PAPER *Special Issue on Internet Technology*

On Scalable Design of Bandwidth Brokers

Zhi-Li ZHANG[†], **Zhenhai DUAN**[†], ***Nonmembers, and*** **Yiwei Thomas HOU**[††], ***Regular Member***

SUMMARY In this paper we study the *scalability* issue in the design of a centralized bandwidth broker model for dynamic control and management of QoS provisioning. We propose and develop a *path-oriented, quota-based* dynamic bandwidth allocation mechanism for efficient admission control operations under the centralized bandwidth broker model. We demonstrate that this dynamic bandwidth allocation mechanism can significantly reduce the overall number of QoS state accesses/updates, thereby increasing the overall *call processing capability* of the bandwidth broker. Based on the proposed dynamic bandwidth allocation mechanism, we also extend the centralized architecture with a single bandwidth broker to a hierarchically distributed architecture with multiple bandwidth brokers to further improve its scalability. Our study demonstrates that the bandwidth broker architecture can be designed in such a manner that it scales with the increase in the network capacity.

key words: *bandwidth broker, admission control, resource reservation, scalability, quality of service, Internet*

1. Introduction

In the IETF Differentiated Services (DiffServ) framework, a centralized model based on the notion of *bandwidth broker* (BB) [4] has been proposed for the control and management of QoS provisioning to reduce the complexity of QoS control plane. Under this centralized model, each network domain has a bandwidth broker (a special network server) that is responsible for maintaining the network QoS states and performing various QoS control and management functions such as admission control, resource reservation and provisioning for the entire network domain. Issues in designing and building such a centralized bandwidth broker architecture have been investigated in several recent studies [1], [6], [8].

This centralized bandwidth broker model for QoS control and management has several appealing features. For example, the centralized bandwidth broker model *decouples* (to a large extent) the QoS control plane from the data plane. In particular, QoS control functions such as admission control and QoS state maintenance are removed from the core routers of a network domain, reducing the complexity of the core routers. Consequently, no hop-by-hop signaling for reservation set-up along the data path is needed, removing the signaling overhead from core routers. Furthermore, because the network QoS states are centrally managed by the bandwidth broker, the problems of unreliable or inconsistent control states are circumvented [5]. This is in contrast to the IETF Integrated Services (IntServ) QoS control model based on RSVP [2], [7], where every router participates in hop-by-hop signaling for reserving resources and maintains its own QoS state database. Hence in this respect, the centralized bandwidth broker model provides a more efficient alternative for QoS control and management.

However, the centralized bandwidth broker model for QoS control and management also introduces its own *scalability* issue, in particular, the ability of the bandwidth broker to handle large volumes of flows as the network system scales. In a DiffServ network where only slow time scale, static resource provisioning and traffic engineering (e.g., those performed to set up virtual private networks) are performed, the scalability problem may not be acute. But with the rapid evolution of today's Internet, many new applications and services such as Voice over IP (VoIP), on-demand media streaming and real-time content delivery (e.g., stock quotes and news) may require dynamic QoS control and management such as admission control and resource provisioning at the time scale of flow arrival and departure. In these circumstances, an improperly-designed centralized bandwidth broker system can become a potential *bottleneck*—limiting the number of flows that can be accommodated into the network system while the network system itself is still under-loaded.

The objective of this paper is to study the scaling issues in the centralized bandwidth broker model for flow-level dynamic QoS control and management. We consider the factors that may potentially affect the scalability of the centralized bandwidth broker model—in particular, we identify two major limiting factors: 1) the memory and disk access speed, and 2) communication capacity between the bandwidth broker and edge routers. Because of the need to access and update the network QoS states during admission control operations, the number of memory and disk accesses/updates plays a dominant role in the time the bandwidth broker takes to process flow reservation requests. Therefore, reducing the overall number of QoS

Manuscript received January 29, 2001.
Manuscript revised March 16, 2001.

[†]The authors are with the Dept. of Computer Science & Engineering, University of Minnesota, Minneapolis, MN, USA.
[††]The author is with Fujitsu Laboratories of America, Sunnyvale, CA, USA.

state accesses/updates is a key means to enhance the overall call processing capability of the bandwidth broker, thereby its scalability. In this paper we develop a *path-oriented, quota-based* dynamic bandwidth allocation approach to address this issue. This approach is designed based on the *two-level representation* of the network QoS states proposed in [8], i.e., a path QoS state database representing the path-level QoS states as well as a link QoS state database representing the link-level QoS states of the network domain. By allocating bandwidth in units of *quota* to paths on demand, the proposed dynamic bandwidth allocation approach limits the majority of flow reservation requests to the *path state* accesses/updates only, avoiding the more time-consuming *link state* accesses and updates. As a result, the overall number of QoS state accesses/updates is significantly reduced, thus increasing the overall *call processing capability* of the centralized bandwidth broker system.

This path-oriented, quota-based dynamic bandwidth allocation approach also leads to a natural architectural extension to the centralized bandwidth broker model: a *hierarchically distributed* bandwidth broker architecture to address the scaling problem caused by the potential communication bottleneck between the centralized bandwidth broker and edge routers. We propose a hierarchically distributed architecture consisting of a number of *edge bandwidth brokers,* each of which manages a (mutually exclusive) subset of path QoS states and performs admission control for the corresponding paths, and a *central bandwidth broker* which maintains the link QoS state database and manages the quota allocation among the edge bandwidth brokers. We conduct extensive simulations to investigate the impact of the proposed mechanisms and architectural extensions on the network system performance, and to demonstrate their efficacy in enhancing the scalability of the centralized bandwidth broker model. Our study shows that the scalability issue of the centralized bandwidth broker model can be addressed effectively, without incurring any additional overhead at core routers.

The remainder of the paper is organized as follows. In Sect. 2, we first present a centralized bandwidth broker architectural model, and then discuss the potential scaling issues of the centralized bandwidth broker architecture. In Sect. 3, we describe the basic path-oriented, quota-based dynamic bandwidth allocation approach, and study its efficacy in enhancing the overall call processing capability of the centralized bandwidth broker system. In Sect. 4, we design a hierarchically distributed multiple bandwidth broker architecture using the path-oriented, quota-based dynamic bandwidth allocation approach. Its impact on the system performance is also investigated. In Sect. 5, we discuss several possible enhancements to our bandwidth brokers architecture. We conclude the paper in Sect. 6.

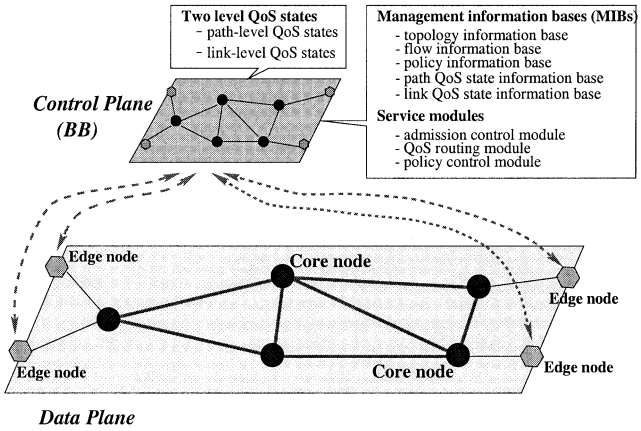


Fig. 1 A bandwidth broker architecture.

2. Bandwidth Broker Architecture: Basic Model and Scaling Issues

As the basis for our study, in this section, we first present a basic centralized bandwidth broker architectural model and describe how *admission control* is performed under such a model. We then discuss the potential scaling issues in this centralized bandwidth broker model, and briefly outline the solutions that we will develop in this paper to address these issues.

2.1 The Basic Bandwidth Broker Model

The basic centralized bandwidth broker model for the management and control of the QoS provisioning of a network domain is schematically depicted in Fig. 1. This model is based on the bandwidth broker architecture proposed in [8]. In this architectural model, the bandwidth broker centrally manages and maintains a number of management information (data) bases (MIBs) regarding the network domain. Among them, the network topology database and network QoS state databases are most relevant to the study of this paper. The network topology database and network QoS state databases together provide a *logical* representation (i.e., a QoS abstraction) of the network domain and its entire state. With this QoS abstraction of the network domain, the bandwidth broker performs QoS control functions by managing and updating these databases. In this sense, the QoS control plane of the network domain is *decoupled* from its data plane. The core routers of the network domain are removed from the QoS control plane: core routers do not maintain any QoS reservation states, whether per-flow or aggregate, and do not perform any QoS control functions such as admission control.

In our centralized bandwidth broker model, the network QoS states are represented at two levels: *link-level* and *path-level.* The link QoS state database maintains information regarding the QoS state of each link

in the network domain, such as the total reserved bandwidth or the available bandwidth of the link. The path QoS state database maintains the QoS state information regarding each path of the network domain, which is *extracted* and *"summarized"* from the link QoS states of the links of the path. An example of the path QoS state is the available bandwidth along a path, which is the minimal available bandwidth among all its links. As shown in [8], by maintaining a separate path-level QoS states, the bandwidth broker can conduct fast admissibility test for flows routed along the path. Furthermore, path-wise resource optimization can also be performed based on the (summarized) path QoS state. As will be demonstrated in this paper, this two-level representation of the network QoS states is also the key means that leads to scalable design of bandwidth broker architectures for dynamic flow-level QoS provisioning. Lastly, we note that both the link QoS states and path QoS states are aggregate QoS states regarding the links and paths. No per-flow QoS states are maintained in either of the two QoS databases. The QoS and other control state information regarding each flow[†] such as its QoS requirement and reserved bandwidth is maintained in a separate flow information database managed by the bandwidth broker [8].

We now briefly describe a simple admission control scheme to illustrate how flow-level dynamic QoS provisioning can be performed under the basic centralized bandwidth broker model. For simplicity of exposition, throughout this paper we assume that bandwidth is the critical network resource that we are concerned about. We consider the flow reservation set-up request first. When a new flow arrives at an edge router, requesting a certain amount of bandwidth to be reserved to satisfy its QoS requirement, the flow reservation set-up request is forwarded by the edge router to the bandwidth broker. The bandwidth broker then applies an admissibility test to determine whether the new flow can be admitted. More specifically, the bandwidth broker examines the path QoS state (obtained from the corresponding link states) and determines whether there is sufficient bandwidth available along the path to accommodate the new flow. If the flow can be admitted, the bandwidth broker updates the path QoS state database and link QoS state database (as well as the flow information database) to reflect the new bandwidth reservation along the path. If the admissibility test fails, the new flow reservation set-up request will be rejected, and no QoS information databases will be updated. In either case, the BB will signal the ingress edge router of its decision. For a flow reservation tear-down request, the bandwidth broker will simply update the corresponding link state database and path state database (as well as the flow information database) to reflect the departure of the flow. Clearly, using the basic admission control scheme presented above, processing either the flow reservation set-up or tear-down request requires access/update to the link QoS state database as well as the path QoS state database. Access and update of the link QoS states are necessary to ensure that the link QoS states are always up-to-date, so that the bandwidth broker can obtain accurate path QoS state information and make correct admission control decisions. We refer to this "naive" admission control scheme that requires per-flow link QoS state access/update, as the *link-update* admission control scheme. In this paper we will present a more efficient approach to performing bandwidth allocation and admission control that can significantly reduce the overall number of QoS state accesses and updates.

2.2 Scaling Issues

The issue of scalability is an important consideration in the design of a (centralized) bandwidth broker system. An important measure of scalability is the ability of the bandwidth broker system to handle large volumes of flow reservation requests, as the network system scales. For example, as the network link capacity increases, the *call processing capability* of the bandwidth broker system, defined as the number of flow requests that can be processed by the bandwidth broker system per unit of time, must scale with the increasing number of flows that can be accommodated in the network system. In particular, the bandwidth broker system should *not* become the bottleneck while the network system has not been overloaded.

Although it is possible to enhance the call processing capability of a bandwidth broker system by simply adding more processing power or increasing memory and disk access speed, such an approach *in itself* in general does not provide a scalable solution. To develop a scalable bandwidth broker architecture, we need to gain a *fundamental understanding* of the potential scaling issues and problems in a centralized bandwidth broker architecture, and then devise appropriate mechanisms and architectural extensions to address these issues and problems. This is precisely the objective of our paper. In this section we will identify two key factors that can potentially limit the scalability of the centralized bandwidth broker architecture, and outline the solutions we will develop in the remainder of the paper.

There are many factors that may potentially affect the call processing capability of a bandwidth broker system. Among them, the number of QoS state access and/or update to handle a flow request plays a prominent role. Recall that when processing a flow reservation set-up request, the bandwidth broker must perform an admissibility test, and if the request can be granted, update the relevant QoS states. Likewise,

[†]In this paper, a flow can be either an individual user flow, or an aggregate flow of multiple users, defined in whatever appropriate manner (e.g., an aggregate flow representing traffic from an institution or a sub-network).

when processing a flow reservation tear-down request, the bandwidth broker needs to update the relevant QoS states. In either case, access and/or update to QoS states, therefore the access to memory/disk, are involved. Since memory/disk access speed is typically much slower than processing speed, we argue that the processing time of flow requests is determined in a large part by the number of memory/disk accesses/updates. Therefore, an important means to enhance the call processing capability of a bandwidth broker system is to reduce the number of accesses/updates to the network QoS states maintained by the bandwidth broker.

Another factor that may affect the overall call processing capability of a centralized bandwidth broker system is the capacity of the communication channels (e.g., the network delay and congestion, or the I/O bandwidth at the bandwidth broker server) between a centralized bandwidth broker system and various edge routers. As the number of flows increases, these communication channels can become a potential bottleneck, limiting or delaying the number of flow requests delivered to the centralized bandwidth broker system, thereby reducing its overall call processing capability. To scale with the demand of the network system, a distributed multiple bandwidth broker architecture may be called for. Therefore, architectural extension to the basic centralized bandwidth broker model must be considered.

The focus of our paper is on the design of mechanistic enhancement and architectural extension to the basic centralized bandwidth broker model to improve its scalability. In particular, we propose and develop a *path-oriented, quota-based* (in short, PoQ) dynamic bandwidth allocation mechanism for performing admission control. This PoQ dynamic bandwidth allocation mechanism exploits the two-level representation of the network QoS states used in the basic centralized bandwidth broker model, and attempts to avoid accessing and updating the link QoS states every time a flow reservation set-up or tear-down request is processed. In other words, this mechanism limits the majority of accesses and updates to *path* QoS states only, thereby reducing the overall number of accesses and updates to the network QoS states. The basic PoQ mechanism is described in detail in Sect. 3. Using the PoQ dynamic bandwidth allocation mechanism, in Sect. 4, we extend the basic single centralized bandwidth broker architecture to a hierarchically distributed architecture with *multiple* bandwidth brokers to address the scaling problem posed by the potential communication bottleneck between the bandwidth broker system and the edge routers. Our results demonstrate that the proposed path-oriented and quota-based dynamic bandwidth allocation mechanism is indeed an effective means to increase the overall call processing capability of the bandwidth broker. Furthermore, the bandwidth broker architecture can be designed in such a manner that it scales, as the network capacity increases.

3. Single Bandwidth Broker Design

In this section, we present the path-oriented, quota-based (PoQ) mechanism for dynamic bandwidth allocation under a single bandwidth broker. We first describe the basic operation of the mechanism (the base scheme), and then analyze its complexity and performance. Simulation results are presented at the end of the section to illustrate the efficacy of the scheme.

3.1 The Basic PoQ Scheme

As pointed out in Sect. 2, using the basic link-update admission control scheme to process a flow reservation set-up or tear-down request, the bandwidth broker needs to access/update the link QoS state of *each* link along the flow's path. Hence the per-flow request processing time is proportional to the number of link state accesses/updates. As the volume of flow requests increases, these per-flow QoS state accesses/updates can slow down the operations of the bandwidth broker, limiting its flow processing capability. Therefore, reducing the number of link QoS state accesses/updates is an important means to prevent the bandwidth broker from becoming a potential bottleneck. In this section, we present the basic PoQ dynamic bandwidth allocation mechanism for a single centralized bandwidth broker, and illustrate how it can be employed to reduce the overall number of link QoS state accesses and updates. We first outline the basic ideas behind the PoQ mechanism, and then provide a more formal and detailed description.

Under the basic PoQ scheme, the total bandwidth of each link of the network is (virtually) divided into *quotas*. A quota is a "chunk" of bandwidth, appropriately chosen, that is much larger than the average bandwidth requirement of typical flows. Bandwidth is normally allocated on-demand to each path in units of quotas. To be more precise, bandwidth allocation along a path operates in two possible modes: the *normal* mode and *critical* mode. During the normal mode, the bandwidth broker allocates and de-allocates bandwidth in unit of one quota at a time. The path QoS state of a path maintains the number of quotas of bandwidth that have been allocated to the path, in addition to the actual bandwidth that has been reserved for the flows routed along the path. When a flow reservation set-up request along a path arrives, the bandwidth broker only needs to check the corresponding path QoS state to see whether the quotas of bandwidth allocated to the path are sufficient to satisfy the flow's request. If the answer is positive, the flow request is accepted, and the relevant path QoS state is updated accordingly (i.e., the actual reserved bandwidth along the path is increased by the amount requested by the flow). Similarly, when

a flow reservation tear-down request arrives, the bandwidth broker simply needs to update the relevant path QoS state (i.e., the actual reserved bandwidth of the path is decreased by the amount reserved for the flow). We see that in the above two cases, flow requests can be processed by accessing and updating the path QoS states only, without the need to access/update the link QoS state database. When there are a large number of flows arriving and departing in a short period of time, with an appropriately chosen quota size, we expect that many of these flow requests (either reservation set-up or tear-down) will be processed by the bandwidth broker using only the path QoS states. This key observation is the major motivation behind our PoQ dynamic bandwidth allocation mechanism.

In the case that the bandwidth allocated to a path is not sufficient to satisfy the reservation set-up request of a flow, the bandwidth broker will attempt to allocate a new quota to the path to accommodate the flow reservation set-up request. In this case, the bandwidth broker needs to check each link QoS state along the path to see whether there is a quota available at all the links. If this is the case, a new quota is allocated to the path, and the number of available quotas at each link of the path is decremented by 1. When there is an extra unused quota available along a path (due to flow departures), the extra quota will be reclaimed by the bandwidth broker, and the extra quota is returned to each link along the path. The available number of quotas at these links will be increased by 1. Clearly, quota allocation/de-allocation will incur some overhead. In particular, the bandwidth broker needs to access and update the link QoS states to keep track of the available quotas at each link. Generally speaking, large quota size tends to reduce the overhead of quota management. On the other hand, large quota size has other performance implications, as we will see later.

Quota allocation for a path can fail if one of the links along the path does not have sufficient quotas left. In this case, bandwidth allocation for the path enters into the *critical* mode. More generally, when the available quota of a link falls below a threshold (say, no quota left), we say that the link is *critically loaded* (or the link is critical). When a link is critically loaded, all paths traversing this link enter the critical mode. Once a path is in the critical mode, the bandwidth broker will cease allocating bandwidth along the path in units of quota. Instead, the bandwidth is allocated/de-allocated on a per-flow basis, as in the basic link-update scheme described in Sect. 2. In particular, it maintains an accurate link QoS state for each critically loaded link (e.g., the precise amount of reserved bandwidth at the link). Hence, when processing a flow reservation set-up/tear-down request for a path in the critical mode, the bandwidth broker must access/update the link QoS states of those critically loaded links along the path. In this way, we ensure that the admission control decision is always made correctly. The reason why we switch to the link-update admission control scheme is that flow reservation set-up request will not be rejected unnecessarily (as the bandwidth is still available). As a result, whenever a flow is admitted using the link-update scheme, it will also be admitted using the basic PoQ scheme. In the next section, we will consider a "lossy-path" model in the context of the multiple bandwidth brokers. The "lossy-path" model can also be used in combination with the basic PoQ scheme to reduce the link QoS state access/update overhead.

In the above we have provided an outline of the basic PoQ dynamic bandwidth allocation scheme. A more formal and detailed description of the scheme is presented in pseudo-code in Figs. 3, 4, and 5. Figure 2 summarizes the notation used in the description. For the ease of exposition, the scheme is divided into three function blocks. Figure 3 describes the path-level admission control for flow reservation set-up and quota allocation management. Figure 4 describes the link-

0. op_p: if $op_p == 0$, path p is in the normal mode.
1. if $op_p > 0$, path p is in the critical mode.
2. cl_p: list of critical links along path p.
3. R_p: total reserved rate along path p.
4. Q_p: number of quotas allocated to path p; it also denotes
5. the total quota bandwidth along p, if no confusion.
6. aqb_p: available quota bandwidth on p: $aqb_p = Q_p - R_p$.
7. op_l: if $op_l == 0$, link l is not critical.
8. if $op_l == 1$, link l is critical.
9. C_l: capacity of link l.
10. Q_l: total quotas of link l.
11. aq_l: available quota of link l. $aq_l = Q_l - \sum_{p:l \in p} Q_p$.
12. rb_l: residual bandwidth of link l. $rb_l = C_l - \sum_{p:l \in p} R_p$.

Fig. 2 Notations.

0. Upon an arrival of a new flow f at a path p:
1. **case 1:** ($op_p == 0$ and $aqb_p \geq r_f$)
2. $R_p \leftarrow R_p + r_f$; accept the flow; End.
3. **case 2:** ($op_p == 0$ and $aqb_p < r_f$)
4. request more quota on all the links l: $l \in p$ (Fig. 4);
5. **case 3:** ($op_p > 0$)
6. request bandwidth r_f on all critical links: $l \in cl_p$ (Fig. 4);
7. **for** $l \notin cl_p$
8. if ($aqb_p < r_f$) request more quota (Fig. 4);
9. **if** (all requests are granted)
10. update Q_p if more quotas are allocated;
11. $R_p \leftarrow R_p + r_f$; accept the flow; End.
12. **else** reject the flow reservation set-up request; End.

Fig. 3 Path-level admission control.

0. Upon a path p requests r_p on a link l:
1. /* r_p can be a quota or a flow's request rate */
2. **case 1:** ($op_l == 0$ and $aq_l < r_p$)
3. collect residual bandwidth: $rb_l \leftarrow C_l - \sum_{p:l \in p} R_p$;
4. **if** ($rb_l < r_p$) reject the request; End.
5. **case 2:** ($op_l == 1$ and $rb_l < r_p$)
6. reject the request; End.
7. /* The request can be honored */
8. **if** ($op_l == 0$ and $aq_l < r_p$)
9. $op_l \leftarrow 1$; /* transition: normal → critical */
10. **for** ($p' : l \in p'$)
11. $cl_{p'} \leftarrow cl_{p'} \cup l$; $op_{p'} \leftarrow op_{p'} + 1$;
12. **case 1:** ($op_l == 0$) $aq_l \leftarrow aq_l - 1$;
13. **case 2:** ($op_l == 1$) $rb_l \leftarrow rb_l - r_p$.

Fig. 4 Link-level bandwidth/quota allocation.

```
0.  Upon an existing flow f departs on a path p:
1.    R_p ← R_p − r_f;
2.    if (op_p > 0)
3.      for (l ∈ cl_p)
4.        rb_l ← rb_l + r_f; re-compute aq_l;
5.        if (aq_l ≥ 0) /* transition: critical → normal */
6.          for (p' : l ∈ p')
7.            op_p' ← op_p' − 1; set Q_p';
8.            cl_p' ← cl_p' − l;
9.    else if (op_p == 0 and p has excess quota)
10.     Q_p ← Q_p − 1; /* return excess quota */
11.     for (l ∈ p)
12.       aq_l ← aq_l + 1;
```

Fig. 5 Scheme for handling flow departure.

level bandwidth allocation and quota allocation management. Finally, Fig. 5 describes both the path-level and link-level bandwidth and quota management operations for handling flow departures.

3.2 Complexity and Performance

In this section, we will provide a simple analysis of the complexity of the basic PoQ dynamic bandwidth allocation scheme, and compare it with the link-update admission control scheme in terms of the QoS state access/update overhead. Since the path QoS states are always accessed/updated for every flow reservation set-up/tear-down request, we focus on the number of link QoS state accesses/updates. We measure the complexity (or cost) of the PoQ scheme by the *expected number of link QoS state access/update per-flow,* i.e., the number of link QoS state accesses and updates incurred by processing a flow arrival and departure.

Consider a network domain whose average path length is P. Let ϕ be the probability that an "average" path of length P is in the critical mode, and γ be the probability a flow is rejected due to unavailability of bandwidth along the path. Note that under the basic PoQ scheme, a flow can only be rejected when the path is in the critical mode. In addition, let φ and χ denote, respectively, the probability that the flow reservation set-up request triggers a quota allocation, and the probability that the flow reservation tear-down request triggers a quota de-allocation, conditioned on that the flow is admitted. Then the expected number of link access/update per-flow, denoted by Θ_{PoQ}, is given by the following expression:

$$\Theta_{PoQ} = P\gamma + 3P\phi(1-\gamma) + P(\varphi+\chi)(1-\gamma). \quad (1)$$

The first term in the above expression is the number of link QoS state accesses for a flow that is rejected. The second term is the number of link QoS accesses and updates for processing a flow arrival in the critical mode plus the number of link QoS state updates for processing a flow departure in the critical mode. Here we assume that to admit a flow in the critical mode, the relevant link states are first accessed for the admissibility test, and then updated after the flow is admitted. Note also that for a flow admitted in the normal mode, no link QoS state is accessed or updated. The last term in (1) reflects the overhead of quota allocation and de-allocation.

Comparing the expected number of link QoS state access/update per-flow of the PoQ scheme with that of the naive link-update admission control scheme, $\Theta_{L-U} = P\gamma + 3P(1-\gamma)$, we see that the reduction in the per-flow number of link QoS access/update under the PoQ scheme is (approximately) proportional to $1-\phi$. Hence if the network system can accommodate N flows, then the reduction in the total number of link QoS access/update is in the order of $N(1-\phi)$. For a large N, this can amount to significant access/update reduction, even when ϕ is fairly close to 1 (say, $\phi = 0.9$). On the other hand, this reduction in the number of link QoS state access/update is offset to some degree by the overhead of quota allocation/de-allocation. Hence, judicious choice of quota size is important in controlling this overhead and balancing the overall access/update reduction. This issue will be investigated in our simulation study in the next section.

Before we leave this section, we comment that the complexity of the PoQ scheme can be analyzed formally using queueing theory. In particular, under the assumption of exponential flow arrival and departure, the PoQ scheme can be modeled as a Markovian system, and the probabilities γ, ϕ, φ and χ can be derived either precisely or approximately. A key result from the analysis is that as the network capacity increases (thus the number of flows that can be accommodated also increases), the probability ϕ that a path enters the critical mode decreases, while the normalized network load (defined as the ratio of the offered load to the network capacity) is fixed. This observation is also supported by our simulation results, as we will see in the next section. Hence the PoQ scheme indeed improves the scalability of the centralized bandwidth broker model.

3.3 Simulation Investigation

In this section, we conduct simulations to study the performance of the basic PoQ scheme. In particular, we will investigate the impact of quota size on the performance of the scheme and its scaling property as the network capacity increases.

Since using the PoQ scheme the QoS states of a link are only accessed/updated when the link becomes critical, in our simulations, we use a simple network topology with one bottleneck link to study the number (or cost) of link QoS state accesses and updates. This simple topology allows us to focus on the key features of the PoQ scheme and provide an adequate environment to explore its performance. The network topology is shown in Fig. 6, where K ingress routers, $I1, I2, \ldots, IK$, are connected via a core router $R1$ to an egress router $E1$. The link $R1 \rightarrow E1$ is the *bottleneck* link of the network topology, and the links $Ii \rightarrow R1$

are assumed to have infinite capacity, $i = 1, 2, \ldots, K$. Flows arriving at the ingress routers have an exponential inter-arrival time with its mean denoted by $1/\lambda$, and an exponential holding time with its mean denoted by $1/\mu$. In our simulations the mean flow holding time $1/\mu$ is fixed at 900 seconds, while we vary the mean flow inter-arrival time to produce different *offered load.* The offered load, ρ, defined as λ/μ, represents the *average* number of flows that may exist in a system (if no flows are blocked). Each flow requests one unit of bandwidth, and the bottleneck link $R1 \rightarrow E1$ has C units of bandwidth. Hence the maximum number of flows that can be accommodated by the bottleneck link is C. We introduce the *normalized* network load, a, defined as ρ/C, as a metric for measuring how heavy the bottleneck link is loaded. For example, if $a < 1$, i.e., the offered load (the average number of flows that may exist at any time) is less than what can be accommodated by the bottleneck link, the system is considered not overloaded. Otherwise, the network system is considered overloaded. In our simulation study, all simulations last 10000 simulated seconds, of which 6000 seconds are the warm-up time. Each value reported in the results is the mean of 5 simulation runs with different random seeds for the mean flow inter-arrival time†.

In the first set of simulations, we examine the impact of quota size on the performance of the scheme. In this set of simulations, the bottleneck link capacity C is 5400. The number of paths K is set to 3, i.e., we have three ingress routers, $I1$, $I2$, and $I3$. Flow arrivals are uniformly distributed onto the three ingress routers. We conduct simulations using five different quota size, namely, 30, 60, 100, 120, and 150. The simulation results are summarized in Table 1 under two different normalized loads ($a = 0.95$ and $a = 1.00$). In this table, we list the total number of flow arrivals at all the ingress routers, the total number of flows accepted into the network system, and among the flows accepted, the total number of flows accepted in the normal mode as well as the total number of flows accepted in the critical mode. We also list the total number of quota allocation/de-allocation operations performed by the bandwidth broker after the warm-up period (i.e., when the network system is in a more stable state).

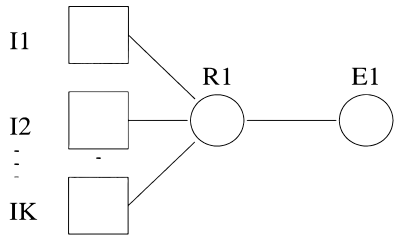


Fig. 6 Network topology used in the simulations.

From Table 1 we see that in the case of $a = 0.95$, i.e., the network is relatively light-loaded, the majority of the flows are accepted in the normal mode. In particular, when the quota size is 30 and 60, respectively, all flows are accepted in the normal mode, whereas when the quota size increases to 100, 120 and 150, only a few hundreds of flows are accepted in the critical mode. Hence, in this light-load case, the proportion of calls accepted in the critical mode is very small. In contrast, in the case of $a = 1.00$, i.e., the network is heavily loaded, the proportion of flows accepted in the critical mode increases significantly. In particular, when the quota size becomes large, the majority of flows are accepted in the critical mode. Figure 7 shows the proportion of flows accepted in the critical mode with five different quota size, as the normalized network load increases. We see that when the network is relatively light-loaded (say, $a \leq 0.95$), the quota size has little impact on the portion of the flows accepted in the critical mode. However, as the network load increases, the impact of the quota size is more significant. In particular, when the normalized load reaches $a = 1.00$, more than half of the flows are accepted in the critical mode for quota size of 60 or larger. Hence when the network is overloaded, the quota size has a significant impact on the performance of the PoQ scheme. It is also interesting to observe that in the heavy-load cases, further increasing the quota size beyond a certain value (say, 100) does not seem to have much further impact†.

We now shift our attention to the cost of the PoQ scheme, namely, the expected number of link QoS state access/update. To simplify discussion, we focus on the number of link QoS state update incurred by

†Given that each value is the mean of 5 simulation runs, the data reported in the paper is only used for the illustration purpose to demonstrate the properties of the schemes under different conditions.

Table 1 Dynamics of flow processing and quota allocation during admission control ($C = 5400$).

Normalized load	$a = 0.95$					$a = 1.00$				
Quota size	30	60	100	120	150	30	60	100	120	150
Total flow arrivals	22946	22946	22946	22946	22946	24099	24099	24099	24099	24099
Total accepted flows	22946	22946	22946	22946	22946	23878	23878	23878	23878	23878
Flows accepted in normal	22946	22946	22570	22464	22395	17519	11204	7582	7370	7319
Flows accepted in critical	0	0	376	482	551	6359	12674	16296	16508	16559
Quotas allocated	736	396	220	155	39	499	114	6	0	0
Quotas de-allocated	739	397	222	156	39	499	114	6	0	0

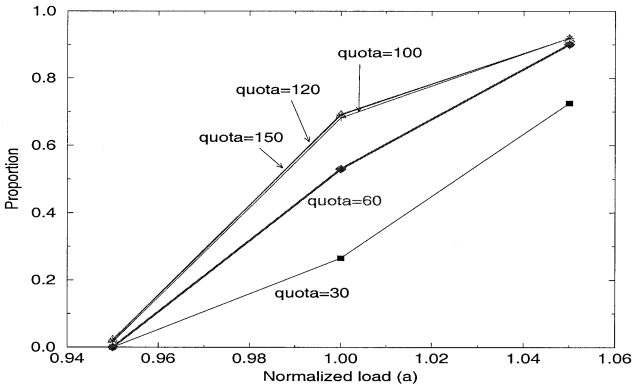


Fig. 7 Proportion of flows accepted in critical mode ($C = 5400$).

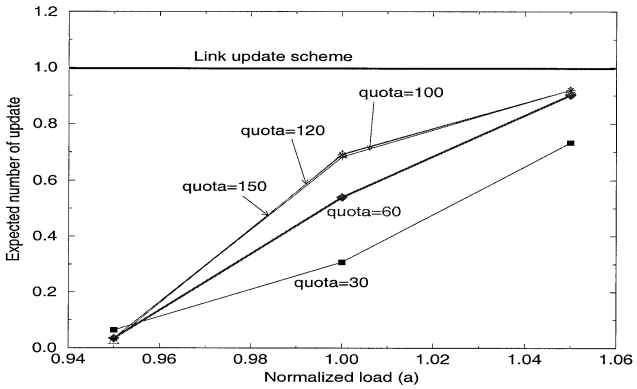


Fig. 8 Expected number of link-level QoS update per-accepted-flow ($C = 5400$).

flow arrivals and the overhead of quota allocation/de-allocation. Hence, instead of using the expected number of link QoS state access/update for each flow (i.e., Θ_{PoQ} defined in Sect. 3.2), we use a simplified cost metric, the expected number of link QoS state update for *accepted* flows, defined below:

$$\hat{\Theta}_{PoQ} = \frac{M + G + L}{N}, \tag{2}$$

where N denotes the total number of accepted flows, M the number of flows accepted in the critical mode, and G and L denote a number of quota allocations and de-allocations, respectively.

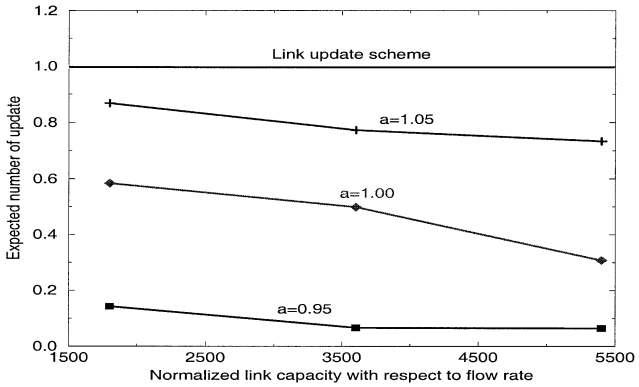
Figure 8 shows the expected number of link QoS state update per-accepted-flow as a function of the normalized network load for various quota size. The bottleneck link capacity C is set to 5400. From the figure we see that when the normalized network load is below 0.98, the expected cost of link QoS state update per-accepted-flow is less than 0.5 for all the quota size. Hence, on average, more than half of the flows accepted do not require any link QoS updates. Even when the network is heavily overloaded, say, $a = 1.03$, the expected number of link QoS state update per-accepted-flow is still less than 0.8. In other words, the PoQ scheme is capable of reducing the overhead of per-flow processing even during heavy load scenarios. In general, smaller quota size tends to have better performance when the network is heavily loaded. This is because the link QoS update cost is dominated by the cost incurred by flows accepted in the critical mode. On the other hand, when the network is not heavily loaded (say $a = 0.95$), smaller quota size (say 30) actually incurs more overheads because of frequent quota allocation/de-allocation operations. These observations are also supported by the data shown in Table 1.

To demonstrate the scalability of our PoQ dynamic bandwidth allocation scheme, we examine how the expected number of link QoS state update per-accepted-flow changes as we increase the network capacity (in this case the bottleneck link capacity C). The results are plotted in Fig. 9 for two different quota sizes: (a) 30 and (b) 100. From the figures, we see that as the network capacity increases, the expected link level QoS update cost per-accepted-flow decreases. This is actually not too surprising (see our comments at the end of Sect. 3.2): with the normalized network load fixed, the probability that a flow is accepted in the critical mode decreases as the link capacity increases, due to the increased multiplexing gain. These results demonstrate that *the PoQ scheme scales well as the network capacity increases.* This is particularly the case, when the network is not heavily overloaded. When the network is heavily loaded, our scheme still leads to some amount of cost reduction (especially with appropriately chosen quota size)—albeit not as significant as when the network is not heavily loaded. Note that when the network is heavily overloaded, some slow-down in flow request processing may not be a severe problem, since the network *itself* may not be capable of accommodating all the incoming flow requests. Furthermore, in this case we can use an extended PoQ scheme (the lossy-path PoQ scheme introduced in the next section) to further improve the flow processing capability of the bandwidth broker.

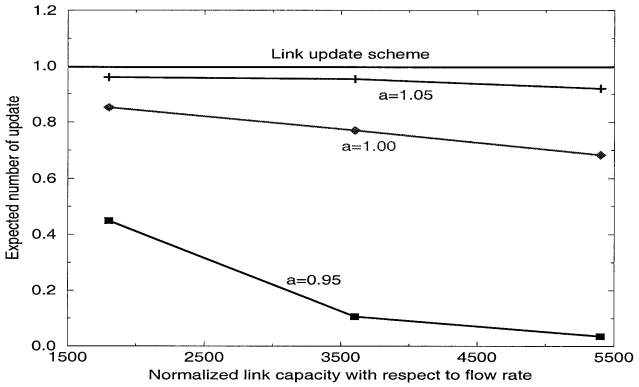
Lastly, we consider the impact of the number of paths sharing a bottleneck link on the performance of the PoQ scheme. Figure 10 shows the proportion of flows accepted in critical mode as we increase the number of paths sharing the bottleneck link. In this set of simulations the normalized load a is set to 0.95. Note that when there are a small number of paths, most of the flows can be accepted in the normal mode. But when the number of paths are large, large quota

[†]One possible reason may be that when the quota size is beyond certain threshold, a path is not able to release a quota as flows come and leave after the system enters a stable state. Therefore using any quota size beyond the threshold will result in the similar results on the proportion of calls accepted in the critical model. More investigation is needed to verify this point.

size causes more flows to be accepted in the critical mode. This is because there are not enough quotas to go around among all the paths. As a general rule-of-thumb, in order to make the PoQ scheme work efficiently, the ratio of the number of quotas a link has over the number of the paths sharing the link should be reasonably large. In particular, in a network where many paths sharing a bottleneck link, smaller quota size is preferred.



(a) Quota size = 30.



(b) Quota size = 100.

Fig. 9 Expected number of link QoS state updates as the network capacity increases.

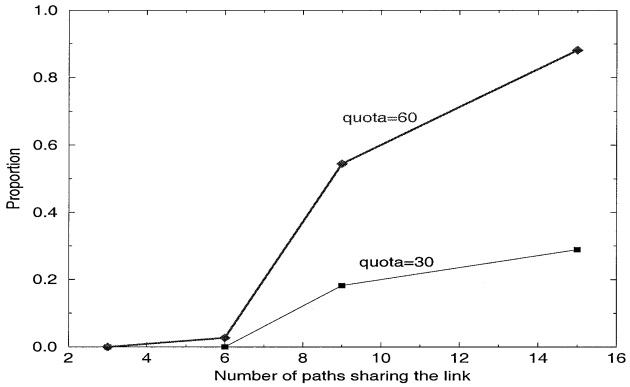


Fig. 10 Proportion of flows accepted in critical mode as the number of paths increases ($C = 5400$, $a = 0.95$).

4. Multiple Bandwidth Broker Design

In this section we extend the centralized bandwidth broker architecture with a single bandwidth broker to a hierarchically distributed architecture with multiple bandwidth brokers. This multiple bandwidth broker (MBB) architecture addresses the scaling problem posed by the potential communication bottleneck between the bandwidth broker system and the edge routers. The MBB architecture is presented in Sect. 4.1, where an extended PoQ mechanism—the *lossy-path* PoQ dynamic bandwidth allocation scheme—is also introduced to further reduce the call processing overhead at the central bandwidth broker. Simulation results are presented in Sect. 4.2.

4.1 Architecture and Mechanisms

The hierarchically distributed multiple bandwidth broker architecture we propose is designed based on the two-level network QoS representation and the PoQ dynamic bandwidth broker architecture. As illustrated in Fig. 11, the proposed MBB architecture consists of a *central* bandwidth broker (cBB) and a number of *edge* bandwidth brokers (eBBs). The central bandwidth broker maintains the link QoS state database and manages quota allocation and de-allocation among the edge bandwidth brokers. Whereas, each of the edge bandwidth brokers manages a *mutually exclusive* subset of the path QoS states and performs admission control for the corresponding paths. The number of eBBs can vary, depending on the size of the network domain. In the extreme case, for example, we can have one eBB for *each* edge router (as shown in Fig. 11), and the eBB can co-locate at the edge router.

When a flow arrives at an edge router, the flow reservation set-up request is forwarded by the edge router to the eBB that is in charge of the flow's path. The eBB will make admission control based on the path state it maintains such as the currently available band-

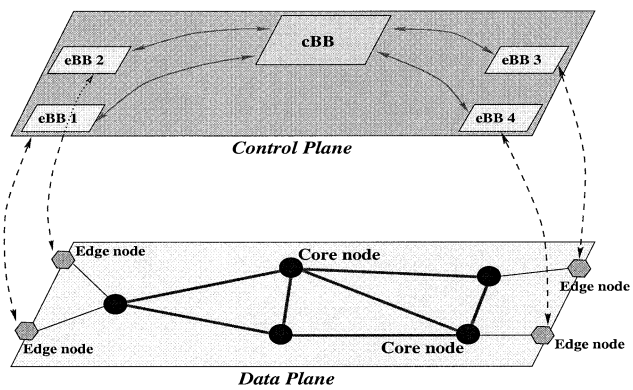


Fig. 11 Multiple bandwidth brokers on the control plane for a network domain.

width allocated to the path. If no sufficient bandwidth is available on the path, the eBB requests a new quota for the path from the cBB. If the request is granted, the eBB admits the flow and updates its path QoS state. If the request fails (i.e., one or more links along the path are critically loaded), we can operate just like the basic PoQ scheme: the eBB forwards the flow reservation request to the cBB, which will perform admission control using the per-flow link-update scheme. We refer to this eBB operation model the *non-lossy-path* model, as no flows will ever be rejected by the eBB, based on its path QoS state. We now introduce an alternative eBB operation model—the *lossy-path* model. Under this model, when a quota request fails, the eBB will simply reject the flow reservation request, instead of passing it to the cBB. We refer to the PoQ dynamic bandwidth allocation scheme under the lossy-path model the *lossy-path PoQ* scheme. With the lossy-path PoQ scheme, the role of cBB is much simpler: *it performs only quota management,* and all admission control decisions are now delegated to the eBBs. Combining the proposed MBB architecture with this lossy-path PoQ scheme, not only can we minimize the communication bottleneck to the cBB, but also we can significantly reduce the processing burden at the cBB. This is particularly desirable in a large network with high traffic intensity. Clearly, the enhanced scalability of the architecture is gained at the expense of some loss in performance, as some flows that are rejected may be accommodated if the non-lossy-path model is used. In the next section we will investigate the performance implication of the lossy-path MBB architecture model.

Before we move on to the simulation investigation, we would like to comment on some of the advantages of the proposed MBB architecture. Note that a straightforward approach to building a distributed bandwidth broker architecture to avoid the communication bottleneck problem would be a replicated bandwidth broker system, with multiple identical bandwidth brokers geographically dispersed in the network domain. However, due to the need to both access and update the network QoS states, maintaining *consistent QoS state databases* requires synchronization among the bandwidth brokers, which can be time-consuming and problematic. In contrast, our hierarchically distributed bandwidth broker architecture does not suffer such a problem, owing to the appropriate partition of the path QoS states and the PoQ dynamic bandwidth allocation mechanism we employ.

4.2 Simulation Investigation

In this section, we conduct simulations to study the performance of the MBB architecture using the lossy-path PoQ dynamic bandwidth allocation scheme. We use the same network topology as shown in Fig. 6, with the number of paths traversing the bottleneck link set to 3. We assume that there is an eBB associated with each path. The normalized link capacity with respect to the flow rate is 5400. All flows have an exponential holding time with a mean of 900 seconds. We again vary the flow arrival rate to produce different network loads.

Recall that under the lossy-path MBB architecture, an eBB will reject a flow when its request to the cBB for a new quota fails, i.e., when the bottleneck link has no quota left. Note that in this case, the total unreserved bandwidth (in b/s) on the link *may* be sufficient to accommodate the flow, since it is possible that not all the paths have used up the bandwidth allocated to it. Hence, in general, the lossy-path model may result in a higher flow blocking rate than the non-lossy-path model. Figure 12 shows the flow blocking rates of the lossy-path model with three different quota sizes, as we vary the network load. The flow blocking rate of the non-lossy-path model is also plotted for comparison. We see that when the normalized network load is below 0.95, all flows are accepted under all the schemes. As the load is increased, a small portion of flows is rejected. The lossy-path model suffers some performance loss compared to the non-lossy-path model. As expected, the larger the quota size is, the bigger the performance loss is. In addition, the performance loss enlarges as the network load increases. However, after the normalized network load reaches 1, the performance loss does not seem to increase visibly, in particular for the two larger quota size. This is likely due to the fact that once the network is overloaded, a large portion of those flow reservation set-up requests that are forwarded by the eBBs to the cBB for admission control under the non-lossy-path model end up being rejected by the cBB. Hence rejecting these flow requests at the eBBs does not degrade the system performance significantly, in particular when the network is highly overloaded. Overall, we observe that the performance loss caused by the lossy-path model is fairly small. We believe that at the expense of a relatively

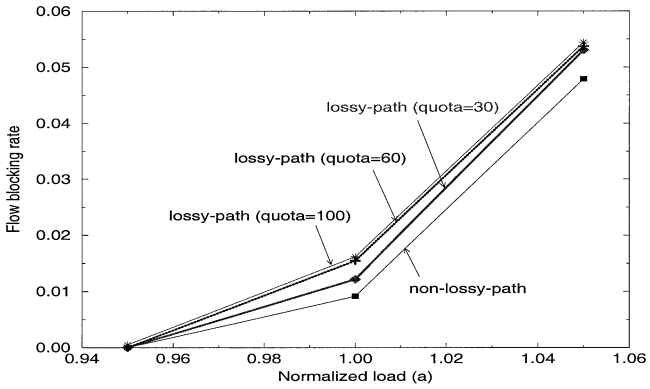


Fig. 12 Flow blocking rate of the non-lossy-path and lossy-path models ($C = 5400$).

small performance loss, the reduced bandwidth broker system overhead may well be worthwhile, in particular when the network system itself is overloaded.

We comment on the advantage of our dynamic bandwidth (in quota) allocation. A straight forward approach to provision bandwidth on a path is the static bandwidth allocation based on measurement of the long-term traffic load on the path. However, such a static provisioning procedure may not be able to reflect the dynamics of bandwidth fluctuation and requirement on the path in a smaller time scale–leading to either permanent bandwidth over-provisioning or temporary bandwidth under-provisioning along the path. The dynamic quota allocation under the proposed hierarchical MBB architecture is capable of adjusting bandwidth allocation on a path according to the offered load along the path. This is particularly important, as inadequate bandwidth allocation under static provisioning can cause flows being rejected unnecessarily, and thus degrade the system performance.

5. Further Enhancements

As we discussed in Sects. 3 and 4, the performance of the PoQ scheme depends critically on the quota management used in the scheme. For example, to control the impact of the scheme on the flow blocking rate of the system, smaller quota size is preferred; on the other hand, smaller quotas introduce more overhead into the system, because of potentially more frequent quota allocation/de-allocation. In this section, we propose several extensions to the quota management of the PoQ scheme, aiming to improve the performance and enhance the flexibility of the PoQ scheme under different environments. These extensions can be applied to both the lossy-path and non-lossy-path models. In the simulation study presented here, however, we will mainly focus on the lossy-path model. Throughout this section, we use the same network topology as depicted in Fig. 6 for the simulations, where the number of paths traversing the bottleneck link is set to three.

5.1 PoQ with Hysteresis

In the PoQ scheme that we presented in Sects. 3 and 4, a path will immediately return an excess quota to the links along the path whenever there is one available. In this way, the returned excess quota can be used by other paths which share a link with the path. However, it is possible that a path has an excess quota only because of *short-term, local* flow fluctuations. That is, the path may need to request a quota shortly after it returns an excess quota to the links.

To have a better understanding of this phenomenon, we introduce the notion of *quota state transition rate* of a path as follows. We say a path is in a quota state i if currently i quotas are allocated on the path. Let T_i denote the total amount of time that a path stays in a quota state i in an arbitrary time interval. Let $N_{i,j}$ be the number of transitions from quota state i to state j during the same time interval. Then the quota state transition rate $\eta_{i,j}$ of the path from state i to j in the time interval is defined as,

$$\eta_{i,j} = \frac{N_{i,j}}{T_i}. \tag{3}$$

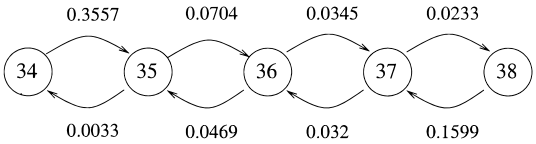


Fig. 13 Quota state transition rate ($C = 3600$, $a = 0.9$, $quota = 30$).

Figure 13 shows the quota state transition rates of the path $I1 \rightarrow E1$ after warm-up of the simulation, where the capacity of the bottleneck link is set to 3600, the normalized network load is 0.9, and the quota size is 30. All flows have an exponential holding time with a mean of 900 seconds. From the figure we see that the quota state transition rates from the state 34 to 35 and from the state 38 to 37 are much higher than the transition rates between other states. A close examination of the trace data reveals that most of the time the path has 35, 36 or 37 quotas and only occasionally goes to the states 34 and 38. Moreover, when the path quota state goes to 34 or 38, it will only stay there for a very short time period. Such a short time state visit is caused by the short-term local flow fluctuations where a small number of consecutive flow arrivals and departures triggers both quota allocation and de-allocation in a very short time. Note that short-term local flow fluctuations also occur in the state transitions between 35, 36 and 37 even though they are not exposed in this simple figure.

Recall that the operations of quota allocation/de-allocation involve *per-link* QoS state updates along the path, which increases the overall overhead of the system. Therefore, we should reduce the number of quota allocation/de-allocation as much as possible. One simple strategy is to allow a path to hold excess quotas without returning them. However, this may cause higher flow blocking rates on other paths which share a link with this path due to a shortage of available quotas, which is undesirable.

To regulate the behavior of a path in handling excess quotas, we develop the following simple mechanism based on the notion of *hysteresis:* each path will maintain a threshold to determine if the path should return an excess quota. Instead of returning an excess quota immediately after it is available, a path will only return an excess quota after the reservation rate along the path is below some threshold with respect to the current allocated quotas on the path. More formally,

let $C^{\mathcal{P}}$ denote the bandwidth allocated on a path $\mathcal{P}$, and $R^{\mathcal{P}}$ be the total reservation rate of flows along the path, respectively. Then the path will only return an excess quota if,

$$R^{\mathcal{P}} \leq C^{\mathcal{P}} - (1+\varepsilon)B^q, \tag{4}$$

where B^q is the bandwidth of a quota, and $\varepsilon \geq 0$ is the hysteresis term.

The intuition of the hysteresis-based scheme is quite straightforward. Instead of returning an excess quota whenever it is available, the hysteresis-based scheme ties the policy of quota de-allocation of a path to the (flow) traffic load on the path. When the reservation rate on a path is below some threshold, it is possible that the path will not use the excess quota in a relatively long time interval. Therefore, it is desirable to release the excess quota so that it can be re-used by other paths.

In the following, we conduct simulations to study the effectiveness of the hysteresis-based scheme. The normalized capacity of the bottleneck link with respect to the flow rate is 3600. All flows have an exponential holding time with a mean of 900 seconds. We vary the flow arrival rate to produce different network loads. In these simulations, the value of ε is set to 0.05.

Figure 14 shows the flow blocking rates of the hysteresis-based PoQ scheme under the lossy-path model for two quota size: 30 and 60. For comparison, we also include the corresponding curves of the PoQ scheme without hysteresis.

As discussed above, the motivation of the hysteresis-based mechanism is to reduce the amount of quota allocations/de-allocations of the PoQ scheme. Table 2 presents the corresponding quota allocation/de-allocation activities after the warm-up of the simulations. From the table we see that there is a significant reductions in quota allocation/de-allocation under the hysteresis-based mechanism. More specifically, compared with PoQ without hysteresis, there are roughly about 3 to 4 times reduction in the number of quota allocations/de-allocations in the hysteresis-based scheme.

Clearly, the performance and effectiveness of the hysteresis-based scheme relies on the value of ε of a path. If the value is too conservative, more flows may get rejected on other paths which share a link with the path because of a shortage of available quotas. On the other hand, an overly optimistic value may undermine the scheme because the threshold is not large enough to hold an excess quota which the path will need shortly after it is available, leading to frequent quota allocation/de-allocation. Figure 15 shows the flow blocking rate of the system for a case where the quota size is 60, and Table 3 presents the corresponding quota allocation/de-allocation activities of the simulations. Note that when the value of ε is 0.01, the hysteresis-based scheme has the same flow blocking rate (i.e., 0) as the PoQ scheme without hysteresis ($\varepsilon = 0$). However, as we can see from the table, when $\varepsilon = 0.01$, the amount of quota allocations/de-allocations is still quite high. As the value of ε increases to 0.2, the number of quota allocations/de-allocations drops dramatically with a slight increase in the flow blocking rate of the system. However, even we further increase the value of ε, the reduction in the amount of quota allocations/de-allocations is not much. Note that when

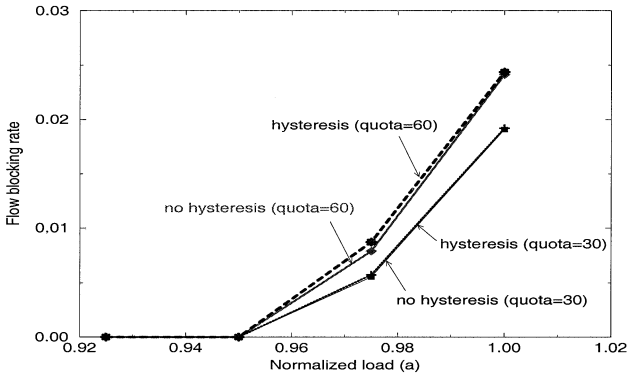


Fig. 14 Effects of hysteresis on flow blocking rates ($C = 3600$, $\varepsilon = 0.05$).

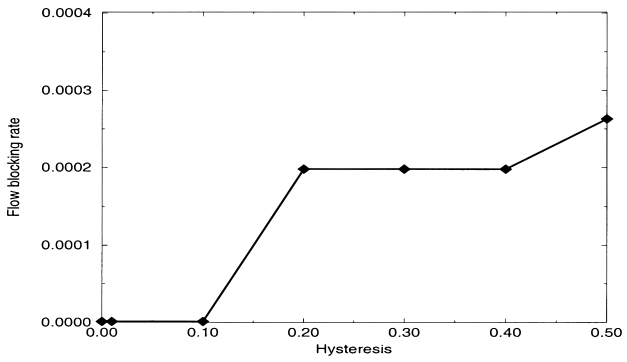


Fig. 15 Flow blocking rates with different hysteresis thresholds ($C = 3600$, $a = 0.95$, $quota = 60$).

Table 2 Effects of hysteresis on quota allocations and de-allocations ($C = 3600$, $\varepsilon = 0.05$).

Scheme	PoQ without hysteresis				PoQ with hysteresis			
Quota size	30		60		30		60	
Normalized load	0.925	1.000	0.925	1.000	0.925	1.000	0.925	1.000
Quotas allocated	517	424	361	123	168	167	92	43
Quotas de-allocated	523	424	364	123	173	167	94	43

Table 3 Effects of different hysteresis on quota allocations and de-allocations ($C = 3600$, $a = 0.95$, $quota = 60$).

Hysteresis (ε)	0	0.01	0.1	0.2	0.3	0.4	0.5
Quotas allocated	172	87	30	16	15	15	13
Quotas de-allocated	173	88	31	18	17	17	15

the values of ε are 0.2, 0.3 and 0.4, the curve of the flow blocking rate of the system is flat. This is partly caused by the fact that the threshold with $\varepsilon = 0.2$ has caught the trend in the flow fluctuation in some degree. That is, when the reservation rate of a path is below the threshold with $\varepsilon = 0.2$, it will soon be below the threshold with ε equal to 0.3 or 0.4. Therefore, they all have similar effects on the quota de-allocations (see also Table 3), and hence the similar flow blocking rates. From the above discussion we can see that over a range of modest values of ε, the system should work well in balancing between the reduction of the amount of quota allocations/de-allocations and the increase in the flow blocking rate of the system.

So far, we have discussed the employment of the hysteresis-based approach for a path to hold an excess quota to accommodate the short-term flow fluctuations. Note that it may be desirable for a path to employ a *forward threshold* to request an extra quota before the path has used up all the allocated quotas on the path [3], especially in the multiple BB architecture. By such a forward threshold, the path may handle flow requests immediately when the requests come instead of waiting for a new quota if the allocated quota bandwidth has been used up on the path.

5.2 PoQ with Variable Quota Size

Allocating bandwidth based on the notion of quota instead of per-flow bandwidth requests provides us with an efficient bandwidth allocation and admission control mechanism. However, as shown in Sects. 3 and 4, the performance and cost of the scheme is closely related to the size of the quota used in the scheme. In general, smaller quotas are preferred to minimize the flow blocking rate; on the other hand, to reduce the amount of quota allocation/de-allocation, larger quota size is preferred. In this section, we propose a scheme based on the notion of *variable* quota size to overcome the dilemma on how to properly choose a fixed quota size.

Let $\delta_1, \delta_2, \ldots, \delta_N$ denote the quota size that will be used in a network domain. Without loss of generality, we assume that $\delta_1 > \delta_2 > \ldots > \delta_N$. Corresponding to each δ_i, there is a link bandwidth allocation threshold θ_i, $i = 1, 2, \ldots, N$, where $\theta_i < \theta_{i+1}$ for $i = 1, 2, \ldots, N-1$. Let C denote the capacity of a link, B the current total bandwidth allocated to all the paths traversing the link. We define the quota allocation mechanism as follows: a quota of size δ_i is allocated upon a quota allocation request if

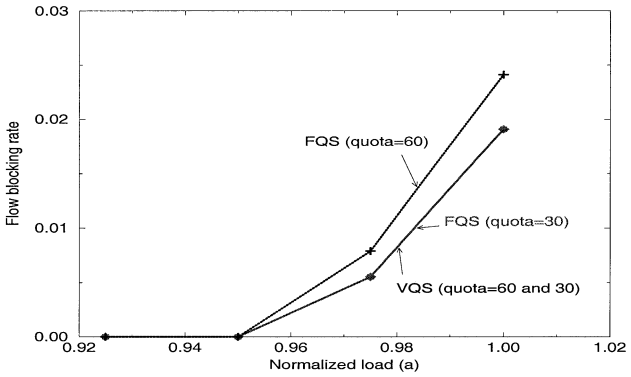


Fig. 16 Effects of variable quota size on flow blocking rates ($C = 3600$, $\theta_1 = 0.9$).

$$\theta_{i-1} < \frac{B}{C} \leq \theta_i$$

for $i = 1, 2, \ldots, N$, where we used the convention that $\theta_0 = 0$.

Figure 16 shows the flow blocking rates of the variable quota size (VQS) scheme, where the link capacity is set to 3600 with respect to the flow rate. In these simulations, the quota bandwidth allocation thresholds are 0.9 and 1. For example, when the configured quota size are 60 and 30 in a network domain, a link will allocate a quota of size 60 to meet a quota request if the proportion of the link bandwidth that has been allocated is less than or equal to 0.9. Otherwise, a quota with size 30 will be allocated. For comparison, the flow blocking rates of the original PoQ under fixed quota size (FQS) bandwidth allocation scheme are also included with a quota size of 30 and 60, respectively.

Note that the curve of the flow blocking rate of VQS with quota size 60 and 30 is almost identical with the curve of FQS with a quota size of 30. These results are not surprising because, when the allocated bandwidth goes beyond 90 percent of the link capacity, the smaller quota size is used under VQS, which is equal to the quota size used in the corresponding FQS scheme.

To examine the advantage of the VQS scheme in quota allocation, we present the quota allocation/de-allocation activities in Table 4 for both VQS and FQS. For VQS, the quota size are 60 and 30, while for FQS the quota size is 30. As shown from the table, when the normalized network load is 0.9, the VQS scheme has a relatively smaller amount of quota allocations/de-allocations compared with the FQS scheme. As the network load increases, both schemes have almost identical amount of quota allocations/de-allocations. Note that the threshold for the VQS scheme to allocate a

Table 4 Effects of the variable quota size scheme on quota allocations and de-allocations ($C = 3600$, $\theta_1 = 0.9$).

Quota size	quota = 30 (FQS)					quota = 60, 30 (VQS)				
Normalized load	0.900	0.925	0.950	0.975	1.000	0.900	0.925	0.950	0.975	1.000
Quotas allocated	483	517	538	523	424	419	510	536	523	424
Quotas de-allocated	483	523	540	524	424	419	516	538	524	424

quota of size 30 instead of 60 is 0.9. Therefore, when the network load is below or equal to 0.9, a quota with larger size (60) is used under the VQS scheme. Hence, the VQS scheme has a relatively small amount of quota allocations/de-allocations in these cases. As the network load increases, the VQS scheme will allocate a quota of smaller size (30), which is identical to the quota size in the FQS scheme. Therefore, they have almost the same amount of quota allocations/de-allocations when the network load is beyond the quota bandwidth allocation threshold (0.9).

From the above discussion and the simulation results, we note that by using variable quota size, the bandwidth broker system has more freedom to allocate large quota size when the network load is low. When the network load becomes higher, it can start to behave more conservatively on bandwidth allocation. In this way, it may reduce the cost of the system while achieving smaller flow blocking rate.

5.3 PoQ with Differentiated Flow Processing

In this section we demonstrate by an example that the bandwidth broker architecture provides us with great flexibility in managing network resource allocation and enforcing various management policies.

In both lossy-path and non-lossy-path models that we have studied, flows are treated equally based on the flow requests and the availability of the resources. For example, in the lossy-path model, as soon as there is no quota available at the central bandwidth broker, a flow will be rejected. However, under certain environment, it may be desirable to give differential treatment to flows based on some pre-defined policies. For example, some types of flows may be more valuable (in terms of revenue or profit margin) to an ISP than some other types of flows. Therefore, instead of rejecting these more valuable flows at an eBB because of the shortage of the quotas, the eBB may forward the flow requests to the cBB, and the cBB can conduct admission control for these flows based on different rules, and perhaps from a special pool of bandwidth reserved in advance.

As an example to illustrate the idea of the differentiated flow treatment with the PoQ scheme, we define two types of flows. One is the long-term *large* flows, and another is the short-term *small* flows. A large flow normally requires a large amount of bandwidth (possibly associated with a higher revenue or profit margin), and a small flow requests a relatively smaller amount of bandwidth. For example, we may consider video applications as large flows, while audio applications like IP telephony as small flows. Instead of conducting admission control for large flows at the edge BBs, an eBB may forward these flow requests to the central BB. The cBB will independently conduct the admissibility test using the *link-update* admission control scheme. That is, no large flows will be rejected as soon as there is enough bandwidth along the path. We call such a scheme as the PoQ with differentiated flow treatment (DFT), and the original lossy-path scheme as the PoQ with uniform flow treatment (UFT). Note that, the PoQ with DFT scheme can be considered as a hybrid of the lossy-path and non-lossy-path models: treating small flows with the lossy-path model while treating large flows with the non-lossy-path model. Moreover, because large flows are forwarded to the cBB, a smaller quota size can be used by the cBB to meet the quota requests from the eBBs (recall that a smaller quota is preferred to reduce the flow blocking rate).

The above PoQ with DFT example clearly demonstrates the *flexibility* of our hierarchical multiple bandwidth broker architecture, which is enabled and built upon the two-level (path and link) representation of the network QoS states.

6. Conclusion

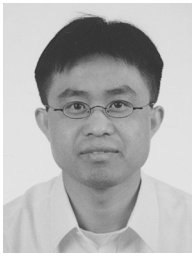
In this paper we studied the scalability issue in the design of a centralized bandwidth broker model for dynamic control and management of QoS provisioning. We identified two major factors that may potentially affect the scalability of the centralized bandwidth broker architecture: the memory and disk access speed and communication capacity between the bandwidth broker and edge routers. To reduce the overall number of QoS state accesses and updates, we developed a path-oriented quota-based (PoQ) dynamic bandwidth allocation mechanism for efficient admission control operations under the centralized bandwidth broker model. Based on the proposed dynamic bandwidth allocation mechanism, we also extended the centralized bandwidth broker architecture to a hierarchically distributed architecture with multiple bandwidth brokers to address the scaling problem posed by the potential communication bottleneck between the bandwidth broker system and edge routers. Our simulation investigation demonstrated that the proposed PoQ dynamic bandwidth allocation mechanism is indeed an effective means to increase the overall call processing capability of the bandwidth broker. Furthermore, the bandwidth

broker architecture can be designed in such a manner that it scales with the increase in the network capacity. Further extensions to the PoQ scheme were also investigated to improve the performance of the PoQ scheme and to enhance the flexibility of the bandwidth broker architecture.

References

[1] P. Aukia, M. Kodialam, P.V.N. Koppol, T.V. Lakshman, H. Sarin, and B. Suter, "RATES: A server for MPLS traffic engineering," IEEE Network Magazine, pp.34–41, March/April 2000.
[2] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," RFC 1633, Internet Engineering Task Force, June 1994.
[3] L. Golubchik and J. Liu, "A fast and accurate iterative solution of a multi-class threshold-based queueing system with hysteresis," Proc. ACM SIGMETRICS, Santa Clara, CA, June 2000.
[4] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," RFC 2638, Internet Engineering Task Force, July 1999.
[5] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," Proc. ACM SIGCOMM, pp.81–94, Cambridge, MA, Aug. 1999.
[6] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A two-tier resource management model for the Internet," Proc. IEEE Global Internet, Dec. 1999.
[7] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A new resource reservation protocol," IEEE Network Magazine, vol.7, no.5, pp.8–18, Sept. 1993.
[8] Z.-L. Zhang, Z. Duan, L. Gao, and Y.T. Hou, "Decoupling QoS control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services," Proc. ACM SIGCOMM, pp.71–83, Stockholm, Sweden, Aug. 2000.

Zhi-Li Zhang received a B.S. degree in Computer Science from Nanjing University, China and his M.S. and Ph.D. degrees in Computer Science from University of Massachusetts, Amherst, in 1992 and 1997 respectively. In January 1997, he joined the Computer Science faculty at the University of Minnesota, where he is currently an Assistant Professor. From 1987 to 1990, he conducted research in the Computer Science Department at Århus University, Denmark, under a fellowship from the Chinese National Commission on Education. Dr. Zhang's current research interests include high speed networks, multimedia and real-time systems, and modeling and performance evaluation of computer and communication system. He received the National Science Foundation CAREER Award in 1997. He is also a co-recipient of 1996 ACM SIGMETRICS Conference best paper award, and a member of IEEE, ACM and INFORMS.



Zhenhai Duan received a B.S. degree from Shandong University, China, and M.S. degree from Peking University, China, in 1994 and 1997, respectively, both in Computer Science. Currently he is pursuing a Ph.D. degree in the Department of Computer Science and Engineering at the University of Minnesota. His research interests are in the areas of computer and communications networks, currently in QoS provisioning over Internet, traffic measurements and modeling. Mr. Duan is a student member of IEEE and ACM.

Yiwei Thomas Hou obtained his B.E. degree (*Summa Cum Laude*) from the City College of New York in 1991, the M.S. degree from Columbia University in 1993, and the Ph.D. degree from Polytechnic University, Brooklyn, New York, in 1998, all in Electrical Engineering. He was awarded a five-year National Science Foundation Graduate Research Traineeship for pursuing Ph.D. degree in high speed networking, and was recipient of the Alexander Hessel award for outstanding Ph.D. dissertation (1997–1998 academic year) from Polytechnic University. Since September 1997, Dr. Hou has been a Research Scientist at Fujitsu Laboratories of America (FLA), Sunnyvale, California. At FLA, he has been conducting and leading research efforts in the areas of scalable architecture, protocols, and implementations for differentiated services Internet; optical networking; quality of service (QoS) support for multimedia over wired and wireless IP-based networks. He has received a few corporate awards for intellectual property contributions. Dr. Hou has authored or co-authored over 50 refereed papers, including over 20 papers in major international journals, and a few patents pending. He is a member of the IEEE, ACM, Sigma Xi, and New York Academy of Sciences.