The 3GPP proposal is mainly specialized on 3G mobile networks and defines its own PIB for interfacing to the Policy Decision Function.

There are a number of issues that have to be resolved for efficient inter-domain reservations to work, which include compatibility in the technology and architectures, SLA negotiation, pricing, routing, restoration and authentication issues. One of the basic aims of the Bandwidth Broker concept was to give a uniform solution to some of these issues. Our proposed admission control algorithm can be adapted in several ways in a multi-domain environment.

A basic criterion is how the routing function for the end-to-end reservation will take place. End-to-end routing that requires full knowledge of intra-domain topology is an efficient but practically not feasible solution. Even if domain managers were willing to share full internal topology, updating and using such information becomes a very complex procedure. A first level of inter-domain routing (where domains are considered black box nodes and only inter-domain links are taken into account) is necessary. This level of 'domain' routing can be performed in two ways:

- The source domain determines all domains that will have to be traversed until the destination domain is reached.
- The source domain only determines the next adjacent domain, and each domain in turn determines the next domain until the destination domain is reached.

Also admission control can take place in two ways:

- All domains that are involved in the reservation simultaneously process the reservation.
- Every domain waits for a positive answer from the preceding domain before it starts its own admission process.

The first option suits better the characteristics of the proposed admission control algorithm, because by simultaneously examining the reservation requests each domain has the capability to gather multiple requests and better utilize its resources without delaying the overall process. However, our solution does not exclude any option for the admission control at the inter-domain level, which is an issue we intend to further investigate in the future.

DOI: 10.1002/nem

Regarding inter-domain security, the effort in this area has concentrated on securing protocols such as the Simple Inter-domain Bandwidth Broker Signaling (SIBBS [39,40]) protocol, which deals with the Bandwidth Broker communication across domains. The SIBBS protocol is proposed by the Internet2 community in order to implement the inter-domain communications of resource reservation between the Bandwidth Brokers. It exchanges two pairs of messages for QoS configuration purposes: the Resource Allocation Request (RAR)/Resource Allocation Answer (RAA) messages to request for a service, and the CANCEL/ACK messages to terminate the requested service. The transmitted information is sensitive and therefore has to be protected against possible security compromises. Lee *et al*. [41] outline the main security threats that inter-domain Bandwidth Broker communication has to protect against, and explain how the Public Key Infrastructure (PKI) can be integrated in order to produce a secure SIBBS implementation. Their implementation offers authentication, integrity, timeliness and non-repudiation services based on the PKI technologies. Communication between Bandwidth Brokers is encrypted in order to protect against the types of attacks described in the previous chapter.

5. PERFORMANCE EVALUATIONS

5.1 Evaluation set-up and admission control module configuration

Before assessing the performance of the algorithm compared with alternative admission control techniques, we decided to examine the adaptive properties of the algorithm and its behavior according to the tuning of the configurable parameters. For this phase of experimentation we used a simulated system that accepted random requests (requests that did not follow a specific pattern in terms of their arrival time or reservation requests), in order to study the performance of the algorithm and the effect of the computation time threshold and adaptation parameter a on the behavior of the algorithm. Our simulated system was running on an Intel-based PC with 512 MB of memory running Windows 2000. The simulations examined a high-level view of the network, without taking into account details at the packet level since our main focus was 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 [42] within suitable boundaries (regarding the total duration of each simulation, the total available bandwidth, and the minimum and maximum reservation requests) for each situation that we wanted to simulate. Maximum available bandwidth for the service were set at 50 Mb/s, while the duration of each simulation was set at 30 time slots. The source code and configurations for our simulations and experiments can be found elsewhere [43].

Figures 3–5 display the adaptive operation of the algorithm. In Figure 3, where a smaller threshold has been defined, we can see that in no case are more than four requests examined simultaneously, and therefore the computation time for approximating the integer linear programming solution is bounded on this threshold of the size of the input. Similarly, in Figure 4, where the threshold has increased, the algorithm can gather more requests and therefore achieve better network utilization. Figure 5 displays the examined requests for an even largest threshold, that still does not permit examining more than eight simultaneous requests. In all cases the algorithm probes for larger sizes of the R set, until the point where the computation time threshold is reached. At that point the algorithm retreats, as can be seen in Figures 3–5.

In the presented simulations the value of the adaptation parameter a was identical at 0.1, which produces a cautious behavior of the algorithm. The size of the examined set R is adapted gradually. Larger values of a produce sharper increases and reductions of the R set. This can be observed in Figures 6–8, which display the adaptive behavior of the algorithm when the a value is doubled at 0.2. The algorithm tends to examine a larger number of requests (using a larger R set), but the threshold is also exceeded more often and the algorithm has to adapt to smaller sets of examined reservation requests. This behavior is magnified depending on the adaptation parameter. As Figure 9 illustrates, our simulations showed that larger values for the adaptation parameter are not suitable for the proposed algorithm. We therefore determined that in order to avoid an oscillatory behavior it is better to keep the adaptive parameter a around the 0.1 range of values and not larger than 0.5.

DOI: 10.1002/nem

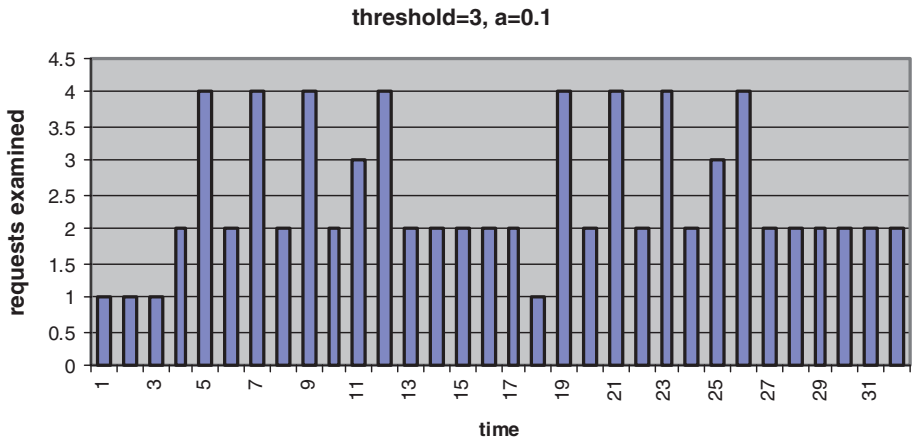

Figure 3. Examined requests for smaller threshold

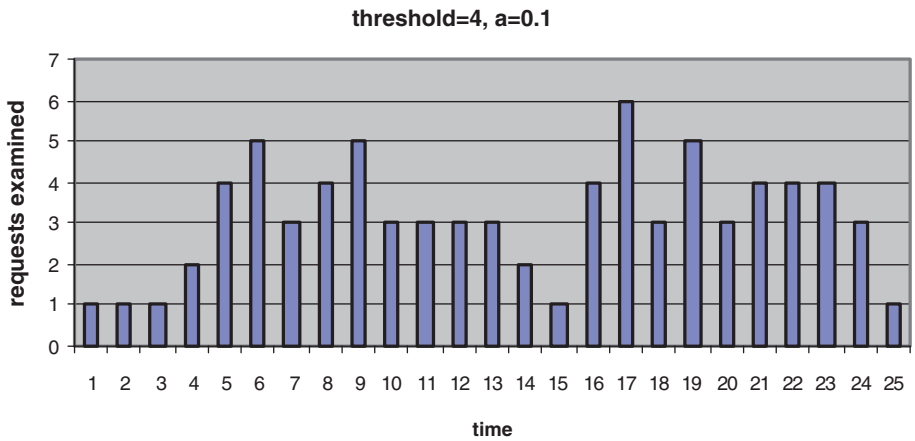

Figure 4. Examined requests for medium threshold

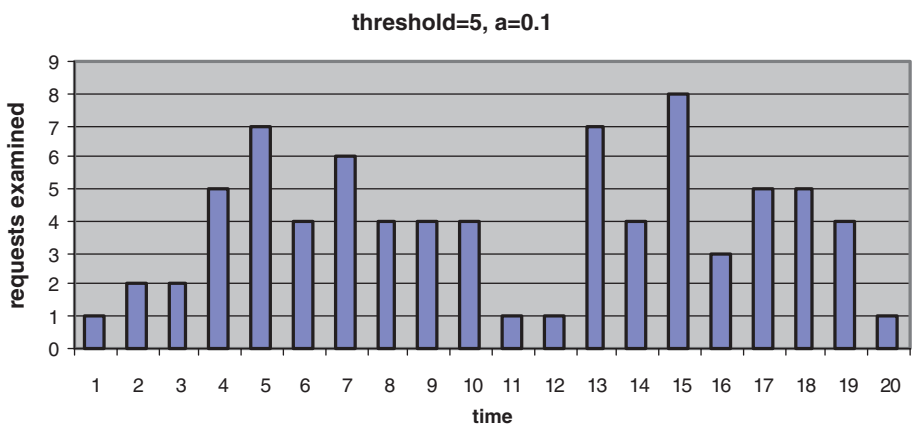

Figure 5. Examined requests for larger threshold

DOI: 10.1002/nem

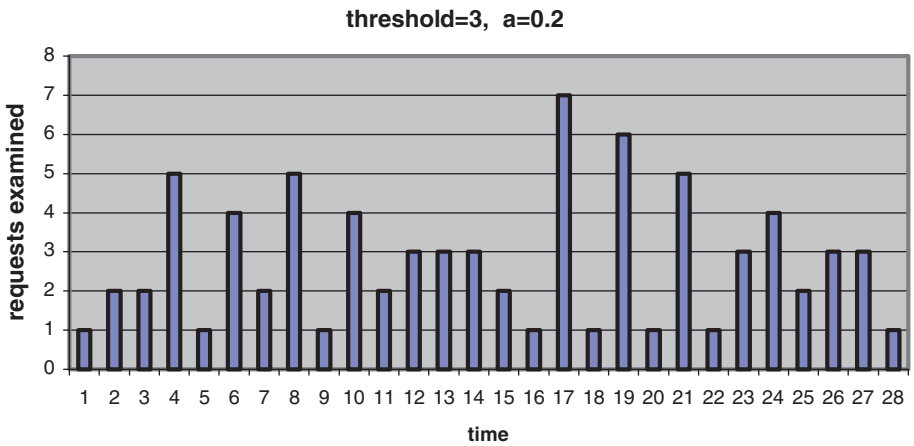

Figure 6. Examined requests with smaller threshold and aggressive adaptation

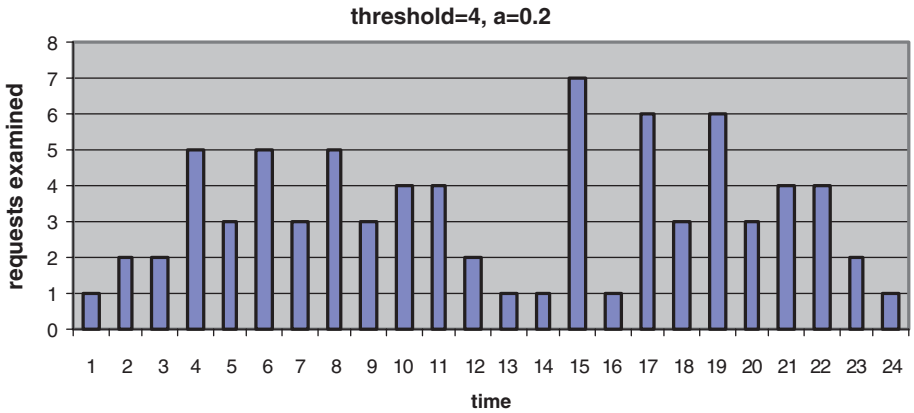

Figure 7. Examined requests with medium threshold and aggressive adaptation

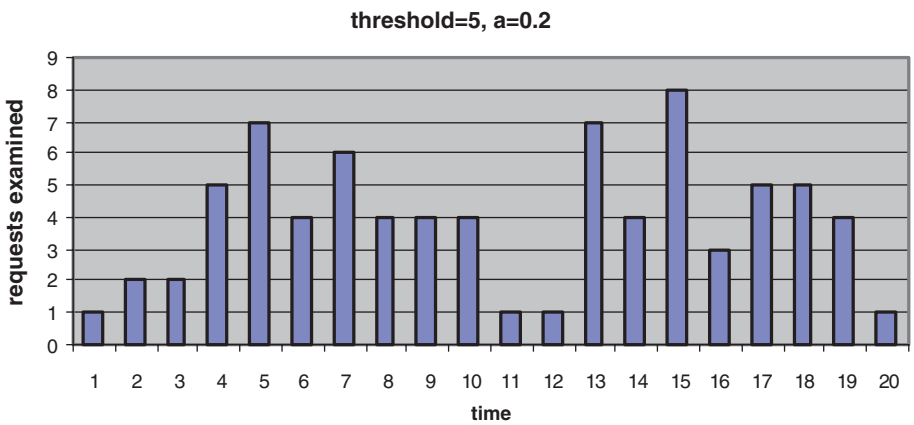

Figure 8. Examined requests with larger threshold and aggressive adaptation

DOI: 10.1002/nem

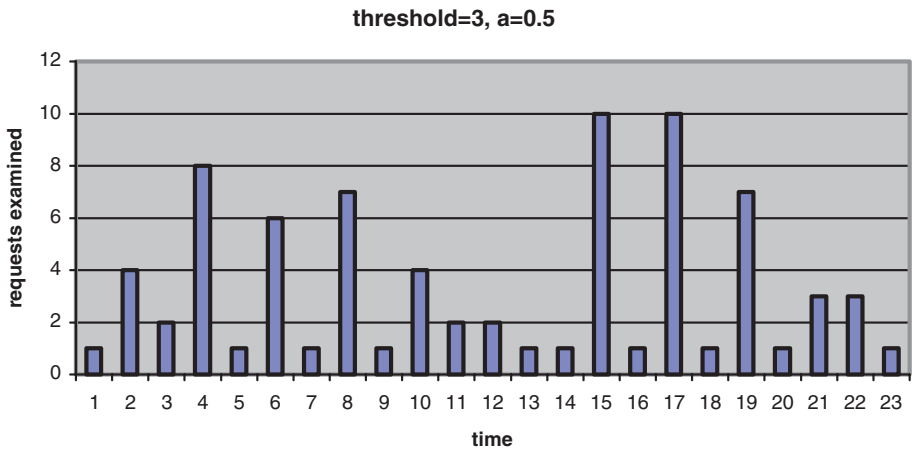

Figure 9. Examined requests with very aggressive adaptation

5.2 Comparative evaluations

Our next set of experiments was designed in order to evaluate the admission control algorithm's performance in terms of the achieved acceptance rate and network utilization. For this reason, we implemented our proposal in the popular ns-2 network simulator [44], 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 700 MHz. 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 [45]. The MRG32k3a generator provides 1.8×10^{19} independent streams of random numbers, each of which consists of 2.3×10^{15} substreams. Each substream has a period (i.e., the number of random numbers before overlap) of 7.6×10^{22}. The period of the entire generator is 3.1×10^{57}. More specifically, the random generator was independently generating numbers that were then assigned to each of the attributes for a new request. If the random combination of attributes was invalid (for example, the start time of the reservation was after the stop time of the reservation) the request was discarded as if it had never been generated. Otherwise, it was generated by the node and sent to the Bandwidth Broker for examination. In order to see how each algorithm could scale, we experimented with generating up to 1000 requests in a 50 s interval. Listings of the random requests generated, as well as the source code for replicating our results in ns-2, can be found elsewhere [43]. The topology defined at the simulator was a star network, with the Bandwidth Broker module being located in the center and requests originating from one leaf node towards another leaf node of the network. The links of the simulation represented the provisioning of hoses as defined in Duffield *et al.* [26]. Maximum available bandwidth for the service was set at 100 Mb/s, while the duration of each simulation was set at 50 time slots. For the adaptive algorithm the results were obtained setting the adaptation parameter a at a value of 0.2 (moderate adaptation) and a computational threshold of both 5 and 10 time slots. The test case examined in our experiments represents the requirements of a network providing QoS to a large number of discrete end users. These end users might require end-to-end QoS guarantees for specific applications, such as VoIP or videoconferencing, with varying start/end times and resources. Such a network has typically performed dimensioning at its core, which is modeled by the VPN model referred in this

DOI: 10.1002/nem

paper. The deployment of an automated resource brokering mechanism is also necessary in this case. The inter-domain negotiations and signaling are more thoroughly covered in the previous sections and related work, while our experiments focus on the admission control procedure.

Our main goal was to study the operation of the adaptive admission control algorithm (from now on called AAC) at a more realistic setting and compare their performance with other alternatives. In particular, the alternatives we studied are the Simple Admission Control (from now on called SAC), which examines each request on its own and accepts it if there are enough resources to satisfy it, and the Price-Based Admission Control algorithm (from now on called PBAC), which 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. This type of admission control is similar to the offline version of the algorithm presented in Chhabra *et al.* [16]. We have selected the above algorithms for evaluation, because their comparison can provide a good insight into 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. 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, which shows the percentage of requests accepted out of the total number of submitted requests. In the 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, which is calculated as the product of the bandwidth consumption of each reservation times its duration. This is, of course, 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. 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. In an environment, 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.

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. The values reported are averages from multiple executions of the simulations, which generally produced identical or very similar results. These results are summarized in Table 2, where average values are presented for each algorithm. Repeating the experiments with the same distribution of requests per time slot produces very similar results (within 5%) regarding the average values, which leads us to believe that the results are representative of the behavior of the algorithms in the ns-2 simulation environment.

Averages per algorithm	Acceptance rate	Average delay (time slots)	Average network utilization (bytes × time slots)
SAC	29.60%	0	3 920 014
PBAC	21.79%	7.08	5 243 307
AAC thr = 5	25.72%	5.44	4 532 672
AAC thr = 10	24.77%	5.48	4 780 385

Table 2. Summary of results

DOI: 10.1002/nem

The following figures display in more detail the behavior of the algorithms under different experiments with different request frequencies, and they help reveal the features of each algorithm, its relative weaknesses and strengths.

As Figure 10 demonstrates, the acceptance ratio of all algorithms 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. 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. 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 in Figures 11 and 12, 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. This metric essentially describes how well the network links are filled with actual traffic, assuming that all simulated users take advantage of the allocated network resources.

The results in Figure 12 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

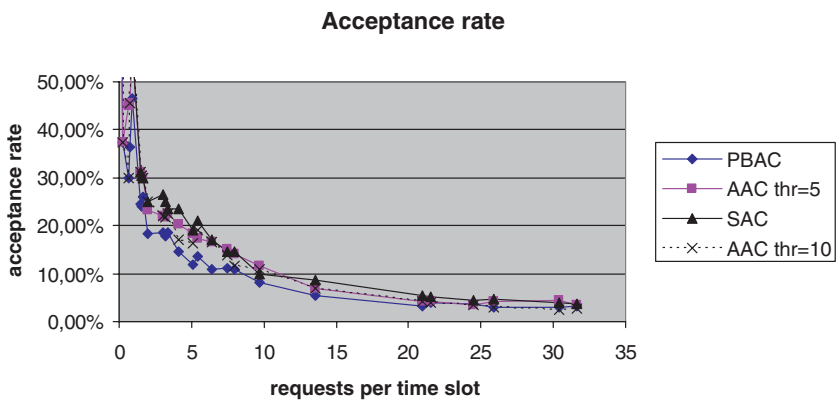

Figure 10: Comparison of acceptance ratios

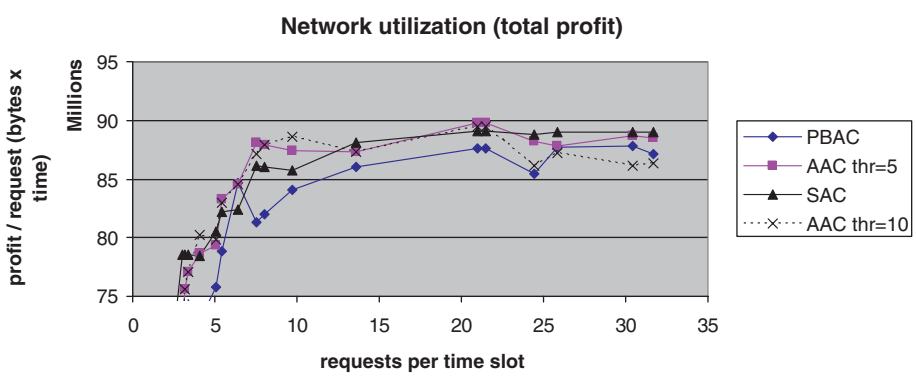

Figure 11: Network utilization

DOI: 10.1002/nem

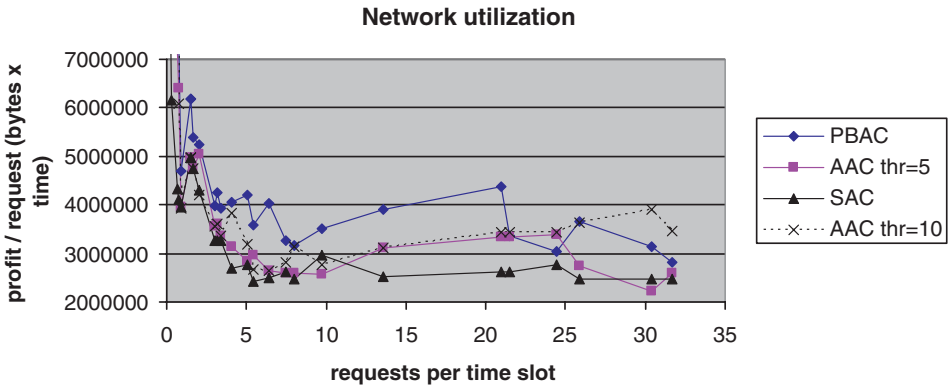

Figure 12: Network utilization per request

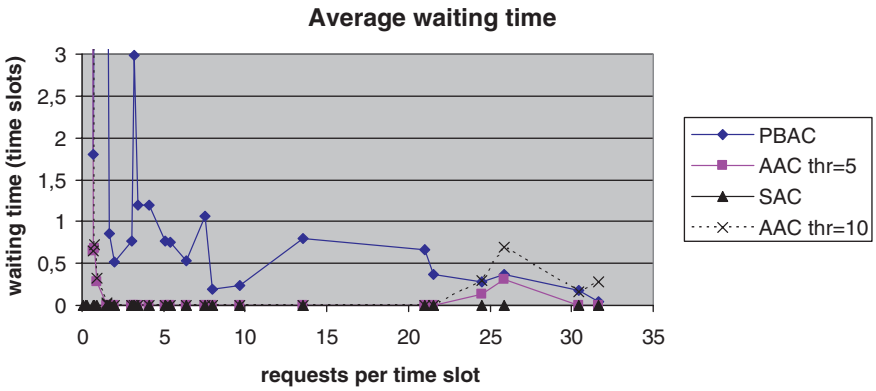

Figure 13. Average waiting time

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.

In most real environments it is expected that a relatively quick response to a request will be essential. As Figure 13 demonstrates, SAC is extremely responsive, as expected. This also means that there is room for a trade-off that can be used 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.

6. CONCLUSIONS AND FUTURE WORK

Our proposed adaptive admission control algorithm improves on the common admission control modules of Bandwidth Brokers on a number of aspects. It offers better utilization of the network

DOI: 10.1002/nem

resources, while keeping a balance between simplicity and functionality. It automatically falls back to a simpler model without trying to optimize the network utilization, if its operating environment indicates that the algorithm is too complex for the circumstances. While we have studied the operation of the algorithm and admission control using the hose model at the domain's ingress points, it can easily be extended and formulated in order to operate on a per flow basis, a variation of the basic algorithm that we intend to simulate and compare with the presented architecture. Our experimental results show that the proposed algorithm compares favorably to other variations of the admission control module when network utilization and reduced computational overhead are important considerations.

A number of improvements can be made to the basic algorithm, depending among other things on the implementation environment, the formulation and the specific circumstances of the admission control problem that we face. For example, in some cases it might be allowed for some requests to not specify their ending time. The Bandwidth Broker algorithm can easily be extended to take into account such requests, by making the conservative assumption that they will reserve the requested bandwidth indefinitely. This flexibility of course can be offered at a higher than normal cost, possibly determined by the SLA parameters. Another idea that we plan to integrate into the basic algorithm is of requests that have been overall rejected, but which can be notified at a later time when they will have better chances of success. This can be achieved by keeping a tentative list of the total bandwidth requested at any time, for both admitted and pending requests.

Our future work will also focus both on the extension of the study on the Bandwidth Broker architectures, and on the issue of securing the Bandwidth Broker's operation from compromised Bandwidth Broker components, from disobedient clients, and from stolen or altered messages while transmitted on the network. Specifically, we intend to evaluate our model on an actual environment and measure our implementation's resilience to various kinds of failures. Finally, we intend to further analyze and document the proper configuration of the operation parameters of the described architecture.

REFERENCES

1. Nichols K, Jackobson V, Zhang L. A two-bit differentiated services architecture for the Internet. *RFC 2638*, July 1999.
2. Fankhauser G, Schweikert D, Plattner B. Service level agreement trading for the differentiated services architecture. *Tech. Rep. 59*, TIK, 1999.
3. Wolf L, Steinmetz R. Concepts for resource reservation in advance. *Journal of Multimedia Tools and Applications, Special Issue on State of the Art in Multimedia Computing*, 1997; May: 255–278.
4. Gupta A. Advance reservation in real-time communication services. In *Proceedings of the IEEE 22nd Annual Conference on Local Computer Networks (LCN'1997)*, Minneapolis, MN, 1997.
5. Ferrari D, Gupta A, Ventre G. Distributed advance reservation of real-time connections. In *Proceedings of the Fifth International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, Durham, NH, April 1995; 16–27.
6. Degermark M, Kohler T, Pink S, Schelen O. Advance reservations for predictive service. In *Proceedings from 5th Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'95)*, Durham, NH, April 1995; 3–14.
7. Hwang J, Chapin SJ, Mantar HA, Okumus IT, Revuru R, Salama A. An implementation study of a dynamic interdomain bandwidth management platform in DiffServ networks. In *Proceedings of the 9th IEEE/IFIP Network Operations and Management Symposium (NOMS 2004)*, Seoul, Korea, April 2004.
8. Wong K, Law KLE. ABB: Active Bandwidth Broker. In *Proceedings of SPIE ITCom 2001*, Vol. 4523, Denver, August 2001; 253–262.
9. Durham D, Boyle J, Cohen R, Herzog S, Rajan R, Sastry A. The COPS (Common Open Policy Service) Protocol. *RFC 2748*, January 2000.
10. Bouras C, Stamos K. An adaptive admission control algorithm for Bandwidth Brokers. In *3rd IEEE International Symposium on Network Computing and Applications (NCA04)*, Cambridge, MA, 30 August–1 September 2004; 243–250.

DOI: 10.1002/nem

11. Schelén O, Nilsson A, Norrga*p26rd J, Pink S. Performance of QoS agents for provisioning network resources. In *Proceedings of IFIP Seventh International Workshop on Quality of Service (IWQoS'99)*, London, June 1999.
12. Burchard L. Analysis of data structures for admission control of advance reservation requests. *IEEE Transactions on Knowledge and Data Engineering* 2005; **17**(3): 413–424.
13. Zhang Z, Duan Z, Hou Y. On scalable design of Bandwidth Brokers. *IEICE Transactions on Communications* 2001; **E84-B**(8): 2011–2025.
14. Machiraju S, Seshadri M, Stoica I. A scalable and robust solution for bandwidth allocation. *Technical Report UCB//CSD02–1176*, University of California at Berkeley, 2002.
15. Greenberg A, Srikant R, Whitt W. Resource sharing for book-ahead and instantaneous-request calls. *IEEE/ACM Transactions on Networking* 1999; **7**(1): 10–22.
16. Chhabra C, Erlebach T, Stiller B, Vukadinovic D. Price-based call admission control in a single DiffServ domain. *TIK-Report No. 135*, May 2002.
17. Bouras C, Kapoulas V, Pantziou G, Spirakis P. Competitive video on demand schedulers for popular movies. *Discrete Applied Mathematics* 2003; **129**: 49–61.
18. Mykoniati E, Charalampous C, Georgatsos P, Damilatis T, Goderis D, Trimintzios P, Pavlou G, Griffin D. Admission control for providing QoS in DiffServ IP networks: the TEQUILA Approach. *IEEE Communications Magazine* 2003; **41**(1): 38–44.
19. QBone Bandwidth Broker architecture. QBone Signaling Design Team. http://qbone.internet2.edu/bb/bboutline2.html [26 January 2007].
20. Bandwidth Broker implementation. Information and Technology Telecommunication Center, University of Kansas. http://qos.ittc.ku.edu [5 February 2007].
21. Braun T, Stattenberger G. Performance of a Bandwidth Broker for DiffServ networks. In *Kommunikation in verteilten Systemen (KiVS03)*, Leipzig, Germany, 25–28 March 2003; 93–104.
22. Ogawa J, Terzis A, Tsui S, Wang L, Zhang L. A prototype implementation of the two-tier architecture for Differentiated Services. In *Proceedings of the Fifth IEEE RealTime Technology and Applications Symposium (RTAS'99)*, Vancouver, Canada.
23. Sohail S, Jha S. The survey of Bandwidth Broker. *Technical Report UNSW CSE TR 0206*, School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, May 2002.
24. Cortese G, Fiutem R, Cremonese P, D'Antonio S, Esposito M, Romano SP, Diaconescu A. CADENUS: creation and deployment of end-user services in premium IP networks. *IEEE Communications Magazine* 2003; **41**(1): 54–60.
25. Bouras C, Stamos K. Examining the benefits of a hybrid distributed architecture for Bandwidth Brokers. In *First IEEE International Workshop on Multimedia Systems and Networking (WMSN'05)*, Phoenix, AZ, 7–9 April 2005; 491–498.
26. Duffield N, Goyal P, Greenberg A, Mishra PP, Ramakrishnan KK, van der Merive JE. Flexible model for resource management in Virtual Private Networks. *SIGCOMM 1999*; 95–108.
27. Gibbens RJ, Hunt PJ. Effective bandwidths for the multi-type UAS channel. *Queueing Systems* 1991; **9**: 17–28.
28. Guerin R, Ahmadi H, Naghshineh M. Equivalent capacity and its application to bandwidth allocation in high-speed networks. *IEEE Journal on Selected Areas in Communications* 1991; **9**: 968–981.
29. Kelly FP. Effective bandwidths at multi-class queues. *Queueing Systems* 1991; **9**: 5–16.
30. Vazirani V. *Approximation Algorithms*. Springer: Berlin, 2001.
31. GEANT2, Bandwidth on Demand. http://www.geant2.net/server/show/nav.756 [26 January 2007].
32. DRAGON project. http://dragon.maxgigapop.net/twiki/bin/view/DRAGON/WebHome [26 January 2007].
33. GEANT2. End-to-End Service Provision. http://www.geant2.net/server/show/nav.893 [26 January 2007].
34. European Telecommunications Standards Institute (ETSI). http://www.etsi.org/ [26 January 2007].
35. ETSI ES 282 003 V1.1.1. Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); Resource and Admission Control Sub-system (RACS); Functional Architecture. European Telecommunications Standards Institute, 2006.
36. Gallon C. Quality of service for next generation VoIP networks. *MSF-TR-QoS-FINAL* February 2003.
37. Gallon C, Schelén O. Bandwidth management in next generation packet networks. *MSF Technical Report, MSF-TR-ARCH-005-FINAL*, August 2005.
38. 3GPP TS 23.207 v6.4.0 3rd Generation Partnership Project. Technical specification group services and system aspects; end-to-end quality of service (QoS) concept and architecture (Release 6) September 2005.
39. QBone Signaling Design Team. Final report. http://qos.internet2.edu/wg/documents-informational/20020709-chimento-etal-qbone-signaling/, 9 July 2002 [26 January 2007].

DOI: 10.1002/nem

40. Sander V, The security environment of SIBBS. http://qbone.internet2.edu/bb/SIBBS-SEC.doc, June 2000 [26 January 2007].
41. Lee B, Woo W-K, Yeo C-K, Lim T-M, Lim B-H, He Y, Song J. Secure communications between Bandwidth Brokers. *Operating Systems Review* 2004; **38**(1): 43–57.
42. True Random Number Service. www.random.org [26 January 2007].
43. Research Unit 6, Research Academic Computer Technology Institute (CTI). http://ru6.cti.gr/ru6/ns_home.php [26 January 2007].
44. The Network Simulator: ns-2. http://www.isi.edu/nsnam/ns/ [26 January 2007].
45. L'Ecuyer P. Good parameters and implementations for combined multiple recursive random number generators. *Operations Research* 1999; **47**(1):159–164.

DOI: 10.1002/nem