An Adaptive Admission Control Algorithm for Bandwidth Brokers

Ch. Bouras, K. Stamos
Research Academic Computer Technology Institute, Riga Feraiou 61, GR-26221 Patras, Greece and Computer Engineering and Informatics Dept., Univ. of Patras, GR-26500 Patras, Greece
Tel:+30-2610-{960375, 960316}
Fax:+30-2610-{969016, 960358}
e-mail: {bouras, stamos}@cti.gr

Abstract

This paper focuses on the operation of the Bandwidth Broker, an entity that is responsible for managing the bandwidth within a network domain and for the communication with Bandwidth Brokers of neighboring domains. A very important aspect of the Bandwidth Broker is its admission control module that determines whether the bandwidth reservation requests are going to be accepted or not. We summarize the status of the current research in this field and propose 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. This is achieved by gathering and examining sets of book-ahead requests and by adapting the size of the set to be examined so that the network utilization and the computation overhead are appropriately balanced.

1. Introduction

The Differentiated Services (DiffServ) framework [1] is one of the basic architectures that have been proposed for QoS provision in the Internet. 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 [2] is an entity responsible for providing QoS within a network domain. The Bandwidth Broker manages the resources within the specific 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. The choice of the Bandwidth Broker to either accept or reject a request is based on the network load and on the Service Level Agreement (SLA) [4]. The SLA is the service contract between the service provider and every customer that indicates the service that the customer is going to receive. The decision to accept or reject a request is made by a module called admission control that takes into account the above data (the network condition and the SLA). A Bandwidth Broker is also responsible for the inter-domain communication with Bandwidth Brokers of adjacent domains. This procedure requires direct communication between the two adjacent Bandwidth Brokers and also a special agreement between the two domains that should be considered on the decision mechanism.

The operation of a Bandwidth Broker depends on the cooperation of a number of modules which include its inter-domain interface, the intra-domain interface, a routing table interface, a user/application interface, a policy manager interface and a network management interface.

Users request network resources from the Bandwidth Broker that manages their local domain by using Resource Allocation Request (RAR) messages. When the Bandwidth Broker receives the request message, it uses the admission control module to determine whether the requested resources can be allocated to the user and whether the SLA requirements are met. If the request is accepted, the Bandwidth Broker uses its intra-domain interface in order to properly configure the routers belonging to the managed domain and notifies the user with a Request Allocation Answer (RAA) message. If the

request is rejected, the answer message informs the user about the rejection.

The resource management in each domain is accomplished mostly via the DiffServ architecture. The DiffServ architecture proposes the provision of service differentiation to the traffic in a scalable manner by suggesting the aggregation of individual application flows with similar quality needs. These aggregations are appointed to different classes of service and the network treats the packets that belong to each class differently. The Differentiated Services Code Point (DSCP) is a field in the IP header that specifies the class of service each packet belongs to. The network classifies and marks the packets in order to provide them differential per hop behavior (PHB) according to the class of service they belong to. The PHB is specified on all the network nodes and includes functions that implement resource allocation and packet drop policy.

In this paper we deal with 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 the best utilization of the network. Our algorithm makes use of the existing literature on scheduling problems where a satisfactory solution can be obtained using approximation algorithms. The algorithm then attempts to balance the calculation of the optimal solution with a limitation on the computation overhead that the algorithm itself incurs at the Bandwidth Broker operation. 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 multitude of real-world cases.

The rest of the paper is organized as follows: Section 2 describes in more detail the admission control module of a Bandwidth Broker and the related work in this area. Section 3 presents our proposed algorithm for the admission control architecture. Section 4 presents some initial performance measurements, while section 5 describes our conclusions as well as the future work that we intend to do on this area.

2. Admission Control Module

Among its tasks, a Bandwidth Broker has to perform admission control for the incoming requests, which means that it has to define whether the 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. 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 the 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.

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. The authors in [12] 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 [13]. 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.

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 [8], 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 of 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), and calculate the resources requested by a new reservation for the time slots that it overlaps [8], [9].

For both immediate and book-ahead requests, it may be possible to either specifically declare the ending time of the reservation, or not. An intermediate case is when a reservation request has to provide both its starting and

ending times, but can make new additional requests that extend its initial reservation period.

In some cases, a book-ahead request may have a flexibility of allowing the Bandwidth Broker to answer the request by either accepting it or rejecting it not immediately, but after a period of time (which can be specified). 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. 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 [3], [6] or on a per-flow basis [11]. 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.

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 [3].

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 [3].

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 [10]. 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. This is the approach we have taken in this paper. 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. 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 [14].

3. Description of 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.

After receiving a request, a routing algorithm has to be invoked in order to determine the path that the requested flow will follow. Admission control is done on a hop-by-hop basis, so for each of the links that the flow will have to traverse, it has to be determined whether there is available bandwidth and whether admitting the flow will increase the utilization of the network.

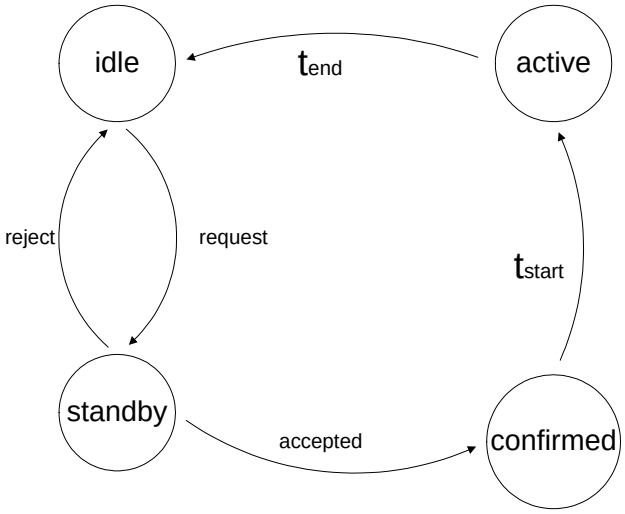

Figure 1. Request states

We suppose a new request has the form of

$$r(t_{start}, t_{end}, b, w)$$

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. The Bandwidth Broker's admission control module keeps a list of unanswered requests, which we call waiting queue W_q, sorted by their waiting time w.

As soon as the first item, 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 [15]:

Suppose n is the cardinality of W_q. We define

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

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 link 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.
An approximation algorithm for this problem can be used solving the following linear programming relaxation in polynomial time [5] and then making the solution discrete regarding the variables x_i, which represent whether r_i is accepted or not:

$$\max \Sigma_{i \in R} b_i (t_{end} - t_{start}) x_i$$

$$\Sigma_{i \in R(t)} b_i x_i \leq B, \text{ for all } t \text{ in the period from the earliest } t_{start} \text{ to the latest } t_{end} \text{ in the } R \text{ set}$$

$$0 \leq x_i \leq 1, \; i \in R$$

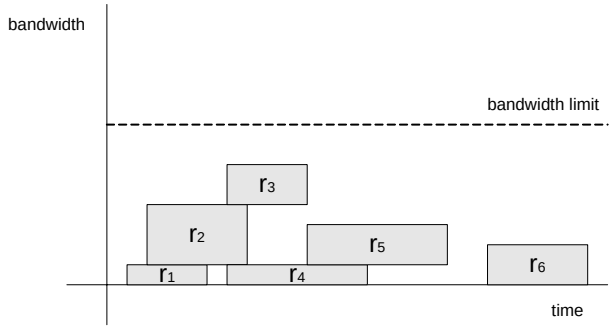

Figure 2. Reservation requests

The important point is how to select the R set. A simple approach would be to simply set $R=W_q$. This solution leads 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 separately be invoked 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
```

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 the algorithm will not last more than an accepted threshold, otherwise it will be simplified to admission control for a single request. 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, using a data structure as in [6]. 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 [15], is equivalent to a simpler version of our scheduling problem where all t_{start} and t_{end} times are equal. Because however it is not critical to have the optimal solution, we can use an approximation scheme that runs in polynomial time and approximates the optimal solution with the desired accuracy.

Following is a summary of the algorithm for calculating the accepted requests on a link:

```
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

    0 ≤ xi ≤ 1, i∈R

    Discretize solution for xi

    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-$\{r_1\}$.

Instead of rejecting an end-to-end request that was not admitted, the algorithm returns to the routing phase and chooses a different path that does not include the rejected link, if available.

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) lead the algorithm to quickly converge to the best approximation of the optimal solution.

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

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

Solving the recursive function gives

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

and therefore

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

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

Therefore, R_{size} converges to the size of W_q as quickly as $(1\text{-}a)^t$ 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 achieves fairness 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.

A very long queue Wq indicates that a specific link is excessively utilized, and we can add a checking mechanism that monitors the length of Wq, and compares it with the corresponding queues for other links. If Wq, exceeds the average queue size by more than a specified threshold, then the routing procedure can be re-invoked and the request rerouted via other links.

4. Performance Evaluation

An important aspect in examining the performance of an admission control architecture is the worst-case

behaviour of the algorithm. In particular, it is not enough to study the behaviour in a normal situation, but we have to examine cases where there is an adversary user issuing requests in a way that tries to minimize the performance of the algorithm. Although in that case our algorithm chooses to sacrifice the optimal network utilization, it does so in order to make sure that the computation overhead will not exceed some acceptable limit. Our intention is to enable the network administrator that configures the Bandwidth Broker operation to specify the acceptable threshold for the overhead due to complexity, and then the algorithm can adapt in order to improve the network utilization under the circumstances.

We experimented with the algorithm using 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.

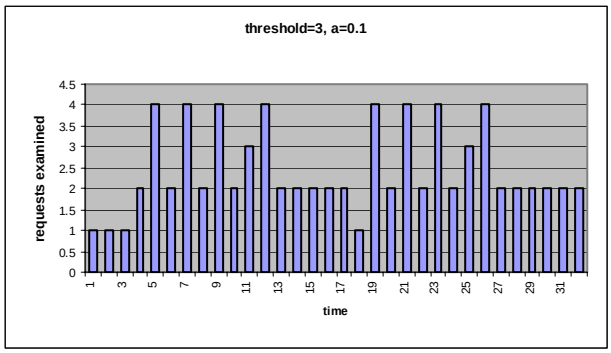

Figure 3. Examined requests for smaller threshold

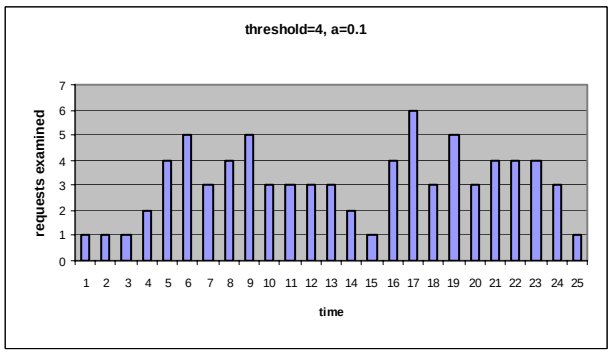

Figure 4. Examined requests for medium threshold

Figures 3 through 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 4 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 8 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, 4 and 5.

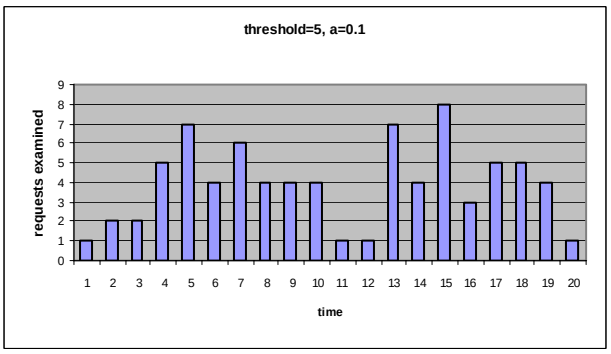

Figure 5. Examined requests for larger threshold

In the presented simulations the value of adaptation parameter *a* was identical at 0.1, which produces a cautious behaviour of the algorithm. The size if 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 through 8, which display the adaptive behaviour 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 behaviour 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 behaviour it is better to keep the adaptive parameter *a* around the 0.1 range of values.

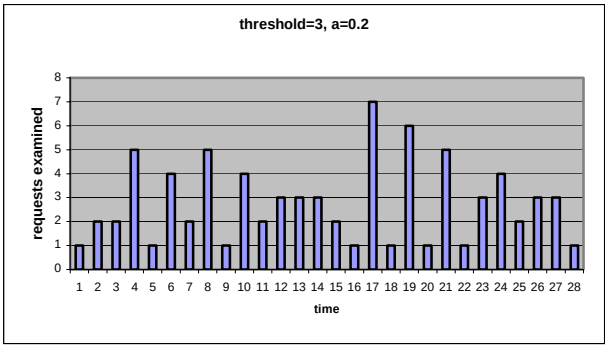

Figure 6. Examined requests with smaller threshold and aggressive adaptation

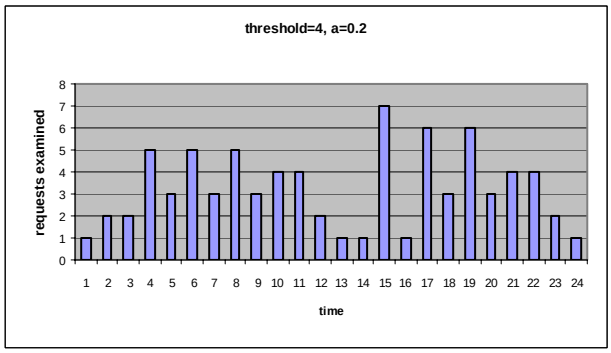

Figure 7. Examined requests with medium threshold and aggressive adaptation

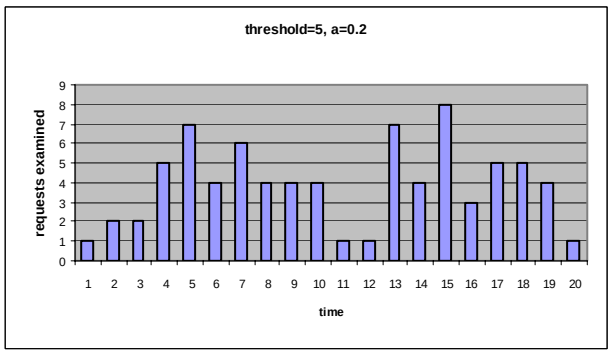

Figure 8. Examined requests with larger threshold and aggressive adaptation

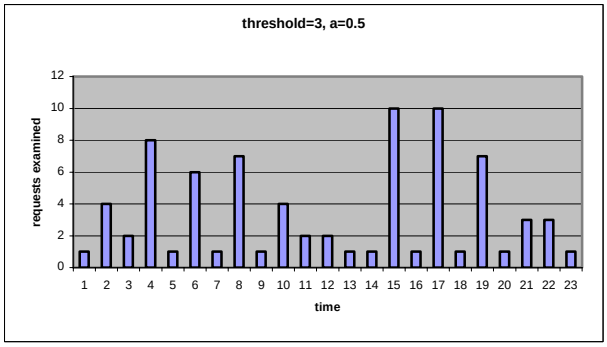

Figure 9. Examined requests with very aggressive adaptation

5. Conclusions - Future work

Our proposed algorithm improves on the common admission control modules of Bandwidth Brokers on a number of aspects. It offers better utilization of the network 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 on a hop by hop basis, 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 future work includes the integration of the proposed algorithm in the well-known ns-2 simulator architecture evaluation. Furthermore we intend to compare the proposed architecture with alternative schemes in terms of efficiency, resource consumption and network utilization. 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 case 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, nut which can be notified at a later time when they will have better success chances. This can be achieved by keeping a tentative list of the total bandwidth requested at any time, for both admitted and pending requests.

Finally, we intend to further analyze and document the proper configuration of the operation parameters of the described architecture.

6. References

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

[2]. RFC 2638 “An Two-bit Differentiated Services Architecture for the Internet”, K. Nichols, V. Jackobson, L. Zhang, July 1999

[3]. Olov Schelén, Andreas Nilsson, Joakim Norrgård, Stephen Pink “Performance of QoS Agents for Provisioning Network Resources”. In Proceedings of IFIP Seventh International Workshop on Quality of Service (IWQoS'99), London, UK, June 1999

[4]. G. Fankhauser, D. Schweikert, and B. Plattner, “Service Level Agreement Trading for the Differentiated Services Architecture”, Tech. Rep. 59, TIK, 1999

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

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

[8]. D. Ferrari, A. Gupta, G. Ventre, “Distributed advance reservation of real-time connections”, 1995

[9]. M. Degermark, T. Kohler, S. Pink, O. Schelen, “Advance Reservations for Predictive Service, 1995

[10]. S. Machiraju, M. Seshadri, I. Stoica, “A Scalable and Robust Solution for Bandwidth Allocation”, 2002

[11]. 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

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

[13]. A. Gupta, “Advance reservations in real-time communication services”

[14]. C. Bouras, V. Kapoulas, G. Pantziou, P. Spirakis, “Competitive Video On Demand Schedulers for Popular Movies”, Discrete Applied Mathematics 129 (2003) pp. 49-61

[15]. V. Vazirani, “Approximation Algorithms”, 1997