Examining the Benefits of a Hybrid Distributed Architecture 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

In this paper the issue of the architectural organization of a Bandwidth Broker with respect to the distribution of functionalities among separate entities is examined. We discuss the benefits of a distributed versus a centralized architecture and study some of the most important implementations. We also present a novel design proposal that builds on our earlier work on the admission control module of a Bandwidth Broker and introduces it in the framework of a distributed Bandwidth Broker architecture. Our conclusions are supported by a series of simulated experiments that we analyze.

1. Introduction

The Differentiated Services (DiffServ) framework 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 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.

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.

Since the concept of the Bandwidth Broker was introduced [1], a number of architectures have been proposed and a number of implementations have been made. In this paper we examine and compare the characteristics of some of the most important already existing architectures, and we study how the way the functionalities are distributed in the network can be improved. A survey on existing Bandwidth Broker implementations can be found in [6].

It is very important while considering the architecture of a Bandwidth Broker to determine whether the Bandwidth Broker is going to be a centralized module with full knowledge of the whole domain that it manages, or whether its operation is distributed among several nodes in the network. Both approaches offer advantages and disadvantages and usually present the engineer with a number of trade-offs.

The basic factor that influents toward a distributed designed Bandwidth Broker is the scalability issue with regard to the distribution of the processing load. As the network size increases, the Bandwidth Broker is likely to face increasingly large numbers of flows. Depending on the approach taken for handling the admission control functionality, this increase can lead to scalability problems in terms of disk access speed and memory requirements. In our previous work [3], we have dealt with how this problem could be tackled by using a more intelligent adaptive admission control module. This solution, however, has the downside of scaling in expense of specific metrics that are associated with the Bandwidth Broker performance, like the achieved network utilization. A different approach that does not have any impact on the Bandwidth Broker performance could be the distribution of the load among separate entities, leading to a distributed Bandwidth Broker design.

On the other hand, such an architecture increases the Bandwidth Broker's need for network communication,

and therefore introduces an overhead that has to be taken into account and examined.

Current literature on Bandwidth Brokers has dealt with both categories of designs. Our description of the existing Bandwidth Broker architectures is presented with regard to the way they distribute the processing load and their potential scalability in terms of disk access speed, memory and communication overhead. The existing architectures can be categorized as follows:

- Centralized architectures: A single Bandwidth Broker handles all requests
- Distributed architectures:
  - Multiple Bandwidth Brokers operating identically, duplicating information
  - Multiple Bandwidth Brokers handling different tasks, e.g. a central Bandwidth Broker and multiple edge Bandwidth Brokers.
- Hybrid architectures: A single Bandwidth Broker that under circumstances can distribute the load.

The rest of the paper is organized as follows: Section 2 presents a comparison between the different architectures, and section 3 presents a hybrid model that extends previous work with a distribution of the Bandwidth Broker components. Section 4 contains extensive simulations in order to evaluate the proposed architecture, while section 5 describes our final conclusions as well as the future work that we intend to do on this area.

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 ([7], [8], [9], [10]). Our intention is to outline the advantages and disadvantages of the different strategies for the Bandwidth Broker design. A lot of researchers have worked on this area producing various solutions, each of which is potentially suitable for several situations and weak on others.

Table 1: Comparison table of various Bandwidth Broker architectures

	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)	–

Some of the architectures presented in the comparison 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 implemented Bandwidth Broker architectures that one can find in the current literature. It should also be noted, that as Table 1 also indicates, the distributed architectures are not as common as the centralized ones.

Most of the existing literature proposes Bandwidth Brokers that comprise 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, the authors in [4] propose a distributed Bandwidth Broker architecture. Their design is comprised of one central Bandwidth Broker (cBB) and a number of edge Bandwidth Broker (eBB) 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 however the link state database is only maintained by the cBB, 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.

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 (for example 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.

At this point we can discover still 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, 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) are usually much more easily discovered and more desirable than the rest of the failure categories, if the Bandwidth Broker architecture is implemented and deployed at an environment where such failures can be ruled out, the relevant types of considerations can naturally be discarded.

3. A hybrid model

In [3], we proposed an adaptive model for the admission control module of a Bandwidth Broker that aims at improving the resource utilization of the admission mechanism while balancing it with the Bandwidth Broker's processing load. The above discussions demonstrate another variation of that algorithm that will also be able to benefit from a distributed Bandwidth Broker architecture. In our previous work, when the admission control module determined that the processing load exceeding a pre-defined level, it chose to relax the approximation of the optimal solution for the resource utilization. Our basic idea here is that a distributed architecture in cases of high processing load, instead of settling for less optimal admission decisions, allows the processing load to be distributed among separate entities that can work in parallel.

Our model uses a central Bandwidth Broker that is typically an available powerful server, and deals with the admission of the requested flows as long as the admission

algorithm computations does not exceed its capabilities. The central Bandwidth Broker has complete knowledge of the managed network and the location of the secondary Bandwidth Brokers. Each secondary Bandwidth Broker is assigned a mutually exclusive subset of the network nodes in its neighborhood and keeps information about the state of the relevant part of the network. As soon as the central Bandwidth Broker is not capable of meeting the computation thresholds, it allocates subsets of the admission requests to the secondary entities. The secondary Bandwidth Brokers use the same admission control algorithm as the central Bandwidth Broker, but the smaller subsets they handle allow them to reside on relatively inexpensive hardware. Figure 1 visualizes an example instance of the described architecture.

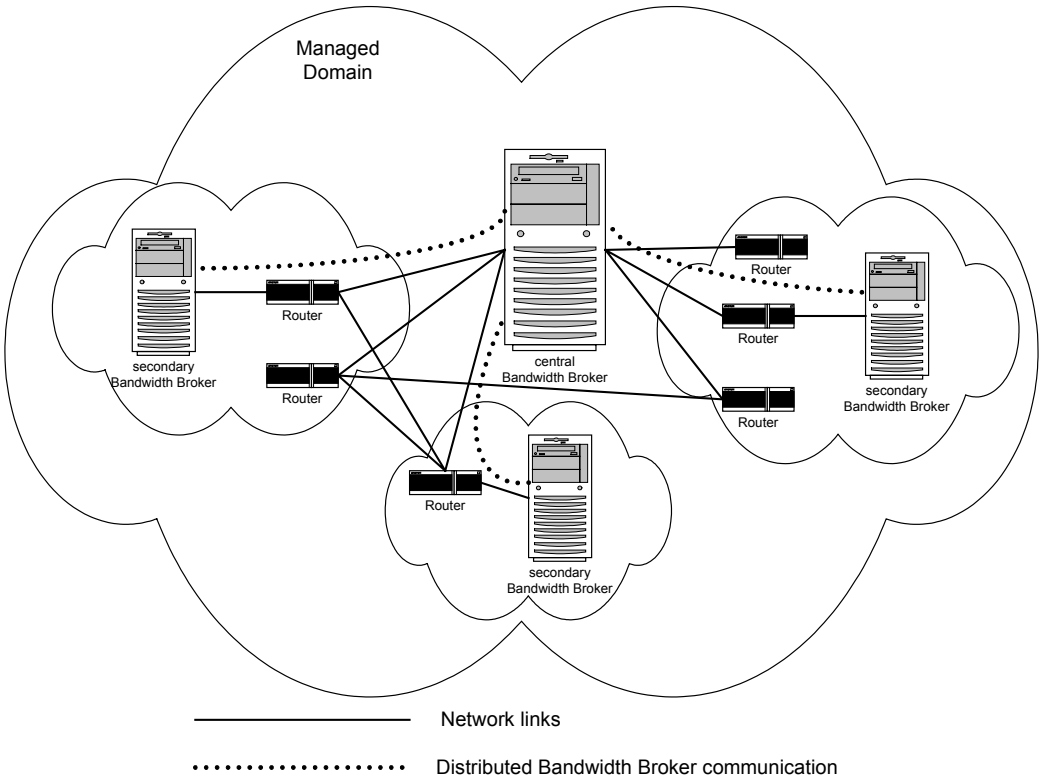

Figure 1: Distributed architecture

We have chosen to distinguish between three cases relative to the processing load on the central Bandwidth Broker. During Phase 1, the central Bandwidth Broker handles all admission requests and manages the domain by itself. When the computation load becomes large enough to exceed a predetermined threshold, the central Bandwidth Broker proceeds to Phase 2 and starts distributing some of the admission requests. In particular, it distributes the admission requests whose source and destination are both managed by the same secondary Bandwidth Broker (we name these local requests). If the remaining load on the central Bandwidth Broker remains large (larger than *a* times the threshold, where *a* can be reasonably set to a value of 1.5 as discussed below), we move to Phase 3, where the central Bandwidth Broker iteratively distributes even more admission requests in chunks. Requests are distributed according to the length of their predicted path (therefore the central Bandwidth Broker will release the control of the requests with the longest paths, for which it is the most suitable of managing, at the very latest stages). In order not to increase the complexity of the distribution mechanism the prediction of the path length is performed efficiently using a matrix that contains average path lengths according to their source and destination. This matrix is aggregated similarly to the segmentation of the domain in the areas managed by the secondary Bandwidth Brokers and therefore is kept small enough for quicker access and smaller memory requirements. The selection of the value of parameter *a* is closely associated to the topology of the specific network. Larger values should be used when the ratio of secondary Bandwidth Brokers to network size is large, while smaller values should be used when there is a relatively small number of secondary Bandwidth Brokers. The reason is that this ratio also implicitly affects the percentage of requests that will be local, and therefore

distributed already from Phase 2. Figure 2 gives an overview of the transitioning mechanism between the three operating phases, while the following pseudocode summarizes the idea behind the proposed model:

```
   Phase = 1
   while incoming requests are accumulated
     try to compute optimal admission decision
     if ComputationTime > Threshold
       if ComputationTime > a * Threshold
         Phase = 3
         sort requests by predicted path length
         chunk = the (chunk size) shortest paths
         distribution set = distribution set + chunk
       else
         Phase = 2
         distribution set = local requests
       end if
       break distribution set in mutually exclusive sets
       for all sets
         activate secondary BB
         distribute set to secondary BB
       end for
     else
       Phase = 1
       Empty distribution set
       increase max number of admission requests concurrently
examined
     end if
   end while
```

Details on how the admission control part of the mechanism operates can be found in [3]. Whether all secondary Bandwidth Brokers are active or not is determined by the central Bandwidth Broker. Secondary Bandwidth Brokers that manage a part of the network where a lot of admission requests are originated from are the ones that are invoked more frequently.

During its operation the central Bandwidth Broker entity makes sure to keep the secondary Bandwidth Brokers updated about the state of the network links and the already admitted flows. In order to keep the communication overhead as low as possible, the central Bandwidth Broker transmits only the absolutely necessary information to the secondary Bandwidth Brokers. In particular, when transitioning from Phase 1 to Phase 2, the central Bandwidth Broker only needs to transfer the local routing and reservation status to some secondary Bandwidth Brokers (since during Phase 2, the secondary Bandwidth Brokers are only going to decide for local requests). While in Phase 2, the central Bandwidth Broker only transmits the routing and reservation status information to the secondary Bandwidth Brokers that will examine requests. Finally, when transitioning to Phase 3, the central Bandwidth Broker transmits the complete routing and reservation status information to some secondary Bandwidth Brokers. While in Phase 3, the central Bandwidth Broker will have to transfer both local and complete information on several secondary Bandwidth Brokers.

Although the secondary Bandwidth Broker entities might seem to require additional resources and additional network management effort, their low requirements allow them to be typically hosted on the machine that accompanies the router software in most practical implementations. In this case, the proposed architecture manages to decrease the load from the central Bandwidth Broker while staying relatively close to a satisfactory network utilization as described in [3], while also taking advantage of existing resources.

We have chosen not to keep a flow database as in [4], in order to save memory, hard disk space and CPU

resources. Furthermore, such an approach would make it more difficult to assign the mutually exclusive admission request sets.

Secondary Bandwidth Brokers can also function as back-up mechanisms in the event that the central Bandwidth Broker fails. In the simple case when the central Bandwidth Brokers ceases to operate, a suitable designated secondary can take its place. For the more serious case of a Byzantine failure, the proposed architecture can easily be designed so that secondary Bandwidth Brokers can detect and overcome an erroneously operating central Bandwidth Broker. This can be achieved if several secondary Bandwidth Brokers have been specifically assigned and monitor the messages originated by the central entity. If these messages become consistently corrupted or incompatible with the algorithm's operation (of which the secondary Bandwidth Brokers are aware since they run the same algorithm on a smaller scale) they can notify the rest and effectively ignore the erroneously behaving central Bandwidth Broker.

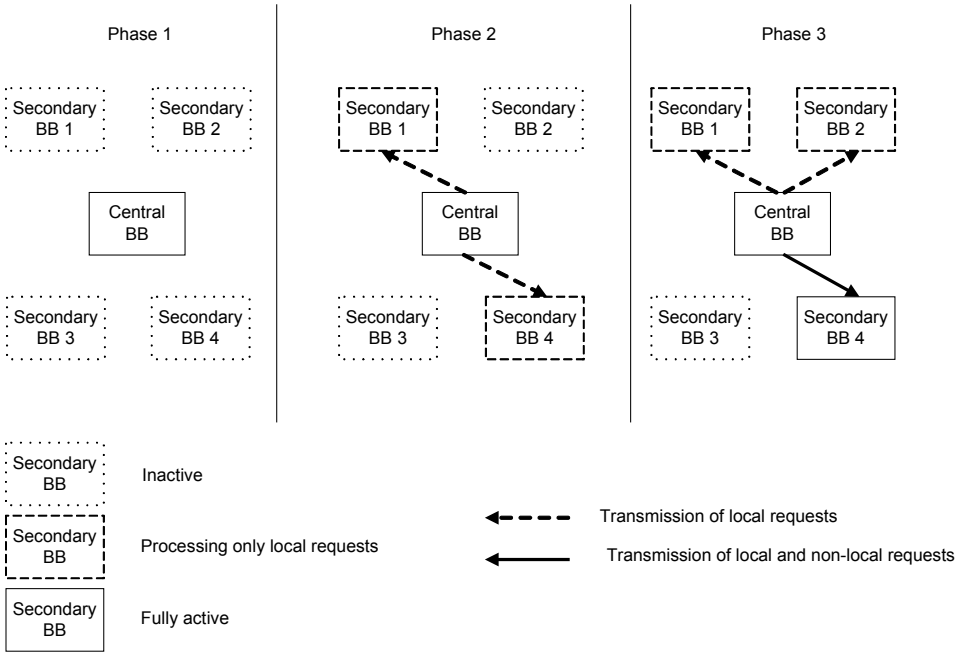

Figure 2: Hybrid model's operation

4. Performance evaluation via simulation

4.1. Simulation setup

In our performance evaluation we have to take into consideration the following factors that affect the scalability capability of a centralized Bandwidth Broker design:

- Disk accesses: This parameter depends on the number of updates that the Bandwidth Broker has to perform on the states that it maintains for managing the network. Reducing this number can significantly enhance the Bandwidth Broker's capability to deal with heavier loads.
- Memory requirements: Memory can also be a limiting factor and has been examined in previous work as in [5].
- Communication overhead: Depending on the distribution architecture, this factor depends on the level of communication between equivalent modules of the Bandwidth Broker or between hierarchically separated components, for example a central Bandwidth Broker communicating with edge Bandwidth Brokers.

The communication model used in an architecture plays a very significant role for the overhead that this communication introduces at the network. Some implementations choose to use the COPS protocol [2] that has been specifically designed for the kind of communication that is necessary for exchanging policy information between the Bandwidth Broker and the router, while others prefer a customized approach.

The hybrid distribution model that was presented in the previous section was evaluated through simulation experiments using a customized simulation environment that we developed for the purposes of this evaluation. This enabled us to run a number of simulations with various topologies and various combinations of central and secondary Bandwidth Brokers. In general, trying to evaluate the number of disk accesses and the memory requirements of a specific algorithm is rather complicated because it can be affected by low level details like the type of operating system used and the quality of the implementation, factors which can not be evaluated satisfactorily in a simulation environment. We therefore chose to use the distribution of the requests as the most appropriate indicator of the overhead that our model imposes on the various components that comprise our architecture, and the one that is least affected by the fact that the results are extracted from a simulation and not from an actual implementation.

For our simulations, we used the initial hybrid model that was presented in section 3. Because our intention was to measure the performance characteristics of the proposed model and not its failure-resilience features, the simulation did not take into account Byzantine or other failures. We believe that such performance evaluations are better suited for actual environments. The simulations were performed on a Pentium 4 2.4 GHz Windows PC and we repeated each experiment in order to certify its repeatability. Reservation requests were simulated using an input file that was gradually processed by the Bandwidth Broker module.

4.2. Simulation results

The initial topology specified one central and 2 secondary Bandwidth Brokers, with each one managing an equal number of users that created reservation requests. For our first experiment we used a random sequence of reservation requests that were infrequent enough for the central Bandwidth Broker to handle them exclusively. The results are shown in Figure 3.

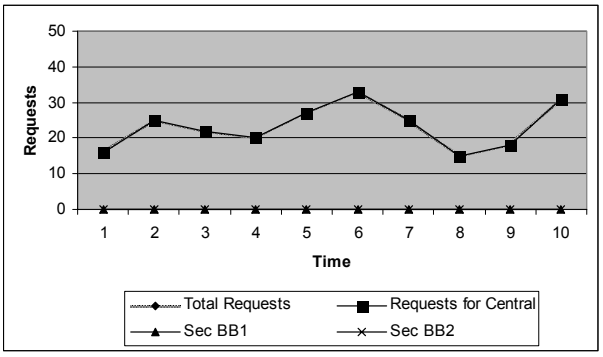

Figure 3: Infrequent requests

Since the central Bandwidth Broker can easily handle all admission decisions, they are optimally decided without resorting to any secondary Bandwidth Broker, which therefore remain inactive. In this case, the central Bandwidth Broker can decide on the admission of requests using an algorithm that approximates the optimal utilization of network resources, as detailed in [3].

For our second experiment we also used a random sequence of reservation requests, this time however they arrived at a higher frequency. The results we obtained can be seen in Figure 4.

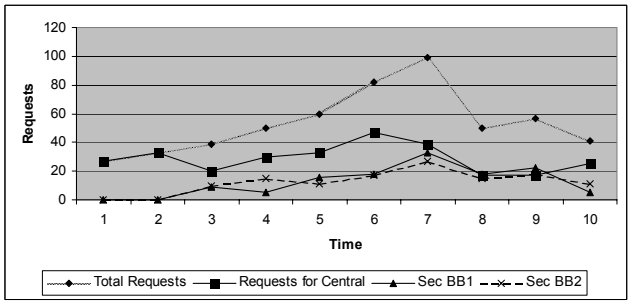

Figure 4: Frequent requests

As the incoming requests increased, the central Bandwidth Broker transitioned from Phase 1 to Phase 2 in the early stages of our experiment. At that point, it started distributing local requests to both the secondary Bandwidth Brokers, which allowed the central Bandwidth Broker to remain within the specified threshold for the computation time. It can also be observed that our distribution scheme offloads some of the computation burden to the secondary Bandwidth Brokers, but avoids placing a significant burden on them, because during Phase 2 the central Bandwidth Broker still handles a significant number of requests. This property of our proposal offers a double advantage over most of the alternative architectures:

- It distributes the processing load and therefore the architecture is more scalable and can cope with more large-scale environments than the single entity architectures
- The central Bandwidth Broker entity is utilized at the maximum of its capabilities, therefore reducing the network communication overhead compared to the distributed architectures.

For our final simulation we used a larger topology with 5 secondary Bandwidth Brokers and an increased rate of reservation requests. As the results in Figure 5 show, the load could easily be handled by the Bandwidth Broker entities. In practice, such large-scale domains would typically have a number of servers available for running the secondary Bandwidth Broker module analogous to their size. This experiment demonstrates that

in such cases, these servers are utilized by our model, allowing the architecture to effectively scale.

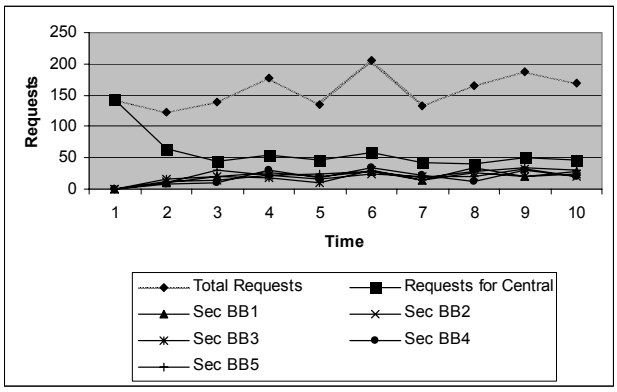

Figure 5: Large-scale topology

Because of the large number of requests, the central Bandwidth Broker was initially overwhelmed. However, it quickly transitioned from Phase 1 to Phase 3 and distributed the requests to the secondary Bandwidth Brokers, and therefore managed to retain a low processing overhead throughout the rest of the experiment, while each secondary Bandwidth Broker was able to deal with its limited admission set.

5. Conclusions - Future work

A distributed architecture can offer a number of advantages over the centralized approach. Our proposal, combined with an adaptive admission control scheme can make use of existing resources in order to both improve the efficiency of the admission decisions and keep the processing requirements low. Furthermore, the distribution of Bandwidth Broker components helps the network deal with a number of failures of varying degree of seriousness.

Our future work will 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.

6. References

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

[2] D. Durham, J. Boyle, R. Cohen, S. Herzog, R. Rajan, A. Sastry, RFC 2748, "The COPS (Common Open Policy Service) Protocol", January 2000

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

[4] Z. Zhang, Z. Duan, Y. Hou, "On Scalable Design of Bandwidth Brokers", IEICE Trans. Communications, Vol. E84-B, No.8 August 2001

[5] L. Burchard, H. Heiss, "Performance Evaluation of Data Structures for Admission Control in Bandwidth Brokers", Technical Report No. 2002-12, TU Berlin, April 2002

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

[7] "QBone Bandwidth Broker Architecture", QBone Signaling Design Team, http://qbone.internet2.edu/bb/bboutline2.html

[8] "Bandwidth Broker Implementation", Information and Technology Telecommunication Centre, University of Kansas, http://www.ittc.ukans.edu/~kdrao/BB/

[9] T. Braun, G. Stattenberger, "Performance of a Bandwidth Broker for DiffServ Networks", Kommunikation in verteilten Systemen (KiVS03), Leipzig, Germany, March 25-28, 2003

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