QoS Based Routing Algorithm in Integrated Services Packet Networks

Chotipat Pornavalai[†] Goutam Chakraborty[‡] Norio Shiratori[†]

[†]Research Institute of Electrical Communication
Tohoku University
Sendai 980-77, Japan
{chotipat,norio}@shiratori.riec.tohoku.ac.jp

[‡]Multimedia Systems Laboratory
The University of Aizu
Fukushima 965-80, Japan
goutam@u-aizu.ac.jp

Abstract

In this paper we study QoS based routing algorithm for supporting resource reservation in high-speed Integrated Services Packet Network (ISPN). Recently, this problem was proved to be NP-complete. However, when the considered QoS constraints are bandwidth, delay, delay jitter, and loss free, we have shown that by employing Weighted Fair Queueing (WFQ) service discipline, the complexity of the problem could be reduced to that of shortest path routing without any QoS constraints. Then such a multiple QoS constrained route could be searched in polynomial time. We also present that routing algorithm (called "$QoSR_{BF}$"), which is a modified version of Bellman-Ford shortest path algorithm. Simulation results show that $QoSR_{BF}$ has better performance compared to the existing QoS routing algorithms.

1 Introduction

Traditional data networks were designed to provide best effort service. This service is suitable only for data communication applications such as ftp, telnet etc. With the recent developments in transmission and computing technologies, distributed multimedia applications, such as video conferencing, video on demand have now become possible and will be widely used in near future. Such applications transmit information which usually has different traffic characteristics and various QoS performance requirements, similar to the case of real-time communication. Meeting all different application requirements by employing a single network, Integrated Services Packet Network (ISPN), would gain the vast economies of scale, ubiquity of access and improved statistical multiplexing [1].

Future packet-switching integrated services networks, including Internet, will be able to support both best effort service and real-time communication or guaranteed service classes [2, 3]. To support guaranteed service, many technical problems are to be solved. In last few years, several new service disciplines or scheduling algorithms, admission controls and resource reservation protocols have been proposed [4, 5, 6]. QoS based routing algorithms and protocols, which can support resource reservation for guaranteed service are needed. In spite of its importance, until recently little interest has been shown by researchers. Without an efficient QoS based routing algorithm network may fail to find a route and reject a request for the call connection, though there may be enough resources to successfully establish that call.

Usually a connection oriented route is preferred because of its simplicity for reserving resources. The route will be fixed before starting of the call and will be continued till the end of the session. Thus, while finding the route, not only all of the required QoS constraints must be satisfied, but also it is important to allocate the resources effectively. QoS routing problem has recently been proved to be NP-complete [7]. Routing algorithms which are currently employed in the networks are either Dijkstra or Bellman-Ford shortest path (SP) algorithms. Routing metric for SP problem is only a single QoS parameter, usually the number of hops or the average path delay. Though Dijkstra or Bellman-Ford algorithms are optimal and run in polynomial time, they could not find the route that satisfy multiple QoS constraints such as bandwidth, delay, delay jitter and loss probability, which are usually required by the multimedia applications. Most of the proposed QoS based routing algorithms either simplify the problem by choosing the route based on only the most stringent QoS parameter, or are too complex to be used in practice.

In this paper, we indicate that conditions imposed by service discipline has great effect on reducing the complexity of QoS based routing algorithm. Thus it is important to consider service discipline while designing the routing algorithm. We have shown that, when the considered QoS constraints are bandwidth, delay, delay jitter, loss free, by employing Weighted Fair Queueing (WFQ) service discipline [5], we can reduce the complexity to a simple SP routing problem without any QoS constraints. This can be solved in polynomial time. We also present that routing algorithm, we named $QoSR_{BF}$, which is a modified version of Bellman-Ford (BF) algorithm. Its worst case computation time is the same as the original BF algorithm, of the order of $O(|E| \cdot H_{max})$, where $|E|$ is

0-8186-8061-X/97 $10.00 © 1997 IEEE

the number of links in the network, and H_{max} is the maximum number of iterations of the BF algorithm, an upper-bound on the number of hops in the QoS constrained route. Extensive simulations results show that $QoSR_{BF}$ can successfully find the minimum-hop route that can satisfy the required QoS constraints, if there exists one. At the same time, it could effectively distribute the traffic throughout the network.

The rest of this paper is organized as follows. In Sect. 2, we define the QoS based routing problem. In Sect. 3, we present an overview of WFQ service discipline. How the complexity of the problem is reduced, the description of $QoSR_{BF}$, correctness and complexity analysis are presented in Sect. 4. Comparison with related works are in Sect. 5. In Sect. 6, detail of simulation setup and results are reported and discussed. Concluding remarks are given in Sect. 7.

2 QoS Based Routing Problem

We simply assume that a communication network can be modeled as a direct connected graph $G = (V, E)$. Here V represents the set of nodes, which could be routers, servers or switches, and E represents the set of edges or links of the network. $h(e) = h(u,v) = 1$, $c(e) = c(u,v)$, $d(e) = d(u,v)$, $b(e) = b(u,v)$, $j(e) = j(u,v)$, $l(e) = l(u,v)$ are the hop, cost, delay, available bandwidth, jitter and loss functions of link e, connecting directly the node u to node v. The link cost function, Eq. (1), is a measure of link's load.

$$c(e) = \frac{1}{b(e)} \tag{1}$$

A source node s and a destination node z, $(s, z \in V)$, are given. The problem is to find a route or path[1] from source node s to destination node z, $P(s,z) = (s, j, k, ..., l, z)$, such that it (i) would consume minimum resources by using minimum hop route (Eq.(2)), (ii) would distribute the traffic or balance the load throughout the network by choosing least cost (least load) route (Eq. (3)), and (iii) would satisfy all of the required QoS requirements (Eq. (4), (5), (6), (7)).

$$H(P) = \underset{P_i \in P'(s,z)}{MIN} \sum_{e \in P} h(e) \tag{2}$$

$$C(P) = \underset{P_i \in P'(s,z)}{MIN} \sum_{e \in P} c(e) \tag{3}$$

$$B(P) = \underset{e \in P}{MIN} [b(e)] \geq \Delta_{bandwidth} \tag{4}$$

$$D(P) = \sum_{e \in P} d(e) \leq \Delta_{delay} \tag{5}$$

$$J(P) = \sum_{e \in P} j(e) \leq \Delta_{jitter} \tag{6}$$

$$L'(P) = \prod_{e \in P} l'(e) \geq (1 - \Delta_{loss}) \tag{7}$$

[1] route and path are used interchangeably.

Here, $H(P)$, $C(P)$, $B(P)$, $D(P)$, $J(P)$ and $L(P)$ are the number of hops, cost, bottleneck bandwidth, delay, jitter, and loss on the path $P(s,z)$ respectively. $P'(s,z)$ is the set of all possible paths from s to z that satisfy the multiple QoS constraints defined in equations (4), (5), (6), (7). The QoS constraints are symbolized by Δ with respective suffixes. The loss probability is symbolized as $l(e)$, and therefore the prob. of successful transmission is $l'(e) = 1 - l(e)$. For the whole path this probability is $L'(P) = 1 - L(P)$. Eq. (2), and Eq. (3) are the optimization functions of QoS based routing problem. These two functions may sometimes be contradictory as minimum hop route will not necessarily minimize the cost. In general, minimum-hop route should be the primary criteria of route selection. If there is more than one minimum-hop QoS satisfying routes, least cost one is selected.

In this paper, we consider the guaranteed service class of the integrated service model in the Internet [2, 3] which is the deterministic guaranteed service. The loss probability would be zero, i.e. we need to ensure that no queueing loss (loss free constraint) occurs. Thus we redefine loss probability constraint, Eq. (7), to be buffer space constraint, Eq. (8), shown below. In other words, every node or switch along the selected route must have enough buffer space, so that no packet queueing loss may occur at that node.

$$\forall e = (u,v) \in P, \quad f(e) \geq \Delta^e_{buffer}, \tag{8}$$

where $f(e) = f(u,v)$. $f(e)$ and Δ^e_{buffer} are the amount of available buffer and required buffer space for no queueing loss at node node v for the incoming link e from node u respectively.

Bandwidth constraint, Eq. (4), is a concave constraint or *link constraint*. Delay constraint, Eq. (5), and jitter constraint, Eq. (6), are additive constraints. Finally, loss probability constraint, Eq. (7), is a multiplicative constraint. Delay, Jitter and loss probability are also called *path constraints*. Wang and Crowcroft [7] presented a proof that searching a route that satisfy n additive and k multiplicative constraints where $(n + k \geq 2)$, is NP-complete problem.

3 Service Disciplines for QoS Based Routing Algorithm

In this section, we briefly describe an overview of WFQ service disciplines which influences the design of our QoS based routing algorithm. For finding a route that satisfy multiple QoS constraints, as already mentioned, is a NP-complete problem [7]. However, the proof has been done without any assumptions of the dependency of routing on service discipline. For any guaranteed communication, some service discipline has to be employed, and that imposes some QoS bound expressions. Thus one important design issue for QoS based routing algorithm is to exploit those expressions derived from the service discipline.

The guaranteed service class of the Integrated Service model in the Internet [2, 3] is based on the Weighted Fair Queueing (WFQ) [5]. It can be used

end-to-end delay bound	$\frac{\sigma_q+HS^q_{max}}{r_q} + \sum_{i=1}^{H}\{\frac{S_{max}}{R_i(e)} + pd_i(e)\}$
end-to-end jitter bound	$\frac{\sigma_q+HS^q_{max}}{r_q}$
buffer space at n^{th} switch	$\sigma_q + nS^q_{max}$

Table 1: End-to-end delay bound, delay-jitter bound, and buffer space requirement of Weighted Fair Queueing (WFQ) service discipline.

to provide a tight upper bound on the end-to-end delay, end-to-end jitter, and ensure that no packet is lost due to non availability of buffer at each switch, assuming no failure of network components. Table 1 shows the expressions to calculate such bounds and buffer space requirement (for no queueing loss) of a flow q, for which the traffic characteristic is in the form $(\sigma_q, \rho_q, S^q_{max})$ [8]. Here σ_q is the token bucket size or maximum burst size, ρ_q is the token generation rate of the leaky bucket, S^q_{max} is the maximum packet size of the requested flow q respectively. r_q is the guaranteed rate for the connection or the amount of bandwidth to be reserved for the requested flow q ($r_q \geq \rho_q$). S_{max} is the maximum packet size allowed in the network. $R_i(e)$ and $pd_i(e)$ are the total bandwidth (link capacity) and propagation delay of link e which is the i^{th} hop on the route traversed by the connection. H is the number of hops of the route.

4 QoS Based Routing Algorithm ($QoSR_{BF}$)

First we explain how to reduce the complexity of the problem by using QoS bound expressions for WFQ service discipline. Then we present informal description, correctness and complexity analysis of $QoSR_{BF}$ algorithm. The pseudocode of the algorithm is available in the Appendix.

4.1 Complexity Reduction

QoS routing problem has been proved to be NP-complete [7]. However, when there are 2 QoS constraints which are either additive or multiplicative, and if the value of at least one of them is the same on every link, we can find QoS constrained route by Bellman-Ford algorithm in polynomial time. Bellman-Ford (BF) is a breadth first search algorithm which proceeds searching the shortest path route by increasing the number of hop-count, which can be considered as one of the additive constraint where every link has the value of 1. The idea of the complexity reduction stems from the fact that it is possible to map (1) *path constraint* to *link constraint*, and (2) express QoS constraints in term of number of hops, i.e. maximum allowed number of hops in the route. We can then map the different constraints to delay constraint and number of hops, and thus reduce the complexity of the problem to that of shortest path routing. By employing WFQ as the service discipline, we can do the complexity reductions (by using expressions in Table 1), when QoS constraints are bandwidth, delay, jitter, and loss free. The complexity reduction procedure is as follows:

Delay constraint: Link delay is composed of queueing, transmission, and propagation delay. Queueing delay of the path is the function of the number of hops H. But transmission and propagation delay may be different in different links. Thus we can not truly reduce it in terms of the number of hops. However, shortest delay can be used as the optimization objective of Bellman-Ford algorithm. From the end-to-end delay bound expression in Table 1, the link delay $d(e)$ of link $e = (u, v)$, is defined by the following equation.

$$d(e) = \begin{cases} \frac{\sigma_k}{r_k} + \frac{S^q_{max}}{r_k} + \frac{S_{max}}{R(e)} + pd(e) & \text{if } u == s \\ \frac{S^q_{max}}{r_k} + \frac{S_{max}}{R(e)} + pd(e) & \text{otherwise} \end{cases} \tag{9}$$

Bandwidth constraint: Because bandwidth is a link constraint, we can simply eliminate all links in the network which can not satisfy bandwidth constraint. Before running the algorithm, we assign infinite delay to those links.

$$d(e) = \begin{cases} \infty & \text{if } b(e) < \Delta_{bandwidth} \\ d(e) & \text{otherwise} \end{cases} \tag{10}$$

Thus bandwidth constraint is mapped to link delay.

Delay jitter constraint: Delay jitter is clearly a function of the number of hops of the route H (see table 1). Before running BF algorithm, the maximum hop path which still satisfy delay jitter can be calculated as shown in Eq. (11). By setting this value to be the maximum number of iteration in BF algorithm, which is the maximum number of hops, we can ensure that the route will have lower delay jitter than the required jitter constraint.

$$H^{jitter}_{max} = \lfloor \frac{\Delta_{jitter} \cdot r_q - \sigma_q}{S^q_{max}} \rfloor \tag{11}$$

Thus the delay jitter is mapped to maximum allowed number of hops in the route.

Loss free or buffer space constraint: This constraint is a special case when loss probability equals to zero. It can be satisfied if there is enough buffer space available at all nodes along the established route. For WFQ, depending on the position of the particular switch from the source node (henceforth we will use the term hop-count), the required buffer size to ensure no packet loss could be determined from Table 1.

We see that the required buffer size increases with this hop-count. Without loss of generality, we assume that each node has its associated buffer corresponding to every incoming link. We can say that a link e has enough buffer if its hop-count is less than or equal to $m(e)_{max}$ defined by the following Eq. (12), where $f(e)$ is the buffer corresponding to link e. Thus, if $m(e)_{max}$ is the maximum allowed hop-count of link e, loss free transmission is ensured. In Bellman-Ford algorithm the number of iteration is the same as the number of hops at that particular stage of searching. So hop-count of every link is the same in a particular iteration, and equal to that iteration number. At the i^{th} iteration a link can be used when $i \leq m(e)_{max}$. Else that link should be ignored. We assume that source node has enough buffer space ($\geq \sigma_q + S^q_{max}$).

$$m(e)_{max} = \lfloor \frac{f(e) - \sigma_q}{S^q_{max}} \rfloor - 1, \quad e = (u, v) \in E \qquad (12)$$

Thus we can map the buffer space constraint to a *link constraint.*

4.2 Informal Description of $QoSR_{BF}$

The basic description of $QoSR_{BF}$ is similar to that of Bellman-Ford (BF) shortest path algorithm [9]. BF algorithm in the first iteration find one-hop shortest paths to all destinations. In the second iteration it tries two-hops shortest path, and continue to higher iterations with longer hops. The algorithm terminates after predefined H_{max} iterations, where H_{max} is the maximum hop allowed, or when the QoS constrained route is found. The cruxes of our $QoSR_{BF}$ are in eliminating those links unsuitable to satisfy QoS, and limiting the number of iterations to the predefined maximum hops. From the previous subsection, these informations could be calculated before running the algorithm. The $QoSR_{BF}$ starts with the calculation of link cost $c(e)$, link delay $d(e)$, the maximum number of hops to satisfy delay jitter H^{jitter}_{max}, the maximum allowed link number to a link $m(e)_{max}$, using equations (1), (9), (11), and (12) respectively. Then it eliminates the links in the network which could not satisfy bandwidth constraint by using Eq. (10). In this paper, we assume that each source has its own policy not to establish a route that has number of hops greater than the predefined maximum value, H^{policy}_{max}. Thus the maximum allowed number of iterations in $QoSR_{BF}$ is defined as,

$$H_{max} = \begin{cases} H^{jitter}_{max} & \text{if } H^{jitter}_{max} < H^{policy}_{max} \\ H^{policy}_{max} & \text{otherwise} \end{cases} \qquad (13)$$

$QoSR_{BF}$ search iterations continue from all 1, 2, ..., H_{max} hops, from source to destination node, until a route is found to satisfy all QoS constraints. At any iteration, before adding a link, those links not having enough buffer with the switches at the end of the links, are eliminated. At each iteration level, if the destination is reached and the paths satisfy the delay constraint, the route which has minimum cost is picked. The algorithm stops either when the QoS constrained route is found, or it completes iteration of H_{max} times.

4.3 Correctness and Complexity Analysis

Theorem 1: *When service discipline used in the network is WFQ, and QoS constraints are bandwidth, delay, jitter and loss free, $QoSR_{BF}$ could always construct a route which satisfy such QoS constraints, if there exists one.*

Proof: The proof is straightforward. Because any links, which do not have enough bandwidth available, will be eliminated before running the algorithm (line 19 of the pseudocode in Appendix A). A route found by $QoSR_{BF}$ will not contain such unsatisfiable links.

Delay jitter of a route solely depends on the number of hops of the route. The maximum allowable value is computed before running the algorithm using Eq. (11) (line 21) to guarantee that delay jitter is satisfied. $QoSR_{BF}$ will search the minimum delay path for which the maximum number of hop is limited by the number of iteration (H_{max}). By setting this number less than the maximum number of hops imposed by delay jitter, when the route is found, would automatically ensure that jitter constraint is satisfied (line 3 and line 22).

The loss free constraint requires that any switch node in the route need to have enough buffer space, to ensure no queueing loss of packet at its incoming link. We have already explained in subsection 4.1, how the maximum allowable hop-count ($m(e)_{max}$) could be calculated for every link depending on its available buffer size. Because Bellman-Ford algorithm searches all possible i hops shortest path from $i-1$ hops shortest path, at each iteration in BF, any links with $m(e)_{max}$ less than the iteration number will be eliminated (the first condition in line 6). This would ensure loss free communication.

In our $QoSR_{BF}$, we search for the shortest delay path. If we fail to find one that satisfies the delay constraint, it is safe to say that there is no such path.

Thus $QoSR_{BF}$ can find a route that satisfy bandwidth, delay, delay jitter and buffer space constraints, if there exists one, for WFQ service discipline.

Theorem 2: *The worst case computation time of $QoSR_{BF}$ is the same as original Bellman-Ford algorithm, i.e. of the order of $O(|E| \cdot H_{max})$, where $|E|$ is the number of links in the network, and H_{max} is the maximum number of iterations of the Bellman-Ford algorithm.*

Proof: In the pseudocode of $QoSR_{BF}$, a loop on line 5 to 7 iterates ($O|E|$) times. In the COMPLEXITY-REDUCTION function, the loop of line 17 to 20 iterates $O(|E|)$ time. The loop of line 28 to 33 in INITIALIZE-SINGLE-SOURCE function also iterates $O(|E|)$. COMPLEXITY-REDUCTION function will be executed only once. INITIALIZE-SINGLE-SOURCE and RELAX functions are executed $H_{max}+1$ and H_{max} times respectively. Therefore the total time would be $O(|E|) + O(|E| \cdot (H_{max}+1)) + O(|E| \cdot H_{max}) = O(|E| \cdot H_{max})$. Thus the worst case computation time of $QoSR_{BF}$ is $O(|E| \cdot H_{max})$.

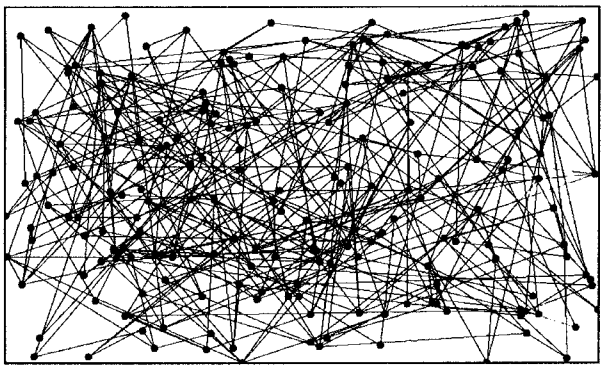

Figure 1: An example of random 200-nodes network.

5 Related Works

Some recent proposals tried to simplify the problem complexity by ranking the importance of the QoS parameters, mostly on delay, hopcount or bandwidth, and use simple SP algorithms to search the route. For example, Wang and Crowcroft [7] proposed Shortest Widest Path (SWP) algorithm . SWP searches the widest (maximum) bottleneck bandwidth route, and if there are more than one such route, the one having minimum delay is chosen. Guerin et. al. [12] proposed Widest Shortest Path (WSP) algorithm for precomputed paths. It searches the minimum hop route that has widest bottleneck bandwidth. Lee et. al. [10] proposed a rule based QoS routing algorithm, called Fallback routing (FB), where a type of routing metric instance is selected at a time. In contrast to SWP, WSP, or FB, which are simple heuristics, Widyono [11] proposed the optimal delay-constrained minimum-cost routing algorithm, called Constrained Bellman-Ford (CBF) algorithm. CBF's running time exponentially increases with the size of the network.

While the above existing QoS routing algorithms did not consider the QoS bound expressions of the service discipline, recently, a few routing algorithms are proposed considering WFQ service discipline. Guerin et. al. [13] proposed to run shortest path algorithm for all of the possible residual bandwidth when the amount of reserved bandwidth or r_q is not known a priori. This is called deterministic case of R-D problem in [13]. However, in their paper, delay jitter and buffer space constraints are not considered. This idea was again suggested by Ma and Steenkiste [14]. They also proposed similar algorithm to $QoSR_{BF}$. However, some detail inside the algorithms are different, and no simulation results were in their paper.

6 Simulation Results

The communication networks used in our experiments are optical fiber, full duplex and directed, with homogeneous link capacity bandwidth of 155.52 Mbps (OC3). Random networks are generated by modifying the routine described in [15]. The positions of the nodes were fixed in a rectangle of size 4000×2400 km^2, roughly the size of the USA. Nodes of all these networks were so connected that the degree of each node is at least 2, and the average node degree is 4, which is close to the average node degree of current networks [16]. Fig. 1 shows a typical example of randomly generated 200-nodes network used in our experiment. Propagation speed is chosen to be two thirds the speed of light. Thus the size of the network in terms of propagation delay is 20×12 msec^2. In every experiment, we initialize the background traffic in each link with random values from 5 to 150 Mbps.

Video traffic was simulated with flow specification of $(\sigma_{video}, \rho_{video}, S_{max}^{video}) = (10S_{max}, 1.5\text{Mbps}, S_{max})$, where S_{max} is equal to the size of an ATM cell, 53 bytes. In our simulations, for every routing algorithm, we restrict the bandwidth reservation to 1.5Mbps, which is the token rate of the requested traffic. The buffer space for each incoming link at a node is randomly selected between 4.6 kb and 128 kb. Thus with this flow specification and initialization, all the links would satisfy the bandwidth constraint. Any node could be selected as the source node.

In each network, we randomly select 10 pairs of sources and destinations. Each plot in the graph is the result averaged over 1000 random 200-nodes networks. We selected H_{max}^{policy} to be one more than the number of hops of the minimum-hop route from source to destination without any QoS considerations. We have implemented an optimal (minimum cost) routing algorithm with delay, delay jitter, and buffer space constraints, which is modification from CBF algorithm. We also implemented FB algorithm, where routing instance is selected in the following order: minimum-hop shortest-cost (MH-SC)[2] route, minimum-hop shortest-delay (MH-SD) route, and shortest delay (SD) route. They are searched by Dijkstra algorithm. SWP and WSP are simulated using Bellman-Ford algorithm. The maximum number of iteration of SWP was set to $|V| - 1$. For WSP, the maximum hop count is restricted to H_{max}.

6.1 Average Success Rate

The average success rate shows how often the network can establish the call that satisfy QoS constraints. In Fig 2, the success rates were plotted varying delay constraint values from 10 ms to 60 ms. Delay jitter was set at a constant value of 10 ms. The simulation results show optimal performance of $QoSR_{BF}$, its success rate is equal to CBF.

When delay constraint is strict, most algorithms, except SC and SWP, have nearly the same success rate. This is because only the destination near the source node could satisfy QoS constraints. Therefore, the selected routes by different algorithms are same in many cases. But when delay constraint is less strict, there are many possible routes to reach the destination. Existing algorithms could not explore all of the routes, so it has lower success rate than $QoSR_{BF}$. Even when delay constraint is not strict, $QoSR_{BF}$ still

[2]MH-SC route is the minimum hop route. However, if there are more than one such routes, it is the one having minimum or shortest cost.

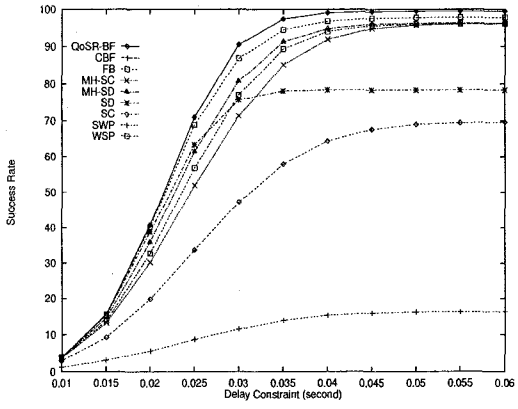

Figure 2: Average success rate.

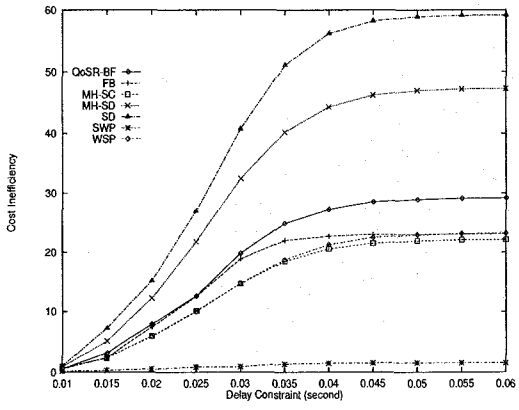

Figure 4: Average cost inefficiency.

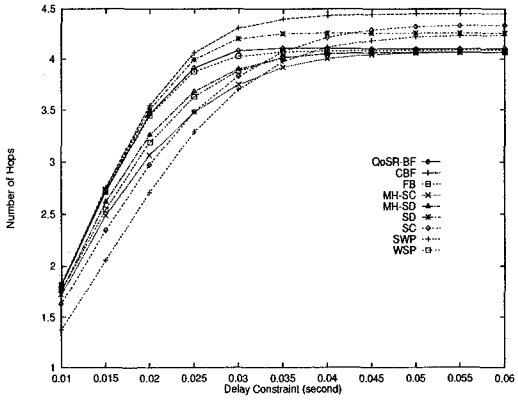

Figure 3: Average number of hops.

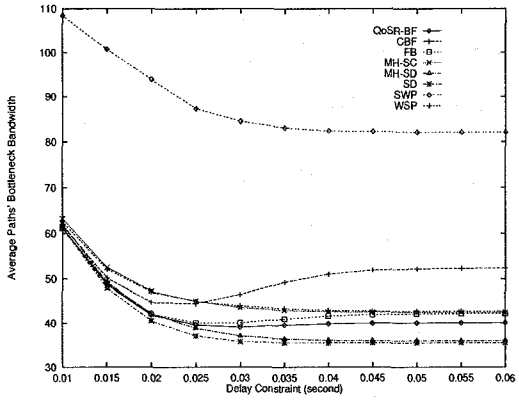

Figure 5: Average path's bottleneck bandwidth.

has higher success rate. This is mainly because the existing algorithms fail to find error free routes. SC, SWP and SD show poor performances as they do not minimize number of hops of the route.

6.2 Average Number of Hops

Figure 3 shows the average number of hops of the routes satisfying QoS constraints. Of the algorithms with quite high success rate of call connection, we see that CBF uses routes with highest average hops. This is because the primary minimization criterion of CBF is the cost. $QoSR_{BF}$ minimizes number of hops. It could find routes with lower average hops. Because only the routes satisfying QoS constraints are used to calculate the average number of hops, and the success rate of SC and SWP are much lower compared to other algorithms, they show lower average number of hops than MH-SC and MH-SD.

6.3 Average Cost Inefficiency

Figure 4 shows the average cost inefficiency or normalized surcharge of $QoSR_{BF}$, FB, MH-SC, MH-SD, SD, SWP and WSP with respect to CBF. The cost inefficiency of algorithm A with respect to CBF is defined as

$$\hat{\delta}_A = \frac{C(P_A) - C(P_{CBF})}{C(P_{CBF})} \tag{14}$$

Simulation results shows that $QoSR_{BF}$ is 29% more expensive than the optimal CBF. It should be mentioned again that cost and hop optimizations are sometimes contradictory. $QoSR_{BF}$ optimizes the number of hops on those routes which satisfy all QoS constraints. Only then it will use cost as the optimization objective. SD has high cost inefficiency ($\hat{\delta}_{SD} \leq 59\%$). The cost inefficiency of SWP is somewhat similar to CBF. Due to the chosen priority of FB, the cost inefficiency of FB, WSP and MH-SC are similar.

6.4 Average of Paths' Bottleneck Bandwidth

Figure 5 shows the average bottleneck bandwidth of the route that satisfy QoS constraints. SWP has highest bottleneck bandwidth, but at the expense of much longer hops than the minimum hop route. And

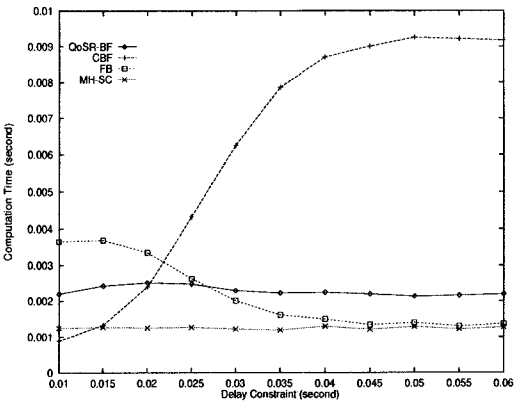

Figure 6: Average computation time.

the success rate is therefore very low. It is about 30 Mbps higher than CBF and about 40 Mbps higher than $QoSR_{BF}$, WSP and FB, when delay constraint is high.

6.5 Average Computation Time

Figure 6 shows the average computation time[3] of different QoS routing algorithms. When the delay constraint is very strict, CBF, which uses breadth first search, will find the route very fast. This is even faster than Dijkstra algorithm for shortest path. The reason is that CBF searches the route by increasing delay until it is greater than the delay constraint. Thus when there is no possible route satisfying the delay constraint, CBF can detect it very fast. FB will first try MH-SC, but can not terminate at this stage. It has to retry MH-SD and SD next. When delay constraint is not strict, CBF has much longer computation time compared to $QoSR_{BF}$. As we restrict the hop count to H_{max}, even for CBF, the search space is restricted and the computation time stabilizes when delay constraint reaches 40ms. FB in most cases could find the route in one step using MH-SC when delay is not strict. So its computation time decreases nearly to that of MH-SC. The computation time of $QoSR_{BF}$ is similar to MH-SC and FB algorithms.

7 Conclusion

Though the complexity of the QoS routing problem was proved to be NP-complete, it has been done without any assumptions of conditions from the service discipline that will be used in the network. The QoS bound values, such as delay bound, jitter bound etc., depend on the service discipline. Thus QoS based routing algorithm should be designed considering the specific service discipline. In this paper, we show that by employing WFQ service discipline, we can reduce the complexity of the problem to that of simple shortest path routing, without any QoS constraints. Therefore, we can find such QoS constrained route in polynomial time.

We then present the QoS based routing algorithm, called "$QoSR_{BF}$". The proposed $QoSR_{BF}$ can find a route that satisfy bandwidth, delay, delay jitter, and buffer constraints, if there exists one. $QoSR_{BF}$ is simple, fast and scale well to large networks. Its computation time in worst case is as fast as original SP Bellman-Ford algorithm, i.e. of the order of $O(|E| \cdot H_{max})$, where $|E|$ is the number of links in network, and H_{max} is the maximum number of iterations of the Bellman-Ford algorithm, an upper-bound on the number of hops of the route. Simulation results confirm that the success rate of the $QoSR_{BF}$ is optimal. It also has quite good performance in terms of number of hops, cost inefficiency, path's bottleneck bandwidth. Finally, its average computation time is still near to Dijkstra shortest path algorithm.

[3]The Dijkstra versions of SC, SD, MH and CBF algorithms were all implemented using binary heap. However, the code for all algorithms may not be speed optimized, though we have carefully implemented.

References

[1] D. D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *Proc. of ACM Sigcomm'92*, pages 14–26, Baltimore, August 1992.

[2] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture: An overview. RFC 1633, June 1994.

[3] S. Shenker, C. Partridge, and R. Guerin. Specification of guaranteed quality of service. *Internet Draft*, August 1996.

[4] H. Zhang and D. Ferrari. Rate-controlled static priority queuing. In *Proc. of IEEE INFOCOM*, 1993.

[5] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Trans. Networking*, 2(2):137–150, April 1994.

[6] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. Rsvp: A new resource reservation protocol. *IEEE Network*, September 1993.

[7] Z. Wang and J. Crowcroft. Quality of service routing for supporting multimedia applications. *IEEE J. Select. Areas Commun.*, 14(7):1228–1234, 1996.

[8] H. Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proc. of the IEEE*, 83(10), October 1995.

[9] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1994.

[10] W. C. Lee, M. G. Hluchyj, and P. A. Humblet. Routing subject to quality of service constraints in integrated communication networks. *IEEE Network*, pages 46–55, July/August 1995.

[11] R. Widyono. The design and evaluation of routing algorithms for real-time channels. Technical Report TR-94-024, University of California at Berkeley, June 1994.

[12] R. Guerin, A. Orda, and D. Williams. Qos routing mechanisms and ospf extensions. *Internet Draft*, March 1997.

[13] R. Guerin and A. Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. In *Proc. of IEEE INFOCOM*, Kobe, Japan, April 1997.

[14] Q. Ma and P. Steenkiste. Quality-of-service routing for traffic with performance guarantees. In *IFIP Fifth International Workshop on Quality of Service (IWQOS'97)*, May 1997.

[15] B. M. Waxman. Routing of multipoint connections. *IEEE J. Select. Areas Commun.*, 6(9):1617–1622, 1988.

[16] H. F. Salama, D. S. Reeves, and Y. Viniotis. A distributed algorithm for delay-constrained unicast routing. In *Proc. of IEEE INFOCOM*, Kobe, Japan, April 1997.

Appendix: $QoSR_{BF}$ Pseudocode

Input: A direct connected graph $G = (V, E), v \in V$, represents a node in the network, and $e = (u, v) \in E$ represents the direct link e, from node u to v.
s, z: source and destination nodes.
$b(e)$: available (unreserved) bandwidth of link e.
$R(e)$: the capacity of link e.
$f(e)$: available input buffer space at node v of link e.
$\tau = (\sigma_q, \rho_q, S^q_{max})$: flow specification (maximum burst size, token rate, maximum packet size).
$pd(e)$: propagation delay of link e.
Δs: QoS constraints (with respective suffixes).
H^{policy}_{max}: is the maximum number of hops determined by source policy.

Output: Boolean TRUE, if a route which could satisfy QoS constraints is found, and FALSE otherwise.
$\pi_i[v]$: the previous node of v on the path P, which has the number of hops equal to i up to node v.
$P(s, z)$, a QoS constrained route from s to z.
$C(P)$, $D(P)$, $B(P)$, $H(P)$, $J(P)$: cost, delay, bottleneck bandwidth, number of hops, and delay jitter of $P(s, z)$ respectively.
$c_i[v]$, $d_i[v]$, $b_i[v]$, $h_i[v]$, $j_i[v]$: cost, delay, bottleneck bandwidth, number of hops, and delay jitter of the path, $(i = h_i[v])$, to node v respectively.

```
QoSR-BF(G, s, z, b, f, τ, pd, Δ, H^policy_max)
1  COMPLEXITY_REDUCTION
   (G, s, b, f, τ, pd, Δ, H^policy_max)
2  INITIALIZE-SINGLE-SOURCE(G, 0, s)
3  for i ← 1 to H_max
4       INITIALIZE-SINGLE-SOURCE(G, i, s)
5       do for each edge e = (u, v) ∈ E[G]
6            if ((i ≤ m(e)_max) and (h_{i-1}[u] == i − 1))
                 and (d_{i-1}[u] ≤ Δ_delay))
7            then RELAX(u, v, i, z, Δ_delay)
8       if ((d_i[z] ≤ Δ_delay)
9            C(P) ← c_i[z]
10           D(P) ← d_i[z]
11           B(P) ← b_i[z]
12           H(P) ← h_i[z]
13           J(P) ← jitter bound in Table 1
14           return TRUE
15 return FALSE
COMPLEXITY_REDUCTION
(G, s, b, f, τ, pd, Δ, H^policy_max)
16 r_q ← Δ_bandwidth
17 do for each edge e = (u, v) ∈ E[G]
18           c(e) ← Eq. (1)
19           d(e) ← Eq. (9), and Eq. (10)
20           m(e)_max ← Eq. (12)
21 H^jitter_max ← Eq. (11)
22 H_max ← Eq. (13)
23 C(P) ← ∞
24 D(P) ← ∞
25 J(P) ← ∞
26 B(P) ← 0
27 H(P) ← ∞
INITIALIZE-SINGLE-SOURCE(G, i, s)
28 do for each vertex v ∈ V[G]
29           c_i[v] ← ∞
30           b_i[v] ← 0
31           d_i[v] ← ∞
32           h_i[v] ← ∞
33           π_i[v] ← NIL
34 c_i[s] ← 0
35 b_i[s] ← ∞
36 h_i[s] ← 0
37 d_i[s] ← 0
RELAX(u, v, i, z, Δ_delay)
38 if v! = z
39      if d_i[v] > d_{i-1}[u] + d(u, v)
40           then d_i[v] ← d_{i-1}[u] + d(u, v)
41                c_i[v] ← c_{i-1}[u] + c(u, v)
42                h_i[v] ← h_{i-1}[u] + 1
43                b_i[v] ← MIN(b_{i-1}[u], b(u, v))
44                π_i[v] ← u
45 else if ((d_{i-1}[u] + d(u, v) ≤ Δ_delay) and
           (c_i[v] > c_{i-1}[u] + c(u, v)))
46           then d_i[v] ← d_{i-1}[u] + d(u, v)
47                c_i[v] ← c_{i-1}[u] + c(u, v)
48                h_i[v] ← h_{i-1}[u] + 1
49                b_i[v] ← MIN(b_{i-1}[u], b(u, v))
50                π_i[v] ← u
MIN(x, y)
51 if x < y
52      return x
53 else
54      return y
```