CLASS OF SERVICE CONCEPTS IN AUTONOMOUS SYSTEMS

Jean-Alexander Müller, Sven Hessler and Klaus Irmscher

Leipzig University, Department of Computer Science
Augustusplatz 10-11, 04109 Leipzig, Germany

e-mail: { jeanm | hessler | irmscher }@informatik.uni-leipzig.de

Abstract

The bandwidth broker architectures introduced to manage diffserv still have scalability flaws in the mean of large volumes of end-to-end tunnels and workload they have to handle. In addition they only provide solutions of inter-domain-management and do not consider the characteristics of intra-domain-management. The new signaling approach for intra-domain-management presented in this paper utilises existing relationships between service providers and service users, which yields to a significant reduction of the number of reservations and the workload of each bandwidth broker. The concept of this signaling proposal will be presented as well as promising test results as proof of concept. Finally further ways to reduce the workload using cache strategies are depicted.

1 Introduction

The Differentiated Services (diffserv) Architecture implements a scalable mechanism for quality-of-service (QoS) provisioning in the data path. The fundamental concept behind the diffserv idea is aggregation of packet flows in the data path. Packets are classified by a single mark (the differentiated services code point, DSCP) and assigned to a particular forwarding behavior. But diffserv itself can only provide different service classes with statistical guarantees[1]. Even if all *diffserv domains* in a *diffserv region* – a network of interconnected diffserv domains – set up their networks properly and established *service level agreements* (SLA) and associated *service level specifications* (SLS) with their neighbors for an expected traffic volume, the traffic volume generated by end-users could exceed these limits.

Bandwidth brokers (BB) have been proposed for access control and resource management to implement dynamic resource provisioning of differentiated services with hard end-to-end service guarantees. Most of the common approaches follow the concept introduced in RFC 2638 [2, 3]. Nevertheless some scalability issues that originally motivated the diffserv framework, remain. In particular, each bandwidth broker has to handle large volumes of end-to-end flows, which is the reason why they do not scale as well as the network does [4]. To solve this issue the aggregation of resource reservation requests with the same source and destination network (or diffserv domain) within a unidirectional end-to-end tunnel[1] was proposed [3]. The number

[1] end-to-end (e2e) tunnels are only logical constructs.

of reservation states to keep in a single bandwidth broker could be reduced to N^2 e2e-tunnels (worse case), where N is the number of diffserv domains within a diffserv region.

As shown in this paper, it is possible to further reduce the number of reservation states using the peering relations between independent organisational units of a network. Such relations can be found between *autonomous systems* (AS), between routing areas and even in intranets between backbone and local networks. If two organizational units are not interconnected neighbors there is at least a third unit providing transport services to both of them (the latter is more common). The common approaches [3, 5, 6] do not consider those relations neither in the process of service provisioning nor in the process of resource reservation. Each organisational unit is seen as service requestor of its successor in the end-to-end path.

Furthermore most of the approaches provide solutions for inter-domain-management with one BB or a shared policy server per autonomous system[3, 5, 6]. Those approaches do not consider the heterogenity of stub autonomous systems – mostly a network of administratively independent intranets (routing areas) and a backbone network that interconnects the intranets. To avoid scalability problems that will occure if all resource reservation requests are processed by a single bandwidth broker, each intranet should have its own set of distributed bandwidth brokers to be able to partition its networks into several diffserv domains and to apply own policies. The intra-domain-management approach presented in this paper addresses those peering relations and allows interworking with solutions for inter-domain-management.

The paper is organised as following. In section 2 our new signaling approach for intra-domain diffserv-management in stub autonomous systems is presented. Section 3 gives an analytical discussion focused on its primary benefit the significant reduction of the number of flows to manage. Further on in section 4 the workload of each bandwidth broker will be analysed. Results obtained from experiments with a prototypical implementation are discussed in section 5. Finally conclusions will be given, together with an outlook of further research.

2 A novel signaling approach

The backbone of an autonomous system is typically seen by the routing areas as service provider. Furthermore, a routing area (campus networks or intranets) can be subdivided into local networks and the intranet backbone. The latter can be considered as the service provider of its interconnected local networks. While the intranet backbone can serve any communication within its own intranet, it becomes a service user of the AS-backbone while communicating with other routing areas. The services provided by the backbone are inherited by the AS-backbone to local networks. Service level specifications are used to describe the service and the diffserv domain that provides the service. This diffserv domain is called *root service provider*.

For each service provided by a root service provider a tree can be constructed, built up following the inheritance of the service[2]. Thus, for each unidirectional path through the network using a particular service, it exists a root service provider both communication endpoints depend on. We propose to split the end-to-end tunnels into upstream and downstream tunnel at the root service provider, to be able to merge tunnels to the same root in each node of the tree (Fig. 1). Once the SLS are inherited from the root service provider to the leaf nodes no further information or mechanisms are required in this process. Our concept provides an aggregation

[2] Additional rules for inheritance are required in multi-path routing environments.

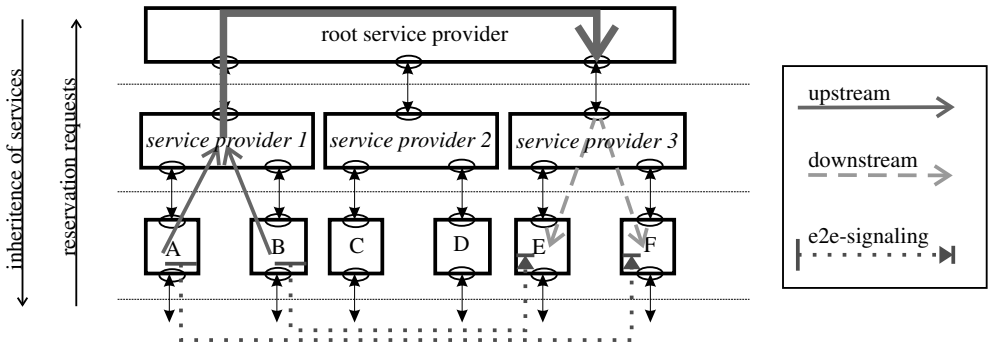

Figure 1: Example network / hierarchy showing aggregation of tunnels

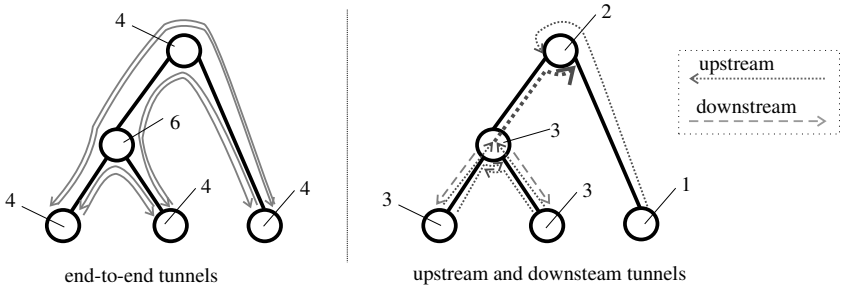

Figure 2: Number of upstream and downstream tunnels to manage per bandwidth broker (or diffserv domain) in comparison to the number of end-to-end tunnels

of resource reservations for unidirectional communication paths into tunnels, to reduce the number of tunnels to be handled in each bandwidth broker — compared to an approach using end-to-end tunnels. An upstream tunnel lasts from the considered diffserv domain in the path from the source to the root service provider up to and throughout the latter. Accordingly the downstream tunnel lasts from the considered diffserv domain in the path from the root service provider to the destination up to but not throughout the root service provider. For an given diffserv domain an upstream tunnel starts at the source where the request came from, goes up to the root service provider and ends at the output interface of the root service provider. Accordingly a downstream tunnel starts at the output interface of the root service provider and goes down to the requested destination. Figure 2 gives an example showing the number of upstream and downstream tunnels to be handled by the bandwidth brokers in comparison to the number of end-to-end tunnels. We count the number of upstream and downstream tunnels to which a bandwidth broker requests resources from its direct service provider. The root service provider is no service user of any other diffserv domain in our example. Nevertheless, we took into account the number of upstream tunnels running throug the root service provider due to its salient feature. The bandwidth brokers at the leaf nodes are required to cooperate in the process of resource allocation. For an unidirectional resource reservation request one of the bandwidth brokers has to request resources for the upstream tunnel while the other one – at the request's destination – has to request resources for the downstream tunnel. Figure 3 shows the way-time-diagram of our signaling approach. The end-to-end signaling is done via direct communication and not relayed by the bandwidth brokers on the path.

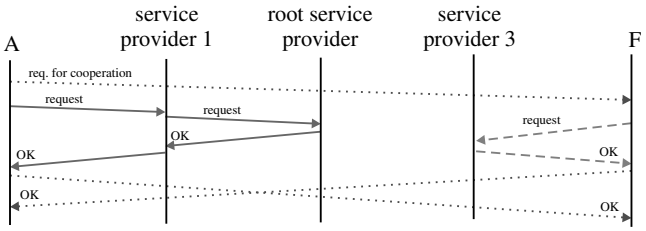

Figure 3: Way-time diagram of end-to-end signaling

As benefit of the separate end-to-end signaling we are flexible to allow sender- and receiver-based reservations, and thus also bidirectional reservations. Furthermore, it is not necessary to have both communication parties involved in the reservation process – unless the peer is a mobile host. It is possible to request e.g. a virtual leased line for a ftp-upstream or a http-downstream at the 'network' layer without notification at the servers in the application layer. The local bandwidth broker is responsible for the proper configuration of the local network and the traffic shaper at the server host. Figure 1 gives an example to illustrate the approach using a simple scenario with one root service provider (level 0) and three diffserv domains inheriting the provided services to their interconnected service users (A. . . F). The latter are seen as local networks (LAN) interconnected by campus backbones which are again interconnected by the backbone of a *national research network* (NREN). We consider resource reservation requests for an unidirectional end-to-end tunnel from A to F and B to E respectively. Both depend on the services provided by the root service provider. The resource reservation requests for the upstream tunnel up to the root service provider are requested by A and B, while F and E are required to request resources for the downstream tunnel "up" to the root service provider. The latter depends on a request for cooperation sent out before by the so called *initiator* of the reservation process from end-to-end (A and B resp.) to its peer (F and E resp.). As both upstream tunnels end at the same gateway it is possible to aggregate or map the tunnels from A and B into one tunnel by the bandwidth broker of their campus backbone. The process is similar for the downstream tunnel, but not required in this example. A downstream tunnel lasts from the service user (LAN) up to the gateway to the root service provider. Thus, it is not necessary to request resources for an downstream tunnel from the latter and thus not necessary to map or aggregate the downstream tunnels in our example.

3 Analytical discussion

This section gives an analytical discussion of the numbers of tunnel to manage per bandwidth broker compared to the numbers of end-to-end tunnel. For a comprehensive discussion a complete n-nary tree is used for the analysis. Nevertheless, it is always possible to rebuild a typical network using overlapping trees. If we assume that the tunnel has to cross the root node — which is the root service provider for the inner nodes and the leaf of the current tree — each end-to-end tunnel could be assigned to exactly one of those overlapping trees. Thus a more general analysis of one tree in a network is applicable for each network with an autonomous system.

In our approach each upstream tunnel is given by the identifier of the root service provider and its direct service user in the downstream path to clearly identify the path through the root service provider. Thus, the number of upstream tunnels is determined by the rank (or

connectivity) r of the root service provider. If each leaf of the tree has an SLS for upstream and downstream services then each leaf and each inner node has to manage at maximum $r-1$ upstream tunnels and one downstream tunnel per root service provider. The number of tunnels to manage for each service does not depend on the number of leafs (N). Compared to an end-to-end tunnel approach the number of tunnels is not higher in the worse case, but normally much less due to the logarithmic growth of the trees high in relation to the number of leafs (Fig. 2). In the worse case the rank of the root node is identical to the number of leaf nodes. That means $r * (r-1)$ upstream tunnels have to be managed by the root node. This happens if all leaf-nodes are children of the root node or each inner node has a connectivity of two.

Table 1 gives the equations to compute the number of tunnels to manage per node if — in addition to the root node — each inner node of the n-nary tree becomes a root service provider of its children and leaf nodes in their subtree. The dimension of the upper bound in the number of end-to-end tunnels to manage is $O(N^2)$ for each inner node and the root node and $O(N)$ for the leaf nodes. In our approach, the dimension is reduced to $O(log(N))$ for the inner nodes and the leaf nodes. As mentioned in the previous section, the root node is no service user of any other diffserv domain. If we take into account the number of upstream tunnels throughout the node there is obviously no reduction in the dimension ($O(log(N)^2)$). Nevertheless, if the height of the tree is greater than 2 then the rank of the root node is always less than the number of leaf nodes in the tree.

type of node	level	upstream tunnel	down-stream tunnel	sum of upstream and downstream	compared to: number of e2e-tunnels
leaf node	n	$n(r-1)$	$n-1$	$nr-1$	$2(N-1)$
inner node	n	$n(r-1)$	$n-1$	$nr-1$	$\left(x := r^{(h-n)}\right)$ $2x(N-x)+x^2\frac{r-1}{r}$
root node	0	$r(r-1)$	0	$r(r-1)$	$\frac{r-1}{r}N^2$

Table 1: Number of upstream and downstream tunnels to which a bandwidth broker requests resources from its direct service provider — and the number to be handled by the broker in the root node respectively — compared to the number of e2e tunnels in an complete n-nary tree with a rank $r >$ and a high $h > 1$ (N gives the number of leaf-nodes and n the level of a node in the tree)

Our approach takes the advantages of the inheritance of services from the root service provider to the leaf nodes to partition a given network of diffserv domains into several (overlapping) trees. It is obvious that complete trees can be denoted as convenient topologies — with one constraint: The rank of the root service provider should be small. As each inner node may become a root service provider itself, it concerns all inner nodes. If we choose a lower rank for the root node and the inner nodes the height of a tree with a fixed number of leaf nodes will grow as well as the number of bandwidth brokers that are involved in the resource-reservation process. The latter not only increases the load at the bandwidth brokers but also the response time due to more instances are involved. Therefore, rank and height should be decreased in the process of partitioning networks into diffserv domains. There are further aspects that have to be considered in this process, e.g the traffic load of the tunnels and the frequency of reservation-requests that cause changes to the resource allocation of those tunnels.

4 Reduction of workload

Splitting of end-to-end tunnels reduces the number of tunnels to be managed and increases the probability that a tunnel is used by more than one service user. It is more likely that the release of tunnel resources is followed by a new request within a short period. Better response times and a lower number of events to be processed could be achieved by caching of resources per tunnel in each bandwidth broker on the path to the root-service-provider. We propose a cache-mechanism that delays the release of to be freed resources combined with pre-allocation. A late-release strategy is is easy to implement and has a further benefit: the number of delete-requests to be forwarded could be reduced due to aggregation of delete requests. The implementation of a pre-allocation mechanism is more difficult as heuristics are required to estimate the amount of resources to request in advance. The following analysis implies that it is possible to implement cache-mechanisms with certain efficiency. A discussion of results obtained in experiments with a prototypical implementation is given further on in section 5.

An M/D/1 model of the network shown in Fig. 4 was used for an analytical study of the caching benefits. Jackson's algorithm [7] was used in the study with a notional exponential arrival rate for resource requests at the leaf nodes and a notional deterministic service rate ($\mu = 500$). The arrivals ($\lambda = 600$) were distiributed equally to the leaf nodes. A cache with a notional efficiency of [0,10,20] percent was modeled at the children of the root node to achieve a reduction of the numbers of events to be processed at the root node.

Table 2 gives the waiting times T_w for the bandwidth brokers in each level of the hierarchy. A significant reduction is already achieved for 10 percent cache efficiency. As the number of messages sent to the root node becomes smaller the number of messages sent by the root node is also reduced, which results in shorter waiting times. Thus the number of events declines in all bandwidth brokers connected to the root node.

efficiency	root	level 1	level 2
0%	0.001500	0.001500	0.000667
10%	0.001059	0.001333	0.000667
20%	0.001200	0.000774	0.000667

Table 2: Waiting time for network in figure 1 with $\lambda = 600$, $\mu = 500$ in dependence of the efficiency that is inputed to the caches implemented to bandwidth brokers at level 1.

Further studies showed that the reduction of the events in the root node does not depend on the position of the cache – with a given efficiency — in the path. Besides a similar reduction in the root node was a obtained by the combination of two caches with a notional efficiency of 10% in sequence compared to one cache with a notional efficiency of 20%.

5 Evaluation

A prototype bandwidth broker has been developed for evaluation purposes. The current implementation features the full signaling approach, a late release cache and provides deterministic service rates. Figure 4 shows the test topology. The leaf nodes represent campus networks and their associated communication end points. A VoIP scenario with 600 bidirectional calls was used to study and compare the cache behavior by experiments. The calls where divided into

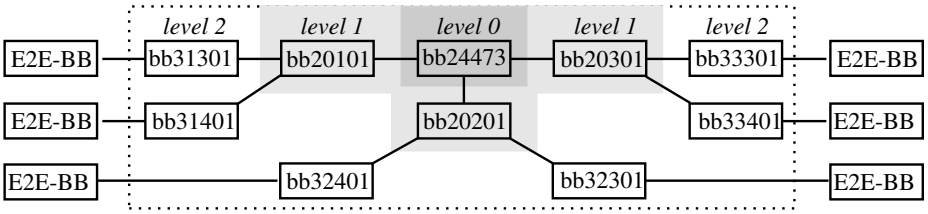

Figure 4: Test topology

80 percent short calls (3 min) and 20% long calls (8 min) and distributed equally to pairs of leaf nodes. The configuration of the test was set so that each request could be fulfilled without rejections. Thereby the specification of parameters for an optimal cache size could have been neglected.[3] The test bed consisted of 5 PCs running RedHat Linux 9 connected over a FastEthernet MAC-switch. The bandwidth brokers were distributed among these computers that CPU load of any machine does not exceed 50%. Several test runs were implemented and five of them will be discussed in this paper (Tab.3), whereby the delay time until the release of the resources was varied. Cache mechanisms were only used at the bandwidth brokers at the level–1 (bb20x01) and level–2 (bb3xx01) but not level–0 (bb24473). Test objectives are the evaluation of different late release delays and their impact on the load behavior. In the first instance (test 1...5) bandwidth brokers of all levels have got equal delay values.

	late release delay values	
Test-no.	bb20x01	bb3xx01
1	20ms	20ms
2	200ms	200ms
3	1s	1s
4	2s	2s
5	6s	6s

Table 3: Delay time until release of resources for each test run

5.1 Load behavior — general results

It is expected that higher delay values will reduce the number of requests and/or forwarded requests in general. The larger the cache the more incoming service requests can be served using the cache only. The positive cache effects will be expected to be low at level–2 bandwidth brokers and the better the nearer it is located to the root service provider (bb24473). According to the given test strategy, reduction comes into effect as recently as the first releases have been made. Hence the request load at the first seconds ($\geq$ 3 sec) of each interval will be higher compared to the rest.

The results shown in Fig. 5–7 show the number of requests on a cumulative basis and support the expectations. The values measured at test No. 1 and 2 have rather the same distribution (Tab. 4). Thus the difference between the release delay of 10ms and 200ms is not significant for the used test scenario.

[3] This is a simple implementation issue, if an appropriate statistical database is given. Generation of such a database will not be covered by this paper.

bandwidth broker (on level)	number of test run				
	test 1	test 2	test 3	test 4	test 5
bb31301 (2)	2260	2236	1773	1692	1634
bb20101 (1)	2699	2696	1546	1326	1112
bb24473 (0)	2356	2344	1471	1290	903

Table 4: Number of all processed requests for each test run

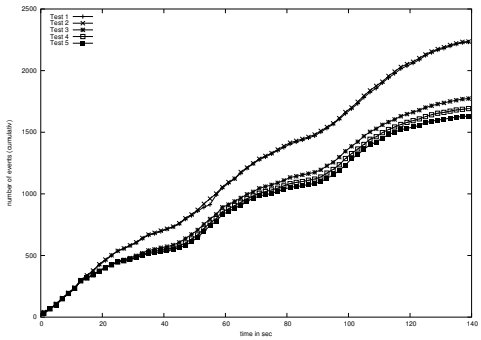

Figure 5: Number of all processed requests (accumulated) at level–2 bandwidth broker (bb31301)

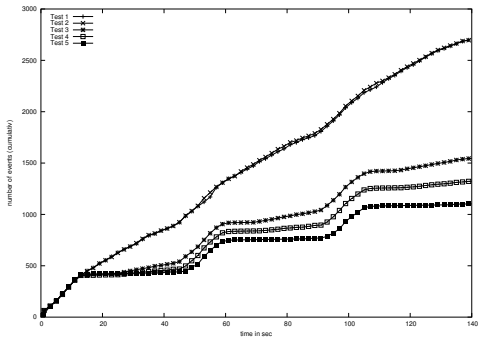

Figure 6: Number of all processed requests (accumulated) at level–1 bandwidth broker (bb20101)

At all levels of bandwidth brokers a distinct cache effect could be observed at the tests 3, 4, and 5 in each case 10 to 15 seconds after starting the next interval. The amount of signaling could be reduced significantly by using a cache strategy. As expected, it could be observed that the efficiency of caching is the better the nearer to the core; bb31301 vs. bb20101 vs. bb24473 (Fig.5–7). The amount of signaling at bb31301 could be reduced up to 15% (test 3) and up to 25% (test 5). That means, a bandwidth broker on level–2 (bb31301) sends up to 25% less requests to the following bandwidth broker on level–1. Besides, the cache positively effects the number of responses sent from core bandwidth broker (bb24473) to its level–1 connectors. Hence, the numbers of processed requests at the level–1 bandwidth broker (bb20101) could be reduced further up to 13% (test 3) and up to 23% (test 5) compared to level–2's bandwidth broker (bb31301).

5.2 Load behavior — differentiated by message types

Further discussion of the test results will differentiate between two types of messages, requests (adds) and releases of resources (frees), to identify reasons for the experienced reduction. From the user's point of view caching should positively effect the delay between sent resource request and received response. Hence, the number of requests as well as the number of releases must be reduced. To address scalability issues of the bandwidth broker itself, the number of pending responses has to be reduced due to the memory and search complexity involved.

Figures 8, 9 show the trend of resource requests incoming at bandwidth broker level–1 (bb20101) and level–0 (bb24473). There is no cache effect visible at bandwidth broker at level–2 (bb31301). Hence the number of received messages is equal to the number of requests sent by e2e-

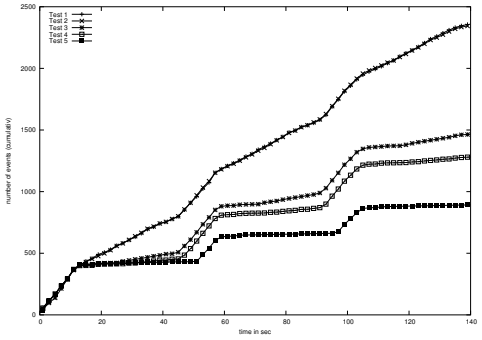

Figure 7: Number of all processed requests (accumulated) at level–0 bandwidth broker (bb24473)

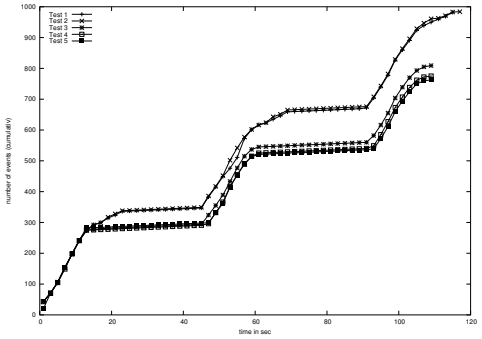

Figure 8: Number of adds (accumulated) at level–1 bandwidth broker (bb20101)

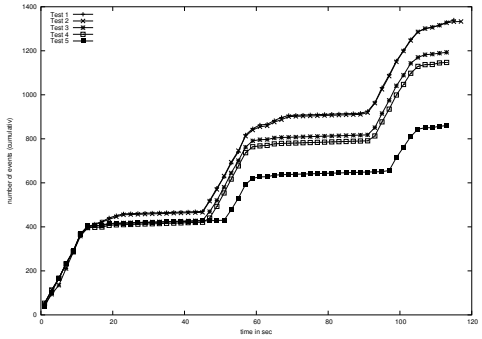

Figure 9: Number of adds (accumulated) at level–0 bandwidth broker (bb24473)

bandwidth brokers. On the contrary the cache at bandwidth brokers at level–1 and level–0 effected the number of messages received by them. The main distinction between the bandwidth brokers at different levels is the number of resource requests to be processed. As a result of message aggregation at the level–1 and level–0 bandwidth brokers the number of resource requests is twice (bb20101) and three times (bb24473) higher than those at level–2's broker. Regardless of the significant reductions of resource requests the cache effect on the whole can not be explained as the over-all reduction at bb20101 and bb24473 is more than twice as high.

As shown in Fig.10, 11 the late release of resources significantly reduces the number of sent release messages. The achieved reduction is up to 80% for test 3, and up to 90% for test 4 and 5 and thus substantial higher than the reduction potential of resource requests (adds). The combination of late release of resources and aggregation of many release messages into a single one is the reason for this significant reduction. The chosen late release parameters of test 5 (Tab.3) yielt to a complete prevention of forwarding release messages. Its effect is clearly visible at level–0 bandwidth broker (Fig.11). Due to the implemented late release strategy at all bandwidth brokers at lower levels (level–1 and level–2), the level–0 broker did not get release messages since the 82nd second. A similar effect could be observed at level–1's broker (bb20101). It received release messages only until the 115th second. This behavior effects the

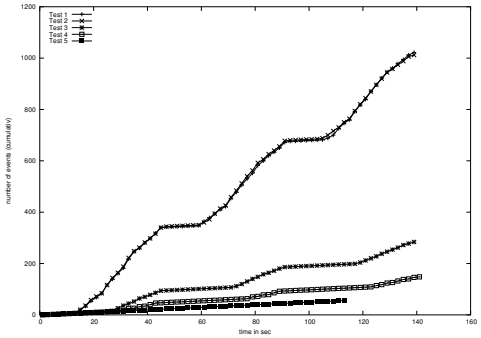

Figure 10: Number of releases (accumulated) at level–1 bandwidth broker (bb20101)

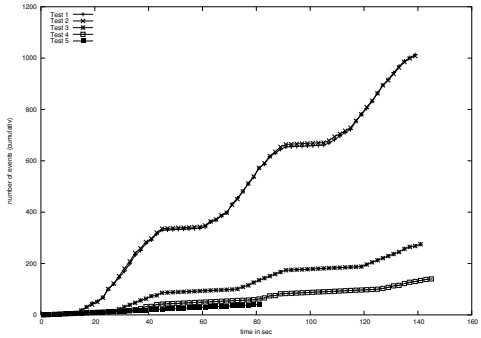

Figure 11: Number of releases (accumulated) at level–0 bandwidth broker (bb24473)

dwell time positively overall. Nevertheless this striate yields to a pool of unused resources at some time. Though it has no effects within the context of this paper due to different objectives, the effect should be considered during design of cache strategies.

5.3 Reduction impact on response time

As shown above, the load behavior of bandwidth brokers could be positively affected by using the implemented cache strategy. It is expected that significant reductions of processed requests will effect response times at level–2 brokers as well. For the tests 3, 4, and 5 faster response times are awaited due to a lower load at the bandwidth brokers and responding requests using the cache.

These expectations could be confirmed by tests. Figure 12 depicts the mean response times for bandwidth broker at level–2 (bb31301) as an example. The observed reduction effects at brokers at level–2 must be the strongest as they depend on response times of higher level (level–1, level–0) bandwidth brokers. Within the first 15 seconds of the first interval no significant differences are visible for test 1 compared to test 3, 4, and 5. After reception of a higher number of release messages (c.f. 10) the a slower growth of curves was noticeable. Within the interval where requests could be answered $adhoc$ using the cache, the curve of test 3, 4, and 5 climbs very slow due to low duration of 0.7ms ($\pm$0.2ms) for processing a request. If the request could be fulfiled using the cache, the waiting time is decreased by the fraction usually needed for the service provider's response. That is the reason for the nearly linear trend of the curves displaying the mean processing time (accumulated) for release-requests as shown in Fig. 13.

The mean response times at test 3, 4, and 5 are rather similar but differ significantly from that in test 1 (Fig.12). The difference between test 1 and test 3,4, and 5 indicates that a reduction of requests results in a lower load of bandwidth brokers, so that lower dwell times yield to faster responses. The further reduction of mean response times — visible at inter–test comparison between tests 3, 4, and 5 — is caused by replying a higher fraction of messages using the cache so that faster response times per request could be realised.

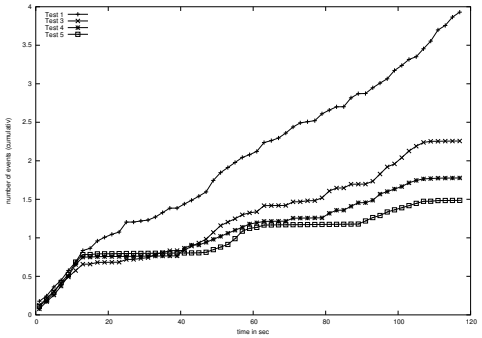

Figure 12: Mean response time for adds (accumulated) at level–2 bandwidth broker (bb31301) for test 1 – 5

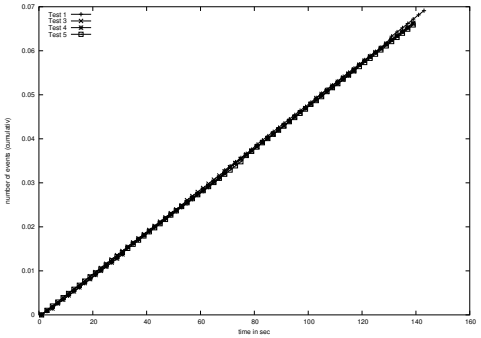

Figure 13: Mean response time for releases (accumulated) at level–2 bandwidth broker (bb31301) for test 1 – 5

6 Outlook on implementation and application

A number of points have to be considered in an application scenario. The application of bandwidth brokers is not limited to huge networks, but may be used to apply access control to better transport services in parts of a campus network. An initial deployment should be limited to subnets within a campus network, to reduce the number of parties involved at one side and to allow further development of applications that can benefit from a higher quality of service with the network at the other side. The application of a bandwidth broker controlled DiffServ environment requires to agree upon the services required or provided within the network — so called *service level specifications, (SLS)*. In the style of similar architectures, we propose that a small number of SLS is required and that those SLS are shared between all routing areas within the autonomous system. Thus, the discussion of such SLS is a first basic step that has to be done before one can decide which subnetworks become a DiffServ domain controlled by its bandwidth broker. We do not propose an partitioning algorithm that can be used for dividing a routing area into DiffServ domains, as it cannot consider the relationships between parts of the network, the proposed number of users and several other management issues. Nevertheless, it should become clear from section 3, that at one side the size of each DiffServ domain should be a large as possible, but a) of limited size — if it incorporates edge devices — with respect to the workload of it bandwidth broker and b) of limited connectivity to adjacent DiffServ domains — in the case of a root service provider — to limit the number of upstream tunnels to keep. Finally, it is important to activate and eventually extend the host operating systems to allow traffic shaping or at least traffic policing in each workstation or at least the first router/firewall.

7 Conclusion

The Differentiated Services Architecture provides a scalable approach for service differentiation in the data path. Bandwidth broker architectures have been developed that supplement diffserv with mechanisms for access control and resource management to allow not only service differentiation but hard service guarantees. Most of the common approaches utilise end-to-end tunnels to reduce the number of reservation states to keep in each bandwidth broker and do not consider the aspects of intra-domain-management. We have designed and implemented a new signaling approach for the latter. It is based on the peering relations between diffserv domains engineered within the routing areas of an autonomous system. Those peering relation

can be used to inherit the services or service level specifications provided by a diffserv domain and, thus, to determine a root service provider for each unidirectional end-to-end tunnel using information that have been inherited with the service level specifications. We propose to split end-to-end tunnels into an upstream and a downstream tunnel at the root service provider to achieve an aggregation of tunnels in each bandwidth broker. The bandwidth brokers at the ends of the end-to-end tunnel cooperate in the process of resource reservation. The latter allows to deploy not only sender-based but also receiver-based reservation requests.

An analytical comparison of the number of end-to-end tunnels to kept in each bandwidth broker compared to the number of upstream and downstream tunnels showed the potential to reduce the number of flows. The scalability of a bandwidth broker architecture is also effected by the workload of each BB. Resource caching was introduced to obtain a reduction in the number of requests to process per BB. The potential to reduce the workload was depicted analytical and by extensive tests. A prototypical implementation of our signaling approach with a late release cache mechanism was used in the tests.

Acknowledgment

This research was done in behalf of the DFN e.V. (Deutsches Forschungsnetz) within the scope of the project 'Class of Service Concepts in the German Research Network', which was partly funded by BMBF (Federal Ministry of Education Research).

References

[1] Charny, A.; Le Boudec, J.-Y. Delay Bounds in a Network With Aggregate Scheduling. In Crowcroft, J.; James Roberts, J.; Smirnov, M. I., editor, *Quality of Future Internet Services, First COST 263 International Workshop, QofIS 2000, Berlin, Germany, September 25-26, 2000, Proceedings*, volume Volume 1922 of *Lecture Notes in Computer Science*, pages 1–13. Springer, September 2000.

[2] Nichols, K.; Jacobson, V.; Zhang, L. A Two-bit Differentiated Services Architecture for the Internet. RFC 2638 (Informational) - The Internet Engineering Task Force, Juli 1999.

[3] Adamson, A.; Bennini, G.; Chimento, P.; Dunn, L.; Frank, R.; Geib, R.; Granzer, H.; Hares, S.; Mantar, H.; Murray, W.; Neilson, R.; Okumus, I.; Reichmeyer, F.; Roy, A.; Sander, V.; Spence, D.; Teitelbaum, B.; Terzis, A.; Wheeler, J. QBone Signaling Design Team: Final Report. http://qos.internet2.edu/wg/documents-informational/20020709-chimento-etal-qbone-signaling/, Juli 2002.

[4] Gunter, M.; Braun, T. Evaluation of Bandwidth Broker Signaling. In *Seventh Annual International Conference on Network Protocols, ICNP 1999, 31 October - 3 November, 1999, Toronto, Canada, Proceedings*, pages 145–152. IEEE Computer Society, 1999.

[5] Koch, B. F. A QoS architecture with adaptive resource control: The AQUILA approach. In Wells, W., editor, *The 8th International Conference on Advances in Communication and Control: Telecommunications/Signal Processing, Crete, Greece, June 25-29, 2001*. OPTIMIZATION SOFTWARE, INC., Juni 2001.

[6] Bless, R. Dynamic Aggregation of Reservations for Internet Services. In *Tenth International Conference on Telecommunication Systems - Modeling and Analysis (ICTSM 10)*, volume 1, pages 26–38, Oktober 2002.

[7] Jackson, J. R. Networks of waiting lines. Operations Reseach 5 (1957), 518-521, 1957.