-

Bandwidth-Delay Based Routing Algorithms

Zheng Wang and Jon Crowcroft

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

Abstract Multimedia applications often require guaranteed quality of service and resource reservation, which has raised a number of challenging technical issues for routing. In this paper, we consider two new routing algorithms based on bandwidth and delay metrics. The implications of routing metrics on path computation are examined and the rationales behind the selection of bandwidth and delay metrics are discussed. Two new routing algorithms based on bandwidth and delay metrics are presented and some of their important properties are investigated.

1. Introduction

In today's computer 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. It is becoming increasingly clear that such routing protocols are inadequate for multimedia applications, such as video conferencing, which often require guaranteed quality of service (QoS). For a network to support QoS requirements, routing must supply explicit information on resources available in the network so that applications can make proper resource reservation. To achieve that, it is necessary for routing to have a more complex model of the network, taking into account important network parameters such as bandwidth, delay and loss probability. This, however, has raised a number of challenging technical issues for routing. In particular, there are two major problems which have to be resolved. First, multiple routing metrics have important implications on the complexity of path computation. Since the problem of finding a path subject to multiple constraints is difficult, it is essential that we find out that which metrics are tractable. Second, new algorithms have to be found that are able to compute paths that satisfy multiple constraints.

In this paper, we consider new routing algorithms for supporting multimedia applications. We first discuss the basic design issues, and then present two bandwidth-delay based algorithms and discuss their properties in detail.

2. Design Issues

The selection of routing metrics is one of the key design issues. Routing metrics determine the criterion by which paths are selected, therefore they have important implications on the complexity of path computation. In this section, we examine various approaches in metric selection, and discuss the rationales behind our design.

2.1. Single-Metric Approach

Single-metric such as delay or hop-count has been widely used in current networks, and path computation algorithms for such metrics are well known. Therefore, the first issue we need to work out is whether a single metric can be found to support multimedia applications.

Using a single primitive parameter such as delay is clearly not sufficient for multimedia applications. We need information on other resources particularly bandwidth for supporting resource reservation. One possible approach might be to define a function and generate a single metric from multiple primitive 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 may be produced with bandwidth B, delay D and loss probability L using 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.

However, a mixed metric only reflects the status of the network at a particular time instance; it does not provide information for making quantitative assessment as to whether application's requirements can be met or not. Take delay for example. Many current routing protocols use measured delay as the metric. However, measure delay is in fact a mixed metric since it consists of queuing delay and propagation delay, both caused by very different factors. Given a value of measured delay, it is not possible to assess whether a path is busy or not, without the knowledge of the queuing delay component or propagation delay component.

Another major problem is that mixing network parameters may invalidate the composition rule. For example, suppose that a path p 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.

Therefore, although the mixed metric approach is a tempting heuristic, it can at best used as an indicator in path selection but is not sufficient for supporting multimedia applications.

2.2. Multiple-Metric Approach

Multiple metrics (e.g., delay, bandwidth and loss probability) can certainly model the characteristics of a network more accurately. However, it may not be feasible to have many parameters as metrics since the problem of finding a path subject to multiple constraints is inherently difficult. A problem with two simple additive constraints called "shortest weight-constrained path" was listed in [2] as NP-complete although the proof has never been published. Jaffe [1] 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 problem is much more complicated in routing as the resource requirements specified by the applications are often diverse and application-dependent. The following set of parameters, though by no means exhaustive, are among the most important and natural ones that may be chosen as metrics: delay, delay jitter, cost, loss probability and bandwidth. Among them, delay, delay jitter and cost are additive while loss probability and bandwidth are not.

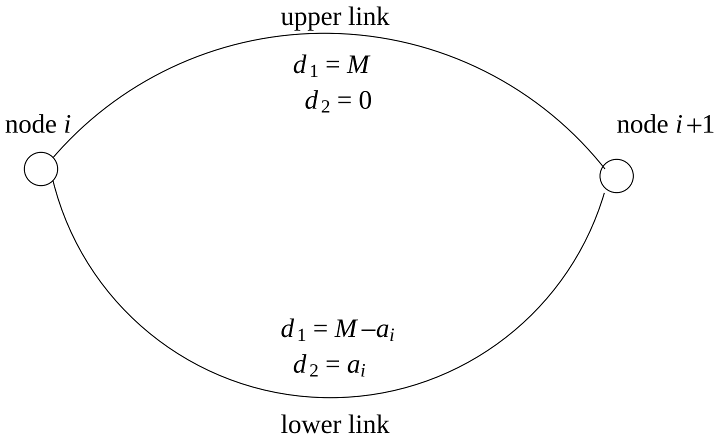

Figure 1: Assignment of Delay and Cost to Two Links Between Node i and $i+1$

Theorem 1: *The problem of finding a path subject to any two of the following metrics: delay, loss probability, cost and jitter, is NP-complete.*

Proof: We first prove that the problem of DELAY AND COST, i.e. finding a path with total delay no more than D and total cost no more than C is NP-complete.

DELAY AND COST is apparently in NP. Since PARTITION is a classic NP-complete problem [2], we now show PARTITION $\propto$ DELAY AND COST to prove 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 $D^*(i, i+1)$ for the two link from node i to node $i+1$ be M and $M - a_i$ respectively, and let metric $C^*(i, i+1)$ be 0 and a_i respectively $(0 \le a\_i \le 0)$.

Consider an instance of DELAY AND COST,

$$D^*(p) \le nM - S/2 \tag{1}$$

$$C^*(p) \le S/2 \tag{2}$$

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

$$D^*(i, i+1) + C^*(i, i+1) = M$$

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

$$D^*(p) + C^*(p) = \sum_{i=1}^{i=n} (D^*(i, i+1) + C^*(i, i+1)) = nM \tag{3}$$

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

$$C^*(p) \ge S/2 \tag{4}$$

From (2) we also have

$$C^*(p) \le S/2$$

Therefore, we get

$$C^*(p) = S/2$$

From (3), we also get

$$D^*(p) = S/2$$

Note that, for the two link from node i to node $i+1$, $C^*(i, i+1)$ be 0 and a_i. Therefore, there must be a subset 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

$$C^*(p) = S/2$$

Since

$$D^*(p) + C^*(p) = nM$$

We also get

$$D^*(p) = nM - S/2$$

This solves the instance of DELAY AND COST.

Note that delay, cost and jitter share the rule for addition is: $f(ab + bc) = f(ab) + f(bc)$, where ab and bc are two adjacent links. Therefore, DELAY AND JITTER and to COST AND JITTER can be proved in a similar way.

Suppose that the loss probability for a link is p, the corresponding success probability as $1-p$. The composition rule for success probability is $f(ab+bc) = f(ab) \times f(bc)$. However, if we define $l(x) = log(f(x))$, we have $l(ab+bc) = l(ab) + l(bc)$. Thus, DELAY AND LOSS PROBABILITY, JITTER AND LOSS PROBABILITY, and COST AND LOSS PROBABILITY can be transformed from DELAY and COST. □

Theorem 1 shows that the problem of finding a path that satisfies two of the following metrics: delay, loss probability, cost and jitter is NP-complete. The possible combination left is bandwidth and one of the other four (delay, loss probability, cost and jitter). Among them, delay is certainly the most important one. As we will show in the next section that efficient algorithms exist for finding a path subject to the constraints on bandwidth and delay metrics. Therefore, as far as complexity is concerned, bandwidth and delay are natural candidates for routing metrics.

The bandwidth of a path is defined as the minimum of the residual bandwidth of all links on the path or the *bottleneck bandwidth*. There are two basic components in delay: propagation delay and queuing delay. Note that the queuing delay is closely related to the bottleneck bandwidth and traffic characteristics. Since queuing delay is already reflected in the bottleneck bandwidth, we only use propagation delay as the delay metric since any inter-dependence among metrics can complicate path computation. Thus, we propose bottleneck bandwidth and propagation delay as the metrics.

We believe that bottleneck bandwidth and propagation delay reflect the most fundamental characteristics of a path in the network, and they provide the basic information for an initial assessment as to whether an application's requirements can be met or not. We can define bottleneck bandwidth and propagation delay of a path as the *width* and the *length* of the path. The problem of routing is then to find a path in the network subject to the constraints on its width and length.

3. Bandwidth-Delay Routing Algorithms

There are two basic types of routing architectures: source routing and hop-by-hop routing [4]. In source routing, a forwarding path is computed on-demand at the source and is specified in the packet header. In hop-by-hop routing, a forwarding path is pre-computed in a distributed fashion across the network. In this section, we present two bandwidth-delay based path computation algorithms, a centralized one for source routing and a distributed one for hop-by-hop routing. and discuss some of the important properties.

3.1. A Centralized Routing Algorithm

We now present a centralized algorithm that can be used for source routing, where the path is computed on-demand at the source node centrally, and is specified in the path. Given the bandwidth and delay constraints for a flow, the algorithm can find 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 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 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 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 B and D, the 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.

Note that 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, which implies that any links with a bandwidth less than B are not parts of the path we want. Hence we can 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 a path exists, and find one if it does exist. Suppose that path p^* is the path with minimum length $B_{\min}$. If $B_{\min} \leq B$, 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 $B_{\min}$.

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

The algorithm is free of loop since the paths selected are shortest paths of the reduced graph. 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.

3.2. A Distributed Algorithm

In this section, we present a distributed algorithm 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. Our approach is similar to that used in existing routing protocols, i.e. computing the best path to every destination. With a single metric, the best path can be easily defined. 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. For example, a path with both maximum bandwidth and minimum delay may not necessarily exist. Thus, we must define a precedence between bottleneck bandwidth and propagation delay.

The precedence between bottleneck bandwidth and propagation delay is somehow application-dependent. Queuing delay is more dynamic, thus bandwidth is probably often considered as more important. If the bandwidth requirement can not be met, the chance is that queuing delay, and probably the loss rate, will also be high. On the other hand, if the propagation delay cannot be met, the overall delay will be higher but the increase will be predictable and stable. Thus, failing to meet either of the two constraints will result in higher overall delay but the increase of queuing delay can have more severe consequences than that of propagation delay. Thus, we choose bottleneck bandwidth over propagation delay.

We first consider the widest path problem, i.e. finding a path between two nodes with maximum bandwidth.

Given any two nodes i and m of the graph, the widest path problem is to find a path from node i to node m with maximum width. Note that the widest path problem is, in many respects, similar to the shortest path problem; the difference lies in the calculation of the width and the length of a path. Therefore, it appears that the widest path problem can be solved with similar technique used in shortest path algorithms. We now consider a version of the widest path first algorithm based on the distributed Bellman-Ford algorithm using distance-vectors.

Suppose that node 1 is the source node and h is the number of arcs away from the source node. Let $B_i^{(h)}$ be the width of the widest path from node 1 to node i within h hops. By convention, $B_1^{(h)} = \infty$. The widest path algorithm is as follows:

```
Step 1:   Initially, $h = 0$ and $B_i^{(0)} = 0$, for all $i \neq 1$
Step 2:   Find $k$ so that $width(1,...,k,i) = \max_{1 \le j \le N}[min[B_j^{(h)}, b_{ji}]], i \neq 1$
Step 3:   $B_i^{(h+1)} = width(1,...,k,i)$
Step 4:   If $h \geq A$, the algorithm is complete. Otherwise, $h = h + 1$ and go to Step 2.
```

The algorithm above can find the widest paths from a given node to all other nodes, and the time complexity is equal to that of the corresponding distributed Bellman-Ford algorithm.

In a centralized computation, the paths produced by the algorithms described above are loop-free widest paths. However, in a distributed computation, loops may be formed. Note that the loops we refer to here are not the temporary ones that only exist during the convergence period [3]. The loops are permanent, formed when there are multiple widest paths with equal width. In a centralized computation, loops avoided with a incremental hop-based scanning of the graph. In a distributed computation, however, when there are paths with equal width, a node may not necessarily select the same path to a destination as its next hop node chooses. An extreme example is a network with all links of same bandwidth. In such a case, any possible path is a widest path as they all have the same width.

Note that shortest path is always free of loops, even in a distributed computation.

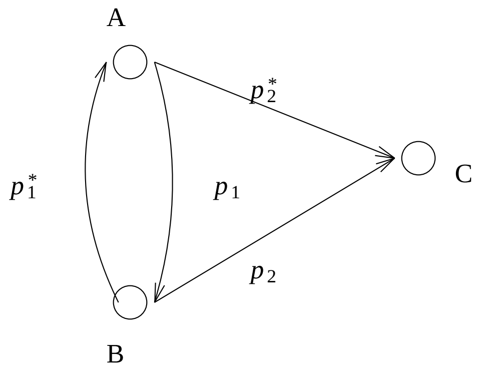

Figure 2: A Loop Involving Node A and Node B

Theorem 2: *Shortest 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 2). Path $p_1 p_2$ is the shortest path from node A to node C and path $p_1^* p_2^*$ is the shortest path from node B to node C. There are two possibilities: 1) $p_2^* = p_1 p_2$, and 2) $p_2^* \neq p_1 p_2$.

If $p_2^* = p_1 p_2$, i.e. path $p_1 p_2$ is used by node B as part of the shortest path to node C, we have

$$length(p_1^* p_2^*) < length(p_2^*) = length(p_1 p_2) < length(p_2)$$

This contradicts the assumption that path $p_1^* p_2^*$ is the shortest path from node B to node C.

If $p_2^* \neq p_1 p_2$, we have

$$length(p_2) < length(p_1 p_2) \leq length(p_2^*) < length(p_1^* p_2^*)$$

This also contradicts the assumption that path $p_1^* p_2^*$ is the shortest path from node B to node C. The proof is completed. □

Note that Theorem 2 is a property of shortest paths, and is independent of the shortest path algorithms. Intuitively, shortest paths have a property that any part of a shortest path is also a shortest path itself. However, such a property does not hold for widest paths. The width of a path is decided by the link. If a loop is not the , it has no impacts on the width of the path. So, a path with a loop and a path without a loop have the same width.

Theorem 2 also leads us to the solution. To eliminate the looping problem, we use delay as the second criteria in the path selection. The path selected are the shortest path among the widest paths, i.e., *shortest-widest path*. The delay criteria also eliminates any loops in widest paths. Shortest-widest paths have the following loop-free property (see [5] for a complete proof).

Theorem 3:*Shortest-widest paths are always loop-free in a distributed computation.*

Note that Theorem 3 is also a property of the shortest-widest paths and is independent of the algorithm used for produced the path. Therefore, we can use shortest-widest paths for hop-by-hop routing.

Let us now look at the algorithms for finding shortest-widest 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 $width(1,...,K,i) = \max_{1 \leq j \leq N}[min[B_j^{(h)}, b_{ji}]],\ i \neq 1$

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

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

Step 6: If $h \geq A$, 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.

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 is simply ignored.

The shortest-widest path algorithm is 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 considered new routing algorithms for supporting multimedia applications. We proposed two new algorithms based on bandwidth and delay metrics. A major area for future research is the behavior of the algorithms during the convergence period. We will investigate if we can use the techniques proposed in [3] can be applied to speed up convergence and eliminate temporary loops may exist during the convergence period.

5. Acknowledgement

We would like to thank Dr David S. Johnson (AT&T Bell Labs) for his excellent book and invaluable help with the proof of NP-completeness.

6. References

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

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

[3] 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.

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

[5] Zheng Wang, Jon Crowcroft, "Quality of Service Routing for Supporting Multimedia Applications", Submitted for publication.