INTERNATIONAL JOURNAL OF NETWORK MANAGEMENT
Int. J. Network Mgmt 2009; **19**: 101–117
Published online 12 May 2008 in Wiley InterScience
(www.interscience.wiley.com) DOI: 10.1002/nem.694

Architectures and performance evaluation of bandwidth brokers

Christos Bouras[1,2,*,†] and Dimitris Primpas[1,2]

[1]*Research Academic Computer Technology Institute, University of Patras Campus, Patras, Greece*
[2]*Department of Computer Engineering and Informatics, University of Patras, Rion, Patras, Greece*

SUMMARY

DiffServ is the basis of contemporary QoS-enabled networks. Setting up DiffServ QoS requires extensive engineering effort in dimensioning and provisioning, especially for adjacent networks under different administrations linked in a 'federated' hierarchy. The bandwidth broker is an entity that is responsible for the management of the resources and the QoS service operation in an automated way. In this paper, we present, test and compare two different architectures of bandwidth brokers: a centralized one and a distributed one. We also deal with the inter-domain operation of the bandwidth broker in order to perform end-to-end provisioning. The paper presents the relevant aspects for inter-domain operation of a bandwidth broker and focuses on pathfinding issues. We discuss two models for inter-domain routing through bandwidth brokers, analyzing their advantages and comparing them. Copyright © 2008 John Wiley & Sons, Ltd.

1. INTRODUCTION

Recently, a number of Quality of Service (QoS) architectures have been proposed aimed at improving the performance of network applications that until now have always been handled by the traditional best-effort service. The most widely used architecture is DiffServ [1], which minimizes the number of actions to be performed on every packet at each node and builds a configuration that, unlike the alternative IntServ architecture, does not use a signalling protocol. Individual DiffServ mechanisms are applied to traffic aggregates rather than individual flows. The operation of the DiffServ architecture is based on several mechanisms. The first mechanism is the classifier that tries to classify the whole traffic into aggregates of flows (traffic classes) and marks them (packet-marking mechanism) mainly using the DSCP field (Differentiated Service CodePoint [1,2]). This field exists in both the IPv4 and IPv6 packet headers, as part of the Type of Service (ToS) field and as part of the Traffic Class field, respectively. The operation of services based on DiffServ architecture also uses several additional mechanisms (packet metering and shaping) that act on every aggregate of flows. In addition, in order to provide QoS guarantees it is necessary to properly configure the queue management and the time routing/scheduling mechanism that finally provide the appropriate 'network treatment' to the incoming packets. The most common DiffServ-based QoS service nowadays is called IP Premium [3]. It classifies the packets using the DSCP values for admitted and downgraded packets. Policing is performed at the edge of the network and high-priority queuing is applied in the core and access routers at the outgoing interfaces. Therefore, under appropriate network dimensioning and admission control, it provides guaranteed bandwidth and low-latency characteristics.

The most critical issue in the deployment of a QoS service is the ability to provision the network and provide the service on an end-to-end basis. Otherwise, the deployment is quite complicated, as many

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

Received 30 November 2006
Revised 31 January 2008
Accepted 10 February 2008

Copyright © 2008 John Wiley & Sons, Ltd.

parties should work and also there are many issues that can finally degrade the overall performance that the applications experience. In this direction, the automatic provisioning and management of a network and ideally all the connected networks through pre-agreed interfaces is the next big challenge. This goal can be achieved through bandwidth brokers. A bandwidth broker (BB) [2,4] is an entity that manages the resources within a specific DiffServ domain by controlling the network load and by accepting or rejecting QoS service requests. Every user (service operator) who is willing to use an amount of network resources, between its node and a destination, sends a request to the BB. For requests that span multiple domains (inter-domain requests), the BB will have to communicate with other BBs in the adjacent domains that are traversed by the requested flow. BBs only need to establish relationships of limited trust with their peers in adjacent domains, unlike schemes that require the setting of flow specifications in routers throughout an end-to-end path (see Figure 1). Therefore, the bandwidth broker architecture archieves to keep state on an administrative domain basis, rather than at every router while the DiffServ architecture makes possible to confine per aggregation of flows state to just the edges routers.

Bandwidth brokers are an intensely studied field and a number of architectures (distributed and centralized), evaluations and surveys have been proposed [5–8,9,10]. Also, extensive research has been conducted on provisioning and resource reservation issues, through bandwidth brokers [11–15]. Researches focus mainly on admission control issues, security as well as technology and vendor independency. Finally, all modern research—academic networks across the world (Internet2 and Geant [6,15])—is already studying and developing such systems, aiming at introducing them in their production network the following years.

A BB contains several modules that are necessary for its operation:

- An inter-domain interface—this is used for communication with adjacent BBs.
- An intra-domain interface—this is used for communication with the service components located inside the domain that the BB controls.

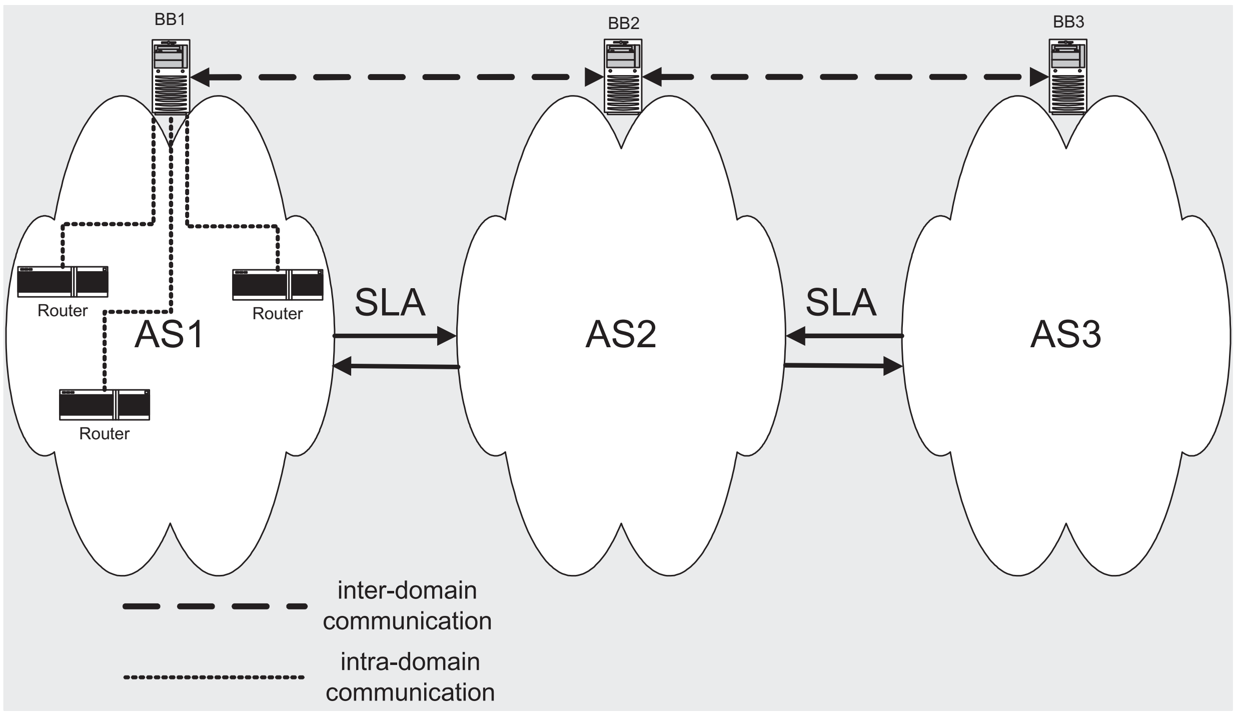


Figure 1. A bandwidth broker controlling a network

- A Network Information System interface—this is used so that the BB knows the information regarding the network topology and operation.
- A user/application interface—the scope of this interface is to allow the user and applications to send requests to the BB.
- A network and policy manager interface—this interface allows implementation of network management, admission control and network provisioning and monitoring.

When a user is willing to use an amount of network resources, he sends a Resource Allocation Request (RAR) to the bandwidth broker of his local domain. The BB receives the RAR, examines whether the requested resources are available and also whether the Service Level Agreement (SLA) requirements are fulfilled (this operation is performed by the admission control module) and sends a Request Allocation Answer (RAA) back to the user informing him whether the RAR was accepted or not. If the RAA contains a positive answer, the network resources that were requested should be reserved. In order to achieve this, the BB uses the intra-domain interface to configure the network elements in order to apply a specified per-hop behaviour, according to the QoS service that the network provides.

The rest of the paper is organized as follows. Section 2 discusses the basic modules of a BB and the implementation of two architectural models in a simulation environment. Section 3 describes a set of tests that aim to validate the operation of the BB and evaluate the performance of the two models. Section 4 discusses issues regarding the inter-domain operation of a BB and presents some simulation results. Finally, Section 5 describes the conclusions and Section 6 is devoted to future work.

2. BANDWIDTH BROKER IMPLEMENTATION

In order to study the operation of the BBs and the possible choices in terms of provisioning, deployment, network management and performance, we have used the ns-2 simulation environment [16] and implemented three variations of BB architectural models [17,18]. These models have been used in order to extensively test a number of aspects of their operation and isolate the architectural choices that influence in a specific way the efficiency of each model.

The three architectural models that have been developed, implemented and tested in the ns-2 simulator environment are the following:

- parallel distributed BB model (PDBB model);
- centralized BB model (CBB model);
- centralized fault-tolerant bandwidth broker model (CFBB model).

In every implementation, two kinds of agents exist: the edge BB (BBedge agent) and the base BB (BBbase agent). A BBedge agent is a module located at each node (router) of the network that represents a client interface (to user or application) which receives and forwards requests for usage of QoS service by traffic with a specific profile for a specific time period. Such a request is received by the BBbase agent, which represents the main BB server, and added as a request with specific parameters (sender, end node, bandwidth, time limit and status, which at this stage is set to pending) at a local database. When the processing of the request finishes, the status request changes according to the result of the processing, either to 'satisfied' or to 'rejected'. The answer is then sent back to the BBedge agent.

Another common element of the BBbase agents for all models is the request buffer. The BBbase agents process one request at a time, so if two requests arrive close together the first one is served and the other one is added to a buffer that exists at the BBbase agents. If a request arrives and the buffer is empty, the request is immediately processed. Otherwise, if the buffer contains previous requests but is not yet full, the request is added at the end of the buffer to be processed in the future. Finally, if the buffer is full, the request is rejected. The length of the buffer is configurable and is a point of investigation for the efficient operation of a BB.

Although the structure of the BBedge agents is the same for all models that have been implemented, each one operates in its own fashion when various packets are received during the simulation. The differences between the architectures have mainly to do with the structure and provisioning model. The different models use different methods for network provisioning and for processing requests (admission control).

The PDBB model keeps information about the state of each link (QoS parameters for existing requests, link dimensioning parameters) and node at the corresponding BBedge agent (which is located on this node). Therefore, the provisioning information is kept locally on the node's agent and is processed through this agent. Conversely, the CBB model stores this information at the BBbase agent and the CFBB model stores the information both at the BBedge agents and at the BBbase agent. The format of this information is as follows. Consider, for example, node 1 that is connected with node 2 and node 2 is connected with node 1. The relevant information for node 1 is that 'the available bandwidth from node 1 to node 2 is b for the time period $t2 - t1$', where $t2$ and $t1$ are points in time, and b is the bandwidth metric that arises from dimensioning of the network and the service. The relevant information for node 2 is that 'the available bandwidth from node 2 to node 1 is b' for the time period $t'2 - t'1$', where $t'2$ and $t'1$ are points in time, and b' is the bandwidth metric that also arises from dimensioning of the network and the service. In our simulation model we make the assumption that b' equals b, $t'82$ equals $t2$ and $t'1$ equals $t1$ between two nodes; in other words, that dimensioning, requests and reservations are bidirectional and symmetric. For the CBB, the BBbase agent's local database has, per each node it manages, complete information of the available bandwidth on every link.

In each one of the implemented models, we use a different communication protocol in order to complete the processing of requests. Communication between a BBbase agent and a BBedge agent is achieved with the use of messages that are always sent either from a BBedge to a BBbase or from a BBbase to a BBedge agent. Two BBedge agents never communicate by sending messages to each other, since the BBbase agent always intervenes in the communication.

For the PDBB model, processing takes place as follows. In this model, all the information about the status of links regarding available bandwidth is stored locally at the BBedge agents, as has already been mentioned. At the beginning, the BBedge agent that received the request forwards it to the BBbase. Then, the BBbase agent querying the network information system retrieves the routing path. Next, the admission control algorithm, which runs in a distributed manner, is executed. In particular, the admission control algorithm requires that the BBbase agent sends a packet to each of the nodes (BBedge agents) that are located on the routing path, querying them for bandwidth at a particular time period. If it receives even a single negative answer, it does not wait for any further response by BBedge agents, but rather it stops the request process, sends a negative answer to the request sender and moves on to another request. If the BBbase agent keeps getting positive answers from the nodes, it waits until all the nodes respond to its query. If all the nodes have answered positively, it sends a positive answer to the request sender and a packet to each node on the request path, in order to update their information about bandwidth availability. After that, the BBbase moves on to process the next request (if any). A positive answer means that the BBbase agent allows the request sender to use the requested bandwidth by guaranteeing the appropriate resource reservation. The PDBB model is fully fault tolerant as, if the BBbase is unreachable (due to system or network failure), the BB system can continue working simply by introducing a new BBbase agent (one of the active BBedge agents become BBbase).

The CBB model is a centralized model compared to the distributed nature of previous models. The BBbase agent stores all information about the status and availability of the links at a local database, which it consults in order to process a request. The response time is therefore reduced by eliminating much of the network communication, but the CBB model also has reduced resiliency because of its dependence on a single node.

The CFBB model combines the advantages of the CBB model and adds some information redundancy for the purpose of increased resilience to node failures. Data regarding the link status and availability is stored both at BBedge and BBbase agents. An incoming request is processed exactly as in the CBB model, but upon a positive answer the BBbase agent not only updates its local database but also sends a packet to each node on the request path to update their relevant information. This information is not retrieved during the request processing, but it can be used in the case of a failure at the node hosting the BBbase agent. Another node can then run a new BBbase agent that is brought up to date by all the BBedge agents (each sending a single message to the new BBbase agent with the relevant information) about the availability of bandwidth and the current reservations. The trade-off for the CFBB model is the generation of some additional network overhead over the CBB model.

The implemented BB models provide a QoS service with the characteristics of bandwidth guarantee as well as minimum delay and jitter. This service is the IP Premium, which follows the classic DiffServ architecture [3]. It classifies the packets using the DSCP values for admitted and downgraded packets. Policing is performed at the edge of the network and high-priority queuing is applied in the core and access routers at the outgoing interfaces.

The original ns-2 functionality supports packet classification at the edge routers using the source–destination pair of the IP header. We have already enhanced the simulator so that the classification is done using the DSCP field of the IP header [17]. This enables packets that have the same source and destination nodes but belong to different applications to belong to different classes as well, and packets with different source and destination nodes to belong to the same class.

The QoS mechanisms at the edge of the network have the responsibility of packet classification and policing. If the BBedge agent receives a positive answer about a request it has submitted, it configures all the relevant edges that it manages, through the network management interface. After the configuration process has been completed, the source can start using the requested and allocated network resources.

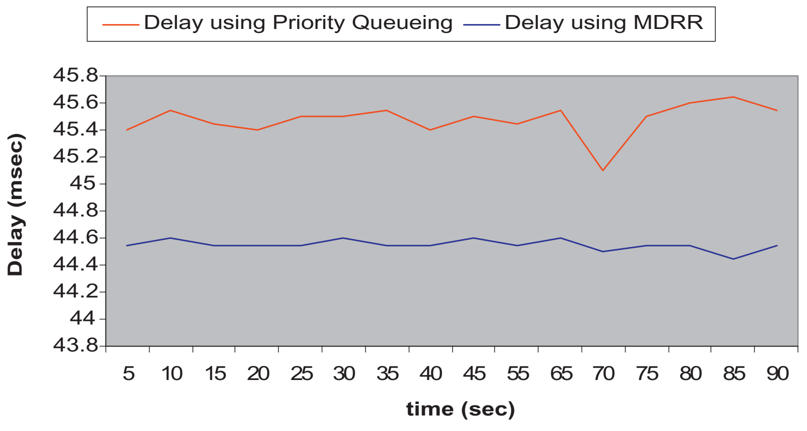
The QoS service, as it has been implemented, classifies the packets that have been admitted with DSCP value 46. Then, when the packets are inserted in the network, a strict token bucket policy is applied in order to be sure that the transmitted rate agrees with the admitted rate (the exceeded packets are remarked to DSCP value 0). Next, on all the network nodes, the queue management mechanism is appropriately configured, using a high-priority queue that takes advantage of the network's dimensioning to provide guaranteed bandwidth and minimum delay and jitter. Other traffic is treated as best-effort using the default queue (FCFS).

3. TESTING OF THE IMPLEMENTED BANDWIDTH BROKER

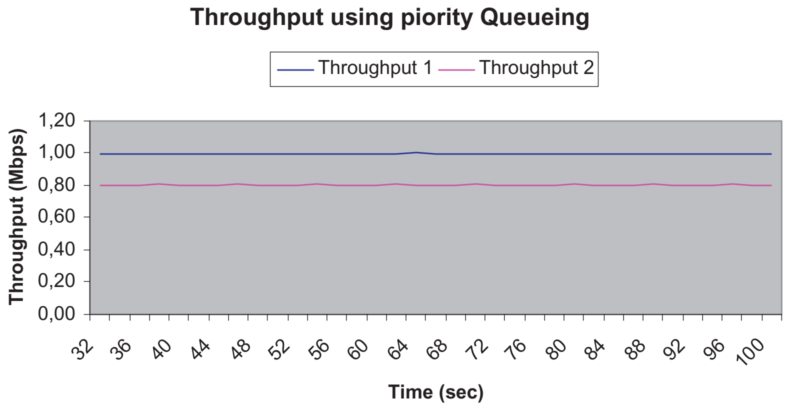
3.1 Testing and validating the QoS service

The BB, as mentioned above, has been configured to manage the IP Premium QoS service. We tested the service using Priority Queuing and the Modified Deficit Round Robin (MDRR) in strict priority mode as the queue-scheduling mechanisms. Priority Queuing is a mechanism that can provide prioritization using a single priority queue while the other packets are served with default queuing (first come, first served) when the priority queue is idle. The MDRR is a newer queue-scheduling algorithm that allows the configuration of several queues. In strict priority mode, MDRR can provide absolute prioritization to the packets of a single queue, while the other queues are managed in round robin manner only when the priority queue is empty. The duration of each queue's serving time is configured by specific attributes (called quantum and deficit).

We performed validation tests of the BB-implemented models, where two sources requested 1Mbps and 800 kbps premium traffic, respectively. Comparing the results from the experiments (Figure 2), it is obvious that the BB manages the IP Premium service very well, either with the MDRR or the Priority Queuing mechanism. The throughput of the admitted flows was ideal and the delay was extremely low in both Priority Queuing and MDRR cases. The only noticeable difference was that the delay was a little bit smaller when the IP Premium service was provided using the MDRR mechanism. Finally, as the MDRR mechanism seems more powerful and provides a smaller delay than Priority Queuing, we decided to use the MDRR mechanism in the next simulations.



Delay (in msec)



Overall throughput

Figure 2. Throughput and delay using the IP Premium service

3.2 *Testing setup*

The performance of the BB models was compared in a number of different network topologies using several criteria and metrics. An important metric is the response time (the time from the moment a request is submitted to the BBbase agent until the moment the BBbase agent responds either by accepting or rejecting it) and the network overhead caused by the control messages exchanged. Requests in our experiments were randomly generated in an ns-2 simulator. The parameters for each request were also randomly produced within suitable boundaries (regarding the total duration of each simulation, total available bandwidth, minimum and maximum reservation requests) for each situation that we wanted to simulate. For the experiments, four different topologies were used.

3.3 *Distributed versus centralized*

For the main set of experiments we used a number of different topologies (Figure 3) in order to obtain more accurate results. The tests aim at measuring and comparing the three implemented models, using as metrics the network overhead, the response time and the acceptance rate.

In Table 1 we have summarized the network overhead caused by each BB model for each topology, measured by the average number of packets exchanged per request.

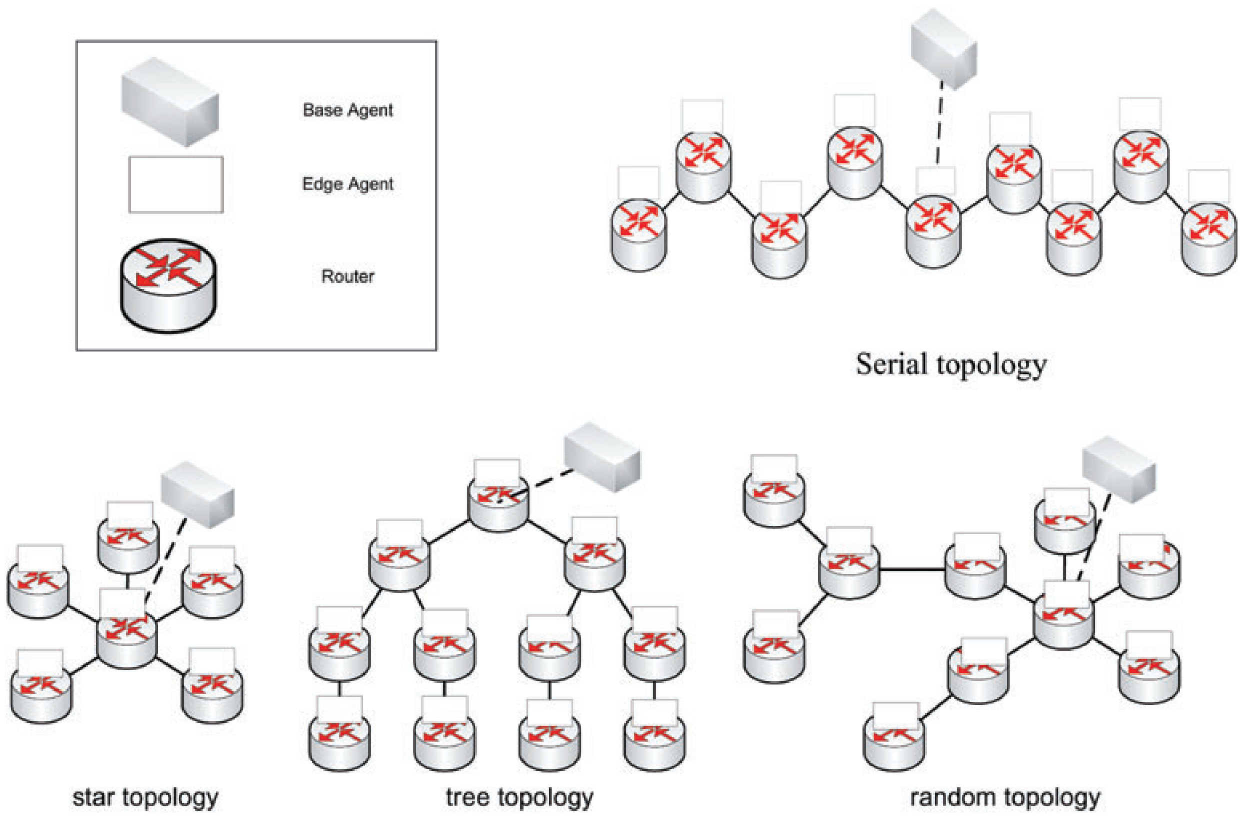


Figure 3. The topologies in simulation tests

Model/topology	Serial	Star	Tree	Random
PDBB	9.68	4.92	9.92	6.61
CBB	2.01	1.70	1.94	1.90
CFBB	3.66	2.44	3.63	3.01

Table 1. Network overhead (average number of packets per request)

Because of the distributed nature, there is much more network overhead for the PDBB compared to the CBB model, especially for the serial and tree topologies. This difference is significantly reduced for the star topology, because the star topology fits better to the implementation algorithm of the PDBB models. CFBB introduce move overhead in network comparing to CBB and in all cases less than PDBB.

In addition, these experiments show that the performance of the PDBB model is affected by network topology and location of the BBbase agent. The distribution of the QoS requests across the network nodes also plays a key role. Therefore, the PDBB model requires an optimal location of the main BBbase agent. Generally, the optimal location of applications is a well-known problem [19] and in the case of the distributed BB it can be addressed by studying the topology and therefore applying periodic optimal selection of host node for the BBbase agent. In Section 3.4.2 we present a relevant module.

The second set of tests aimed at measuring the acceptance rate per model for each topology, together with the results, are presented in Table 2. All models benefit from the star topology and produce better

Model/topology	Serial	Star	Tree	Random
PDBB	0.2029	0.2227	0.2171	0.2130
CBB	0.2035	0.2259	0.2088	0.2106
CFBB	0.2060	0.2323	0.2083	0.2118

Table 2. Acceptance rate (ratio of accepted to submitted requests)

Model \ Topology	Serial	Star	Tree	Random
PDBB	22.19	2.520	9.382	6.041
CBB	5.502	1.597	4.097	2.879
CFBB	5.502	1.597	4.097	2.879

Table 3. Response time (average time passed until the answer returns to the request sender) (10^{-3} s)

Model \ Topology	Serial	Star	Tree	Random
PDBB	19.83	1.485	6.404	4.726
CBB	2.576	0.8128	2.176	1.770
CFBB	2.576	0.8128	2.176	1.770

Table 4. Standard deviation of the response time (10^{-3})

results. Because of the lack of potential bottleneck links, the star topology allows the largest percentage of requests to be accepted. Since in general the topology is typically fixed and the only realistic choice is between the BB models, Table 2 suggests that for a random topology all models display similar behaviour, which hints that this is probably the best behaviour one can expect, short of using more sophisticated admission control algorithms or more sophisticated positioning of the BBbase agent.

Finally, we measured the average response time for all models, and Table 3 presents the results. We have also calculated the standard deviations for the response times, which are presented in Table 4. In all cases, the CBB and CFBB models are identical, as expected, since their request processing procedures are exactly the same.

The close proximity of the nodes to the BBbase agent in the star topology favour the PDBB distributed model, which closes the gap with the CBB/CFBB centralized models. But also for the rest of the topologies the response time remains within two to four times that for the CBB/CFBB models, which can be a reasonable trade-off for the distributed advantages of the PDBB model.

3.4 Discussion

3.4.1 Optimization of bandwidth broker

Examining the admission control algorithm, a possible drawback is that the decisions concerning a request that have been taken at a given time include the routing path. The latter can cause some rejections of requests, when the network resources are not reserved in a balanced way (using alternative paths too) in order to allow free resources in all links where possible. This problem can be solved by running an optimization algorithm when the network approaches such unbalanced conditions (for example, when the reserved resources on a link exceed a specific threshold: 80% of the available resources). This optimization algorithm can run periodically, reconfigure some admitted requests using alternative paths with the same guarantees and examine again the rejected requests. Such an optimization algorithm should use traffic engineering characteristics with the use of MPLS and other advanced mechanisms that have been proposed [20,21].

3.4.2 A host selection model

Another important point in the operation of the distributed bandwidth broker is to decide which node should host the BBbase agent. Many research teams have addressed the issue of optimal location of critical applications in a network [19]. Regarding the BB, this problem is more complicated due to the fact that its usage is not standard and depends on user premises.

We tried to approach this problem by examining a host selection model that evaluates the nodes and adjacent links and tries to find the best node to locate the base BB. In other words, the problem is to find the root of a graph, where the root is the most important node in the network, and therefore most of the packets for the operation of the BB will reach it quickly. The proposed model is divided into two phases that should run sequentially.

Phase 1

Phase 1 tries to evaluate the importance of each node in the network and finally assigns a weight to all nodes. The evaluation is based on the following criteria:

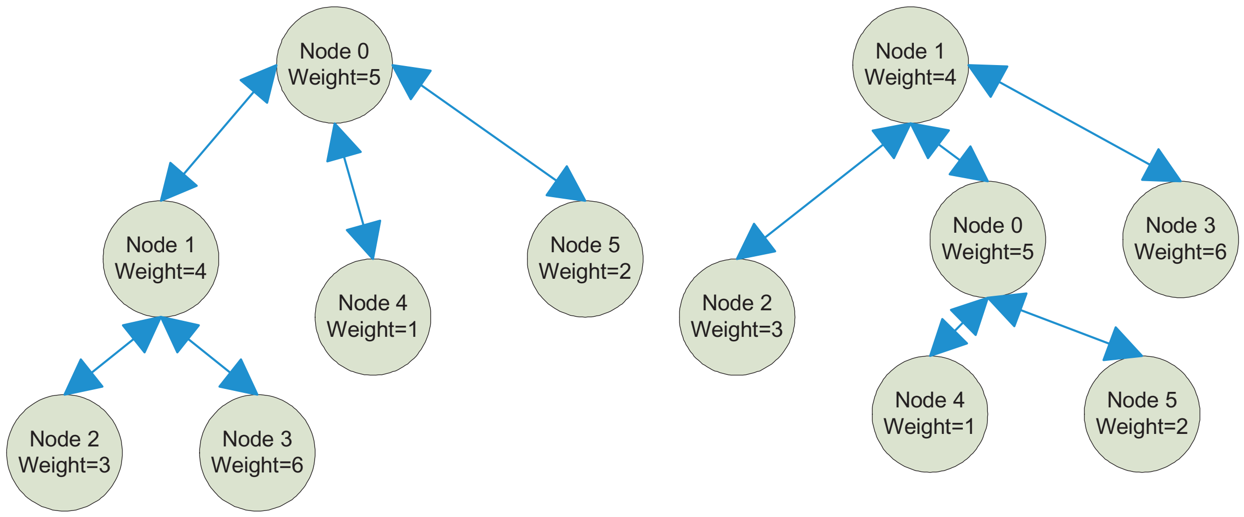
- the number of access interfaces in this node and the corresponding traffic that these interfaces insert in the network;
- the maturity and capabilities of the equipment of the node in order to handle the traffic;
- the role of the node in the routing schema. This criterion is related to the network topology and means the importance of the node in the operation of the network (for example, in a star topology, the root is a critical node);
- finally, the interconnection point—a node has an important role if it is the interconnection point with a bigger backbone network and comprises a hierarchical federation.

Phase 2

In phase 2, for each node we create the 'routing' graph, which means that we place each node as root and create all the paths to all the other nodes, using the current network's routing scheme. Therefore, there are N graphs (where N is the number of nodes in the network) that should be examined. Then, for every graph, we calculate the total cost, which is formulated as the sum of the cost for every node to communicate with the root. This communication cost arises as the weight of this node multiplied by its depth in the graph. Finally, the problem is to find the graph that has the minimum total cost.

Next we tried to simulate the above model and measure its accuracy. The idea of the simulation was to find the two best locations for the base BB in a given network topology (see Figure 4) and in these two cases to make random requests between nodes according to their weights. We measured the execution time, with the assumption that it is affected mainly by the required packet transmissions. First, we calculated the percentage of host selection success, as determined by the execution time (thus we measured for every request whether the selected host or the alternative (second option) gave the better performance). The simulation test was of 50 iterations of 100 random requests and the results show that the success rate (in comparison with its major candidate host) was between 90% and 96%.

Next, we tried to identify how elastic the model is according to the criteria. We used 10%, 20% and 30% variation in nodes' weights (as declared in phase 1) and we compared the operation when BBbase agent is host in the best location (provided by model) than its major candidate (second best location by model). The set of tests was of 30 iterations of 100 random requests. In this case, we noticed that the average reduction of the measured execution time is 8.2%, 22% and 26% for weights' variation 10%, 20% and 30% respectively." Also, Figure 5 shows the distribution of the reduction in execution time. Therefore, this model seems to improve the performance of the PDDB model, while it is tolerant to weight calculation. In summary, it can be combined with the PDBB model, relocating the host node periodically, in order to allow the model to achieve its best performance every time.



Node 0 “special-weight” = 0
Node 1 “special-weight” = weight*depth = 4*1=4
Node 2 “special-weight” = weight*depth = 3*2=6
Node 3 “special-weight” = weight*depth = 6*2=12
Node 4 “special-weight” = weight*depth = 1*1=1
Node 5 “special-weight” = weight*depth = 2*1=2
Graph’s “special-weight” = SUM (Node “special-weight”) = 0+4+6+12+1+2 = 25

Node 0 “special-weight” = weight*depth = 5*1=5
Node 1 “special-weight” = 0
Node 2 “special-weight” = weight*depth = 3*1=3
Node 3 “special-weight” = weight*depth = 6*1=6
Node 4 “special-weight” = weight*depth = 1*2=2
Node 5 “special-weight” = weight*depth = 2*2=4
Graph’s “special-weight” = SUM (Node “special-weight”) = 5+0+3+6+2+4 = 20

Figure 4. The best two locations for BBbase hosting

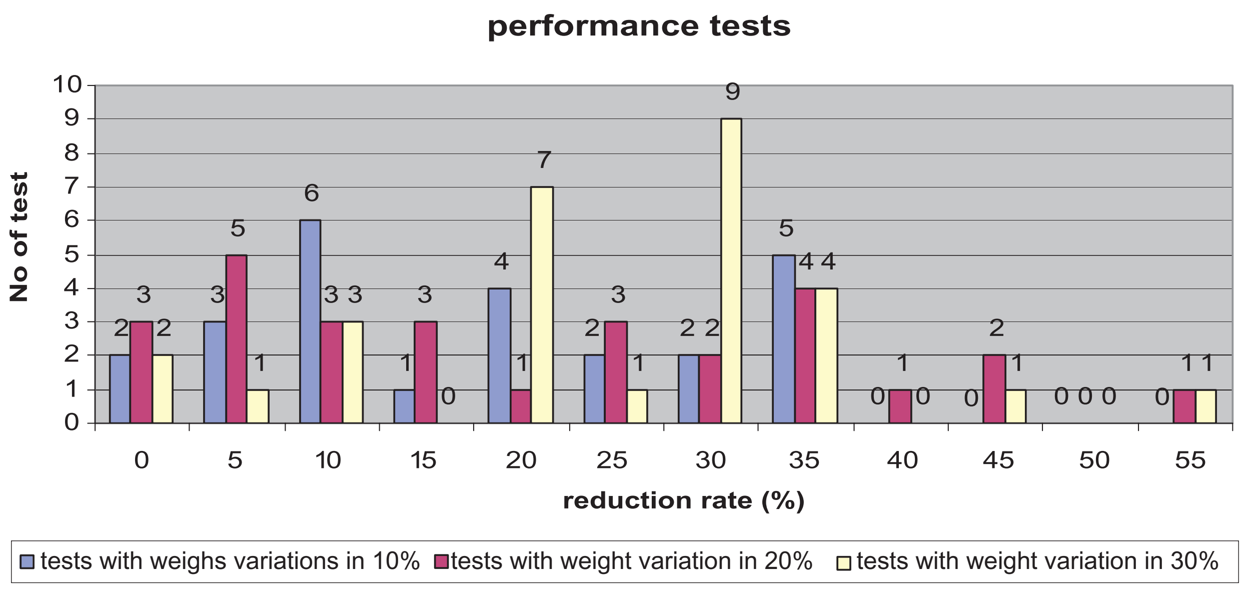


Figure 5. Distribution of the reduction in execution time between the optimal selection and the major candidate

4. INTER-DOMAIN OPERATION

A bandwidth broker should be able to communicate with adjacent domains that work in a hierarchy or peer connection, in order to serve requests for end-to-end QoS services.

In general, when independently managed networks are interconnected, additional difficulties exist in ensuring interoperability of DiffServ-based QoS across network boundaries. This involves (a) adopting interoperable conventions about DiffServ traffic classes and the respective PHBs, (b) adopting interoperable dimensioning and provisioning mechanisms and (c) linking together functions of the two domains, such as provisioning, policing and admission control functions. During the last few years, several research teams and research projects have worked on this area [3,6,14,15,22,23]. Assuming that compatible DiffServ-based QoS service definitions and mechanisms exist in all domains, we focus and study the issue of service provisioning across adjacent boundaries and the issue of advanced admission control.

Provisioning in a topology with independently managed domains requires interoperation between the various domains and the management infrastructure each domain has developed. This interoperation between such heterogeneous applications is ideally achieved by the use of industry standards such as XML [24] and the Web Services framework [25,26]. The most common and promising approach [15] is the use of cooperating agents, each managing provisioning within a domain network ('intra-domain'), while all agents communicate to implement inter-domain provisioning requests. In particular, specific entities called inter-domain managers (IDMs) are used in each network ensuring provisioning of the end-to-end service. These IDMs collaborate on a peer-to-peer basis over the topology and handle SLA and peering agreements too. The actual implementation in each domain is described in another entity called Domain manager. Every domain is free to implement its own DM (following its design principles and infrastructure capabilities) that should communicate with its IDM using a specific interface.

In this case, the most challenging issue is the actual admission control and investigation of the path (pathfinding) from the source domain to the destination one for every request. A BB should calculate the 'best' path that provides the QoS guarantees from source to destination through the intermediate domains, taking into account the admission control 'answers' from the intermediate domain as well as the possible SLAs between domain peerings. The outcome (the preferred path) should finally be given as input to traffic engineering mechanisms that will route the traffic of the specific request through this path. Many researchers have studied traffic engineering issues on intra-domain and inter-domain operation of bandwidth brokers [8,11,12,27]. In this section of the paper, we discuss two models that approach end-to-end traffic engineering (pathfinding) on inter-domain operation.

4.1 Pathfinding approaches

4.1.1 Centralized pathfinding model

According to the centralized pathfinding model, the decision concerning the routing of the request, in particular the domains and the internal paths that the traffic will traverse in order to reach the destination domain, is made by the source domain. In order to take this decision, a central provisioning system is necessary that will keep topology, peering (SLAs between ISPs) and technology information. In particular, the domains that take part in this model announce their topology and resource status to a common network information system (NIS) that is used for BB operation. All domains should always keep the relevant information in this system synchronized, otherwise the BB will work on inconsistent data, which can lead to incorrect paths and reservations. The implementation of such systems is an open issue and several approaches have been presented [11,12].

After a request has been submitted, the source domain makes 'queries', asking for the requested resources across the paths. It follows this procedure sequentially until it finds all paths from source to destination that have the requested resources. Next it decides which path is the best one, according to criteria that have been specified in the BB's policy. The criteria should be defined by the domains and should indicate the cost that they have in order to provide the resources.

This centralized pathfinding model can be parallelized with the operation of RSVP-Traffic Engineering protocol on a single managed domain [20]. The RSVP-TE on a single MPLS-enabled domain provisions the network resources and provides Label Switched Paths (that can use the resources) using various selection criteria in the case of many candidate paths. Generally it is close to the centralized pathfinding model but it is not applicable, as the inter-domain model traverses independently managed networks and therefore the end-to-end operation of an RSVP-TE is not possible. Also, the RSVP-TE operates only on the runtime of network devices and its operation can not be exported for usage on offline or complementary tools. Additionally, the same operation can be achieved through other proposed approaches [11], but they still have the limitation that the freedom of each domain is eliminated (since it announces its local policy) or require common network mechanisms that are not applicable on independently managed networks (and especially if networks use different vendors' equipment).

4.1.2 Distributed pathfinding model

The second model is 'peer to peer' and in this case the pathfinding service of the source domain forwards the requests that have destination to another domain to every adjacent domain, under the condition that there are available resources from the source to the respective egress points. Next, every domain receiving such a message checks if it is the destination domain or an intermediate one. In the case of intermediate domain, the BB server of the domain checks whether it can guarantee the requested resources from the ingress point to the egress point to all other domains it has a connection with. If there are the necessary resources, then the message is broadcast to the next domain. In order to avoid loops, when a domain receives a request it checks the combination of the sequence number of the request and the domain's ingress point that received it. If this combination has already been processed, the domain discards the request. When the destination domain is reached, the BB server of this domain checks for the requested resources from the ingress point where it received the message to the destination.

This procedure is repeated and finally all the possible routes between the source and destination are found and declared to the source domain. Then, the source domain decides the best routing according to specific criteria that have been specified in the pathfinding model. In the case where there are not any paths with the requested resources due to SLA restrictions, the model may request some SLA renegotiation between the domains and take a final decision. Also, when that the destination is unreachable due to network failure, the module may find itself in a deadlock, as there are no return paths, or even rejected due to insufficient resources. Therefore, the module should have an expiration timer and, in the case that this period expires, the module assumes that the destination is unreachable.

The complexity of this distributed pathfinding module is quite large and its performance depends on the current topology between the domains where the module is applied. Actually, this problem relies on Breadth First Search (BFS) on a graph that it is formatted in every request using as root the source domain [28]. Time complexity of the BFS algorithm is proportional to the number of vertices plus the number of edges in the graphs it traverses.

Additionally, a very important issue in the whole operation of the distributed model is the definition of criteria that should be applied on the selected paths (if the model provides more than one) in order to decide the best one. The proposed model uses a combination of criteria: minimum hops, minimum SLA fulfilment and the overall cost of the path. The overall cost is the summary of the costs from all the domains in the path. The cost of each domain is declared by the domain itself and indicates the actual cost to provide the resources. It has to be time-independent in order not to further complicate the path computation process. The aim of the formula is to decide the best path from the available ones while performing load balancing and low cost according to the domains' premises. The model, as described above, assumes symmetric SLA on transmit-and-receive direction between domains, but it also works when the SLAs are asymmetric.

In order to implement this model, each domain has the freedom to use any open-source or proprietary provisioning tool for its network. The only requirements are the implementation of the overall module that will provide the synchronization of the operation and also a common format of the data that each domain provides. Additionally, this format is proposed to be based on XML as the overall pathfinding module can

```
<pathfinding-message>
  <request-sequence-number>DATA</request-sequence-number>
  <domain>DATA</domain>
  <ingress_point>DATA</ingress_point>
  <outgress_point>DATA</outgress_point>
  <resources>DATA</resources>
  <cost>DATA<cost>
  <destination_reached>YES/NO</destination_reached>
  <other-domain-data>DATA</other-domain-data>
</pathfinding- message >
```

Figure 6. XML example message for pathfinding module

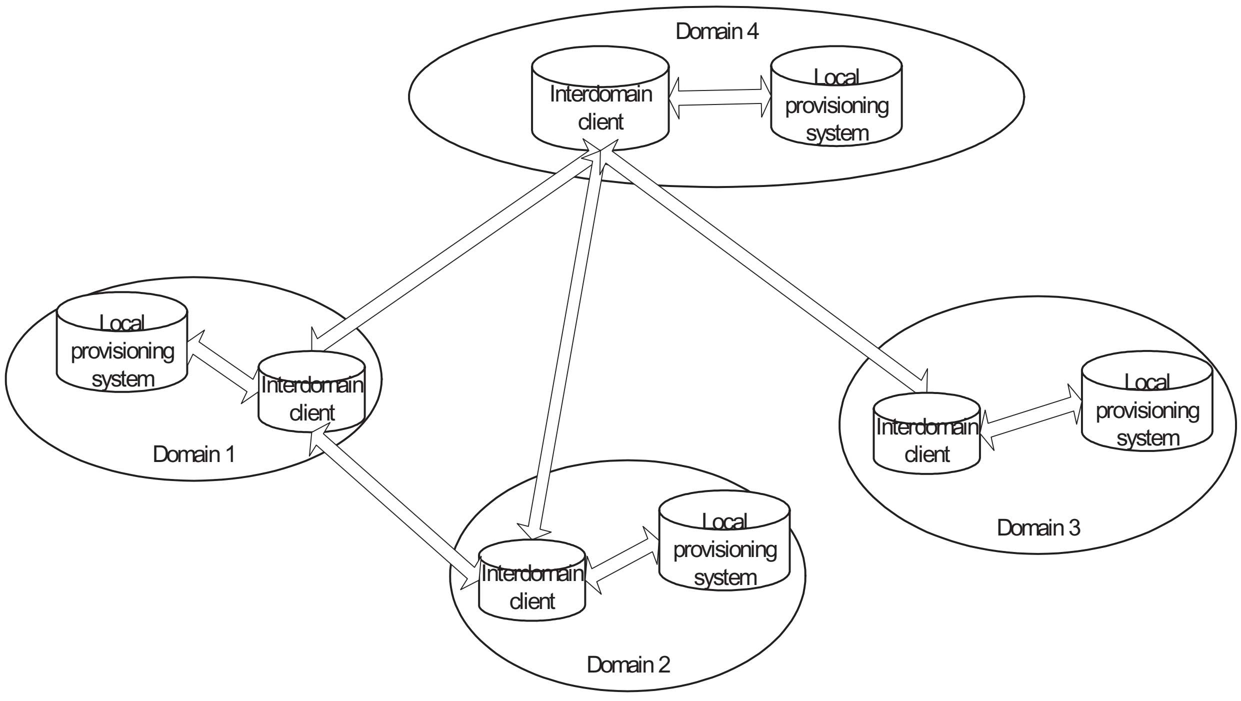


Figure 7. An inter-domain approach using distributed pathfinding

work using Web Services. An example XML message for the distributed pathfinding search is presented in Figure 6. In addition, Figure 7 describes the architecture of the model in the inter-domain environment.

4.1.3 Comparison

Both models (centralized and distributed) need updated information for the network's condition and its policy. The first algorithm requires every domain to announce that information centrally, where all the other domains 'query' the central system in order to process inter-domain requests. The second algorithm eliminates this announcement (of network's condition and operation), by forcing that only the bandwidth broker server of each domain has access to internal data and is responsible to answer about the possible paths with available resources in its domain. Therefore the operation of each network remains internal and every request that may pass through a domain is processed by domain's bandwidth broker only. On the other hand, the second algorithm has greater complexity and also needs more time to answer a request, as all the possible paths should be checked though a BFS-like search by asking domains' BBs. Therefore, the response time of the pathfinding module includes the transmission delay of the requests for pathfinding between the BBs of the domains and their execution time locally in each domain. The first algorithm has a big advantage in this issue, as all the information about all domains and their operation is stored locally in a database and therefore all the alternative routing paths can be found by searching the new graph that is produced by adding all the domains' topology, the available resources as well as the SLAs between adjacent domains.

In conclusion, the first algorithm has a better response time but needs to announce internal information periodically in order to keep the global 'provisioning system' updated. This requirement introduces a multitude of scaling and security problems, which make this solution in most cases infeasible. The second algorithm keeps that internal information hidden but the process of every request engages all the BBs of the involved domains. As this inter-domain operation should be deployed on independently managed domains, the second model (distributed) is more suitable due to the fact that it preserves the independence of the management and local policy in the domain.

4.2 Pathfinding simulation

In order to study the distributed pathfinding module further, we simulated its operation and executed the model using 170 random requests between domains in the three topologies presented in Figure 8.

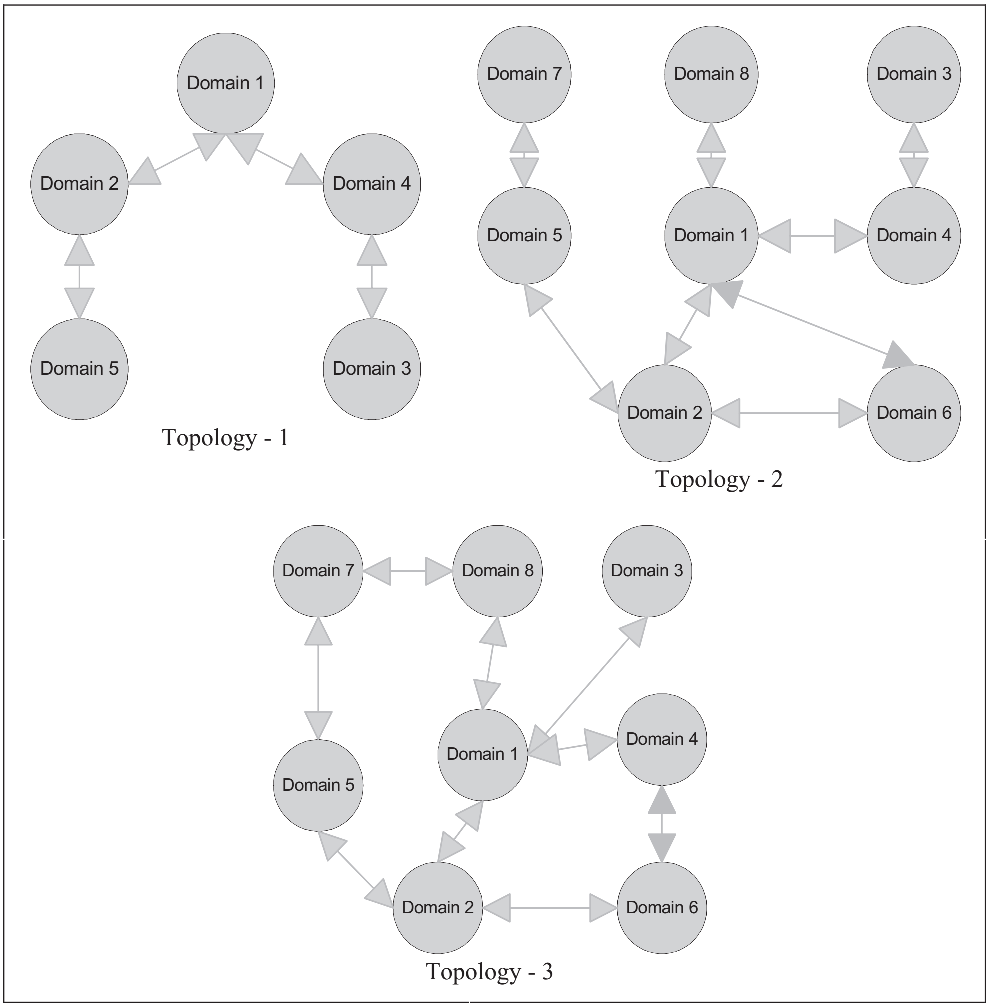


Figure 8. Simulated domain topologies

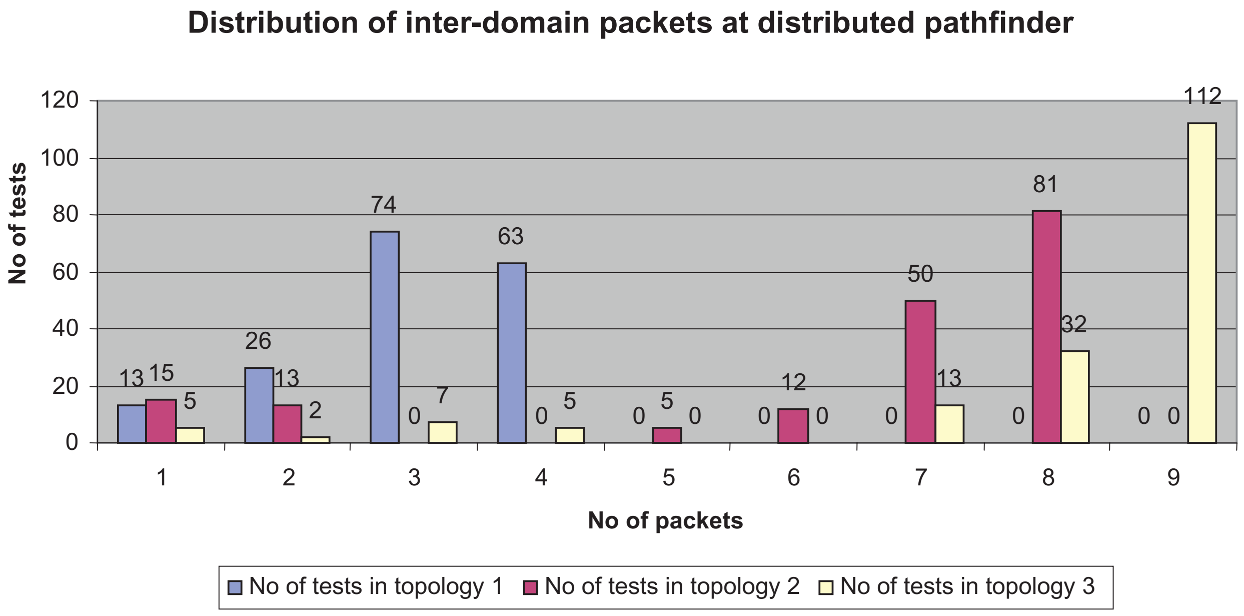


Figure 9. Distribution of exchanged packets for the pathfinding module

We tried to measure the exchanged 'inter-domain' packets per request, as they indicate the overall response time, using the assumption that the response time of each domain's BB is quite similar. According to the results (see Figure 9), the number of packets that the pathfinding module needs to exchange is quite small and depends on the overall topology and location in the topology of the source and destination domain on every request. The figure presents the exchanged packets between the domains and not the internal packets in each domain (if any), as it depends on the provisioning service that each domain has selected to use. After the pathfinding module, the algorithm returns the available paths that reach the destination domain (and their characteristics), waiting for the final decision about the preferred path that will be produced after setting up the pathfinding criteria. The overall request process will finish when the appropriate path has been selected and the relevant domains have been informed to reserve the resources and perform the necessary configuration on network devices.

Comparing the results for the needed packets (communication) for pathfinding in the three topologies, we notice that the average number of exchanged packets increases proportionally to the complexity of the topology (and in particular when it contains many links that lead to cycles). This result was expected due to the fact that the distributed pathfinding module is based on the BFS algorithm [28].

5. CONCLUSIONS

Summarizing the work that this paper presents, we tried to study and simulate the basic functionality of a BB that provides a DiffServ-based QoS service using two different architectures; a centralized one and a distributed one. The tests described above demonstrated first that the BB operates well in providing the requested resources. Additionally, we compared two different models, a centralized versus a distributed one, evaluating the performance of each model by measuring the exchanged packets as well as the average processing time of a new request on the BB. In small topologies, the centralized model had better performance, but the distributed one is more powerful in large topologies, especially if it is combined with a host selection model, like the one proposed in this paper.

This paper also deals with the inter-domain operation of BBs in order to perform end-to-end provisioning and therefore end-to-end guarantees through QoS services. Today's networks do not yet have automatic procedures to provide such functionality, but many approaches are under design or testing. The paper presents the relevant aspects for inter-domain operation of a BB and focuses on pathfinding issues. We describe two models that can be used (centralized and peer-to-peer), analyse and compare them. Also, we simulated the peer-to-peer model on three different topologies. The peer-to-peer model inserts a communication overhead as it exchanges many packets (the actual number depends on the topology and the location of the source and destination domain). But this overhead can be characterized as low because the exchanged data are small (the packets are of very small size). On the other hand, the peer-to-peer model suits commercial ISPs and academic networks better as it allows them to manage their network independently and interact through specific procedures based on standards (like Web Services and XML schemas) without announcing much information about internal topology and policy.

The inter-domain operation and approaches described in this paper are applicable to both IP QoS BBs as well as those that provide resources on an optical layer. It is obvious that the pathfinding module might be similar in both cases (if some L2 provisioning mechanisms such as MPLS-enabled core are used to support the QoS service) unlike the other components of the BB, such as provisioning, technology-based reservation and establishment of guaranteed connections, which are very different when implemented at L2 or L3.

6. FUTURE WORK

The main directions for future work are the extension of the implementation in NS-2 simulator and the setup of large-scale experiments. We intend to perform a number of large-scale simulation tests that will include the PDBB model, accompanied by the optimal host selection model running periodically. In addition, the extension of the implementation includes implementation of the inter-domain operation and its migration in the current implemented BB models. Also, we plan to study and implement other advanced provisioning models and admission control algorithms in order to perform comparison tests.

Finally, another direction for future work will be the investigation of SLA establishment and automatic negotiation between domains in the context of BBs that operate in a federated environment.

REFERENCES

1. Blake S, Black D, Carlson M, Davies E, Wang Z, Weiss W. An architecture for differentiated services. *RFC 2475*, December 1998.
2. Nichols K, Jacobson V, Zhang L. A two-bit differentiated services architecture for the Internet. *IETF RFC 2638*, July 1999.
3. SEQUIN. Service quality across independently managed networks. *IST Project IST-1999–20841*. http://local.dante.net/sequin/.
4. Vollbrecht J, Calhoun P, Farrell S, Gommans L, Gross G, de Bruijn B, de Laat C, Holdrege M, Spence D. AAA authorization application examples. *RFC 2905*, August 2000.
5. Hashmani M, Yoshida M. ENICOM's bandwidth broker. In *Saint 2001 Workshops*, San Diego, CA, 8–12 January 2001; 213–220.
6. QBone Bandwidth Broker Architecture, http://qbone.internet2.edu/bb/bboutline2.html [15 March 2008].
7. Ogawa J, Terzis A, Tsui S, Wang L, Zhang L. A prototype implementation of the two-tier architecture for differentiated services. *RTAS 99*, Vancouver, Canada.
8. 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, May 2002.
9. Zhang Z, Duan Z, Hou Y. On scalable design of bandwidth brokers. *IEICE Transactions on Communications* 2001; E84-B: 2011–2025.
10. Pagani E, Rossi GP. Distributed bandwidth broker for QoS multicast traffic. *Distributed Computing Systems* 2002; 319–326.

11. Mantar HA, Okumus I, Chapin SJ, Hwang J. A scalable model for inter-bandwidth-broker resource reservation and provisioning. *IEEE Journal on Selected Areas in Communications* 2004; **22**: 2019–2034.
12. Hwang J, Revuru R. Inter-domain Diffserv dynamic provisioning and interconnection peering study using bandwidth management point: a simulation evaluation. In *2003 International Conference on Information Systems and Engineering*, 2003.
13. 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.
14. Jaskola P, Malinowski K. Two methods of optimal bandwidth allocation in TCP/IP networks with QoS differentiation. In *2004 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS' 04)*, San Jose, CA, 25–29 July 2004.
15. GÉANT: Pan-European Academic Network. http://www.geant2.net [15 March 2008].
16. The Network Simulator: Building Ns. http://www.isi.edu/nsnam/ns/ns-build.html [15 March 2008].
17. Ceid DiffServ NS web site (2006). DiffServ Quality of Service in NS. http://ru6.cti.gr/diffserv-ns/default.htm [24 March 2006].
18 Bouras C, Pappas I, Primpas D, Stamos K. Using the ns-2 simulation environment to implement and evaluate bandwidth broker models. In *2nd Conference on Next Generation Internet Design and Engineering (NGI 2006)*, Valencia, 3–5 April 2006; 285–291.
19. Qiu L, Padmanabhan VN, Voelker GM. On the placement of Web server replicas. In *INFOCOM 2001: Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings, IEEE* Vol. 3, 22–26 April 2001; 1587–1596.
20. Awduche D, Berger L, Gan D, Li T, Srinivasan V, Swallow G. RSVP-TE: extensions to RSVP for LSP tunnels. *IETF RFC 3209*, December 2001.
21. Gojmerac I, Ziegler T, Ricciato F, Reichl P. Adaptive multipath routing for dynamic traffic engineering. In *IEEE Globecom*, San Francisco, November 2003.
22. Trimintzios P, Andrikopoulos I, Pavlou G, Flegkas P, Griffin D, Georgatsos P, Goderis D, T'Joens Y, Georgiadis L, Jacquenet C, Egan R. A management and control architecture for providing IP differentiated services in MPLS-based networks. *IEEE Communications*, special issue on IP-Oriented Operations and Management 2001; **39**: 80–88.
23. EUQoS Project. http://www.euqos.eu/ [15 March 2008].
24. Yergeau F, Bray T, Paoli J, Sperberg-McQueen CM, Maler E. *Extensible Markup Language (XML) 1.0* (3rd edn). W3C, February 2004. http://www.w3.org/TR/2004/REC-xml-20040204/ [15 March 2008].
25. *SOAP/1.1 TR*. W3C, June 2003. http://www.w3.org/TR/soap/ [15 March 2008].
26. Christensen E, Curbera F, Meredith G, Weerawarana S. *Web Services Description Language (WSDL) 1.1*. W3C TR, March 2001. http://www.w3.org/TR/wsdl/.
27. Okumus I, Hwang J, Chapin SJ, Mantar H. Inter-domain traffic engineering on a bandwidth broker supported Diffserv Internet. In *Applied Telecommunication Symposium (ATS '03)*, part of the *Advanced Simulation Technologies Conference 2003 (ASTC'03)*.
28. Cormen TH, Leiserson CE, Rivest RL, Stein C. Depth-first search. In *Introduction to Algorithms* (2nd edn). MIT Press/McGraw-Hill: Cambridge, MA, 2001; 540–549.

AUTHORS' BIOGRAPHIES

Christos Bouras obtained his Diploma and PhD from the Department Of Computer Engineering and Informatics of Patras University (Greece). He is currently and Associate Professor in the above department and a scientific advisor of Research Unit 6 in Research Professor in the above department and a scientific advisor of Research Unit 6 in Research Academic Computer Technology Institute (CTI). He has published over 200 papers in various well-known refereed conferences and journals. He is a co-author of seven books in Greek. He has been a PC member and referee in various international journals and conferences and he has participated in numerous R&D projects. His research interests include Analysis of Performance of Networking and Computer Systems, Computer Networks and Protocols, Telematics and New Services, QoS and Pricing for Networks and Services.

Dimitris Primpas obtained his Diploma, MSc and PhD from the Computer Engineering and Informatics Department of Patras University (Greece). He works in the Research Unit 6 of CTI and on Telematics, Distributed Systems and Basic Services Laboratory of the Computer Engineering & Informatics Department, in Patras University and has participated in numerous R&D projects. His interests include: networks, protocols, Quality of Service and Network applications.