Aggregation-Based QoS Routing in the Internet

Ronghui Hou[1], Ka-Chung Leung[1], King-Shan Lui[1], Ka-Cheong Leung[1], and Fred Baker[2]
[1]Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong
[2]Cisco Research Center, Cisco, USA
Email: {rhhou, kachung, kslui, kcleung}@eee.hku.hk; fred@cisco.com

***Abstract*—In this paper, we study the problem of QoS routing with two concave constraints in the Internet. We propose an efficient approach for computing the supported QoS across domains based on the aggregated intradomain topology. The time complexity of our approach is polynomial. Moreover, our approach can be incorporated in the distance-vector based routing protocol, such as BGP, the de facto interdomain routing protocol in the Internet. Our simulation results show that our interdomain QoS routing protocol can successfully serve more than 80% of the total connection requests and is very scalable.**

***Index Terms*—QoS routing, hierarchical networks, concave constraints, topology aggregation.**

I. Introduction

With the popularity of distributed multimedia applications, the Internet has to be enhanced to provide the desired quality-of-service (QoS) guaranteed connection for the multimedia applications [1]. Very often, users have more than one QoS requirements, such as machine capacity (CPU, memory, or disk storage), bandwidth, and delay, etc. The machine capacity and bandwidth are concave QoS metrics [2], [3], while delay is an additive QoS metric. The capacity of a path is the minimum of the capacities for all nodes on this path, and the bandwidth of a path is the minimum of the bandwidths for all links on this path. On the other hand, the delay of a path is the sum of the delays for all links on this path. In this work, we assume that each link is associated with two independent concave QoS metrics. A metric, such as the machine capacity, that is originally associated with a node, can be transformed to a link metric where the capacity of a link is the minimum of the capacities of the two end nodes of the link. We denote two concave QoS metrics as metric S and metric W in our subsequent discussion.

In this work, we consider the problem of routing with two concave constraints in the Internet. That is, we want to find a path that can support two concave requirements of a request. In the Internet, nodes are grouped into different domains or autonomous systems (ASes). A node in a domain has no topology information of other domains. Some nodes are called border nodes if they are connected with other border nodes in other domains. Given two border nodes, if they are in the same domain or are directly connected with each other, we call one is the *border neighbor* of another. Therefore,

This work is supported in part by the Cisco Research Initiative Award.
Manuscript received January 22, 2009; revised July 21, 2009; accepted August 15, 2009.

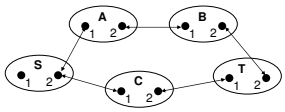
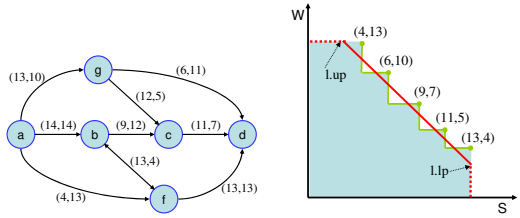


Fig. 1. A simple Internet topology.

there may be several hops between two border neighbors. Fig. 1 illustrates a simple Internet topology. There are six domains and only border nodes are shown. Fig. 2(a) is the topology of Domain C where Node a is border node $C.1$ and Node d is border node $C.2$, respectively. $C.1$ and $C.2$ are border neighbors of each other, but not directly connected as shown in Fig. 2(a).



(a) intradomain topology. (b) S-W plane.

Fig. 2. illustration for supported QoS.

In the Internet, we call a pair of border nodes in the same domain are connected via a *logical link*. A link that connects two nodes directly is a *physical link*. According to BGP, the de facto interdomain routing protocol in the Internet, each border node has to compute the QoS metrics of the logical link from itself to each border node in the same domain, and advertise it to its border neighbors. The QoS information advertised should reflect the QoS supported by the paths between the two border nodes in the same domain. When the physical links are associated with one QoS metric, a single best path can be identified and its QoS metric is advertised. However, when each physical link is associated with two concave QoS metrics, we may not be able to find the best path between any two border nodes. For instance, let (x, y) associated with each edge in Fig. 2(a) represent the QoS metrics of S and W, respectively. We also call tuple (x, y) a QoS parameter. Since the QoS metrics are concave, the QoS parameter of path $p_1 = a \rightarrow g \rightarrow d$ is (min(13, 6), min(10, 11)) = $(6, 10)$ while the QoS parameter of path $p_2 = a \rightarrow b \rightarrow c \rightarrow d$ is $(9, 7)$. p_1 is better in terms of W and p_2 is better in terms of S. No matter which

doi:10.4304/jcm.5.3.239-246

path is selected as the "best", either path cannot represent accurately the supported QoS from a to d. In Section II, we will show that, if the logical link from a to d in Fig. 2(a) is associated with the QoS parameters $(4, 13)$, $(6, 10)$, $(9, 7)$, $(11, 5)$, and $(13, 4)$, node a can always identify whether a request is feasible and never rejects a feasible request. In other words, these QoS parameters define the supported QoS of routing from a to d.

In the distance-vector based approach, such as BGP, each border node computes the (intradomain) supported QoS between any two border nodes in the same domain and advertises this information to other border nodes in other domains. Based on the received supported QoS, each border node can compute the supported QoS of routing across multiple domains. We now use an example in Fig. 1 to illustrate the process of inter-domain supported QoS computation. We consider the process of computing the supported QoS from $S.2$ to the destination domain T. In the first step, $B.2$ and $C.2$ are directly connected to domain T, and they advertise the supported QoS from themselves to T to $B.1$ and $C.1$, respectively. In the second step, $B.1$ and $C.1$ compute supported QoSes from themselves to domain T, and advertise them to $A.2$ and $S.2$, respectively. In step 3, $S.2$ receives the supported QoS from $C.1$ to domain T, and computes the supported QoS from itself to domain T, via $C.1$. In step 4, $S.2$ receives the supported QoS from $A.1$ to T, and computes the supported QoS from itself to domain T via $A.1$. Now, there are two different supported QoSes from $S.2$ to domain T. One goes through $A.1$ and another goes through $C.1$. By aggregating these two supported QoSes, border node $S.2$ obtains the total supported QoS from itself to domain T. Since the supported QoS is represented by a set of QoS parameters and the number of these QoS parameters depends on the network size and topology, it is not scalable for each border node to advertise all these QoS parameters. The work in [4] proposes the *line-segment* approximation method which applies a line segment to approximately represent the supported QoS, which will be discussed in Section II. By applying the line segment aggregation mechanism, the supported QoS can be defined by two QoS parameters, which is scalable.

In this work, we apply a line segment to approximately define the supported QoS between two border nodes. Accordingly, there are two issues we need to consider.

1) How to compute the supported QoS from s to a destination d based on the supported QoSes from s to u and from u to d, which are defined by line segments?
2) Assume a border node has several border neighbors and has computed the supported QoS to a destination via each border neighbor. How does this border node obtain the total supported QoS from itself to the destination?

A preliminary version of this paper can be found in [5]. This paper enhances the mechanisms, and provides formal proofs and complexity analysis. Note that when a border node finds a feasible path for a connection request, it should reserve the network resources for this connection, and a link state update process should be initiated for obtaining the accurate network topology information. In this work, our focus is on the problem of computing the supported QoS information based on the accurate QoS metrics of each link, and we would not consider how to reserve the network resources, update the network state information, and estimate the QoS metric of each link.

The rest of the paper is organized as follows. Section II gives some notations and definitions in this paper, and introduces the existing line segment aggregation method proposed in [4]. In Section III, we propose the mechanism to cope with the first issue, formally prove the correctness of our mechanism, and analyze its computational complexity. In Section IV, we consider the second issue. We present the mechanism for finding the aggregated supported QoS and analyze its computational complexity. Our simulation results are discussed in Section V. Section VI compares and discusses some existing works on QoS routing in the Internet. Finally, we conclude our work in Section VII.

II. LINE SEGMENT AGGREGATION METHOD

We mentioned earlier that the logical link between any two border nodes in the same domain may be associated with several QoS parameters. Now, we describe how to find the QoS parameters of the logical link between two border nodes in the same domain. For the ease of discussion, we plot the QoS parameter on the S-W plane, as illustrated in Fig. 2(b), where each point denotes the QoS parameter of a path from a to d in Fig. 2(a). The shaded area is called the *feasible region*. Any connection request with the QoS requirements falling in the feasible region can be supported by at least one physical path.

Definition 1: An QoS parameter p is more representative than the QoS parameter p', denoted by $p \succ p'$ or $p' \prec p$, if and only if

- $p.s \neq p'.s$ *or* $p.w \neq p'.w$, *and*;
- $p.s \geq p'.s$ *and* $p.w \geq p'.w$.

In Fig. 2(a), there are two paths $p_1 = a \rightarrow g \rightarrow d$ with the QoS parameter $(6, 10)$ and $a \rightarrow f \rightarrow b \rightarrow c \rightarrow d$ with the QoS parameter $(4, 4)$. We have $(6, 10) \succ (4, 4)$. As illustrated in Fig. 2(b), the feasible region defined by $(4, 4)$ is included in that of $(6, 10)$. We cannot find any path whose QoS parameter is more representative than $(6, 10)$, and we say that p_1 is a *non-dominated path*. The feasible region is defined by five points $(4, 13)$, $(6, 10)$, $(9, 7)$, $(11, 5)$, and $(13, 4)$, which are the QoS parameters of all the non-dominated paths from a to d. Therefore, computing the supported QoS between any two border nodes in the same domain is to find all the non-dominated paths, which can be solved in polynomial time [6].

The work in [4] proposes a line segment to approximate the staircase. Thus, the feasible region is approximately defined by a line segment, such as line segment l in Fig. 2(b). Such line segment is found by applying the *Method of Least Squares*.

For the ease of discussion, we would like to introduce some notations used in the context. Given a QoS parameter p, denote $p.s$ as the metric S of p, and $p.w$ as the metric W of p. Denote a line segment l as $[l.up, l.lp]$, where $l.up$ and $l.lp$ are called the upper and lower endpoints of l, respectively. The feasible region defined by l is compassed by the three line segments $[(0, l.up.w), l.up]$, $[l.up, l.lp]$, and $[l.lp, (l.lp.s, 0)]$, as illustrated in Fig. 2(b). When $l.lp = l.up$, the supported QoS can be represented by a point $l.lp$, which means that there is only one non-dominated path between the source and the destination.

III. QOS JOIN OPERATION

Assume that we have found the supported QoS from border node a to its border neighbor u, and the supported QoS from border node u to a border node d. Note that there may exist several border nodes between u and d, that is, the paths between u and d go across several domains, as discussed in Section I. In this section, we discuss how to find the supported QoS supported by all paths from a to d via u. Denote $\mathbb{R}_1$ and $\mathbb{R}_2$ as the feasible region supported by all paths from a to u and from u to d, respectively. We have the following lemma.

Lemma 1: Let $\mathbb{R}_1$ and $\mathbb{R}_2$ be the feasible region supported by all paths from a to u and from u to d, respectively. The feasible region supported by all paths from a to d via u, denoted by $\mathbb{R}$, is $\mathbb{R}_1 \cap \mathbb{R}_2$.

Proof: Given any point (c_s, c_w) located in $\mathbb{R}_1 \cap \mathbb{R}_2$, we can find a path $\mathcal{P}_1$ from a to u and a path $\mathcal{P}_2$ from u to d, both of which have the metric S not less than c_s and the metric W not less than c_w. Let path $\mathcal{P}$ be the path from a to d which is concatenated with $\mathcal{P}_1$ and $\mathcal{P}_2$. Path $\mathcal{P}$ can satisfy the QoS requirement (c_s, c_w). This means that given any QoS requirement falling in $\mathbb{R}_1 \cap \mathbb{R}_2$, we can find a feasible path from a to d satisfying this request. We thus have $\mathbb{R}_1 \cap \mathbb{R}_2 \subseteq \mathbb{R}$.

For any point $p = (c_s, c_w)$ that is outside $\mathbb{R}_1 \cap \mathbb{R}_2$, it is either not in $\mathbb{R}_1$ or $\mathbb{R}_2$, or not in both. Without loss of generality, let p be not in $\mathbb{R}_2$. Thus, we cannot find a feasible path from u to d satisfying the QoS requirement (c_s, c_w). In other words, all paths from s to d via u cannot support any request with the QoS requirement falling outside $\mathbb{R}_1 \cap \mathbb{R}_2$. This implies that $\mathbb{R} \subseteq \mathbb{R}_1 \cap \mathbb{R}_2$.

Based on the arguments above, we have $\mathbb{R} = \mathbb{R}_1 \cap \mathbb{R}_2$. ■

If $\mathbb{R}_1$ and $\mathbb{R}_2$ are represented by points p_1 and p_2, respectively, $p_1 \oplus p_2$ defines the feasible region $\mathbb{R}_1 \cap \mathbb{R}_2$. We call $\oplus$ the *join* operation. If $\mathbb{R}_1$ is represented by a line segment l and $\mathbb{R}_2$ is represented by a point p, $p \oplus l$ defines the feasible region $\mathbb{R}_1 \cap \mathbb{R}_2$. The solutions for computing $p_1 \oplus p_2$ and $p \oplus l$ can be referred to [5]. In this section, we focus on the more general situation that $\mathbb{R}_1$ and $\mathbb{R}_2$ are defined by line segments l_1 and l_2, respectively. Let $l_1 \oplus l_2$ represent the region $\mathbb{R}_1 \cap \mathbb{R}_2$.

Given two specified line segments l_1 and l_2, by plotting l_1 and l_2 on the S-W plane, we can easily find the result of $l_1 \oplus l_2$ based on Lemma 1. The work in [5] also describes how to compute the feasible region based on the endpoints of the line segments, but the solution is not complete. In the following, we provide a complete mechanism for computing $l_1 \oplus l_2$ and formally prove its correctness. Based on Lemma 1, we have the following corollary.

Corollary 1: Define $su = \min\{l_1.lp.s, l_2.lp.s\}$ and $wu = \min\{l_1.up.w, l_2.up.w\}$. The feasible region for $l_1 \oplus l_2$ must be inside the region spanned by $[0, su] \times [0, wu]$.

Proof: By Lemma 1, the feasible region defined by $l_1 \oplus l_2$ is the intersection of the feasible region for l_1 and that for l_2. Therefore, the new feasible region would be bounded by the minimum W of the upper endpoints and the minimum S of the lower endpoints. ■

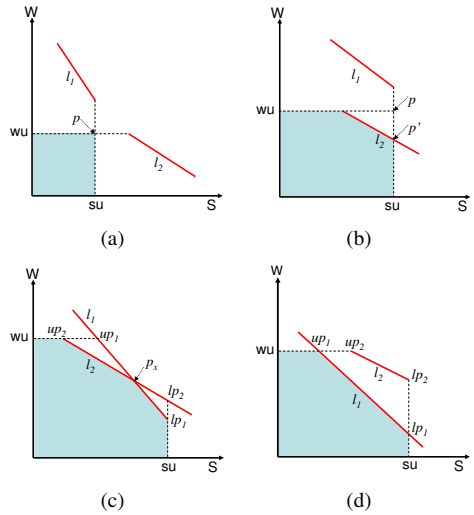


Fig. 3. An illustration of the join operation for the two line segments l_1 and l_2.

We new describe the details on how to identify $l_1 \oplus l_2$. From the illustration in Fig. 3, both l_1 and l_2 may or may not intersect the region $\mathbb{R} = [0, su] \times [0, wu]$, where $su = \min\{l_1.lp.s, l_2.lp.s\}$ and $wu = \min\{l_1.up.w, l_2.up.w\}$. There are three different cases:

Case I: Both l_1 and l_2 do not intersect the region $\mathbb{R}$, as illustrated in Fig. 3(a). The feasible region defined by $l_1 \oplus l_2$ is thus $[0, su] \times [0, wu]$. That is, $l_1 \oplus l_2$ is a point (su, wu).

Case II: Only one line segment, say l_2, intersects the region $\mathbb{R}$ while the other is outside the region $\mathbb{R}$. Let l_2' be the segment of l_2 which is located in the region $[0, su] \times [0, wu]$. In this case, the feasible region defined by $l_1 \oplus l_2$ is represented by l_2'. For instance, in Fig. 3(b), $l_1 \oplus l_2 = [l_2.up, p']$.

Case III: If both l_1 and l_2 intersect the region $\mathbb{R}$. Let l_1' and l_2' be the segments of l_1 and l_2, respectively, which are located in the region $[0, su] \times [0, wu]$. For instance, in Figs. 3(c)-3(d), $l_1' = [up_1, lp_1]$ and $l_2' = [up_2, lp_2]$. We need to consider two cases.

1) l_1 does not intersect with l_2. In that case, the line segment that is located at a lower position defines

the feasible region. For instance, in Fig. 3(d), l_1 is at a lower position than l_2. Therefore, $l_1 \oplus l_2 = l'_1$.

2) l_1 intersects with l_2. In this case, we denote p_x as the intersection point. Since $p_x.s < su$ and $p_x.w < wu$, p_x is located in the region $[0, su] \times [0, wu]$. p_x divides l'_1 into two parts $[l'_1.up, p_x]$ and $[p_x, l'_1.lp]$. Similarly, p_x divides l'_2 into two parts $[l'_2.up, p_x]$ and $[p_x, l'_2.lp]$. Without the loss of generality, assume that $[l'_1.up, p_x]$ is sitting above $[l'_2.up, p_x]$. Then, $[p_x, l'_2.lp]$ must be located at a higher position than $[p_x, l'_1.lp]$, as shown in Fig. 3(c). In this case, two line segments $[l'_2.up, p_x]$ and $[p_x, l'_1.lp]$ form the boundary of the feasible region defined by $l_1 \oplus l_2$. Unfortunately, this region cannot be represented using two points as a line segment. For advertisement purposes, we can apply the method of least squares to get an approximate line segment.

To determine $l_1 \oplus l_2$, we need only a few operations on checking whether two line segments intersect and finding the intersection points. These operations can be done in $\mathcal{O}(1)$ time [7]. Therefore, the computational complexity for computing $l_1 \oplus l_2$ is $\mathcal{O}(1)$.

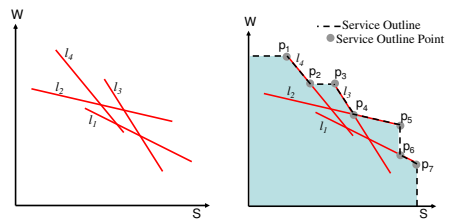
IV. QoS Aggregation

In the previous section, we discussed how to compute the supported QoS from border node b_a to border node b_d via border neighbor u of a. In practice, border node b_a may have several border neighbors. A request can be served by a path via either one of the neighbors. Therefore, the supported QoS from b_a to b_d should be the union of the supported QoSes provided by the neighbors. For example, b_a has computed four different line segments which represent the supported QoSes from itself to b_d through different border neighbors, as illustrated in Fig. 4(a). The feasible region from b_a to b_d is the union of the different feasible regions defined by these four different line segments. The shaded area in Fig. 4(b) illustrates the aggregated feasible region.

We call the boundary of the aggregated feasible region the *service outline*, which is composed by several line segments. As shown in Fig. 4(b), the service outline is composed by eight line segments. We call the points defining the service outline the *service outline points*. In Fig. 4(b), the service outline points is $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$.

We presented a mechanism to find the service outline in [8] but for a concave and an additive QoS metric pair, which cannot be directly applied in the situation studied in this paper. In this section, we describe how to identify the service outline when two concave metrics are considered and provide the complexity analysis. As a matter of fact, our mechanism can also be applied for finding the aggregated supported QoS with additive-concave or two additive QoS metrics.

Given a line segment l, the service outline of a line segment l is composed by three line segments $[(0, l.up.w), l.up]$, $[l.up, l.lp]$, and $[l.lp, (l.lp.s, 0)]$, as illustrated in Fig. 2(b). Suppose that b_a has m neighbors and let $\mathcal{L} = \{l_1, \ldots, l_m\}$ be the set of line segments that



(a) The QoS lines. (b) The total service area.

Fig. 4. An illustration for QoS aggregation.

define the supported QoS from b_a to b_d via each of the neighbors.

We adopt an iterative approach that expands the existing feasible region by considering the line segments one by one. First, we compute the feasible region defined by l_1 only. We then consider l_2 and include the feasible region induced by l_2 if necessary. After m steps, we can find the service outline defining the aggregated feasible region. We now describe each step in details.

In step k, the *current service outline* defines the total feasible region, denoted by $\mathbb{R}_c$, for all line segments $\{l_1, \ldots, l_{k-1}\}$. That is, in the first step, the current service outline is the service outline of l_1. We call the line segments forming the service outline the *service outline segments*.

If the service outline of l_k is completely located in $\mathbb{R}_c$, l_k does not induce any new feasible region, and the current service outline should not be changed. To check whether this is the case, we verify whether the service outline segments of l_k ($[(0, l_k.up.w), l_k.up]$, l_k, and $[l_k.lp, (l_k.lp.s, 0)]$) intersect with the current service outline segments one by one in ascending S order. After identifying all the intersection points, we can determine the new region defined by l_k to be included in the feasible region. Finally, we remove all the current service outline segments located in the feasible region defined by l_k, and put all the service outline segments of l_k located outside the existing feasible region into the current service outline.

We would like to use the examples in Fig. 5 to illustrate how to identify the new service outline segments, where the dashed line segments are the current service outline segments. We first consider the example in Fig 5(a). In the first step, we identify the intersection points p_1, p_2, and p_3. The points divide the service outline segments of l_k into several segments. We then check whether each individual segment is inside the current feasible region or on the outside. For example, the segment $[l.up, p_1]$ is outside and defines a new feasible region, while the segment $[p_1, p_2]$ is inside the current feasible region. Finally, we expand the existing feasible region by including the feasible region induced by l_k. The new service outline is formed by replacing the segment $[(0, w_1), p_1]$ with $[(0, w_2), l.up]$ and $[l.up, p_1]$, and replacing the segment $[p_2, p_3]$ with $[p_2, l.lp]$ and $[l.lp, p_3]$. In Fig. 5(b), there are only two intersection points between the service outline segments of l_k and the current service outline segments.

l_k defines a new region defined by p_1, p_2, and p_3. The new service outline can be formed by replacing the segments $[p_1, p_3]$ and $[p_3, p_2]$ with $[p_1, p_2]$.

Now, we analyze the time complexity for finding the aggregated supported QoS. A point on the service outline is either an intersection point of two line segments in the service outline segments of $\mathcal{L}$ or an endpoint of a line segment in $\mathcal{L}$, where $\mathcal{L} = \{l_1, \ldots, l_m\}$. As a result, there are at most $\mathcal{O}(m^2)$ service outline segments in each step of our algorithm. Since it takes $\mathcal{O}(1)$ time to find the intersection between two line segments, it takes $\mathcal{O}(m^2)$ time to find all the service outline segments of l_k. Therefore, the time complexity for finding the aggregated supported QoS defined by $\mathcal{L}$ is $\mathcal{O}(m^3)$.

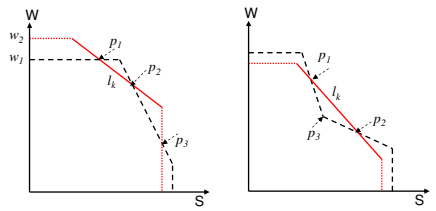


Fig. 5. Illustrations for identifying the service outline.

After finding all the service outline points, Node a will perform the method of least squares on them in order to obtain a line segment to approximate the service outline. As line segments only provide estimates of supported QoS, routing loops may occur due to the estimation errors and it leads to slow convergence. Therefore, we also apply the *threshold checking* and *advertisement history checking* techniques in [8] in our mechanism. Interested readers can refer to [8] for more detailed discussions.

V. Simulation

In this section, we present the performance evaluation and compare our protocol with the existing mechanisms. To our best knowledge, there are two existing QoS aggregation mechanisms which can be implemented in the distance-vector routing model. They are the best-point algorithm and the worst-point algorithm [4]. The best-point algorithm uses the largest metric W and the largest metric S in all the paths from a source to a destination to denote the supported QoS, while the worst-point algorithm uses the smallest metric W and the smallest metric S to denote the supported QoS. Our simulation experiments aim at comparing the performance of our routing protocol with those of the existing mechanisms.

A. Simulation configurations

We evaluate our protocol on two network types: (1) BRITE [9] topology generated by the BRITE software (version 2.1b Java generator), and (2) random topology. For the BRITE topology, both the inter-domain and the intra-domain topologies are generated using the Waxman [9] model. In random topology, any two nodes in the same domain are directly connected with the probability $p = 0.15$. We randomly select four nodes in each domain as the border nodes. For both topology types, we generate networks with 10 domains and 20 domains, each of which contains 50 nodes. The average number of border nodes in a domain is four. In the network with 10 domains, there are in average 40 interdomain links. In the network with 20 domains, there are in average 80 interdomain links. The metric S and metric W of each link independently fall in $[5, 10]$ and $[1, 10]$.

When the network contains 10 domains, we generate one QoS request for every possible "source node to destination domain" pair. This contributes to $50 \times 10 \times 10 = 5000$ QoS requests. While in the network with 20 domains, we randomly select 25 nodes from each domain, and generate one QoS request from each node to each domain in the network. This contributes to $20 \times 20 \times 25 = 10000$ QoS requests. We simulate 10 network instances, and the simulation results are the average values over these 10 topologies.

Two performance metrics are used to evaluate our routing protocol: *success ratio* and *crankback ratio*. The routing protocol may reject some feasible requests, or cannot successfully establish a connection for a feasible request. *Success ratio* is defined as the ratio of the number of requests successfully served by a protocol to that of the feasible requests actually supported by the network. On the other hand, the routing protocol may accept some infeasible request. *Crankback ratio* is defined as the number of infeasible request accepted by a protocol to the total number of feasible requests.

From the above definitions, it is easy to see that a good QoS routing protocol should have a high success ratio and a low crankback ratio. We can easily verify that the worst-point algorithm does not accept any infeasible request, and the best-point algorithm does not reject any feasible request. This implies that the crankback ratio of the worst-point is zero, and the success ratio of the best-point is 1. That is to say, we should evaluate the success ratio of the worst-point and the crankback ratio of the best-point.

B. Simulation Results

Fig. 6 and Fig. 7 show the performances of our routing protocol and the existing mechanisms with the change of the Metric W requirement of the generated connection requests under the BRITE topology type. Tables I and II show the average success ratios and the average crankback ratios of our protocol and the existing mechanisms. Each column corresponds to a network topology. For instance, "Domain 10(5-9)" means that the network contains 10 domains and the link metrics fall in the range $[5, 9]$. From Fig. 6, our protocol has a success ratio of 90% in many different settings of W requirement. However, the success ratios of the worst-point are often less than 50%, as illustrated in Fig. 7. Table I shows that the average success ratio of our protocol is larger than 80%, but that of the worst-point is less than 40%, which is very low. Moreover, we can observe that the average crankback

ratio of our protocol is less than 8%, but that of the best-point is larger than 40%, which is very high. From Table I, we also observe that the average success ratio of our protocol with the link metric range $[1, 10]$ is higher than that with the link metric range $[5, 10]$. As the number of link metrics increases, our routing protocol not only saves more advertisement overhead but also achieves a better performance.

Fig. 8 and Fig. 9 show the performances of our protocol and the existing mechanisms in randomly generated topologies. Tables III and IV show the average success ratios and the average crankback ratios of our protocol and the existing mechanisms. Under the random topology, we can get the similar observations as those under the BRITE topology. By comparing with Tables I and III, under the random network topology, the crankback ratio produced by our protocol is very small. As we know, the distortion introduced by the aggregation mechanism depends on the specific network topology and the current network state. Therefore, the performance of our protocol under the random topology is a little different than that under the BRITE topology. Generally speaking, our protocol achieves the average success ratio larger than 80% and the average crankback ratio smaller than 8%. Unfortunately, from Tables III and IV, the best-point and the worst-point still do not work well under the random network topology. Therefore, we can say that our protocol outperforms the existing mechanisms with the better tradeoff between the success ratio and the crankback ratio.

VI. Related Work

Provisioning QoS in the Internet has been paid much attention. Some works, such as [10], [11], consider that the QoS requirements can be divided into multiple quality levels. In this work, we consider that each QoS requirement is represented using a real number. For instance, a request can specify its bandwidth request to be 1Mbps. Some works, such as [12], [13], just consider one QoS metric. Although [13] studies the case of multiple QoS metrics, the path selection is just based on only one metric.

The work in [14] considers the interdomain routing with the delay and bandwidth metrics. However, this work does not consider the QoS information aggregation, and so the routing protocol is not scalable. Ref. [15] proposes a link-state based interdomain QoS routing architecture, where each domain is abstracted into a single routing agent. This work uses the worst-case scenario, such as the maximum delay or minimum bandwidth, to represent the supported QoS of each domain. In [4], it has been shown that the worst-case based aggregation mechanism produces huge distortion. The work in [16] considers the QoS routing with the bandwidth and cost metrics. This work assumes that an existing aggregation mechanism is applied, and gives the general model of computing the supported QoS across multiple domains. However, this work does not give any solution for designing the QoS routing algorithm with a specified topology aggregation mechanism. Ref. [17] presents the existing QoS information aggregation mechanisms.

In [4], the line segment aggregation method is shown to have the best performance by comparing with the existing aggregation methods. We would like to adopt the line segment aggregation mechanism for designing the QoS routing protocol with two concave metrics. The work in [8] also applies the line segment aggregation method to design the QoS routing protocol in the Internet, but it considers the additive and concave QoS metrics.

The work in [18] considers the QoS routing with two concave QoS metrics, computational capacity and bandwidth, in the Internet. There are two differences between [18] and our work. Firstly, [18] assumes that each physical link is associated with one QoS metric, bandwidth, and each border node is associated with another QoS metric, capacity. In this case, we can find the best path between any two border nodes in the same domain. Our work considers that each physical link is associated with two simultaneous concave QoS metrics, and we may not be able to find the best path between any two border nodes in the same domain. In other words, [18] does not involve the QoS aggregation problem. Secondly, [18] develops a mixed formula combining two independent concave QoS metrics, so that the shortest path algorithm can be used to select an "optimal" path. The path selection mechanism in our work considers two independent QoS metrics simultaneously. Moreover, the mixed formula proposed in [18] is based on the specified QoS requirements, while our work focuses on finding the supported QoS which has no specified QoS requirements. To the best of our knowledge, this paper presents the only work on the interdomain routing with two concave QoS metrics in the Internet.

VII. Conclusion

In this paper, we developed the mechanisms for computing the supported QoS between two border nodes in different domains based on the distance-vector approach. By comparing with the existing mechanisms, our protocol has the higher success ratio and the lower crankback ratio. The success ratio measures how well our routing protocol serves the feasible requests, while the crankbacked ratio evaluates how serious the network resources are wasted by the routing protocol. Therefore, our protocol outperforms the existing protocols.

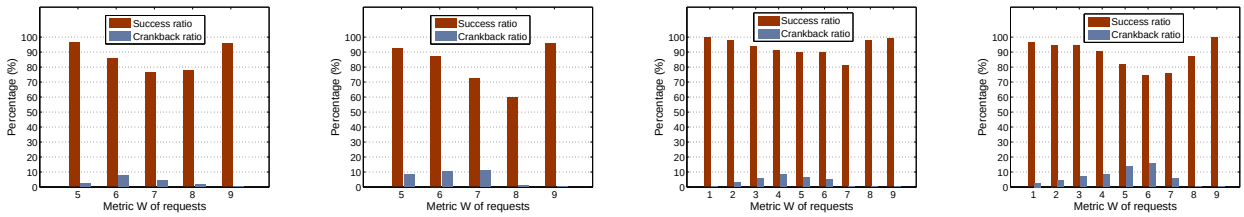
References

[1] S. Chen and K. Nahrstedt, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions," *IEEE Network Mag.*, pp. 64–79, November 1996.

[2] X. Gu and K. Nahrstedt, "A scalable QoS-aware service aggregation model for peer-to-peer computing grids," *Proc. of High Performance Distributed Computing*, pp. 73–82, July 2002.

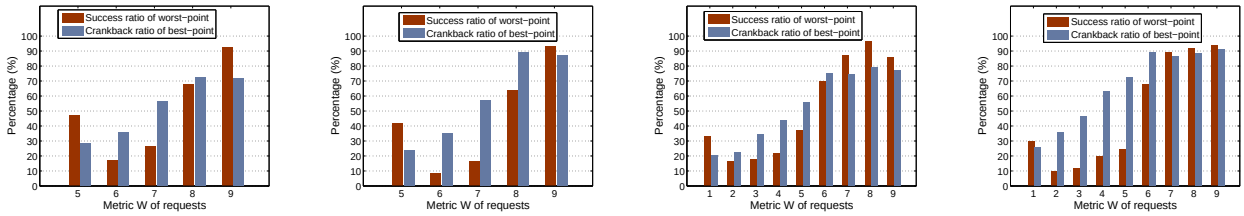
[3] J. Jin and K. Nahrstedt, "A distributed approach for QoS service multicast with geometric location awareness," *IEEE Distributed Systems Online*, vol. 4, no. 6, June 2003.

[4] K.-S. Lui, K. Nahrstedt, and S. Chen, "Routing with Topology Aggregation in Delay-Bandwidth Sensitive Networks," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 17–29, February 2004.



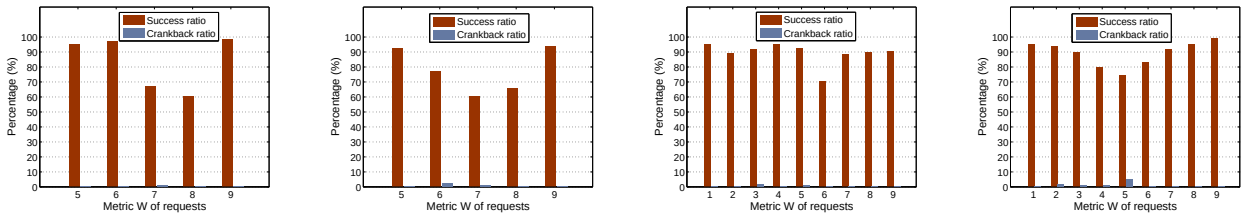
(a) 10-domain networks with the link metric $w \in [5, 9]$. (b) 20-domain networks with the link metric $w \in [5, 9]$. (c) 10-domain networks with the link metric $w \in [1, 9]$. (d) 20-domain networks with the link metric $w \in [1, 9]$.

Fig. 6. The performance of our protocol with BRITE topology.



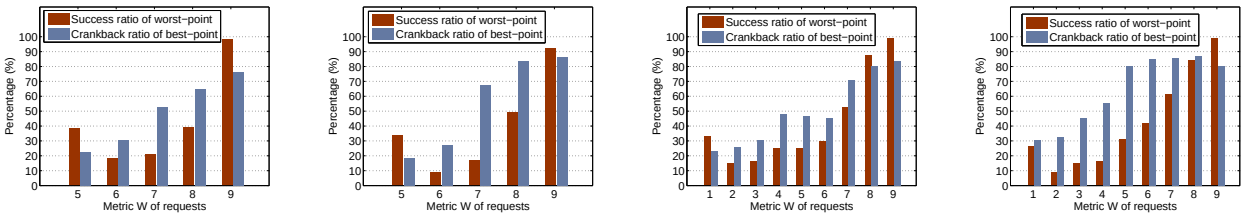
(a) 10-domain networks with the link metric $w \in [5, 9]$. (b) 20-domain networks with the link metric $w \in [5, 9]$. (c) 10-domain networks with the link metric $w \in [1, 9]$. (d) 20-domain networks with the link metric $w \in [1, 9]$.

Fig. 7. The performance of the existing mechanisms with BRITE topology.



(a) 10-domain networks with the link metric $w \in [5, 9]$. (b) 20-domain networks with the link metric $w \in [5, 9]$. (c) 10-domain networks with the link metric $w \in [1, 9]$. (d) 20-domain networks with the link metric $w \in [1, 9]$.

Fig. 8. The performance of our protocol with random topology.



(a) 10-domain networks with the link metric $w \in [5, 9]$. (b) 20-domain networks with the link metric $w \in [5, 9]$. (c) 10-domain networks with the link metric $w \in [1, 9]$. (d) 20-domain networks with the link metric $w \in [1, 9]$.

Fig. 9. The performance of the existing mechanisms with random topology.

	Domain 10 (5-9)	Domain 20 (5-9)	Domain 10 (1-9)	Domain 20 (1-9)
Our protocol	86.95	83.73	93.11	88.76
Worst-point	36.32	27.58	30.4	21.72

TABLE I
THE AVERAGE SUCCESS RATIOS (%) WITH BRITE TOPOLOGY.

	Domain 10 (5-9)	Domain 20 (5-9)	Domain 10 (1-9)	Domain 20 (1-9)
Our protocol	4.66	7.59	4.44	7.77
Best-point	45.16	46.66	43.51	55.18

TABLE II
THE AVERAGE CRANKBACK RATIOS (%) WITH BRITE TOPOLOGY.

	Domain 10 (5-9)	Domain 20 (5-9)	Domain 10 (1-9)	Domain 20 (1-9)
Our protocol	83.72	80.57	90.06	89.63
Worst-point	30.99	24.22	27.47	21.62

TABLE III
THE AVERAGE SUCCESS RATIOS (%) WITH RANDOM TOPOLOGY.

	Domain 10 (5-9)	Domain 20 (5-9)	Domain 10 (1-9)	Domain 20 (1-9)
Our protocol	0.2	0.94	0.45	1
Best-point	42.57	46.56	42.43	56.55

TABLE IV
THE AVERAGE CRANKBACK RATIOS (%) WITH RANDOM TOPOLOGY.

[5] K.-C. Leung, K.-S. Lui, K.-C. Leung, and F. Baker, "Quality-of-Service Routing with Two Concave Constraints," *2008 IEEE International Conference on Communications (ICC 2008)*, pp. 5746–5750.

[6] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE J. Select. Areas Comm.*, vol. 14, pp. 1228–1234.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, "*Introduction to Algorithms*," September 2001.

[8] W.-Y. Tam, K.-S. Lui, S. Uludag, and K. Nahrstedt, "Quality-of-Service Routing with Path Information Aggregation," *Elsevier Computer Networks Journal*, vol. 51, no. 12, pp. 3574–3594, August 2007.

[9] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," August 2001.

[10] M. P. Howarth, P. Flegkas, G. Pavlou, N. Wang, P. Trimintzios, D. Griffin, J. Griem, M. Boudcadair, A. Asgari, and P. Georgatsos, "Provisioning for Interdomain Quality of Service: the MESCAL Approach," *IEEE Communications Magazine*, pp. 129–137, June 2005.

[11] D. Griffin, J. Spenser, J. Griem, M. Boucadair, P. Morand, M. Howarth, N. Wang, G. Pavlou, A. Asgari, and P. Georgatsos, "Interdomain Routing through QoS-Class Planes," *IEEE Communications Magazine*, pp. 88–95, February 2007.

[12] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz, "OverQoS: An Overlay Based Architecture for Enhancing Internet QoS," pp. 71–84, March 2004.

[13] M. Boucadair, "The meta-QoS-class concept: a step towards global QoS interdomain services," *IETF Internet draft*, July 2005.

[14] D. Bauer, J. Daigle, I. Iliadis, and P. Scotton, "Topology aggregation for combined additive and restrictive metrics," *Computer Networks*, vol. 50, no. 17, pp. 3284–3299, 2006.

[15] I. T. Okumus, H. A. Mantar, J. Hwang, and S. J. Chapin, "Inter-Domain QoS Routing on Diffserv Networks: A Region-Based Approach," *Computer Communications*, vol. 28, no. 2, pp. 174–188, February 2005.

[16] A. Orda and A. Sprintson, "Precomputation schemes for QoS routing," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 578–591, August 2003.

[17] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of Topology Aggregation Techniques for QoS Routing," *ACM Computing Surveys (CSUR)*, vol. 39, no. 3, 2007.

[18] Z. Li and P. Mohapatra, "QRON: QoS-Aware Routing in Overlay Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 1, pp. 29–40, January 2004.

Ronghui Hou obtained her B.Eng., M.Eng., and Ph.D. degrees in Communication Engineering from Northwestern Polytechnical University, China, in 2002, 2005, and 2007, respectively. She is a Post-Doctoral Fellow in the University of Hong Kong. Her research interests include network quality of service issues and routing algorithm design.

Ka-Chung Leung received the B.Eng. degree in Computer Engineering from the University of Hong Kong in 2007. He is currently a postgraduate student in the Department of Electrical and Electronic Engineering at the same University. His research interests include quality of service routing and network protocol design.

King-Shan Lui received the B.Eng. and M.Phil. degrees in Computer Science from the Hong Kong University of Science and Technology. After receiving her PhD degree from the University of Illinois at Urbana-Champaign, USA, she joined the Department of Electrical and Electronic Engineering of the University of Hong Kong. Her research interests include network protocol design and analysis, sensor networks, and Quality-of-Service issues.

Ka-Cheong Leung received the B.Eng. degree in Computer Science from the Hong Kong University of Science and Technology, Hong Kong, in 1994, the M.Sc. degree in Electrical Engineering (Computer Networks) and the Ph.D. degree in Computer Engineering from the University of Southern California, Los Angeles, California, USA, in 1997 and 2000, respectively. He worked as Senior Research Engineer at Nokia Research Center, Nokia Inc., Irving, Texas, USA from 2001 to 2002. He was Assistant Professor at the Department of Computer Science at Texas Tech University, Lubbock, Texas, USA, between 2002 and 2005. Since June 2005 he has been with the University of Hong Kong, Hong Kong, where he is currently Research Assistant Professor at the Department of Electrical and Electronic Engineering. His research interests include transport layer protocol design, wireless packet scheduling, routing, congestion control, and quality of service guarantees in high-speed communication networks, content distribution, high-performance computing, and parallel applications.

Fred Baker has been working in networking technology, including the Internet, since 1978. He is a Fellow at Cisco Systems, and participates in the IETF.