On Scalable Network Resource Management Using Bandwidth Brokers

Zhi-Li Zhang, Zhenhai Duan
Dept. of Computer Science & Engineering
University of Minnesota
Minneapolis, MN 55455
{zhzhang,duan}@cs.umn.edu

Y. Thomas Hou
Fujitsu Laboratories of America
595 Lawrence Expressway
Sunnyvale, CA 94085
thou@fla.fujitsu.com

Abstract

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.

Keywords

bandwidth broker, admission control, resource reservation, scalability, differentiated services, quality of service, Internet

1 Introduction

In the IETF Differentiated Services (DiffServ) framework, a centralized model based on the notion of *bandwidth broker* (BB) [3] 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, 5, 7].

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 [4]. This is in contrast to the IETF Integrated Services (IntServ) QoS control model based on RSVP [2, 6], 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 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 [7], 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.

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. To address the scaling problem caused by the potential communication bottleneck between the centralized bandwidth broker and edge routers, the path-oriented, quota-based dynamic bandwidth allocation approach is then extended to a *hierarchically distributed* bandwidth broker architecture. In particular, the hierarchically distributed architecture consists 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 Section 2, we first present a centralized bandwidth broker architectural model, and then discuss the potential scaling issues of the centralized bandwidth broker architecture. In Section 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 Section 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. We conclude the paper in Section 5.

2 Bandwidth Broker Architecture: the Basic Model

As the basis for our study, in this section, we present a basic centralized bandwidth broker architectural model and describe how *admission control* is performed under such a model. The basic centralized bandwidth broker model for the management and control of the QoS provisioning of a network domain is schematically depicted in Figure 1. This model is based on the bandwidth broker architecture proposed in [7]. 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 net-

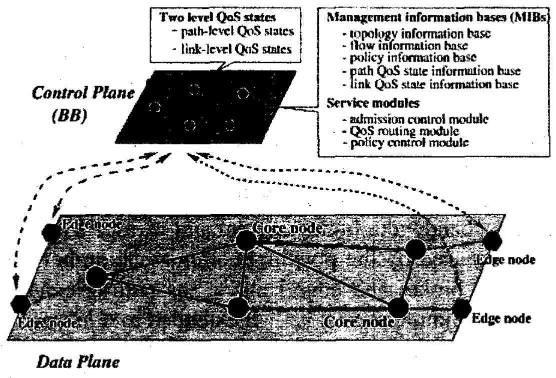


Figure 1: A bandwidth broker architecture.

work 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, 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 [7], by maintaining a separate path-level QoS states, the bandwidth broker can conduct fast admissibility test. Furthermore, path-wise resource optimization can also be performed based on the (summarized) path QoS state. As will be demonstrated later, *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[1] such as its QoS requirement and reserved bandwidth is maintained in a separate flow information database managed by the bandwidth broker [7].

[1] 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).

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.

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 illustrate how it can be employed to reduce the overall number of link QoS state accesses and updates, and then we present the simulation results to illustrate the efficacy of the scheme.

3.1 The Basic PoQ Scheme

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

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

Figure 2: Notation used in the algorithm.

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

Figure 3: Path level admission control.

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 re-claimed 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

```
Upon a path $p$ requests $r_p$ on a link $l$:
  /* $r_p$ can be a quota or a flow's request rate */
  **case 1:** ($op_l == 0$ and $aq_l < r_p$)
    collect residual bandwidth: $rb_l \leftarrow C_l - \sum_{p:l \in p} R_p$;
    **if** ($rb_l < r_p$) reject the request; return.
  **case 2:** ($op_l == 1$ and $rb_l < r_p$) reject the request; return.
  /* The request can be honored */
  **if** ($op_l == 0$ and $aq_l < r_p$)
      $op_l \leftarrow 1$; /* transition: normal $\rightarrow$ critical */
    **for** ($p' : l \in p'$)
      $cl_{p'} \leftarrow cl_{p'} \cup l$; $op_{p'} \leftarrow op_{p'} + 1$;
  **case 1:** ($op_l == 0$) $aq_l \leftarrow aq_l - 1$
  **case 2:** ($op_l == 1$) $rb_l \leftarrow rb_l - r_p$.
```

Figure 4: Link level bandwidth/quota allocation.

```
Upon an existing flow $f$ departs on a path $p$:
  $R_p \leftarrow R_p - r_f$;
  **if** ($op_p > 0$)
    **for** ($l \in cl_p$)
      $rb_l \leftarrow rb_l + r_f$; recompute $aq_l$;
      **if** ($aq_l \geq 0$) /* transition: critical $\rightarrow$ normal */
        **for** ($p' : l \in p'$)
          $op_{p'} \leftarrow op_{p'} - 1$; set $Q_{p'}$;
          $cl_{p'} \leftarrow cl_{p'} - l$;
  **else if** ($op_p == 0$ and $p$ has excess quota)
    $Q_p \leftarrow Q_p - 1$; /* return excess quota */
    **for** ($l \in p$)
      $aq_l \leftarrow aq_l + 1$;
```

Figure 5: Scheme for handling flow departure.

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

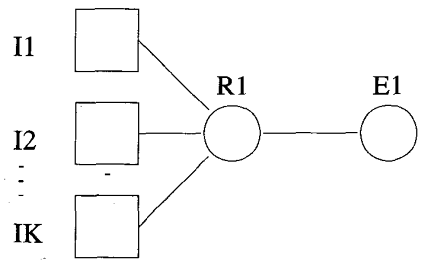


Figure 6: Network topology used in the simulations.

3.2 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 Figure 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. Flow arrivals are uniformly distributed onto the

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

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 $a = 1.00$).

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 over-loaded, 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 flow arrivals and the overhead of quota allocation/de-allocation. We use a simple cost metric, the expected number of link QoS state update for *accepted* flows, defined below:

$$\hat{\Theta}_{PoQ} = \frac{M + G + L}{N}, \tag{1}$$

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.

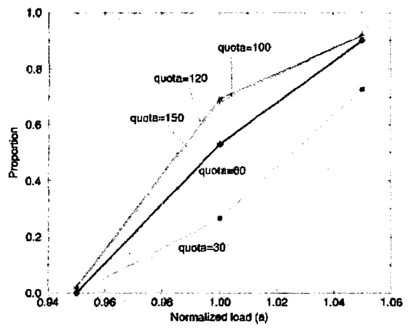


Figure 7: Proportion of flows accepted in critical mode.

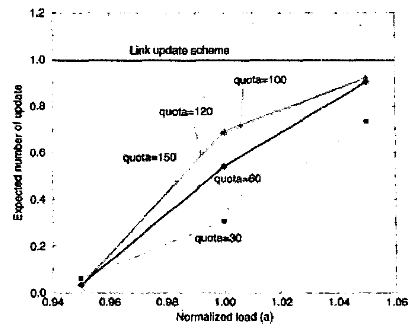
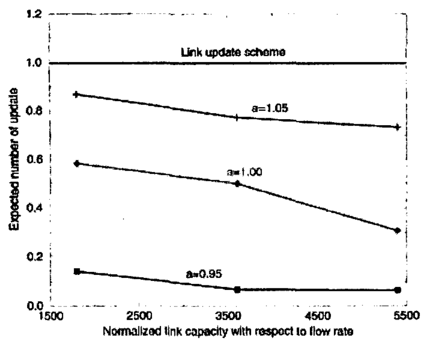


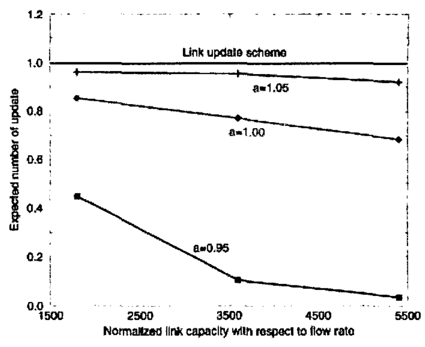
Figure 8: Expected number of link-level QoS update per-accepted flow.

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 Figure 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: 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



(a) Quota size = 30.



(b) Quota size = 100.

Figure 9: Expected number of link QoS state updates *vs.* network capacity

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 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.

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 Section 4.1, where an extended PoQ mechanism — the *lossy-path* PoQ dynamic bandwidth alloca-

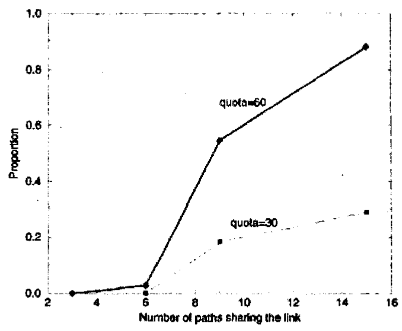


Figure 10: Effects of number of paths sharing a link (C = 5400, a = 0.95).

tion scheme — is also introduced to further reduce the call processing overhead at the central bandwidth broker. Simulation results are presented in Section 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 Figure 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 Figure 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 bandwidth 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

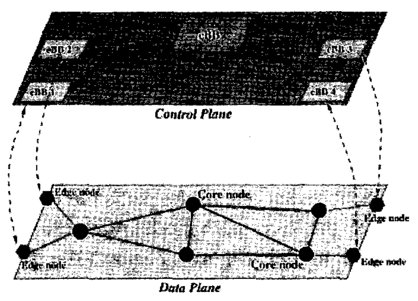


Figure 11: Multiple bandwidth brokers on the control plane for a network domain.

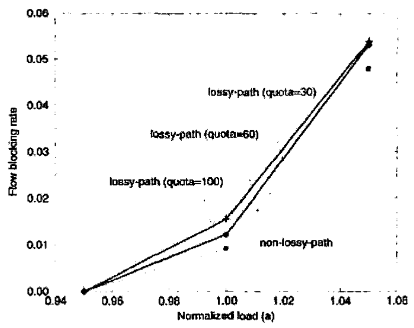


Figure 12: Flow blocking rates ($\check{C} = 5400$).

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 Figure 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 sizes. 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 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 provisioning bandwidth on a path is the static bandwidth allocation based on measurement of the long-term traffic load on the path. However, such static provisionings 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, in constrast, is capable of adjusting bandwidth allocation on a path according to the offered load along the path, and therefore, overcoming the above mentioned problems associated with static provisionings.

5 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.

To further improve the performance of the PoQ scheme and to enhance the flexibility of the bandwidth broker architecture, we have also investigated possible extensions to the basic scheme, for example, PoQ with variable quota sizes, treating long-term large flows (elephants) differently from short-term small flows (mice), and forward threshold for an eBB to request an extra quota and backward threshold for an eBB to de-allocate an excess quota. We studied their impacts on the performance of the system. However, due to the page limit constraint, we omit them here and refer interested readers to the technical report version of this paper [8].

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] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *RFC 2638,* Internet Engineering Task Force, July 1999.

[4] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in: *Proc. ACM SIGCOMM,* pp. 81–94, August 1999, Cambridge, MA.

[5] A. Terzis, L. Wang, J. Ogawa, and L. Zhang, "A two-tier resource management model for the Internet," in: *Proc. IEEE Global Internet,* Dec., 1999.

[6] 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.

[7] 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," in: *Proc. ACM SIGCOMM,* pp. 71–83, August 2000, Stockholm, Sweden.

[8] Z.-L. Zhang, Z. Duan, and Y.T. Hou, "On scalable design of bandwidth brokers," *Technical Report,* Dept. of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, July 2000.