Quality of Service Routing for Supporting Multimedia Applications

Zheng Wang and Jon Crowcroft

Department of Computer Science, University College London
Gower Street, London WC1E 6BT, United Kingdom

ABSTRACT

In recent years, several new architectures have been developed for supporting multimedia applications such as digital video and audio. However, quality of service routing is an important element that is still missing from these architectures. In this paper we consider a number of issues in QoS routing. We first examine the basic problem of QoS routing, namely, finding a path that satisfy multiple constraints, and its implications on routing metric selection, and then present three path computation algorithms for source routing and for hop-by-hop routing.

1. Introduction

Multimedia applications such as digital video and audio often have stringent quality of service (QoS) requirements. For a network to deliver performance guarantees it has to make resource reservation and excise network control. In the past several years, there have been much discussion and research in the area of resource setup, admission control and packet scheduling, and many new architectures have been proposed [1-3, 5-6, 9-14, 17-19].

One important element that is still missing from these architectures is quality of service (QoS) routing, namely, routing based on QoS requirements. A typical resource reservation process has two essential steps: finding resources and making reservations. Resource reservation can only be made when routing has found paths with sufficient resources to meet user requirements. Therefore, to support resource reservation, routing has to take into consideration the wide range of QoS requirments.

In traditional data networks, routing is primarily concerned with connectivity. Routing protocols usually characterize the network with a single metric such as hop-count or delay, and use shortest-path algorithms for path computation. However, in order to support a wide range of QoS requirements, routing protocols need to have a more complex model where the network is characterized with multiple metrics such as bandwidth, delay and loss probability. The basic problem of QoS routing is then to find a path that

satisfies multiple constraints. As current routing protocols are already reaching the limit of feasible complexity, it is important that the complexity introduced by the QoS support should not impair the scalability of routing protocols.

In this paper, we examine a number of issues in QoS routing in detail. We first look at the complexity of finding paths subject to multiple constraints, the selection of metrics for QoS routing, and then present three path computation algorithms both for source routing and hop-by-hop routing.

2. Complexity Analysis and Metric Selection

In this section, we first present some results on the problem of finding a path subject to multiple constraints, and then discussion metric selection based on our analysis.

2.1. Selection Criterion

Routing metrics are the representation of a network in routing; as such, they have major implications not only on the complexity of path computation, but also on the range of QoS requirements that can be supported. A number of factors have to be taken into consideration here:

1) For any metrics selected, efficient algorithms must exist for path computation, so that the routing protocol is able to scale to large networks such as the Internet. The complexity of the algorithms for path computation should preferably be comparable to that of current routing algorithms. It is also desirable that any algorithms should be able to work both in a centralized environment and a distributed environment.

2) The metrics must reflect the basic characteristics of a network. The information they contain should make it possible to support basic QoS requirements. Note that any QoS requirements have to be mapped onto the constraints on a path expressed in terms of the metrics, thus the metrics, to some extend, determine the types of QoS that the network can support. For example, if cost and bandwidth are the metrics, all QoS requirements have to be mapped onto cost and bandwidth. Some requirements such as reliability obviously can not be supported by such metrics.

3) Metrics should be orthogonal to each other so that there should no redundant information among the metrics. Redundant information can introduce inter-dependence among the metrics which makes it impossible to evaluate each metric independently. Recursive evaluation among metrics can substantially complicate path computation.

2.2. Single Mixed Metric

Path computation algorithms for a single metric, such as delay and hop-count, are well known and have been widely used in current networks. Thus, a natural question is whether a single metric can support user QoS requirements.

One possible approach might be to define a function and generate a single metric from multiple parameters. The idea is to mix various pieces of information into a single measure and use it as the basis for routing decisions. For example, a mixed metric M may be produced with bandwidth B, delay D and loss probability L with a formula $f(p) = \frac{B(p)}{D(p) \times L(p)}$. A path with a large value is likely to be a better choice in terms of bandwidth, delay and loss probability.

Single mixed metric, however, can only be used as an indicator at best as it does not contain sufficient information to assess whether user QoS requirements can be met or not. Another problem has to do with mixing parameters of different composition rules. For example, suppose that a path has two segments ab and bc. If metric $f(p)$ is delay, the composition rule is $f(ab+bc) = f(ab) + f(bc)$. If metric $f(p)$ is bandwidth, the rule is $f(ab+bc) = min[f(ab), f(bc)]$. However, if $f(p) = \frac{B(p)}{D(p) \times L(p)}$, neither of the above are valid. In fact, there may not be a simple composition rule at all.

We believe that the mixed metric approach is a tempting heuristic but it can at best be used as an indicator in path selection.

2.3. Multiple Metrics

Multiple metrics can certainly model a network more accurately. However, the problem is that finding a path subject to multiple constraints is inherently hard. Polynomial-time algorithms for the problem may not exist. A simple problem with two constraints called "shortest weight-constrained path" was listed in [8] as NP-complete but the proof has never been published. Jaffe [11] investigated this particular problem further and proposed two approximation algorithms that solve the problem in pseudopolynomial-time or polynomial-time if the lengths and weights have a small range of values. The running time of such NP-complete problems for real-world network topologies is investigated in [15].

The problem in QoS routing is much more complicated since the resource requirements specified by the applications are often diverse and application-dependent. The computation complexity is primarily determined by the composition rules of the metrics. There are three basic composition rules we are most interested in:

Definition: Let $d(i, j)$ be a metric for link (i, j). For any path $p = (i, j, k, ..., l, m)$, we say metric d is *additive* if

$$d(p) = d(i, j) + d(j, k) + \cdots + d(l, m)$$

We say metric d is *multiplicative* if

$$d(p) = d(i, j) \times d(j, k) \times \cdots \times d(l, m)$$

We say metric d is *concave* if

$$d(p) = min[d(i,j), d(j,k), ..., d(l,m)]$$

Let us now look at some parameters that are likely to be considered as routing metrics: delay, delay jitter, cost, loss probability and bandwidth. It is obvious that delay, delay jitter and cost follow the additive composition rule, and bandwidth follows the concave composition rule. The composition rule for loss probability is more complicated.

$$d(p) = 1 - ((1-d(i, j)) \times (1-d(j, k)) \times \cdots \times (1-d(l, m)))$$

However, loss probability metric can be easily transformed to an equivalent metric (the probability of successful transmission) that follows the multiplicative composition rule.

We now present three general NP-completeness Theorems for additive and multiplicative metrics. They form the foundation for our metric selection.

Theorem 1: Give a network $G = (N, A)$, n additive metrics $d_1(a), d_2(a), ..., d_n(a)$ for each $a \in A$, two specified nodes i, m, and n positive integers $D_1, D_2, ..., D_n$, ($n \geq 2, d_i(a) \geq 0, D_i \geq 0$ for $i = 1, 2, ..., n$), the problem of deciding if there is a simple path $p = (i, j, k, ..., l, m)$ which satisfies the following constraints $d_i(p) \leq D_i$ where $i = 1, 2, ..., n$ (the *n Additive Metrics Problem*) is NP-complete.

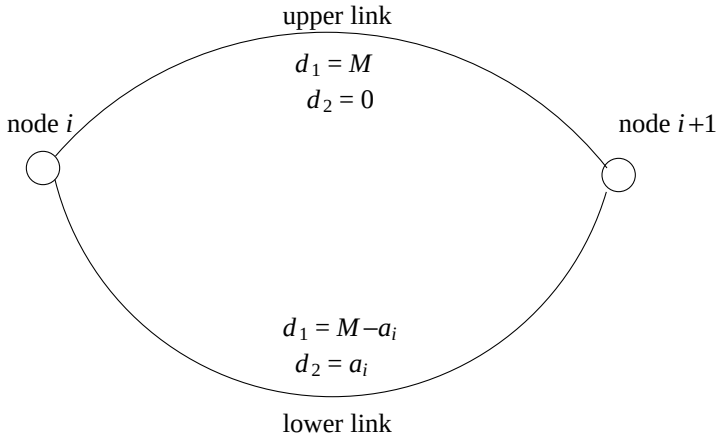

Figure 1: Assignment to Two Links Between Node i and $i+1$

Proof: We proceed by induction. First we show that *2 Additive Metrics Problem* is NP-complete. It is easy to see that *2 Additive Metrics Problem* $\in$ NP. Since *Partition* is a well-known NP-complete problem [8], we show *Partition* $\propto$ *2 Additive Metrics Problem* to prove its NP-completeness.

Given an instance of *Partition*, a set of numbers $a_1, a_2, ..., a_n$, construct a network with $n+1$ nodes and $2n$ links, two each from node i to $i+1$, $1 \le i \le n$ (see Figure 1). Let $S = \sum_{i=1}^{i=n} a_i$ and $M=2nS$. Let metric $d_1(i,i+1)$ for the two link from node i to node $i+1$ be M and $M-a_i$ respectively, and let metric $d_2(i,i+1)$ be 0 and a_i respectively ($0 \le a\_i \le 0$).

Consider an instance of *2 Additive Metrics Problem*:

$$d_1(p) \le nM - S/2 \tag{1}$$

$$d_2(p) \le S/2 \tag{2}$$

where p is a path between node 1 and node n. Note that for both the upper link and the lower link between i and $i+1$, we have

$$d_1(i,i+1) + d_2(i,i+1) = M$$

Therefore, for any possible path p between node 1 and node n,

$$d_1(p)+d_2(p) = \sum_{i=1}^{i=n}(d_1(i, i+1) + d_2(i, i+1)) = nM \tag{3}$$

From (1), we know $d_1(p) \le nM - S/2$. Thus,

$$d_2(p) \ge S/2 \tag{4}$$

From (2) we also have

$$d_2(p) \le S/2$$

Therefore, we get

$$d_2(p) = S/2$$

From (3), we also get

$$d_1(p) = S/2$$

Note that, for the two link from node i to node $i+1$, $d_2(i,i+1)$ be 0 and a_i. Therefore, there must be a sub-set of the original numbers with total exactly $S/2$. This solves the instance of *Partition*.

Conversely, if there is a subset of the original number set with total exactly $S/2$. For the two links between i and $i+1$, choose the lower link if a_i is in the subset. Otherwise, choose the upper link. For the

resulting path p, we get

$$d_2(p) = S/2$$

Since

$$d_1(p) + d_2(p) = nM$$

We also get

$$d_1(p) = nM - S/2$$

This solves the instance of *2 Additive Metrics Problem*.

We now show that *n Additive Metrics Problem* $\propto$ *n+1 Additive Metrics Problem*. Consider an instance of *n+1 Additive Metrics Problem*,

$$d_i(p) \le D_i, \; i=1, 2, ..., n+1 \tag{5}$$

Let D_{n+1} be a large number, say $D_{n+1} = \sum_{a \in A} d_{n+1}(a)$. Thus, we have $d_{n+1}(p) \le D_{n+1}$ for any path p. So $d_i(p) \le D_i, \; i=1, 2, ..., n$ if and only if (5) holds. This completes the proof. □

Theorem 2: Give a network $G = (N, A)$, n multiplicative metrics $d_1(a), d_2(a), ..., d_n(a)$ for each $a \in A$, two specified nodes i, m, and n positive integers $D_1, D_2, ..., D_n$, ($n \ge 2, d_i(a) \ge 1, D_i \ge 1$ for $i = 1, 2, ..., n$), the problem of deciding if there is a simple path $p = (i, j, k, ..., l, m)$ which satisfies the following constraints $d_i(p) \le D_i$ where $i = 1, 2, ..., n$ (the *n Multiplicative Metrics Problem*) is NP-complete.

Proof: It is easy to see that *n Multiplicative Metrics Problem* $\in$ NP. We show *n Additive Metrics Problem* $\propto$ *n Multiplicative Metrics Problem* to prove its NP-completeness.

Given an instance of *n Additive Metrics Problem* with $d_i(a)$ and D_i ($i=1, 2, ..., n$), define

$$d_i^*(a) = e^{d_i}$$

$$D_i^* = e^{D_i}$$

where $i=1, 2,..., n$

Consider an instance of *n Multiplicative Metrics Problem*

$$d_i^*(p) \le D_i^* \; i=1, 2, ..., n$$

Note that $D_i \ge 1$, $d_i^*(p) = \prod d_i^*(a)$ and $d_i(p) = \sum d_i(a)$ for $i = 1, 2, ..., n$ and $a \in A$. Therefore, $d_i^*(p) \le D_i^*$ if and only if $d_i(p) \le D_i$, where $i=1, 2, ..., n$ and $a \in A$. This completes the proof. □

Theorem 3: Give a network $G = (N, A)$, n additive and k multiplicative metrics $d_1(a), d_2(a), ..., d_{n+k}(a)$ for each $a \in A$, two specified nodes i, m, and $n+k$ positive integers $D_1, D_2, ..., D_{n+k}$, $(n \geq 1,\ k \geq 1, d_i(a) \geq 1, D_i \geq 0$ for $i = 1, 2, ..., n$, $D_i \geq 1$ for $i = n+1, 2, ..., n+k)$, the problem of deciding if there is a simple path $p = (i, j, k, ..., l, m)$ which satisfies the following constraints $d_i(p) \leq D_i$ where $i = 1, 2, ..., n+k$ (the *n Additive and k Multiplicative Metrics Problem*) is NP-complete.

Proof: It is easy to see that *n Additive and k Multiplicative Metrics Problem* $\in$ NP. We show *n+k Additive Metrics Problem* $\propto$ *n Additive and k Multiplicative Metrics Problem* to prove its NP-completeness.

Given an instance of *n+k Additive Metrics Problem* with $d_i(a)$ and D_i $(i=1, 2, ..., n+k)$, define

$$d_i^*(a) = d_i \tag{6}$$

$$D_i^* = D_i \tag{7}$$

where $i=1, 2, ..., n$, and

$$d_i^*(a) = e^{d_i}$$

$$D_i^* = e^{D_i}$$

where $i=n+1, n+2, ..., n+k$.

Consider an instance of *n Multiplicative Metrics Problem* with $d_i^*(p) \leq D_i^*$ $(i=1, 2, ..., n)$. Note that $D_i \geq 1$, $d_i^*(p)=\prod d_i^*(a)$ and $d_i(p)=\sum d_i(a)$, where $i=n+1, n+2, ..., n+k$ and $a \in A$. Therefore, $d_i^*(p) \leq D_i^*$ if and only if $d_i(p) \leq D_i$, where $i=n+1, n+2, ..., n+k$ and $a \in A$. From (6) and (7), we know $d_i^*(p) \leq D_i^*$ if and only if $d_i(p) \leq D_i$, where $i=1, 2, ..., n+k$ and $a \in A$. This completes the proof. □

The three Theorems above show that the problem of finding a path subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete. The results are applicable to to any metric that follows additive or multiplicative composition rules, and to any metrics that can be transformed to equivalent metrics that follow the additive or multiplicative composition rule.

2.4. Bandwidth and Delay as Metrics

Let us come back to delay, delay jitter, cost, loss probability, and bandwidth. It is clear that any two or more of delay, delay jitter, cost, loss probability in any combination as metrics are NP-complete. The only feasible combinations are bandwidth and one of the four (delay, loss probability, cost and delay jitter). Although delay, loss probability, cost and delay jitter are all very useful parameters, we believe that for the majority of applications, delay is comparatively more important than the others. For the rest of the paper,

we choose that bottleneck bandwidth and propagation delay as the routing metrics. However, the algorithms presented in the next sections are generic and apply to other routing metrics with similar composition rules, for example, bandwidth and jitter or bandwidth and cost.

The bandwidth we are interested here is the residual bandwidth that is available for new traffic. We define the bandwidth of a path as the minimum of the residual bandwidth of all links on the path or the *bottleneck bandwidth*. The delay has two basic components: queuing delay and propagation delay. Note that the queuing delay is determined by bottleneck bandwidth and traffic characteristics. Since queuing delay is already reflected in the bandwidth metric, we only need to consider propagation delay in the delay metrics. This way, we can make sure that the two metric are not inter-dependent.

Bottleneck bandwidth and propagation delay reflect some fundamental characteristics of a path in the network. We can view bottleneck bandwidth and propagation delay as the *width* and the *length* of a path. The problem of QoS routing is then to find a path in the network given the constraints on its width and length.

For most applications, particularly real-time ones, the end-to-end delay is one of the most important QoS requirements. The bottleneck bandwidth and propagation delay metrics provides the two pieces of essential information for applications and the network to work out the end-to-end delay, and also the relationship between the reserved bandwidth and the end-to-end delay.

Using bottleneck bandwidth and propagation delay as metrics is a compromise between complexity and optimality. As it is hard to find a path in a network which satisfies all requirements, we first find some candidate paths based on the bandwidth/delay metrics where efficient algorithms exist. Other requirements, for example, loss probability, jitter and cost, can still be considered in the admission control and resource setup protocols.

3. Path Computation Algorithms

In this section, we first examine the implications of source routing and hop-by-hop routing for QoS routing, and present three path computation algorithms for finding a path in a network for any given constraints on bottleneck bandwidth and propagation delay.

3.1. Source Routing and Hop-by-Hop Routing

Source routing and hop-by-hop routing are the two basic routing architectures for data networks. Hop-by-hop routing is the common form of general-purpose routing in current networks while source routing is mainly used for network diagnosis and special policy routes [4]. We now examine their implications for

path computation for QoS routing.

In source routing, a forwarding path is computed on-demand at the source and listed in the packet header. Packets are forwarded according to the path in the packet. Since the computation is is done for each individual request in a centralized fashion, source routing is very flexible; a source can any algorithm of its choice, or use a couple of algorithms for different purposes. However, a source must have access to full routing information for each link for path computation, and packets have a larger packet header. There is also an initial computation delay during the setup.

In hop-by-hop routing, packets are forwarded hop-by-hop at each node. Each node has a routing table with next hops for all destinations, and this table is usually computed periodically in response to routing updates. When a packet is received, hop-by-hop routing only requires a table lookup to find the next hop and send the packet to it. The packet header can be much smaller compared with source routing as the packets do not have to carry the full forwarding path. Hop-by-hop routing can use fully distributed computation algorithms [7] which has lower memory requirements for the routers.

We believe that both source routing and hop-by-hop routing architectures have important roles to play in QoS routing. Since QoS requirements are diverse, it is difficult to specify a set of general requirements that can apply to most applications. Therefore, source routing, which computes forwarding paths on-demand on a per-flow basis, fits very well. On the other hand, hop-by-hop routing allows distributed computation and has the advantage of smaller overhead and little setup delay. Thus, we can use hop-by-hop routing for general-purpose QoS routing, and use source routing for handling special cases and as a mechanism to override general routing.

3.2. Source Routing Algorithm

We now present a centralized algorithm suitable for source routing. Given bandwidth and delay constraints, the algorithm finds a path that satisfies both constraints, if such a path exists.

Consider a directed graph $G = (N, A)$ with number of nodes N and number of arcs A, in which each arc (i, j) is assigned two real numbers, b_{ij} as the available bandwidth and d_{ij} as the propagation delay. To simplify the notation, let $b_{ij} = 0$ and $d_{ij} = \infty$ if (i, j) is not an arc of the graph. Given any directed path $p = (i, j, k, ..., l, m)$, the width of the path $width(p)$ is defined as the bottleneck bandwidth of the path, i.e. $width(p) = min[b_{ij}, b_{jk}, \ldots, b_{lm}]$, and the length of the path $length(p)$ is defined as the sum of propagation delay, i.e. $length(p) = d_{ij} + d_{jk} + \cdots + d_{lm}$.

Given any two nodes i and m of the graph, and two constraints W and D, the QoS routing problem is then to find a path p^* between i and m so that $width(p^*) \geq B$ and $length(p^*) \leq D$. We refer to such paths

as bandwidth-delay-constrained paths.

Theorem 4: *A path has a width no less of B, if and only if each link in the path has a bandwidth no less than B*

Proof: If each link in p has a bandwidth no less than B, it is obvious that $width(p) \geq B$. Suppose that $width(p) \geq B$ but there is a link b_{ij} in p has a bandwidth less than B. We then have $width(p) = b_{ij} < B$, which contradicts the assumption that $width(p) \geq B$. □

Theorem 4 implies that any links with a bandwidth less than W are not parts of the path we want. Hence we may find the path in two steps. First, we eliminate any links with a bandwidth less than B so that any paths in the resulting graph satisfy $width(p) \geq B$. Then, we can simply try to find a path that satisfies $length(p) \leq D$. To do that, we can try to find the path with minimum length. In a single search, we can then determine whether such path exists, and find one if it does exist. Suppose that path p^* is the path with minimum length $D_{\min}$. If $D_{\min} \leq D$, then path p^* is a path that satisfies the two constraints. Otherwise, we can conclude that no such a path exists as all other paths have a length no less than $D_{\min}$. Suppose that node 1 is the source node and node m is the destination. The following algorithm finds a path between node 1 and m that has a bandwidth no less than B and a delay no more than D, if such a path exists. Let D_i be the estimated length of the bandwidth-delay-constrained path from node 1 to node i.

Step 1: Set $d_{ij} = \infty$, if $b_{ij} < B$.

Step 2: Set $L = \{1\}$, $D_i = b_{1i}$ for all $i \neq 1$

Step 3: Find $k \notin L$ so that $D_k = \min_{i \notin L} D_i$.

If $D_k > D$, no such a path can be found and the algorithm terminates.

If L contains node m, a path is found and the algorithm terminates.

$L := L \bigcup \{k\}$.

Step 4: For all $i \notin L$, set $D_i := min[D_i, D_k + d_{ki}]$

Step 5: Go to Step 3.

Step 1 eliminates all links that do not meet the bandwidth requirement by setting their delay to ∞. Step 2-5 find the minimum delay path to node m using Dijkstra's algorithm. Note that we do not have to find the minimum delay paths to all nodes. The algorithm can be terminated either when node m is permanently labled or the delay exceeds the threshold before reaching node m.

Each step in the above algorithm requires a number of operations proportional to N, and the steps are, in the worst case, iterated $N-1$ times. Thus, the computation in the worst case, is $O(N^2)$, which is the same as the Dijkstra's algorithm.

3.3. Hop-by-Hop Routing Algorithms

We now present two distributed algorithms suitable for hop-by-hop routing.

Since hop-by-hop routing pre-computes forwarding entries for every destination, it has to accommodate all possible resource requirements. The usual approach in current hop-by-hop routing algorithms is to compute the best path to every destination. With a single metric, the best path can be defined easily. For example, if delay is the metric, the best path is the one with optimal delay (i.e. shortest delay). With multiple metrics, however, the best path with all parameters at their optimal values may not exist at all. For example, a path with both maximum bandwidth and minimum delay may not necessarily exist. Thus, we must decide the precedence among the metrics in order to define the best path.

The precedence of bottleneck bandwidth and propagation delay is somehow application-dependent. But general speaking, queuing delay is more dynamic and traffic-sensitive, thus bandwidth is often more critical for most multimedia applications. If there is no sufficient bandwidth, queuing delay, and probably the loss rate as well, will be very high. In contrast, if the propagation delay cannot be met, the overall delay will be higher but the increase will be predictable and stable. Thus, although failing to meet either of the two constraints will result in higher overall delay, the lack of bandwidth may have more severe consequences. In this paper, we define the precedence as bottleneck bandwidth and then propagation delay. Our search strategy is to find a path with maximum bottleneck bandwidth (a *widest path*), and when there are more than one widest path, we choose the one with shortest propagation delay. We refer to such a path as the *shortest-widest* path.

An important property of widest paths is that they are decided by bottleneck links; non-bottleneck links have no effects on widest paths. Therefore, for a given topology, there are usually many widest paths with equal width, and loops can be formed as a result. Note that if a loop is not a bottleneck, a path with the loop and a path without the loop has the same width, hence the loop can not be easily detected. With a centralized algorithm, an ordered scanning of the nodes should avoid such loops. In a distributed algorithm, however, loops can occur. However, we can show that one of the widest paths, the shortest-widest path, is always free of loops. Intutively, the delay meteric eliminates the loops.

Theorem 5: *Shortest-widest paths are loop-free in a distributed computation.*

Proof: By contradiction. Suppose that node A and node B are involved in a loop for destination C (Figure 5). Path $p_1 p_2$ is the shortest-widest path from node A to node C and path $p_1^* p_2^*$ is the shortest-widest path from node B to node C.

By the definition of shortest-widest paths, we have

$$width(p_2^*) \leq width(p_1 p_2) \tag{8}$$

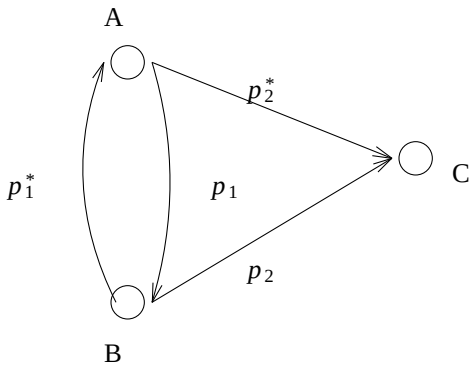

Figure 2: A Loop Involving Node A and Node B

$$width(p_2) \leq width(p_1^* p_2^*) \tag{9}$$

Note that

$$width(p_1^* p_2^*) = min[width(p_1^*),\ width(p_2^*)] \leq width(p_2^*) \tag{10}$$

Similarly,

$$width(p_1 p_2) \leq width(p_2) \tag{11}$$

From (8), (10) and (11), we have

$$width(p_1^* p_2^*) \leq width(p_2) \tag{12}$$

Comparing (12) with (9), we have

$$width(p_1^* p_2^*) = width(p_2) \tag{13}$$

Similarly, we have

$$width(p_1 p_2) = width(p_2^*) \tag{14}$$

Equation (13) shows that path $p_1^* p_2^*$ and path p_2 are equal widest paths. Since path $p_1^* p_2^*$ is the shortest-widest path, we have

$$length(p_2) \geq length(p_1^* p_2^*) > length(p_2^*) \tag{15}$$

Similarly, Equation (14) shows that path $p_1 p_2$ and path p_2^* are equal widest paths. Since path $p_1 p_2$ is the shortest-widest path, we have

$$length(p_2^*) \geq length(p_1 p_2) > length(p_2) \tag{16}$$

Equation (15) and (16) contradict each other. This completes the proof. □

Note that Theorem 5 is also a property of the shortest-widest path itself and is independent of any particular algorithms for finding such paths.

Suppose that node 1 is the source node and h is the number of arcs away from the source node. Let $B_i^{(h)}$ and $D_i^{(h)}$ be the width and length of the chosen shortest-widest path from node 1 to node i within h hops. By convention, $B_1^{(h)} = \infty$ and $D_1^{(h)} = 0$ for all h. The shortest-widest path algorithms can be produced by adding the length checking when there are multiple equal widest paths. The shortest-widest path algorithm based on distance-vectors is as follows:

Step 1: Initially, $h = 0$ and $B_i^{(0)} = 0$, for all $i \neq 1$

Step 2: Find set K so that $width(1,...,K,i) = \max_{1 \le j \le N}[min[B_j^{(h)}, b_{ji}]]$ $i \neq 1$

Step 3: If K has more than one element, find $k \in K$ so that $length(1,...,k,i) = \min_{1 \le j \le N}[D_j^{(h)} + d_{ji}]$, ~$i \neq 1$

Step 4: $B_i^{(h+1)} = width(1,...,k,i)$ and $D_i^{(h+1)} = length(1,...,k,i)$

Step 5: If $h \ge A$, the algorithm is complete. Otherwise, $h = h + 1$ and go to Step 2.

Step 2 finds all widest path from node 1 to each node i. If there are more than one widest path found, Step 3 chooses the one with minimum length. Step 4 updates the width and length for the shortest-widest path from node 1 to each node i.

Suppose that node 1 is the source node Let B_i and D_i be the width and length of the chosen shortest-widest path from node 1 to node i. By convention, $B_1 = \infty$ and $D_1 = 0$. The shortest-widest path algorithm based on link-states is as follows:

Step 1: Initially, $L = \{1\}$, $B_i = b_{1i}$ and $D_i = d_{1i}$ for all $i \neq 1$

Step 2: Find set $K \notin L$ so that $B_K = \max_{i \notin L} B_i$.

Step 3: If K has more than one element, find $k \in K$ so that $length(1,...,k,i) = \min_{j \in K}[D_{(1,...,j,i)}]$.

$L := L \cup \{k\}$. If L contains all nodes, the algorithm is completed

Step 4: For all $i \notin L$, set $B_i := max[B_i, min[B_k, b_{ki}]]$

Step 5: Go to Step 2.

Step 2 finds the nodes with maximum width among the tentatively labelled nodes. If there are more than one node found, Step 3 chooses one with minimum length and permanently labels it. Step 4 updates the tentatively labelled nodes around the newly permanently labelled node.

Under some circumstances, such as the case where all links have the same amount of bandwidth, shortest-widest path algorithms are effectively reduced to shortest path algorithms. In this sense, we can view shortest path algorithms as a special case of shortest-widest path algorithms. So we can also use shortest-widest algorithms to compute shortest paths by simply setting the bandwidth of all links to the same amount. In this case, the constraint on the bandwidth requirement has no effects.

The two shortest-widest path algorithms are scalable. Note that, in the two versions of shortest-widest algorithms, the number of operation required in each iteration is proportional to that in the corresponding versions of shortest path algorithms. Therefore, the time complexity of the two shortest-widest algorithms is equal to that of the shortest path algorithms.

4. Future Work

In this paper, we examined a number of important issues in QoS routing and presented three path computation algorithms. There are a number of areas for future research:

- QoS routing is an integrated part of a resource management system. We will look into ways of integrating our algorithms with other components in resource management architectures such as admission control and resource setup.
- Although the research was done in the context of datagram networks such as the Internet, many of the results and algorithms are general, and can be readily applied to connection-oriented networks such as ATM networks. We will examine some issues in this area.
- We will study the convergence speed of our algorithms after link or node failures, and work out a revised algorithm based on the diffusing computation approach suggested by Garcia-Luna-Aceves [7].
- We will investigate approximation algorithms for metrics with NP-complete search problems, and carry out simulation experiments to evaluate their performance.

5. Acknowledgement

We would like to thank Dr David S. Johnson (AT&T Bell Labs) invaluable help with the problem of NP-completeness, and anonymous reviewers for their useful comments.

6. References

[1] D. Clark and V. Jacobson. "Flexible and Efficient Resource management for Datagram Networks", unpublished paper, 1991.

[2] D. Clark, S. Shenker, and L. Zhang. "Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism", in Proceedings of SIGCOMM'92, pp 14-26, 1992.

[3] A. Demers, S. Keshav, and S. Shenker. "Analysis and Simulation of a Fair Queueing Algorithm", In Journal of Internetworking: Research and Experience, 1, pp. 3-26, 1990.

[4] D. Estrin, Y. Rekhter, S. Hotz, "Scalable Inter-Domain Routing Architecture", In Proceedings of ACM SIGCOMM'92, Maryland, August 1992.

[5] D. Ferrari. "Distributed Delay Jitter Control in Packet-Switching Internetworks", In Journal of Internetworking: Research and Experience, 4, pp. 1-20, 1993.

[6] S. Floyd. "Link-sharing and Resource Management Models for Packet Networks", unpublished paper, 1993.

[7] J. Garcia-Luna-Aceves, "A Unified Approach to Loop-Free routing Using Distance Vectors or Link States", in Proc. Sigcomm'89, Texas, USA, Sept 1989.

[8] M. R. Garey, D. S. Johnson, "Computers and Intractability - A Guide to the Theory of NP-Completeness", Freeman, California, USA, 1979.

[9] S. J. Golestani. "Duration-Limited Statistical Multiplexing of Delay Sensitive Traffic in Packet Networks", In Proceedings of INFOCOM '91, 1991.

[10] R. Guerin and L. Gun. "A Unified Approach to Bandwidth Allocation and Access Control in Fast Packet-Switched Networks", In Proceedings of INFOCOM'92, 1992.

[11] Jeffrey Jaffe, "Algorithms for Finding Paths with Multiple Constraints", Networks, Vol 14, pp 95-116, 1984.

[12] J. Kurose. "Open Issues and Challenges in Providing Quality of Service Guarantees in High-Speed Networks", In Computer Communication Review, 23(1), pp 6-15, 1993.

[13] J. Hyman, A. Lazar, and G. Pacifici. "Real-Time Scheduling with Quality of Service Constraints", In IEEE JSAC, Vol. 9, No. 9, pp 1052-1063, September 1991.

[14] C. Kalmanek, H. Kanakia, and S. Keshav. "Rate Controlled Servers for Very High-Speed Networks", In Proceedings of GlobeCom'90, pp 300.3.1-300.3.9, 1990.

[15] C. Partridge, I. Castineyra, "Routing Flows in an internet", unpublished paper, April 1993.

[16] A. Parekh. "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks", PhD Thesis, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992.

[17] S. Shenker, D. Clark, L. Zhang, "A Service Model for an Integrated Services Internet", unpublished paper, October 1993.

[18] D. Yates, J. Kurose, D. Towsley, and M. Hluchyj. "On Per-Session End-to-End Delay Distribution and the Call Admission Problem for Real Time Applications with QOS Requirements", In Proceedings of SIGCOMM '93, 1993.

[19] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol", IEEE Network, Sept, 1993.