Distributed QoS Routing for Backbone Overlay Networks

Li Lao[1], Swapna S. Gokhale[2], and Jun-Hong Cui[2]

[1] Computer Science Dept., University of California, Los Angeles, CA 90095
[2] Computer Science & Engineering Dept., University of Connecticut, CT 06029
llao@cs.ucla.edu, {ssg,jcui}@engr.uconn.edu

Abstract. In recent years, overlay networks have emerged as an attractive alternative for supporting value-added services. Due to the difficulty of supporting end-to-end QoS purely in end-user overlays, backbone overlays for QoS support have been proposed. In this paper, we describe a backbone QoS overlay network architecture for scalable, efficient and practical QoS support. In this architecture, we advocate the notion of QoS overlay network (referred to as QSON) as the backbone service domain. The design of QSON relies on well-defined business relationships between the QSON provider, network service providers and end users. A key challenge in making QSON a reality consists of efficiently determining routes for end user QoS flows based on the service level agreements between the QSON provider and network service providers. In this paper, we propose and present a scalable and distributed QoS routing scheme that can be used to efficiently route end user QoS flows through QSON. We demonstrate the effectiveness of our solution through simulations.

1   Introduction

With the dramatic advances in multimedia technologies and the increasing popularity of real-time applications, Quality of Service (QoS) support in the Internet has been in a great demand. However, due to many historical reasons, today's Internet primarily provides best-effort connectivity service. To enhance the current service model to provide QoS, researchers have proposed many seminal architectures represented by IntServ and DiffServ Unfortunately, realizing these QoS architectures in the Internet is unlikely to be feasible in the long run. In addition, there are no appropriate economic models for these architectures: although some ISPs might be interested in providing QoS in their own domains, there are no strong incentives for them to support QoS for users in other domains who are not their customers.

In the past few years, overlay networks have emerged as an alternative mechanism for supporting value-added services such as fault tolerance, multicasting, and security [3, 5, 12]. Many of these overlays are end-user overlays, namely, overlays are constructed purely among the end hosts without support from intermediate nodes. Due to the difficulties of supporting end-to-end QoS purely in end-user overlays, some recent work [7, 9, 15, 13, 16] proposes using backbone

overlays for QoS support, where overlays are managed by a third party provider such as an ISP.

In this paper, we adopt the approach of backbone overlays, and present a QoS overlay network architecture for scalable, efficient and practical QoS support. In this architecture, we advocate the notion of a QoS overlay network (referred to as QSON) as the backbone service domain. The design of QSON relies on well-defined business relationships between the QSON provider, network service providers (i.e., the underlying network domains which we also refer to as underlying ISPs for short), and end users: the QSON provider provisions its overlay network according to end user requests, purchases bandwidth from the network service providers based on their service level agreements (SLAs), and sells its QoS services to end users via service contracts. A key challenge in making QSON a reality consists of efficiently determining routes which satisfy the QoS requirements of end user flows based on the SLAs with the underlying ISPs. In a QSON, a QoS flow will routinely straddle multiple domains. Thus, existing QoS routing techniques which are designed primarily for flows within a single a domain may not be applicable in QSON. In this paper, we present a scalable and distributed QoS routing scheme that can be used to efficiently route end user QoS flows through QSON. We demonstrate the effectiveness of our solutions through simulations.

The layout of the paper is as follows. Section 2 describes the QSON architecture and discusses its main characteristics and advantages. Section 3 describes the routing scheme for QSON and provides a formal analysis of the scheme. Section 4 presents the simulation results to evaluate the scalability of the scheme. Section 5 provides an overview of the related work in the areas of QoS architectures and QoS routing. Section 6 offers concluding remarks and directions for future research.

2 QoS Overlay Network Architecture

In this section, we describe the QoS overlay architecture for scalable, efficient, and practical QoS support. In this architecture, a QSON (QoS Overlay Network) is constructed as the backbone service domain, which consists of many strategically deployed proxies by the QSON provider. The overlay paths between proxies are composed based on the SLAs between the QSON provider and the underlying ISPs. Outside the QSON, end hosts in access domains subscribe to the QSON by connecting to some edge proxies advertised by the QSON provider. A high level illustration of QSON is shown in Fig. 1. Though QSON is an overlay network across multiple domains, it is actually a single logical domain from the end user point of view. End user flows are managed by QSON, and the underlying ISPs only see "aggregated" flows traversing overlay paths.

2.1 Physical Network Structure

The underlying physical network structure from which QSON will be constructed is composed of multiple domains, each of which is managed by an underlying
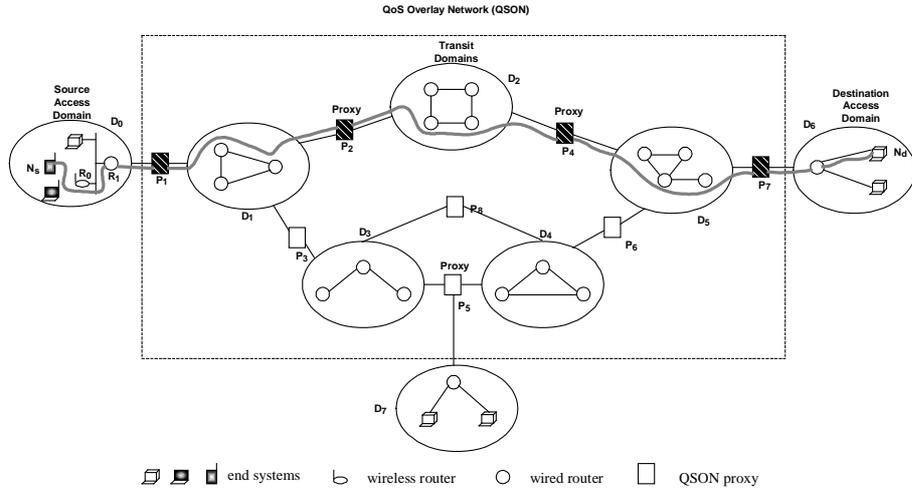
Fig. 1. The QSON Architecture.

ISP. QSON proxies are strategically deployed between domains, and each proxy may belong to multiple domains. We refer to non-proxy nodes as *internal nodes*. Internal nodes can only be linked to nodes (internal nodes or QSON proxies) within its domain, whereas a QSON proxy can be linked to nodes in all the domains it belongs to. For example, in Fig. 1, QSON proxy P_2 belongs to both domain D_1 and domain D_2. We refer to domains hosting end users as *access domains*, such as domains D_0, D_6, D_7, and other domains used for data delivery as *transit domains*, such as D_1 through D_5. The QSON proxies in access domains are called *edge proxies*, which are usually advertised to end users by the QSON provider. In addition, we refer to the edge proxy in the source (or destination) access domain as *source (or destination) edge proxy*. Furthermore, the two proxies in a transit domain along a path from the source to the destination are referred to as *ingress proxy* and *egress proxy*. As shown in Fig. 1, if a connection originates in access domain D_0 and terminates in access domain D_6, P_1 is a source edge proxy, and P_7 is a destination edge proxy. Also, P_1 and P_2 are respectively the ingress and egress proxies for domain D_1.

2.2 QSON Logical Network Structure

In order to route end user QoS flows through the QSON, for each domain the QSON provider needs to know about the possible alternate paths between ingress/egress proxy pairs and the amount of bandwidth available on each one of these paths, which can be obtained from the SLAs negotiated between the QSON provider and the underlying ISP. Note that the QSON provider does not need to know the underlying topology. The logical view of the QSON thus consists of paths between pairs of proxies in each domain, and each path may be annotated by the bandwidth allocated by the ISP to the QSON. It is worth pointing out

that the ISP may also provide some other information about the quality of the paths, such as the number of hops, which can be used by the QSON provider to guide the selection of one path if multiple suitable paths exist. Generally speaking, *the more information available for the possible paths in ISP domains, the better QoS support can be provided by the QSON provider to the end users*. In later sections, for the ease of presentation, we mainly use the bandwidth metric along with the number of hops to demonstrate our routing scheme. We have also investigated how the QSON can be incrementally deployed in the current "best-effort" Internet using the metrics of delay/jitter, bandwidth capacity, etc. and implemented a prototype system in PlanetLab. Due to space limit, we will not present these results in this paper. Interested readers can find more in our technical report [6].

2.3 Network State in QSON

The QSON uses the bandwidth allocated along the paths between the proxies to route end user QoS flows. As end user flows arrive and depart, the amount of available bandwidth along each one of the paths between the proxies changes dynamically. To handle such dynamic situations, each QSON proxy maintains the amount of available bandwidth on all the paths to other proxies in the same domain. Each proxy also stores a list of logical paths to the proxies in other domain(s). A logical path between the pair of proxies which do not belong to the same domain consists of a sequence of proxy nodes. To limit the amount of information to be maintained, the QSON provider can eliminate some "lengthy" paths by defining an appropriate management policy. It is usually useless to maintain very lengthy paths: the end-to-end delay may become too long and end users may not be satisfied with the service even though the available bandwidth meets the basic request. For example, in Fig. 1, proxy P_1 may know about the following logical paths $P_1P_2P_4$, $P_1P_2P_4P_7$, $P_1P_3P_5$, $P_1P_3P_5P_6$, $P_1P_3P_5P_6P_7$, $P_1P_3P_8$, $P_1P_3P_8P_6$, and $P_1P_3P_8P_6P_7$. However, the paths $P_1P_3P_5P_6P_4$ and $P_1P_3P_8P_6P_4$ may be eliminated, since these paths have twice the length (in terms of the number of logical hops) of the shortest logical path $P_1P_2P_4$. Thus, the QSON provider can pre-define a logical path length threshold lp_{th}. For a logical path between two proxies P_A and P_B, if its length is bigger than $d(1 + lp_{th})$, where d is the length of the shortest logical path between P_A and P_B, then this logical path is eliminated from P_A and P_B.

2.4 Advantages of QSON

The QSON architecture combines the benefits from overlay networks and QoS-aware IP networks. On the one hand, it does not require the global deployment of QoS-aware routers. On the other hand, it can take advantage of the information obtained from intermediate nodes (proxies) to facilitate QoS support. In addition, it offers many other advantages. First, unlike end user overlays (which can only support one application), a QSON provider can support a variety of applications simultaneously. This provides an additional incentive for ISPs to

adopt QSON. Second, it simplifies the management of resources in the underlying networks, since network service providers only need to provide services to a limited number of QSON providers instead of millions or billions of individual users. This is facilitated because QSON decouples the end user service management and network resource management. This level of traffic aggregation, in the long run, will make IntServ-like architectures practical.

3 Routing in QSON

In this section, we describe a distributed and scalable routing scheme, which is based on a probing technique, to efficiently route end user QoS flows in QSON.

3.1 Description of the Scheme

An end user QoS flow originates at the source node in the source access domain, traverses one or more QSON proxies in the transit domains, and terminates at the destination node in the destination access domain. In order to route such a QoS flow, an end-to-end path which satisfies the QoS constraints is obtained by composing the paths through the source and destination access domains and one or more transit domains in the QSON as explained below. In this section, we assume that the end user flow expresses its QoS constraints in terms of bandwidth for the purpose of demonstration.

Routing in Source Access Domain: To route an end user QoS flow, the source node forwards *probes* along all the existing paths to the source edge proxy within its domain. These probes are loaded with the bandwidth constraints of the flow. Each probe computes the bottleneck bandwidth of the path it traverses in the forward direction. Therefore, for each path between the source node and the source edge proxy, a probe will reach the source edge proxy. The source edge proxy then selects a suitable path that has sufficient bandwidth to satisfy the constraints of the connection. If multiple suitable paths are available, then one can be selected either randomly, or based on other criteria such as the number of physical hops along the path, or the actual bottleneck capacity of the path.

Routing Across Transit Domains: Departing from the source access domain, the probe is forwarded by the source edge proxy to other proxies in the same domain. The proxies chosen to forward the probes are such that they lie along the possible multi-domain logical paths leading to the destination edge proxy. To choose the possible multi-domain paths, we suggest a criteria of coarse-grained delay threshold based on the user request (especially if the user has explicit delay requirement): for a possible multi-domain logical path, the number of logical hops should not exceed the specified delay threshold. In this manner, the overhead incurred in forwarding the probe on paths that may not satisfy the user requirements is reduced. Before forwarding the probe to the next proxy, the source edge proxy composes the bandwidth of the path carried by the probe with the bandwidth of a suitable path between itself and the next proxy, and loads the probe with this bandwidth. A suitable path in a transit domain can be

selected based on criteria similar to those described for the selection of a path in the source access domain. If no path between the chosen ingress/egress proxy pair has sufficient bandwidth to satisfy the requirement, the probe is *pruned* and not forwarded further. Otherwise it is forwarded to the next proxy along the selected path. This continues until the probe reaches the destination edge proxy.

Note that, starting from the source edge proxy, it may be likely that multiple possible logical paths leading to the destination edge proxy exist and these paths share a common portion at the beginning. For example, in Fig. 1, $P_1 P_3 P_5 P_6 P_7$, and $P_1 P_3 P_8 P_6 P_7$ share the first logical link $P_1 P_3$ (for the case when P_1 is the source edge proxy, and P_7 is the destination edge proxy). In this case, the probe is forwarded only once along the shared path. This technique is called *probe aggregation*, which aids in the reduction of the overhead associated with forwarding the probes.

In summary, for each domain along a possible logical path, the ingress proxy forwards the probe to the egress proxy in the domain. Before forwarding the probe, the ingress proxy updates the bandwidth of the path carried by the probe with the bandwidth of a suitable path between itself and the egress proxy.

Routing in Destination Access Domain: When a probe reaches the destination edge proxy, it carries the bandwidth of a path between the source node and itself. The destination edge proxy then forwards the probe to the destination node along all the possible paths (probe flooding as in the source access domain). After receiving the probes, the destination node then selects a path based on the QoS metrics of the path(s) (i.e., the bottleneck bandwidth) and the bandwidth requirement of the connection.

3.2 An Illustrative Example

We explain the QSON routing scheme described above with the help of an example. Referring to Fig. 1, we consider a flow originating at the source node N_s in domain D_0 which is to be routed to the destination node N_d in domain D_6. The QoS constraints of the flow are expressed in terms of the required bandwidth, say b units. In order to route this flow, the source node N_s floods probes along the path $N_s R_0 R_1$ towards source edge proxy P_1. At node N_s, the bandwidth of the path $N_s R_0$ is compared with b units, and since this bandwidth is higher than b units, the probe is forwarded to node R_0. At node R_0, the bandwidth of the path $N_s R_0 R_1$ is composed, and upon determining that it is greater than b units, the probe is forwarded to node R_1. The same process is repeated at node R_1, and the probe is forwarded to the source edge proxy P_1.

At proxy P_1, three possible paths which satisfy the pre-specified delay threshold exist to the destination edge proxy P_7: $P_1 P_2 P_4 P_7$, $P_1 P_3 P_5 P_6 P_7$, and $P_1 P_3 P_8 P_6 P_7$. For the first path, proxy P_1 composes the bandwidth of the path $N_s R_0 R_1 P_1$ with the bandwidth of the path between itself and P_2, finds that the bandwidth of the entire path $N_s R_0 R_1 P_1 P_2$ to be greater than b units and hence forwards the probe to proxy P_2. The second and the third paths share a common egress proxy P_3. As a result, a single probe is forwarded to proxy P_3 by proxy P_1 upon determining that the bandwidth of $N_s R_0 R_1 P_1 P_3$ is greater than b units.

For the first path $P_1 P_2 P_4 P_7$, when the probe reaches proxy P_2, P_2 determines that the path between N_s and P_4 satisfies the bandwidth constraints and forwards the probe to proxy P_4. For the second and third paths, however, the probe is pruned because the bottleneck bandwidth of the paths between P_3 and the egress proxies P_5 and P_8 is not sufficient.

The probe that reaches proxy P_4 continues to traverse to the destination proxy P_7. When a probe reaches the destination proxy P_7, probes are once again flooded to the destination node N_d along all the existing paths.

3.3 Correctness and Complexity Analysis

Complexity Analysis Given an overlay network $G = (V, E)$, where V is the set of QSON proxies, and E is the set of logical links connecting QSON proxies. Let $|V| = n$ and $|E| = m$. For each logical link $e_i \in E$ ($1 \le i \le m$), it is assigned a value ph_i, which represents the number of physical paths connecting the two QSON proxies of e_i. We assume ph_i is bounded by ph_B. The diameter of G, i.e., the length of the longest shortest logical path between any pair of QSON proxies, is represented by D. Further, we denote the pre-defined logical path length threshold (compared with the shortest logical paths) as lp_{th} and the maximum number of logical paths between any pair of proxies as W.

In the proposed QSON routing scheme, for any end user QoS flow, the number of probe messages can be bounded by $WD(1 + lp_{th})$, where $D(1 + lp_{th})$ denotes the maximum length of a logical path. To set up a reference point, we choose an intuitive inter-domain QoS distributed routing scheme, in which probes are flooded along all paths across the domains. We refer this approach to as *all-path-flood inter-domain distributed routing* or *all-path-flood routing* in short. In this algorithm, the number of probe messages has an upper bound of $W ph_B^{D(1+lp_{th})3}$. This simple analysis demonstrates the dramatic difference between these two schemes: the control message overhead of the QSON routing scheme increases much more slowly to the delay threshold lp_{th} than that of the all-path-flood routing scheme. Note that the probe messages included here are only those inside QSON, i.e., in transit domains, and they do not include probe messages in access domains, which are actually the same for both QSON routing and all-path-flood routing. In fact, in QSON routing, the probe overhead can be reduced further by employing probe aggregation and pruning techniques. In Section 4, we will evaluate the scalability of QSON routing using simulations.

Correctness Analysis *Claim 1: Given the logical path length threshold lp_{th}, if there exits one path lp_1 which satisfies the end user request (say, b units of bandwidth), then QSON routing will find at least one suitable path.*

This claim can be easily proved as follows: If we assume QSON routing can not find any path for the end user QoS flow, then the probe along the path lp_1

---

[3] Various techniques have been developed to improve the performance of the all-path-flood routing algorithm. However, unless a similar hierarchical routing structure to that of QSON is adopted, its routing overhead can not be reduced significantly by orders of magnitude.

must have been pruned (probe pruning is the only possible exit point for a probe before reaching the destination). However, according to the given property: lp_1 satisfies the end user request, i.e., the bottleneck bandwidth along lp_1 is greater than b. In order words, the probe cannot be pruned along the path lp_1, as it conflicts with the above premise. Thus, QSON routing will find at least one suitable path.

4 Performance Evaluation

In this section, we evaluate the scalability of QSON routing via simulations. We compare QSON routing with all-path-flood inter-domain distributed routing. Further, we investigate how the probe aggregation and pruning techniques help to improve the performance.

4.1 Simulation Settings

We implement the QSON routing in a simulator built using C++. In the simulations, we use two types of network topologies, a real AT&T backbone network with 120 nodes [1] and 10 random networks with 100 nodes generated using the Waxman model [17]. Each node in the topologies represents a proxy, and each edge denotes a logical link between two proxies in the same domain. We use the delay threshold lp_{th} to compute the logical paths: a logical path from a source proxy to a destination proxy should be no more than lp_{th} logical hops longer than the shortest logical path between them. We vary the delay threshold from 0 to 4 logical hops.

We assume that the number of physical paths connecting two neighbor proxies follows a uniform distribution in the range of $[1, 10]$. To evaluate the effectiveness of the probe pruning technique, we set the available bandwidth between two neighbor proxies to be uniformly distributed between 1 and 20 units. Unless otherwise specified, an end user QoS flow requires 10 units of bandwidth. For each topology, we generate 1000 QoS flows in each simulation run. For each flow, the source and the destination are chosen randomly from all the nodes in the network. We conduct 1000 simulation runs by varying the available bandwidth of the logical paths. The results are averaged over the 1000 runs.

We use *control overhead* as the performance metric to evaluate the scalability of QSON routing. It is defined as the total number of logical hops that probes traverse. The lower the control overhead, the less the consumed bandwidth, and hence the more efficient the scheme.

4.2 Results and Analysis

We conduct three sets of simulation experiments to examine the performance of QSON routing, the probe aggregation and pruning techniques, respectively.
QSON routing We plot the results of QSON routing without probe aggregation and pruning vs. all-path-flood routing (denoted as "Non-QSON") for the
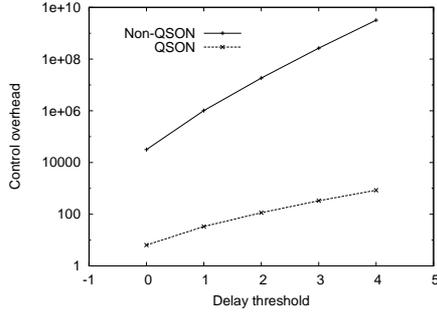
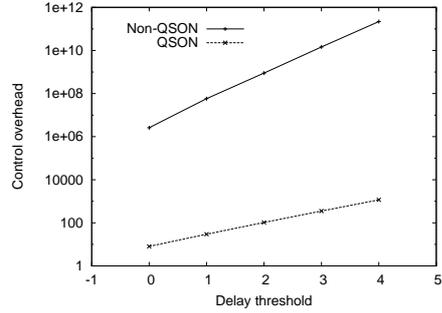**Fig. 2.** QSON vs. Non-QSON routing in the AT&T topology.

Fig. 3. QSON vs. Non-QSON routing in the Waxman topologies.

AT&T backbone and Waxman topologies in Fig. 2 and 3, respectively (Note that the control overhead is plotted in the log scale). These two figures show the same trends. First, it is clear that QSON routing effectively reduces the control overhead by sending only one probe between every involved ingress/egress pair. For instance, when the delay threshold is 4, QSON decreases the control overhead from 3.20×10^9 to 848 in the AT&T topology, and from 2.2×10^{11} to 1166 in the Waxman topologies, both corresponding to more than 99% overhead reduction.

Second, as the delay threshold increases, more logical paths between neighbor proxies become eligible. Consequently, both routing schemes result in a higher control overhead. However, comparing the increasing trend of the two curves in each figure, we observe that the overhead of all-path-flood routing grows almost exponentially with the delay threshold, whereas that of QSON routing grows much less rapidly. This difference is caused by the uncontrolled flooding involved in the former scheme, and it is indeed consistent with our analysis in Section 3.3.
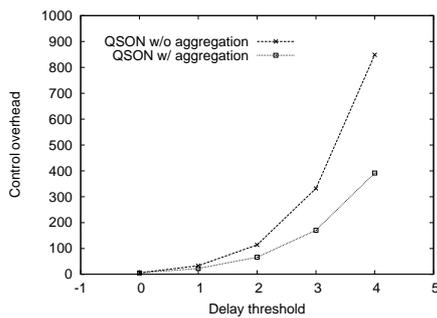

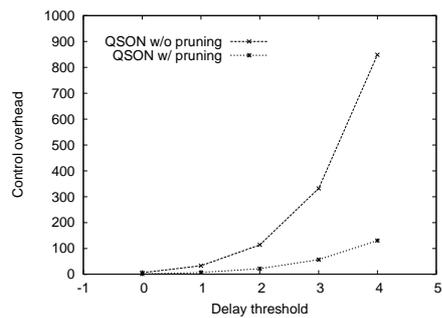**Fig. 4.** Effectiveness of the probe aggregation technique in the AT&T topology.

Fig. 5. Effectiveness of the probe pruning technique in the AT&T topology.

Probe Aggregation The results of QSON with and without the probe aggregation technique in the AT&T topology are shown in Fig. 4. This figure demonstrates that probe aggregation helps reduce the control overhead of QSON routing. In addition, the reduction in the overhead increases with the delay threshold, since more logical paths are likely to share a larger common portion at the beginning of these paths. For example, when the delay threshold is 0, probe aggregation reduces the control overhead by less than 10%; when the threshold is raised to 4, it decreases more than half of the overhead. The results for Waxman graphs are similar and thus are not shown here to save space.

Probe Pruning Fig. 5 illustrates the benefit of using the probe pruning technique in the AT&T backbone when the QoS flows require 10 units of bandwidth. Obviously, this technique improves the performance of QSON routing with respect to control overhead: when the delay threshold is varied, probe pruning consistently decreases the overhead by more than 70%. Furthermore, this benefit becomes very distinguished when the delay threshold is higher: approximately 85% of the control overhead is reduced at a delay threshold of 4. In this case, the logical paths tend to go through a larger number of proxies and it is more likely that some portions of these paths can be pruned due to bandwidth deficiency.

Summary The results of our simulations reported in this section indicate that QSON routing generates significantly less control overhead than all-path-flood routing. In addition, both the probe aggregation and pruning techniques are very effective in further reducing the control overhead, especially when the delay threshold is high.

5   Related Work

In this section, we briefly review some related QoS overlay architectures and QoS routing schemes along with their pros and cons.

5.1 QoS Overlay Architectures

Recently, overlay networks are proposed to support value-added services without making changes to network routers. End-user overlays rely on end systems to implement QoS features. For example, Resilient Overlay Networks (RONs) are proposed to detect and recover from Internet path failures by actively monitoring the quality of overlay links and routing packets based on application-specific metrics [3]. The Spine architecture applies TCP-like loss recovery and congestion control on each overlay link to reduce the latency and jitter of reliable connections [2]. Though highly flexible, end-user overlays usually cannot provide end-to-end QoS guarantees, since they normally cross many uncontrolled intermediate domains. Moreover, it is difficult to design an effective economic model for ISPs to adopt end-user overlays.

To solve these problems, backbone overlays managed by third party providers are advocated. Some example proposals are Service Overlay Network or SON [7],

OverQoS [15], QRON [13], and QUEST [9]. Unlike QSON, which well combines the benefits of overlay networks and QoS-aware IP networks, these proposals either assume the guaranteed services from underlying networks, such as SON [7], QRON [13], and QUEST [9], or try to infer the statistical bandwidth and loss rate assurance along overlay links, e.g., OverQoS [15]. Moreover, none of these proposals addresses the QoS routing issue in backbone overlays, which in fact is the main problem of this paper.

5.2 QoS Routing

QoS routing techniques can be categorized into source routing and distributed routing. In the former case, the source node is responsible for determining a suitable path by applying graph algorithms to the link state information stored at the source node [4]. A link state protocol is used to broadcast link state information through the network, which consumes an enormous amount of resource. On the other hand, distributed routing is achieved by probe flooding, where the source node floods probes towards the destination node searching for suitable paths [4]. If multiple suitable paths satisfying the constraints exist, then the shortest path is chosen to reduce delay and the probability of path degradation.

When used for inter-domain routing in large networks, the overhead associated with source and distributed routing is even aggravated. Aggregation techniques for source routing have been proposed to alleviate this issue [11], but it may lead to inaccuracies, crankback, and reaggregation [10, 8]. The scalability of inter-domain routing via probe flooding can be improved by precomputing only the shortest paths [14]. However, even the number of shortest paths in the case of a large network is likely to be very high. When a QoS flow is to be routed through QSON, it will typically cross multiple domains. If aggregation techniques are used, they will lead to inaccurate and sub-optimal solutions especially when the resource availability is low. If distributed routing via probe flooding is to be used, then flooding probes across all the possible paths will consume resources that could be otherwise used for supporting additional flows. Due to these reasons, the existing QoS routing techniques cannot be used for routing end user flows through QSON.

6   Conclusions and Future Work

In this paper, we presented a backbone QoS overlay network (QSON) architecture for scalable, efficient and practical QoS support. The major contributions of this paper can be summarized as follows. (1) We advocate backbone overlays for scalable QoS support, and present an integrated QSON architecture which involves access domains and transit domains. (2) We propose a scalable and distributed QSON routing scheme, which can reduce the probe overhead significantly compared with all-path-flood routing. (3) We conduct simulations to evaluate the performance of QSON routing, and the results show that its probe overhead is significantly reduced by more than 99% in the simulated scenarios.

We plan our future work in the following two directions. (1) Network design for QSON: In this paper, we assume the overlay network is known. We do not consider the overlay network design, i.e., where to place proxies and which logical links to select. Clearly, overlay network design for QSON is a challenging issue to investigate. (2) Comparison with various multihoming proposals: Multihoming route control is closely related to overlay routing. It is worth investigating how QSON performs compared with multihoming routing schemes.

References

1. AT&T IP Backbone. *http://www.ipservices.att.com/backbone/*, 2001.
2. Y. Amir and C. Danilov. Reliable communication in overlay networks. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks, June 2003.*
3. D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of Symposium on Operating Systems Principles*, pages 131–145, 2001.
4. S. Chen and K. Nahrstedt. An overview of Quality–of–Service routing for the next generation high–speed networks: Problems and solutions. *IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, 1998.
5. Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
6. J.-H. Cui, S. Gokhale, L. Lao, and J. Lu. Distributed QoS Routing for Backbone Overlay Networks. UCONN CSE Technical Report: UbiNet-TR06-01, Jan. 2006.
7. Z. Duan, Z.-L. Zhang, and Y. T. Hou. Service overlay networks: SLAs, QoS, and bandwidth provisioning. *IEEE/ACM Transactions on Networking*, 11(6):870–883, 2003.
8. E. Felstaine, R. Cohen, and O. Hadar. Crankback prediction in hierarchical ATM networks. In *Proc. of INFOCOM*, 1999.
9. X. Gu, K. Nahrstedt, R. Chang, and C. Ward. Qos-assured service composition in managed service overlay networks. In *Proceedings of IEEE 23rd International Conference on Distributed Computing Systems, Providence, May 2003.*
10. R. Guerin and A. Orda. QoS–based routing in networks with inaccurate information: Theory and algorithms. In *Proc. of INFOCOM*, 1997.
11. F. Hao and E. Zegura. On scalable QoS routing: Performance evaluation of topology aggregation. In *Proc. of INFOCOM*, 2000.
12. A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proceedings of ACM SIGCOMM'02, (Pittsburgh, PA), August 2002.*
13. Z. Li and P. Mohapatra. QRON: QoS-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications, January*, 2004.
14. S. Norden and J. Turner. Inter–domain QoS routing algorithms. Technical Report WUCS-02-03, Department of Computer Science, WUCS, 2002.
15. L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Overqos: An overlay based architecture for enhancing internet qos. *In Proceedings of USENIX NSDI'04*, pages 71–84, 2004.
16. S. Vieira and J. Liebeherr. Topology design for service overlay networks with bandwidth guarantees. In *IEEE IWQoS*, 2004.
17. B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, December 1988.