

Distributed QoS Routing for Backbone Overlay Networks

Jun-Hong Cui, Swapna S. Gokhale, Li Lao and Jijun Lu
{*jcu*,*ssg*}@*cse.uconn.edu*, *llao@cs.ucla.edu*, *jijun.lu@uconn.edu*

UConn CSE Technical Report: UbiNet-TR06-01

Last Update: January 2006

Abstract

In recent years, overlay networks have emerged as an attractive alternative for supporting value-added services. Many of these overlays are end-to-end overlays, but 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. 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, and sells its QoS services to end users via service contracts. A key challenge in making QSON a reality consists of efficiently determining routes for end user QoS flows based on the service level agreements with the underlying network service providers. Existing QoS routing techniques may not be applicable in QSON, since they are primarily designed for routing flows that reside wholly within a domain, while a QoS flow in a QSON will routinely straddle multiple domains. Thus, 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 use simulations to evaluate the scalability of the proposed routing scheme. We also discuss how to implement it in an incremental fashion. We implement a prototype protocol in PlanetLab, and by real experiments, we demonstrate the effectiveness of our solution.

I. INTRODUCTION

The Internet is one of the most successful technologies in last century. Within less than four decades (starting from the first packet-switched computer network, ARPAnet), it has evolved into an extremely popular commercial infrastructure, and has a significant impact on almost all aspects of our lives and our society. 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 best-effort service model to provide QoS, researchers have proposed many seminal architectures, represented by IntServ [9] and DiffServ [8]. Unfortunately, due to many critical factors, 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 proposed 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 which 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 [6], [36], [40], [13], [38], [24]. Many of these overlays are end-user overlays, namely, overlays are constructed purely among the end hosts without support from any other intermediate nodes. Due to the difficulties of supporting end-to-end QoS purely in end-user overlays, some recent works [15], [17], [35], [28] propose 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 lower tier ISPs, or 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 QoS flows based on the SLAs with the underlying ISPs. Existing QoS routing techniques may not be applicable in QSON, since they are designed primarily for flows that lie wholly within a single domain. However, in a QSON, a QoS flow will routinely straddle multiple domains. Thus, 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 use simulations to evaluate the scalability of the proposed routing scheme. We also discuss how the proposed scheme can be deployed in an incremental fashion. We implement a prototype of the proposed scheme in PlanetLab, and through real experiments, we demonstrate the effectiveness of the scheme.

The layout of the paper is as follows: Section II provides an overview of the related work in the areas of QoS architectures and QoS routing. Section III describes the QSON architecture and discusses its main characteristics and advantages. Section IV describes the routing scheme for QSON, provides a formal analysis of the scheme, and discusses the issue of incremental deployment. Section V presents the results of the simulations conducted to evaluate the scalability of the scheme, and Section VI provides results from real experiments. Section VII offers concluding remarks and directions for future research.

II. BACKGROUND AND RELATED WORK

A. QoS Architectures

QoS architectures have evolved from IntServ, DiffServ, to the recently proposed overlay model. In this subsection, we briefly review some representative architectures along with their pros and cons.

Integrated Service Architecture: Integrated Service Architecture (IntServ) [9] is designed to provide different types of QoS guarantees for data flows. It establishes connections through the network using a resource reservation protocol, such as RSVP [39], and an admission control mechanism, such as [22], based on network information. Then, the resource state information for each flow needs to be maintained at routers in order to ensure that sufficient resources are available during the lifetime of the flow. Since the number of data flows in the Internet can be very huge, the main criticism of IntServ is its poor state scalability.

Differentiated Service Architecture: Differentiated Service architecture (DiffServ) [8] is proposed for scalable service differentiation in the Internet. It categorizes data flows into a number of classes. Data in the same class receives same QoS. In a DiffServ domain, packets crossing a link and requiring the same behavior (e.g., scheduling treatment and drop probability, or in other words, in the same class) constitute a Behavior Aggregate (BA). At the ingress nodes, the packets are classified and marked with a Diff-Serv Code Point (DSCP) according to their Behavior Aggregate. At each transit node, the DSCP is used to determine the behavior for each packet. By flow aggregation, DiffServ is much more scalable than IntServ, however, it can only provide very coarse-granularity services.

End-User Overlays for Value-added Services: Several end-user overlays have been designed to support value-added services. Resilient Overlay Networks (RONs) are proposed to detect and recover from Internet path failures [6]. They route packets on paths optimized for application-specific metrics. RON nodes actively monitor the quality of the paths to their neighbors and decide where to route packets based on collected information and application requirements. Amir *et al* propose an overlay architecture, called Spine [5], [4], which uses the hop-by-hop reliability approach on overlay links to reduce the latency and jitter of reliable connections. By applying TCP-like loss recovery and congestion control on each overlay link, this approach can detect packet loss faster and recover the packets locally. Though highly flexible, end-user overlays usually cannot provide end-to-end QoS guarantees, since this type of overlays typically cross many intermediate domains, and the uncontrolled domain peering structure is unlikely to provide direct QoS support to the end users. Moreover, it is difficult to design an effective economic model for ISPs to adopt end user overlays.

Backbone QoS Overlays Some recent works [15], [17], [35], [28] propose using backbone overlays for QoS

support, and the backbone overlays are managed by a third party provider. In Service Overlay Networks or SONs [15], bandwidth is provisioned with certain QoS guarantees from individual network domains to build a logical end-to-end service delivery. While SONs rely on underlying networks to provide QoS services, OverQoS [35] presents a Controlled Loss Virtual Link (CLVL) abstraction to provide Internet QoS (e.g., statistical bandwidth and loss rate assurance) using overlay networks and performing bundle loss control on each virtual link. QRON [28] introduced the concept of overlay brokers (OBs) and a general unified framework for an overlay network. Another proposal called QUEST (QoS assured composEable Service InfrasTructure) [17] goes further and presents solutions to compose qualified service paths with multiple QoS constraints and load balancing from SLA contracts of individual service components. From a high-level, conceptual point of view, QSON shares some similarity with SON, QRON, and QUEST. However, these backbone overlay research efforts either focus on different aspects of overlay provisioning (e.g., bandwidth dimensioning [15] for overlay links, or overlay path composition [17] based on centralized SLA information) or dedicate to introducing an overlay architecture (e.g., [28]). To the best of our knowledge, our work is the first effort to tackle the distributed QoS routing problem in backbone overlay networks.

B. QoS Routing

QoS routing has been researched extensively, and in this subsection we describe the aspects that are pertinent to the proposed QoS overlay architecture.

QoS routing consists of determining a path through the network which has adequate resources to satisfy the QoS requirements of a connection, while simultaneously achieving global efficiency in network resource utilization. QoS routing can be classified into unicast routing and multicast routing [11]. It can be further classified as intra-domain routing and inter-domain routing. To facilitate QoS routing, the state of every link in the network must be expressed in terms of a set of QoS metrics such as delay, bandwidth, jitter and cost [20]. The metrics of the links along a path are composed to obtain QoS metrics of the path. A QoS connection expresses its QoS requirements in the form of constraints on one or more path QoS metrics [11]. These constraints are then compared to the QoS metrics of the paths through the network in order to select one that satisfies the constraints.

QoS routing techniques can be categorized into source routing and distributed routing. In the case of source routing, the source node is responsible for determining a suitable path by applying graph algorithms to the network topology and link states that are stored at the source node [11]. The link state protocol is used to maintain link states. It broadcasts link state information through the network which consumes an enormous amount of resource. As a result, there is a tradeoff between the frequency of link state broadcasts, the staleness of the link states, and the optimality of the paths [29]. Another drawback of source routing is that it has to employ heuristic algorithms to solve the k -constrained routing problem [30]. Distributed routing is achieved by probe flooding, where the source node floods probes through the network towards the destination node searching for suitable paths [11]. The overheads incurred in flooding probes can be reduced by bounded and selective flooding [26]. If multiple suitable paths which satisfy the constraints exist, then the shortest path is chosen to achieve global efficiency in network

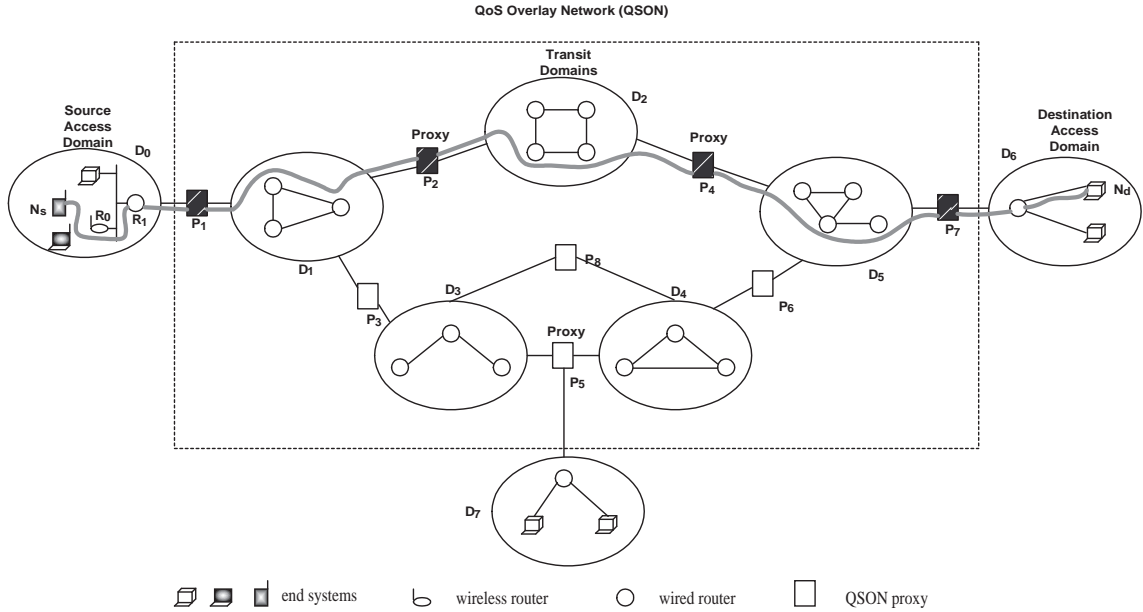


Fig. 1. QSON Architecture

resource utilization [12], a reduction in delay and a reduction in the probability of path degradation.

The overheads associated with source and distributed routing are aggravated to a large extent when used for inter-domain routing in large networks. Aggregation techniques for source routing have been proposed to alleviate this issue [19], [27], but it may lead to inaccuracies, crankback, and reaggregation [10], [7], [18], [16]. The scalability of inter-domain routing via probe flooding can be improved by precomputing only the shortest paths [31], [26], [33]. 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. For each one of the domains the QSON provider may have a different level of SLAs, resulting in a different level of resource availability. When the level of available resources is high, the inefficiencies introduced by aggregation (if aggregation techniques are used) may not be significant. However, in order to maximize the revenue, the QSON provider may be interested in admitting as many connections as possible which may result in a low level of resource availability. In such cases, aggregation will lead to inaccurate and sub-optimal solutions. If distributed routing via probe flooding is to be used to route end user flows, then flooding probes across all the possible paths will consume resources that could be otherwise used for supporting additional flows to increase revenue. Due to these reasons, the existing QoS routing techniques cannot be used for routing end user flows through QSON.

III. 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 service level agreements (SLAs) between the QSON provider and the underlying ISPs. Outside QSON,

end hosts in access domains subscribe to QSON by proper paths 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, from the end user point of view, it is actually a single logical domain. End user flows are managed by QSON, and the underlying ISPs only see “aggregated” flows traversing overlay paths.

A. Physical Network Structure

The underlying physical network structure from which QSON will be constructed is composed of multiple domains. We assume that each one of the domains is managed by a network service provider (underlying ISP). QSON proxies are strategically deployed between domains. Each proxy may belong to multiple domains. In each domain, we refer to QSON proxies as border nodes, and all other nodes as internal nodes. An internal node can only be linked to internal nodes or QSON proxies within its domain, whereas a QSON proxy can be linked to any node (internal nodes or QSON proxies) in 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 (QSON) edge proxies, which are usually advertised to end users by the QSON provider. In addition, we refer to the (QSON) edge proxy in the source (or destination) access domain as source (or destination) (QSON) 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 in the forward direction. For example, in Fig. 1, if a connection originates in access domain D_0 and terminates in access domain D_6 , P_1 is a source QSON edge proxy, and P_7 is a destination QSON edge proxy. Also, in the above example P_1 and P_2 are respectively the ingress and egress proxies for domain D_1 .

B. 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. It does not need to know the topology of the underlying ISP domains. The QSON provider thus enters into a SLA with an underlying ISP, where the SLA specifies the set of alternate paths between all the pairs of proxies that belong to the ISP’s domain, and the amount of bandwidth allocated by the ISP to the QSON provider on each one of the paths. The initial amount of bandwidth purchased by the QSON provider can be adjusted based on the demands of the end users by modifying the SLA. The logical view of the QSON thus consists of paths between pairs of proxies in each domain, and each one of these paths may be annotated by the bandwidth allocated by the ISP to the QSON. For each path through the domain, the ISP may also provide the QSON information about the number of hops along the path. Information regarding the number of hops along the paths can be used to guide the selection of one path if multiple suitable paths exist, so that higher efficiency in resource utilization, lower delay, and lower probability of path degradation can be achieved. Thus, each path

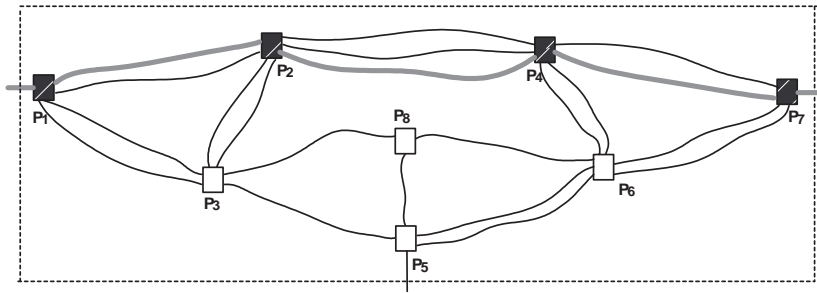


Fig. 2. A logical view of QSON

may also be annotated with the number of hops along the path in addition to the bandwidth¹. The logical network structure from the point of view of the QSON is shown in Fig. 2.

C. Network State in QSON

The QSON uses the bandwidth allocated along the paths between the proxies to route end user QoS flows. As a result, as end user flows arrive and depart, the amount of available bandwidth along each one of the paths between the proxies changes dynamically. In the QSON, each proxy maintains the amount of available bandwidth on all the paths to other proxies in the domain that are included in the SLA. Each proxy also maintains 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 some extent, each proxy knows all the logical paths to all the proxies which are not in the same domain(s). This type of information can be obtained when the QSON is deployed. However, to limit the amount of information that needs to be maintained, the QSON provider can eliminate some “lengthy” paths by defining an appropriate management policy². 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 smallest logical path $P_1P_2P_4$. Thus, the QSON provider can pre-define a logical path length threshold, say, lp_{th} , compared with the shortest logical paths between proxies. In this way, for a logical path between two proxies A and B , if its length (in terms of the number of logical hops) is bigger than $D(1 + lp_{th})$, where D is the length of the shortest path between A and B , then this logical path is eliminated, and not included either A or B .

¹In fact, our design does not constrain the path metrics underlying ISPs may provide. An ISP may simply supply delay information for each available path in its domain. Generally speaking, the higher the information available for the possible paths in ISP domains, the better would be the QoS support QSON can provide to the end users. In the later sections, we mainly use the bandwidth metric along with the number of hops to demonstrate our routing scheme. However, we will also discuss how the QSON can be incrementally deployed in the current “best-effort” Internet using the metrics of delay/jitter, bandwidth capacity, etc.

²In general, the more the available logical paths, the higher the possibility of routing end user QoS flows. However, it is usually useless to maintain very lengthy paths; for 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.

D. 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 or service nodes) 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 and is based on well-defined business relationships via SLAs with network service providers and service contracts with end users. This level of traffic aggregation, in the long run, will make IntServ-like architectures practical. A good analogy to this scenario is the relationship between manufacturers, dealers, and consumers in our daily life.

IV. ROUTING IN QSON

In this section, we describe a distributed and scalable routing scheme that can be used to efficiently route end user QoS flows on overlay paths (between proxies).

A. 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 QSON 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 among the possibly multiple paths. A suitable path between the source node and the source edge proxy is the one 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 as the source edge proxy. 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). This delay threshold can be expressed in terms of the number of proxies along the path. Then a possible multi-domain (logical) path is thus a path where the number of proxies along a path is less than the specified limit. In this manner, the overhead incurred in forwarding the probe on paths that may not satisfy the user requirements is avoided. 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 domain³. If no path between the chosen ingress/egress proxy pair has sufficient bandwidth to satisfy the bandwidth requirements, the probe is *pruned* and not forwarded further. If the probe is not pruned, it is then forwarded to the next proxy along the selected path. After the next proxy receives the probe, it repeats the same process as that of the source edge proxy. 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, $P_1P_3P_5P_6P_7$, and $P_1P_3P_8P_6P_7$ share the first logical link P_1P_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 overheads 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 paths (probe flooding as in the source access domain). Thus, each probe that reaches the destination node carries the bandwidth of a path between the source and the destination nodes. The destination node then compares the QoS metrics of the path(s) (i.e., the bottleneck bandwidth) with the bandwidth requirement

³Additional administrative factors included in the SLA between the QSON and the underlying ISP (for the domain) may also be considered in the selection. For example, some paths through the domain may have a higher cost, in which case these paths may not be chosen unless it is absolutely necessary.

of the connection and selects one. The final path selection may also be based upon other administrative factors enforced by the QSON.

SLA Renegotiation: During the routing of an end user QoS flow, if the QSON provider encounters a situation where the available bandwidth along all the possible logical paths between the source and the destination edge proxies is less than the bandwidth requirements of the flow, the provider has two options. In the first option it can choose to block the connection. This option though simple is not desirable. In the second option, it chooses a possible path and determines the domain(s) in which the available bandwidth is less than the bandwidth requested by the flow. It can then negotiate with the ISPs to upgrade the bandwidth in those domains.

B. 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 limit on the number of proxies (delay-constrained threshold) exist to proxy P_7 , which is the destination edge proxy. These three paths are $P_1 P_2 P_4 P_7$, $P_1 P_3 P_5 P_6 P_7$, 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 suitable 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 probe is forwarded to proxy P_2 along the chosen suitable path.

The destination edge proxy can be reached along the second and the third path by a single egress proxy, namely, P_3 . As a result, a single probe is forwarded to proxy P_3 , by proxy P_1 by composing the bandwidth of the path $N_s R_0 R_1 P_1$ with that of the suitable path between P_1 and P_3 and upon determining that it is greater than b units.

For the first path $P_1 P_2 P_4 P_7$, when the probe reaches proxy P_2 , the same process is repeated for the path between P_2 and P_4 , and upon determining that the overall path between N_s and P_4 satisfies the bandwidth constraints, the probe is forwarded to proxy P_4 . For the second and third paths, however, at proxy P_3 , it is determined that the bottleneck bandwidth of the paths between P_3 and the egress proxies P_5 and P_8 is not sufficient to satisfy the bandwidth constraints of the flow. As a result, the probe that reaches proxy P_3 is pruned and not forwarded any further.

The probe that reaches proxy P_4 continues to traverse to the destination proxy P_7 since the bottleneck bandwidth

of the path traversed by the probe satisfies the QoS constraints. 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.

C. Correctness and Complexity Analysis

In this section we analyze the correctness and the complexity of the proposed QSON routing scheme.

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, which connect QSON proxies. We assume $|V| = n$, and $|E| = m$. For each logical link $e_i \in E$, where $1 \leq i \leq m$, it is assigned a value ph_i , which represents the number of physical paths connecting the two QSON proxies of e_i . The diameter of G , i.e., the longest shortest path between any pair of QSON proxies, is D . Further, we assume ph_i is bounded by ph_B , and the pre-defined logical path length threshold (compared with the shortest logical paths between proxies) is lp_{th} . Again, the maximum number of (logical) paths between any pair of proxies is 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. In traditional inter-domain distributed routing algorithms, the number of probe messages has an upper bound of $Wph_B^{D(1+lp_{th})}$. 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 traditional inter-domain distributed routing. In fact, in QSON routing, the probe overhead can be reduced further by employing probe aggregation and pruning techniques. In Section V, we will extensively evaluate the scalability of QSON routing using simulations.

We further illustrate the efficiency of QSON routing using the following simple example. Consider the scenario where there are three transit domains between the source and destination proxies, such that the first domain has three alternate paths, the second one has four paths and the third domain has five paths. In the conventional schemes, probes would be flooded across all the sixty paths that result from the combination of the paths in the three domains, whereas in the proposed scheme a probe would traverse across only one path in each one of the domains.

Correctness Analysis:

Claim 1: Given the logical path length threshold lp_{th} , if there exists 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 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 other 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.

D. Advantages of QSON Routing

The QSON routing scheme offers several advantages over traditional inter-domain routing schemes as described below.

- It is a distributed routing scheme using probe flooding. Compared with centralized source routing, it provides better robustness and enables quicker adaptation to network dynamics.
- On-demand, uncontrolled probe flooding via all the existing paths is used only in the source access and destination access domain. Since the access domains are usually smaller in size, uncontrolled flooding is affordable.
- In a transit domain, probes are not flooded along all the existing paths between every ingress/egress proxy pairs that lie along a possible logical path. Instead, a probe is sent only once along the selected path between each possible ingress/egress proxy pair. This significantly mitigates the overheads of conventional inter-domain routing via probe flooding where probes are forwarded along all the inter-domain paths that exist between the source and destination proxies. Probe pruning and probe aggregation mitigate the overheads further through the transit domains.
- Another significant advantage of this scheme is that it facilitates the determination of multiple candidate paths for routing a QoS connection without incurring any additional overhead. Thus, if resource reservation fails along the selected path, it can be attempted along another path which is readily available.

E. Incremental Deployment

Due to its formidable size, it is feasible to employ a QoS scheme in the Internet only if it can be deployed in an incremental manner. In the previous sections, we assume each domain including source access domain and destination access domain are QoS-aware⁴. However, it is possible that some domains may not be capable of providing QoS services. For the domain which is not QoS-aware, the QSON proxies in the domain could use measurements to obtain QoS metrics, such as bandwidth capacity (using CapProbe [23] or Pathrate [14] for example), delay/delay jitter (using Ping for instance), etc. In such situations, although QSON cannot provide guaranteed services, significantly enhanced services may be possible. In Section VI, we will describe a real implementation of QSON in PlanetLab [2], and demonstrate the effectiveness of QSON solution in the current Internet scenario.

V. PERFORMANCE EVALUATION

In this section, we evaluate the scalability of QSON routing via simulation. We compare QSON routing with conventional inter-domain routing (via probe flooding). Further, we investigate how the probe aggregation and pruning techniques help to improve the performance.

⁴In fact, many traffic engineering and provisioning techniques, such as MPLS (MultiProtocol Label Switching) [32], empower more and more domains with QoS support.

A. Simulation Settings

We implement the QSON routing in a simulator built using C++. The simulation settings are described in detail as follows.

Network Topologies In the simulations, we use two types of network topologies, a real AT&T backbone network with 120 nodes [3] and 10 random networks with 100 nodes generated using the Waxman model [37]. 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 using the logical links. 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 [1, 10]. To evaluate the effectiveness of the probe pruning technique, we set the available bandwidth between two neighbor proxies to a number drawn from a uniform distribution in the range between 1 and 20 units. Unless otherwise specified, 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.

Performance Metric We use control overhead as the performance metric to evaluate the scalability of QSON routing. The control overhead is defined as the total number of logical hops that probes traverse. The lower the control overhead, the less bandwidth the QoS routing scheme consumes, and hence more efficient is the scheme. For each topology, the control overhead is computed for each one of the 1000 runs, and the average control overhead is reported.

B. Results and Analysis

We conduct three sets of simulation experiments to examine the performance of QSON routing, the probe aggregation and pruning technique, respectively.

QSON routing We plot the results of QSON routing without probe aggregation and pruning vs. conventional inter-domain routing (denoted as “Non-QSON”) in the AT&T backbone and Waxman topologies in Fig. 3 and 4, respectively. 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. Note that the control overhead is plotted in the log scale.

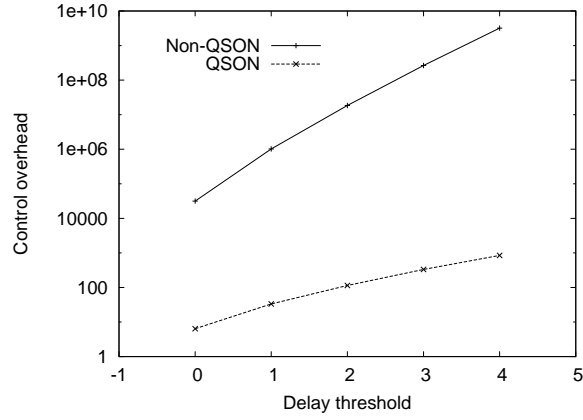


Fig. 3. QSON vs. Non-QSON routing in the AT&T topology.

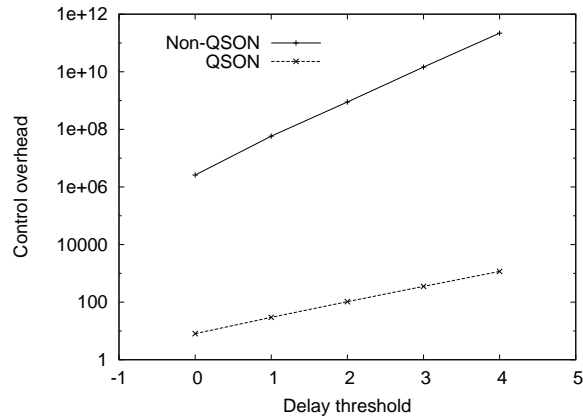


Fig. 4. QSON vs. Non-QSON routing in the Waxman topologies.

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 these two curves in each figure, we observe that the overhead of conventional 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 IV-C.

Probe Aggregation The results of QSON with and without the probe aggregation technique in the AT&T topology are shown in Fig. 5. 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 consistent with the AT&T backbone network and are not shown here to conserve space.

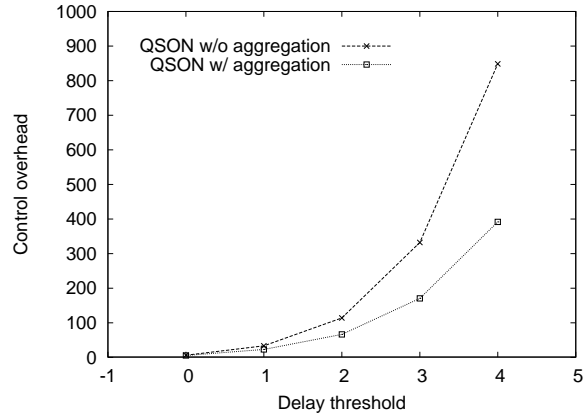


Fig. 5. Effectiveness of the probe aggregation technique in the AT&T topology.

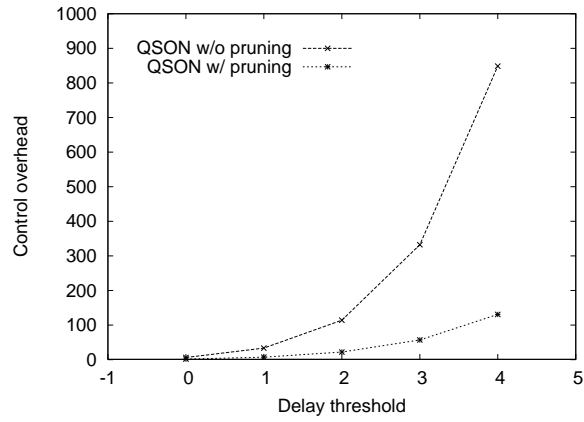


Fig. 6. Effectiveness of the probe pruning technique in the AT&T topology.

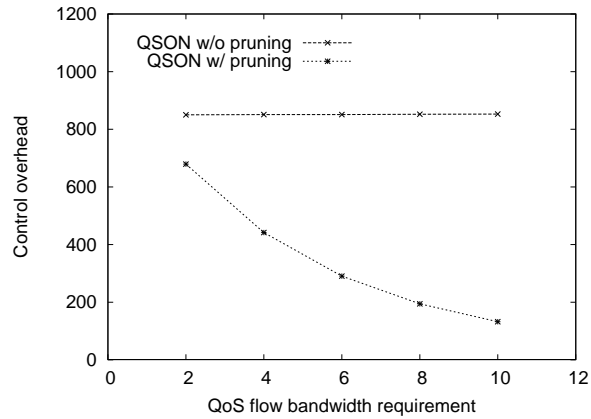


Fig. 7. Impact of QoS flow bandwidth requirement on path pruning in the AT&T topology.

Probe Pruning Fig. 6 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.

It is also worth pointing out that the effectiveness of the probe pruning mechanism depends on the available bandwidth and QoS flow requirements. Therefore, we vary the bandwidth requirement of QoS flows from 2 to 10 units and plot the results when the delay threshold is set to 4 in Fig. 7. As the QoS flows require more bandwidth, the available bandwidth on the logical paths becomes less and less abundant; thus, more and more paths can be pruned. As a result, the control overhead remains fairly stable when the probe pruning technique is not adopted, whereas it decreases dramatically when the pruning is enabled. This is exactly what we observe in this figure. When 2 units of bandwidth is required by QoS flows, approximately 20% of the control overhead is reduced. In contrast, when the QoS flows require 10 units of bandwidth, probe pruning successfully reduces the overhead by 85%.

C. Summary

The results of our simulations reported in this section indicate that QSON routing generates significantly less control overhead than conventional 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. We also show that the performance of probe pruning depends heavily on whether there is adequate available bandwidth on the logical paths.

VI. IMPLEMENTATION IN PLANETLAB

To test the effectiveness of the proposed QSON routing scheme, we implemented a prototype in PlanetLab. In this section, we discuss the results of the experiments conducted using the prototype.

A. Experimental Settings

The prototype QSON routing scheme is developed with Java 2 Platform. The QoS metrics used are the end-to-end delay and bandwidth capacity between the source node and destination node. The basic measurement technique for delay is *Ping*, a program which allows us to measure the delay on each (logical) link between two proxies. The technique used for measuring the bandwidth capacity on each logical link is CapProbe [23]. The process used to obtain delay and bandwidth capacity measurement techniques using the above two approaches is described below.

Delay Measurement: The delay measurements are carried out by Ping, which sends packets to neighbors that echo the packets back. A packet is time stamped at each node and when it returns the round trip delay is calculated. The packet also counts the hops while travelling this round trip. In our implementation, each node in the overlay runs “Ping” program every 5 minutes. It “pings” each one of its neighbors and stores the delay for each one in a file named “delay.log”. The stored delay is computed using an exponential weighted moving average [25], which is explained below.

The exponential moving average allows a percentage of the old delay to average in to the new delay and mitigates unfavorable spikes in the measurement. In our implementation, we used the following equation to compute QoS metrics: $Delay_{new} = 80\%(Measurement_{new}) + 20\%(Delay_{old})$.

When the probe originating from the source node propagates through the proxies along the path, it records the stored delays and computes a cumulative delay which is use for routing decisions.

Bandwidth Measurement: We integrated CapProbe into our prototype to measure bandwidth capacity. CapProbe combines delay measurements and packet pair dispersion to estimate the bandwidth capacity. Using the measured bandwidth capacity, CapProbe allows a user to intelligently select the best links for routing.

Experimental Topology The QSON prototype is deployed in PlanetLab. We selected 8 nodes from MIT, University of Massachusetts (Amherst), University of Michigan, Iowa State University, University of Utah, Georgia Institute of Technology, Rice University (Texas), and UCLA. These eight nodes form 4 paths from MIT to UCLA. The nodes and the paths are shown in Fig. 8⁵, and the four paths are described as follows:

Path 1: mit-umass-umich-iowas-utah-ucla (Red line)

Path 2: mit-git-rice-ucla (Blue line)

Path 3: mit-umass-rice-ucla (Green line)

Path 4: mit-git-utah-ucla (Teel line)

B. Results and Analysis

We conduct four sets of experiments. In all the experiments, every 5 minutes, each node measures the QoS metrics on all of its connected links. Every 15 minutes a request is initiated from MIT to UCLA.

Selecting the best path to route (from 2 paths) based on end-to-end delay: In the first set of experiments, we use 2 paths, Path 1 and Path 2 in Fig. 8, to test our routing scheme. 40 requests are initiated from the source node, and each request results in a probe on each link. Further, we use 400 msec as the QoS requirement (for audio streaming).

⁵The topology in Fig. 8 is generated based on a map from map.google.com by the courtesy of Google [1].



Fig. 8. Selected nodes and paths in PlanetLab

The results are plotted in Fig. 9. The “star” curve represents the delay estimate on Path 1 which has 4 intermediate nodes. The “circle” curve represents the delay estimate on Path 2 which has 2 intermediate nodes. It can be observed that both paths satisfy the QoS requirement as there is no end-to-end delay greater than 400 msec. Also, it can be observed from the figure that in the first period (from 1st round to 17th round), the end-to-end delays on the path given by the “circle” line are smaller than those on the path given by the “star” line. Thus we should choose the path given by “circle” line as the routing path. For the following 23 sampling points (from 18th round to 40th round), the cumulative delays on the path given by the “star” line dropped dramatically and became smaller than those on the path given by the “circle” line. So the “star” line should be chosen as the routing path. We can see that the “circle” line is much smoother than the “star” one. It is very likely that during the first period, there were some applications running on the middle nodes on the path represented by the “star” line, or the traffic situation on the path given by the “star” line is worse than that of the path given by the “circle” line. Later after that period, both the paths have relatively stable cumulative delay.

Evaluating the correctness of the programs:

This set of experiments is to further test the correctness of our programs. At the end of the first set of experiments, the end-to-end cumulative delay on Path 1 is 88.274 msec.; for Path 2, it is 49.397 msec. We set the QoS requirement to be 40 msec., 60 msec., 100 msec. respectively and initiated 3 requests accordingly. These three requirements can represent three classes of QoS requirements, namely, the requirements that cannot be satisfied by any of the paths, requirements that can be satisfied by only one of the paths, and requirements that can be satisfied by both the paths. For the first request (40 msec), the feedback from the destination node indicates that there is no path which satisfies this QoS requirement. For the second one (60 msec.), the feedback indicates that Path 2 satisfies the QoS requirement. For the 3rd request (100 msec.), we get the same feedback as for the second request, which is correct (here both paths satisfy the QoS requirement and the destination node selects the better one, i.e., Path 2, and sends it back to the source node). It can be observed, in all the three experiments the expected feedback is

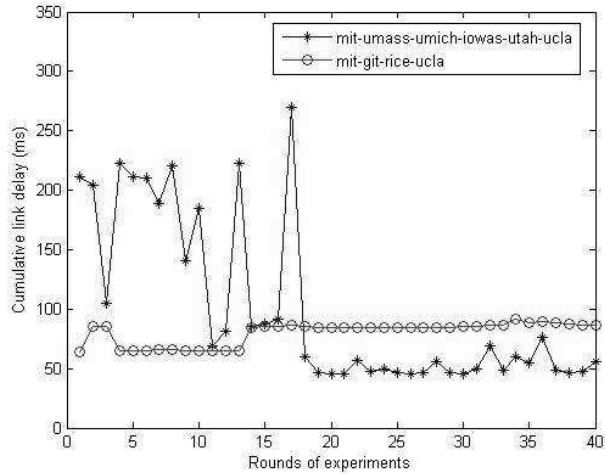


Fig. 9. Experiment (i), selecting the best path from 2 available paths

received and the right paths are chosen.

Selecting the best path to route (from 4 paths) based on end-to-end delay:

In the third set of experiments, we selected all 4 paths in Fig. 10 to test our routing mechanism. 40 requests were initiated from the source node, and each request resulted in a probe on each link. In this case, we use 400 msec. as the QoS requirement.

The results are plotted in Fig. 10. We can observe that all the 4 paths satisfy the QoS requirement and there is no end-to-end delay greater than 400 msec. Also from this figure, it can be seen that the end-to-end delays on Path 1 and Path 4 are more stable than those on the other two paths. Further notice that the only node that is located both on Path 2 and Path 3, but not on Path 1 and Path 4, is the node from Rice University in Texas. Thus it is very likely that during this experiment, the Rice node was experiencing higher working load or the traffic situation at that node was not desirable. From this observation, it is reasonable to suggest that taking delay jitter into account would result in better routing paths.

In the fourth set of experiments, we test our routing scheme using bandwidth capacity as the QoS metric. We obtain results similar to the case where the delay metric was used. These results are not presented in this paper for conciseness.

C. Lessons Learned

In this section, we demonstrated the feasibility of our QSON routing scheme through real implementation. On the one hand, we can see that the incremental deployment of QSON depends heavily on measurement techniques. The more information measurements can provide, the better service QSON can support. For example, if there are efficient techniques to measure available bandwidth (which is very challenging, though there has been significant progress recently [21], [34]), QSON can then find suitable paths which satisfy bandwidth requirement as is discussed

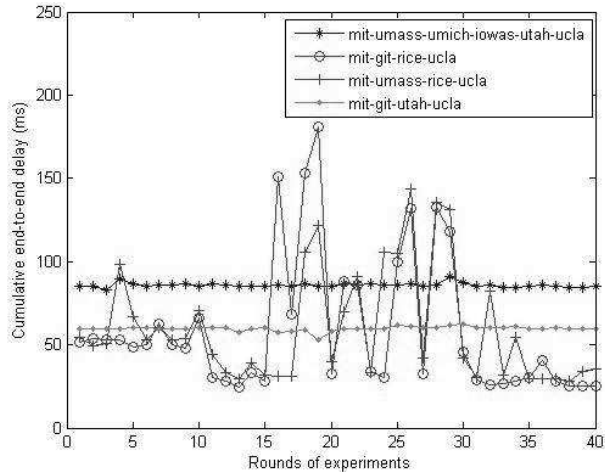


Fig. 10. Experiment (iii), selecting the best path from 4 available paths

earlier. On the other hand, MPLS (MultiProtocol Label Switching) [32] techniques may enable more underlying ISPs to provide QoS support.

VII. 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.

- We advocate backbone overlays for scalable QoS support, and present an integrated QSON architecture which involves access domains and transit domains.
- We propose a scalable and distributed QSON routing scheme, which can reduce the probe overhead significantly compared with traditional inter-domain routing via probe flooding.
- 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 implement a prototype QSON routing protocol in PlanetLab, and the results demonstrate the feasibility of incremental deployment of QSON routing.

We plan our future work in the following two directions. (1) Fault Tolerance: QSON is an overlay network, which is usually not as robust as IP networks. Thus, the issue of providing fault tolerance needs to be considered. (2) Overlay Design: In this paper, we do not consider the overlay network design, i.e., where to deploy proxies and the logical links that are selected. Instead, we assume the overlay network is ready, and focus on distributed QoS routing. Obviously, overlay network design for QSON is a challenging issue to investigate.

REFERENCES

- [1] Google. <http://www.google.com/>.
- [2] Planetlab. <http://www.planet-lab.org/>.

- [3] AT&T IP Backbone. <http://www.ipservices.att.com/backbone/>, 2001.
- [4] Y. Amir and C. Danilov. Reliable communication in overlay networks. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks, June 2003.*, 2003.
- [5] Y. Amir, C. Danilov, and C. Nita-Rotaru. High performance, robust, secure and transparent overlay network service. In *Proceedings of International Workshop on Future Directions in Distributed Computing (FuDiCo), June 2002.*, 2002.
- [6] 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.
- [7] B. Awerbuch, Y. Du, B. Khan, and Y. Shavitt. Routing through networks with hierarchical topology aggregation. *Journal of High-Speed Networks*, 7(1), 1998.
- [8] S. Blake, D. Black, and et al. An architecture for Differentiated Services. *IETF RFC 2475*, 1998.
- [9] R. Braden, D. Clark, and S. Shenker. Integrated Services in the internet architecture: an overview. *IETF RFC 1633*, 1994.
- [10] B. Chang and R. Hwang. “Dynamic update of aggregated routing information for hierarchical routing in ATM networks”. In *Proc. of Eighth Intl. Conference on Parallel and Distributed Systems*, pages 653–660, June 2001.
- [11] 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.
- [12] S. Chen and K. Nahrstedt. Distributed Quality-of-Service routing in high-speed networks based on selective probing. In *Proc. of Local Computer Networks*, October 1998.
- [13] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
- [14] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and capacity estimation. *IEEE/ACM Transactions of Networking*, 12(6), Dec. 2004.
- [15] 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.
- [16] E. Felstaine, R. Cohen, and O. Hadar. Crankback prediction in hierarchical ATM networks. In *Proc. of INFOCOM*, 1999.
- [17] 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.*, 2003.
- [18] R. Guerin and A. Orda. QoS-based routing in networks with inaccurate information: Theory and algorithms. In *Proc. of INFOCOM*, 1997.
- [19] F. Hao and E. Zegura. On scalable QoS routing: Performance evaluation of topology aggregation. In *Proc. of INFOCOM*, 2000.
- [20] J. Hong, S. Kim, and K. Lee. QoS routing schemes for supporting load balancing. In *Proc. of 5th Intl. Conference on High Speed Networks and Multimedia Communications*, 2002.
- [21] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput”. *IEEE/ACM Transactions of Networking*, 11(4), Aug. 2003.
- [22] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A measurement-based admission control algorithm for integrated services packet networks. *Proceedings of ACM SIGCOMM’95*, Sept. 1995.
- [23] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi. Capprobe: a simple and accurate capacity estimation technique. *SIGCOMM Comput. Commun. Rev.*, 34(4):67–78, 2004.
- [24] A. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *Proceedings of ACM SIGCOMM’02, (Pittsburgh, PA), August 2002.*, 2002.
- [25] J. F. Kurose and K. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [26] S. Kweon and K. Shin. Distributed QoS routing with bounded flooding for real-time applications. Technical Report CSE-TR-388-99, University of Michigan, 1999.
- [27] W. Lee. Topology aggregation for hierarchical routing in ATM networks. In *Proc. of SIGCOMM*, pages 82–92, 1995.

- [28] Z. Li and P. Mohapatra. QRON: QoS-aware routing in overlay networks. *IEEE Journal on Selected Areas in Communications*, January, 2004.
- [29] X. Masip-Bruin, S. Sanchez-Lopez, J. Sole-Pareta, and J. Domingo-Pascual. “QoS routing algorithms under inaccurate routing information for bandwidth constrained applications”. In *Proc. of Intl. Conference on Communications*, pages 1743–1748, 2003.
- [30] S. Ming-Hong, W. Si-Bing, and B. Ying-Cai. “A bandwidth constrained QoS routing optimization algorithm”. In *Proc. of Intl. Conference on Communication Technology*, pages 491–494, April 2003.
- [31] S. Norden and J. Turner. Inter-domain QoS routing algorithms. Technical Report WUCS-02-03, Department of Computer Science, WUCS, 2002.
- [32] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching architecture. *IETF RFC 3031*, 2001.
- [33] A. Shaikh, J. Rexford, and K. Shin. Efficient precomputation of Quality-of-Service routes. In *Proc. of NOSSDAV*, pages 15–27, July 1998.
- [34] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Proceedings of the 3rd ACM SIGCOMM/USENIX conference on Internet measurement (IMC 2003)*, 2003.
- [35] 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.
- [36] W. Wang, C. Jin, and S. Jamin. Network overlay construction under limited end-to-end reachability. In *Proceedings of IEEE INFOCOM*, 2005.
- [37] B. M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Communications*, 6(9):1617–1622, Dec. 1988.
- [38] S. Yuen and B. Li. Strategyproof mechanisms for dynamic multicast tree formation in overlay networks. In *Proceedings of IEEE INFOCOM*, 2005.
- [39] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: a new resource reservation protocol. *IEEE Network*, Sept. 1993.
- [40] S. Zhuang, D. Geels, I. Stoica, and R. H. Katz. On failure detection algorithms in overlay networks. In *Proceedings of IEEE INFOCOM*, 2005.