Routing Guaranteed Quality of Service Connections in Integrated Services Packet Networks *

Wei Zhao
Mobile Computing and Multimedia Lab
Department of Computer Science
University of Maryland
College Park, MD 20742
zw@cs.umd.edu

Satish K. Tripathi
Bourns College of Engineering
University of California
Riverside, CA 92521-0425
tripathi@engr.ucr.edu

Abstract

A critical functional component of quality-of-service (QoS) deployment in packet-switched networks is QoS-based routing. In this paper, we present a routing solution for guaranteed quality-of-service connections in integrated services packet networks (ISPN) – the future QoS-capable Internet proposed by the IETF. The problem is in essence a path finding problem with both end-to-end delay and per-node buffer constraints, in networks with heterogeneous intermediate switching nodes. We present a polynomial time algorithm using a capacity plane decomposition technique combined with a per-node constrained shortest path algorithm. We further propose an efficient route computation architecture, based on a novel metric-separation approach, to a slightly restricted version of the problem. The strategy is somewhat similar to "route caching", but in a new and broader sense.

Keywords: *QoS-based routing, quality of service, guaranteed service, admission control, resource allocation, integrated services*

1 Introduction

Network infrastructure and protocols are undergoing fundamental changes driven by the need to support network-performance-sensitive, such as distributed multimedia and real-time applications. An essential part of the QoS architecture that has not received a lot of attention until recently is the problem of QoS-based routing. Lack of good QoS routing algorithms may cause problems such as high call blocking probabilities, low network utilizations and long connection setup delays. These detrimental effects may to some extent compromise the potential benefits brought forward by the QoS deployment in future networks.

In this paper, we address the problem of quality-of-service based routing for *Guaranteed Service* in *Integrated Services Packet Networks* (ISPN) [1]. ISPN is a recent proposal by the *Internet Engineering Task Force* (IETF) aimed at establishing a quality-of-service architecture in the Internet. The proposal defines two new classes of connections with different levels of QoS guarantees, *guaranteed service* and *controlled-load service*, in addition to the traditional best effort service. Guaranteed service [14] ensures a mathematically provable delay bound and a zero packet loss for the data packets it carries, while controlled-load service [18] is loosely defined as providing a packet delivery performance comparable to that of a network in a noncongested state. Given a connection setup request, a network path is *feasible* if sufficient resources exist along the path to meet the connection's QoS requirements. The QoS-based routing problem is thus to find a path with sufficient resources for the connection to be routed along [4, 9, 17]. Among the above three service classes, routing for guaranteed service is the most critical problem due to its inherently stringent resource requirements on the network components along the connection path.

In this paper, we formulate the guaranteed service routing problem and present solutions to it. We present a polynomial time solution based on capacity plane decomposition and a per-node constrained shortest path algorithm. When the buffer constraints are negligible, we are able to take advantage of a very useful

*This research was supported in part by the U.S. Department of the Army, Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium.

metric-separation technique. The technique separates the original problem into three subproblems along the temporal dimension, based on the types of metrics involved. Algorithms are presented to solve the three separate subproblems. By integrating these algorithms into an integrated QoS routing architecture based on the dynamics of the routing messages, we are able to greatly improve the efficiencies of route computations.

The rest of the paper is organized as follows. Section 2 outlines the specification of the guaranteed service. Section 3 formulates the guaranteed service routing problem. Section 4 presents a routing algorithm for guaranteed service. An efficient routing architecture and related algorithms are proposed in Section 5. Section 6 presents a result on the related problem of resource allocation. After discussing some related work in Section 7, we conclude the paper with a summary.

2 Guaranteed Quality of Service

The IEFT proposed a packet delivery service called guaranteed service with firm end-to-end delay and zero packet loss [1]. It is defined with reference to a fluid model of service, which is essentially the service provided by a dedicated wire of bandwidth R between a pair of end points. A flow characterized by a token bucket (b, r) served with constant service rate R yields a delay bound of b/R for any $R \geq r$, where b and r are the token bucket size and token generation rate, respectively. The ideal fluid model relies on the infinite divisibility of the service unit which is impossible to achieve in reality. There are a number of packet scheduling disciplines [19] that approximate the fluid model. *Weighted Fair Queueing* (WFQ) [5, 12], *Virtual Clock* [20], *Self-Clocked Fair Queueing* [7], *Deficit Round Robin* [15] are some of the packet scheduling disciplines that approximate the fluid model. In order to treat various scheduling disciplines that may be present in a network in a unified manner, two similar frameworks, *Latency-Rate Servers* ($\mathcal{LR}$-Servers) [16] and *Generalized Guaranteed Rate Servers* (GR-Servers) [8] have been proposed. Both frameworks encompass all the scheduling disciplines listed above. In essence, the frameworks define a lag factor that characterizes how far behind the service of a fluid-approximating scheduling discipline trails behind the ideal fluid model. By accumulating the lag factors of servers along a connection path, the end-to-end delay bound and the required buffer spaces can be determined.

In particular, guaranteed service defines two non-negative constants, a rate-dependent error term C_i and a rate-independent error term D_i for each network node (e.g. switch or router) n_i, to characterize the lag of the service of the network node. The values of C_i and D_i depend only on the specific scheduling discipline or other specific mechanisms (such as packet fragmentation and reassembly) employed at the particular network node n_i and are independent of the network state. Service at a network node is characterized by a service rate R and a buffer size B. With a token bucket (b, r) constrained input traffic entering network node n_i, a delay bound of $b/R + C_i/R + D_i$ can be guaranteed. In other words, term $C_i/R + D_i$ represents the server's lag behind the ideal fluid model. By employing the above scheduling frameworks, the end-to-end delay bound along a path P from source s to destination t consisting of network nodes $n_0 = s, n_1, \ldots, n_{l-1}, n_l = t$ can be shown to be the following, [1]

$$D = b/R + \sum_{i=0}^{l-1} C_i/R + \sum_{i=0}^{l-1} D_i \qquad (1)$$

Note that the delay bounds along the path is not additive: the end-to-end delay bound is in fact much tighter than the sum of the individual delay bounds.

In addition to satisfying the end-to-end delay bound, guaranteed service must also ensure that no buffer overflow occurs in the network. For node n_j along the connection path, the buffer space required for zero packet loss is

$$B(j) = b + (\sum_{i=0}^{j} C_i/R + \sum_{i=0}^{j} D_i)r \qquad (2)$$

This buffer bound is tighter than the bound given in [14], the proof is given in the appendix. Throughout the paper, we assume a guaranteed service connection request from source s to destination t with a traffic token bucket descriptor (b, r) and a delay bound d.

3 Problem Definition of Guaranteed Service Routing

We formally define the problem of guaranteed service routing and establish some preliminary properties. Let the network be represented by a directed graph $G = (V, E)$, $|V| = n$ and $|E| = m$. Vertex set V represents network nodes and edge set E represents links between network nodes. Associated with each vertex n_i are two scheduling error terms C_i and D_i, and an available buffer size B_i. A residual capacity $R_{i,j}$ is associated with each link (i, j), representing the currently available bandwidth on the link. Given a guaranteed service connection routing request, we want to find a path and a service rate allocation R along the path

[1] We express the max propagation delay of a link as part of D_i, see the discussion later in the paper for its implications.

that can accommodate its requirements. The following lemma holds by the above formulae on delay and buffer bounds.

Lemma 1 *For a guaranteed service connection setup request from source s to destination t with traffic descriptor (b, r) and delay bound d, a path $P = (s = n_0, n_1, \ldots, n_l = t)$ can serve the connection with service rate R, if the following hold:*

$$R \geq r \quad (3)$$

$$R \leq R_{j,j+1} \quad (4)$$

$$D = b/R + \sum_{i=0}^{l-1} C_i/R + \sum_{i=0}^{l-1} D_i \leq d \quad (5)$$

$$B(j) = b + (\sum_{i=0}^{j} C_i/R + \sum_{i=0}^{j} D_i)r \leq B_j \quad (6)$$

for all $j = 1, 2, \ldots, l-1$.

The last two conditions (5) and (6) are for the end-to-end delay bound and the per-hop buffer requirements defined earlier. Condition (4) requires that the allocated service rate R not exceed the residual capacity of any link along the path. Condition (3) simply requires that the service rate R be at least the average input traffic rate r, a precondition for the queue stability. The following lemma establishes a relationship among service rates with which a path can serve a connection.

Lemma 2 *If a path $P = (n_0, n_1, \ldots, n_l)$ can serve a connection with a service rate R, then it can serve the connection with any service rate R', $R \leq R' \leq R^*$. $R^* = \min_{0 \leq i \leq l-1} R_{i,i+1}$ is the bottleneck residual capacity of the path.*

Proof: Replacing R by a larger R' in Lemma 1, both delay bound D and buffer requirements $B(j)$ decrease, thus conditions (5) and (6) still hold. Conditions (3) and (4) also hold for R' by

$$r \leq R \leq R' \leq R^* = \min_{0 \leq i \leq l-1} R_{i,i+1} \leq R_{j,j+1}.$$

□

Theorem 1 *A path is feasible for a guaranteed service connection if and only if it can serve the connection with its bottleneck residual capacity R^*.*

Proof: By Lemma 1 and Lemma 2. □

As a result of Theorem 1, the problem of finding a feasible path reduces to the problem of finding a path satisfying the conditions in Lemma 1 with R as the path's bottleneck residual bandwidth R^*.

Corollary 1 *A path $P = (s = n_0, n_1, \ldots, n_l = t)$ with bottleneck residual capacity R^* is feasible if and only if the following hold:*

$$R^* \geq r \quad (7)$$

$$D = b/R^* + \sum_{i=0}^{l-1} C_i/R^* + \sum_{i=0}^{l-1} D_i \leq d \quad (8)$$

$$B(j) = b + (\sum_{i=0}^{j} C_i/R^* + \sum_{i=0}^{j} D_i)r \leq B_j \quad (9)$$

for all $j = 1, 2, \ldots, l-1$.

4 Guaranteed Service Routing Algorithm

Let K be the number of different values of $R_{i,j}$'s in the graph, then the number of values that the bottleneck rate R^* can take is bounded by K. Note that $K \leq m$ since each edge contributes at most one distinct R^*. For each distinct value $R^{(k)}$ of R^*, we construct a subgraph $G^{(k)}$ of the original graph by keeping only the edges with residual capacities $R_{i,j}$'s *at least as high as* $R^{(k)}$: $E(G^{(k)}) = \{(i,j) | R_{i,j} \geq R^{(k)}\}$. Now for each subgraph $G^{(k)}$, we define the cost of path P as

$$Cost^{(k)}(P) = \sum_{n_i \in V(P)} (C_i/R^{(k)} + D_i) \quad (10)$$

where $V(P)$ is the vertex set of P. For simplicity, we let the C_i and D_i of the destination be zero. We first define the following subproblem.

Problem 1 *(Per-hop Constrained Shortest Path) Find a shortest path (path with the lowest cost) $P = (s = n_0, n_1, \ldots, n_l = t)$ satisfying the following per-hop constraints: for any path prefix P' of P ending at node n_j,*

$$Cost(P') \leq (B_j - b)/r.$$

Lemma 3 *A path P in $G^{(k)}$ satisfies the per-hop constraints in Problem 1 if and only if P satisfies the buffer bound conditions in Lemma 1 with service rate $R^{(k)}$.*

Proof: Assume P is a path in $G^{(k)}$ satisfying the per-hop constraints. For any network node n_j on path P, let P' be the prefix of P ending on n_j. By the definitions of the cost function (10) and Problem 1, the following inequalities are equivalent:

$$\begin{aligned}
Cost(P') &\leq (B_j - b)/r \\
b + (Cost(P'))r &\leq b + ((B_j - b)/r)r \\
b + (\textstyle\sum_{i=0}^{j} C_i/R^{(k)} + \sum_{i=0}^{j} D_i)r &\leq B_j \\
B(j) &\leq B_j
\end{aligned}$$

□

Theorem 2 *There is a feasible path for the guaranteed connection request if and only if, there exists a subgraph $G^{(k)}$ with $R^{(k)} \geq r$ such that the path P found by Problem 1 on $G^{(k)}$ satisfies $b/R^{(k)} + Cost^{(k)}(P) \leq d$. In particular, P is a feasible path for the connection.*

Proof: Sufficiency. We show that P found by Problem 1 on graph $G^{(k)}$ can serve the connection with service rate $R^{(k)}$ by Lemma 1. Since P is in graph $G^{(k)}$, only edges with residual capacity at least $R^{(k)}$ are present. So $R^{(k)} \leq R_{j,j+1}$ is satisfied. The delay bound condition

$$\begin{aligned} D &= b/R^{(k)} + \sum_{i=0}^{l-1} C_i/R^{(k)} + \sum_{i=0}^{l-1} D_i \\ &= b/R^{(k)} + Cost^{(k)}(P) \leq d \end{aligned}$$

is satisfied. By Lemma 3, the buffer bound conditions also hold.

Necessity. Let P be a feasible path with bottleneck capacity R^* satisfying the conditions in Corollary 1. Choose k such that $R^{(k)} = R^*$. We show that P is a path in $G^{(k)}$ satisfying the per-hop constraints of Problem 1. First, P is a path in $G^{(k)}$ since $R_{j,j+1} \geq R^* = R^{(k)}$. By Lemma 3, P also satisfies the per-hop constraints in Problem 1. The validities of $b/R^{(k)} + Cost^{(k)}(P) \leq d$ and $R^{(k)} \geq r$ are directly implied by Corollary 1. Since Problem 1 finds the shortest path under the constraints, it must also satisfy the above conditions. □

Therefore, we can solve the routing problem by solving the per-hop constrained shortest path problem in each subgraph and checking the delay conditions of the resulting paths. A *Per-hop Constrained Shortest Path Algorithm* in Figure 1 solves the subproblem on graph $G^{(k)}$ with $R^{(k)} = R$. The correctness proof is given in the appendix. The algorithm has the same worst case complexity as the Dijkstra's algorithm, $O(m + n \log n)$ when a Fibonacci-Heap is used.

The *Guaranteed Service Route Computation* algorithm in Figure 2 computes a feasible path based on the solutions to the subproblems.

Corollary 2 *The algorithm in Figure 2 solves the guaranteed service routing problem.*

The corollary is implied by Theorem 2. The worst case complexity of the algorithm is $O(K(m + n \log n))$, where K is the number of different $R_{i,j}$ values, $K \leq m$. Note that the algorithm checks paths with residual bandwidth in descending order. This ensures that the path found by the algorithm is the one with the largest residual capacity, or the "widest", among all feasible paths. Thus the algorithm achieves some degree of load balancing.

Per_Hop_Constrained_Shortest_Path(G)
$S \leftarrow \{s\}$, $Cost(0) \leftarrow C_0/R + D_0$, $Cost(i \neq 0) \leftarrow \infty$
for each adjacent node n_i of s,
  $Cost(i) \leftarrow \min\{Cost(i), Cost(0) + C_i/R + D_i\}$
while true
  find k such that $Cost(k) = \min_{n_i \notin S}\{Cost(i) | Cost(i) \leq (B_i - b)/r\}$
  if k is not found, return null
  if node n_k is t, return the path
  $S \leftarrow S \bigcup \{n_k\}$, extend the path with n_k
  for each adjacent node n_i of n_k
    $Cost(i) \leftarrow \min\{Cost(i), Cost(k) + C_i/R + D_i\}$

Figure 1. Per-hop Constrained Shortest Path

Guaranteed_Service_Route_Computation(G)
sort the set $\{R_{i,j}\}$ in descending order
for every distinct $R \in \{R_{i,j}\}$
  if $R < r$ return null
  construct graph G' by deleting from G all edges with $R_{i,j} < R$
  call Per_Hop_Constrained_Shortest_Path(G')
  if P found and $b/R + Cost(P) \leq d$, return P

Figure 2. Guaranteed Service Routing

5 Efficient Routing Architecture

The previous algorithm solves the route computation problem with a computation time of K times that of the Dijkstra's algorithm. The worst case of K is m. In real networks, the interface hardware limits the degree of each network node to a constant. In fact, the average node-degree in the Internet is very small (2-4). Thus m is of the same order of magnitude as n. The worst case running time of the algorithm becomes $O(n^2 \log n)$. This should be good enough for link-state based intra-domain routing in average size networks with relatively infrequent guaranteed service routing requests.

In this section, we improve the running time of guaranteed service routing, in order to scale it to larger networks with more frequent routing requests. We make the assumption that if we select a path that satisfies the delay bound condition, it satisfies the buffer bound conditions as well. By the facts that we select "widest" paths with lowest delays and that guaranteed service connections typically request low delays, this might be a reasonable assumption, based on the observation that low delays usually imply low buffer occupancies.

Proper network capacity planning can also contribute to this assumption. Even when the assumption is not true, a set of paths satisfying the delay bound can be found very quickly by our algorithm, and subsequently be subjected to buffer bound tests. Only when all of these paths violate the buffer bound conditions will the original algorithm be executed as a final resort.

5.1 Problem Decomposition by Metric Separation

We separate the terms in the delay bound condition in Corollary 1 into a connection request dependent component b/R^* and d, and a connection request independent component $\sum_{i=0}^{l-1} C_i/R^* + \sum_{i=0}^{l-1} D_i$. The connection independent component depends only on the network state.

By metric separation, we decompose the problem into two subproblems: one that computes a partial result using only the connection independent metrics and another that computes the final path based on the connection dependent information. In this particular problem, the partial result is a set of candidate paths with "low" connection-independent part, $\sum_{i=0}^{l-1} C_i/R^* + \sum_{i=0}^{l-1} D_i$, of the total delay. This can be computed based only on the network state information. When a connection routing request arrives, the final route is *selected* from the candidate paths using the per-connection metrics. Note that the first subproblem essentially corresponds to the existing routing solutions in the current non-QoS Internet.

5.2 Candidate Paths

As before, we construct a subgraph $G^{(k)}$ for every capacity level $R^{(k)}$. We define the set of *candidate paths* to be *the set of shortest paths at each capacity level*, according to the same cost functions as in (10).

Corollary 3 *There exists a path satisfying the delay bound condition, if and only if, there exists an $R^{(k)}$, such that the candidate path belonging to $G^{(k)}$ satisfies*

$$b/R^{(k)} + Cost^{(k)}(P) \leq d$$

The corollary follows from Theorem 2. Consequently, keeping this set of candidate paths is sufficient for finding a path satisfying the delay bound whenever one exists. The algorithm for finding candidate paths is simply a Dijkstra's algorithm in each $G^{(k)}$ with cost function $Cost^{(k)}$. Once the set of candidate paths is found, the path computation for each connection request is simply to check each candidate path P in $G^{(k)}$ whether $b/R^{(k)} + Cost^{(k)}(P) \leq d$ is true.

5.3 Routing Algorithm Architecture

Any routing entity in the network receives three categories of input messages: *initial network state information*, *connection routing request* and *network state update*. Initial network state arrives only once at the routing entity start-up time when it receives the flooded topology information for the first time. Network state changes occur at different time-scales: the C_i and D_i terms associated with each network node never changes; the up/down state of a link changes rarely; while the residual bandwidths change more frequently, depending on the state propagation-triggering thresholds. Most importantly, the response time to a routing request is the determining factor in routing performance and thus has the highest priority for optimization. Figure 3 shows the routing algorithm architecture.

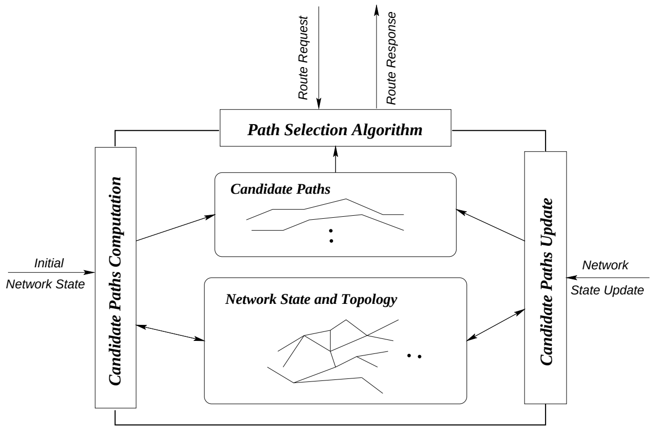

Figure 3. A Routing Algorithm Architecture

As in Figure 3, we integrate three separate algorithms into a unified routing architecture. The *candidate paths computation algorithm* computes candidate paths based only on the metrics associated with the network. The *path selection algorithm* selects a path from the set of candidate paths based on the connection-specific parameters, in response to a routing request. Route selection is expected to be very fast since it only checks the feasibilities of the candidate paths, thus resulting in a significant routing performance improvement. The *candidate paths update algorithm* updates the candidate paths upon network state changes, such as a residual bandwidth increase/decrease.

5.4 Candidate Paths Computation Algorithm

The Dijkstra-like algorithm in Figure 4 computes the shortest path in the subgraph with capacity R. The

Candidate Paths Computation Algorithm in Figure 5 calculates all candidate paths. The worst case running time is $O(K(m+n\log n))$. K is the number of distinct capacity levels, which is also the bound on the number of candidate paths.

Candidate_Path_In_Subgraph(G)
$S \leftarrow \{s\}$, $Cost(0) \leftarrow C_0/R + D_0$, $Cost(i \neq 0) \leftarrow \infty$
for each adjacent node n_i of s,
  $Cost(i) \leftarrow \min\{Cost(i), Cost(0) + C_i/R + D_i\}$
while true
  find k such that $Cost(k) = \min_{n_i \notin S} Cost(i)$
  if k is not found, return null
  if node n_k is t, return the path
  $S \leftarrow S \bigcup \{n_k\}$, extend the path with n_k
  for each adjacent node n_i of n_k
    $Cost(i) \leftarrow \min\{Cost(i), Cost(k) + C_i/R + D_i\}$

Figure 4. Candidate Path in Subgraph

Candidate_Paths_Computation(G)
sort the set $\{R_{i,j}\}$ in descending order
for every distinct $R \in \{R_{i,j}\}$
  construct graph G' by deleting from G all edges with $R_{i,j} < R$
  call Candidate_Path_In_Subgraph(G')
  if path P is found, record P as a candidate path

Figure 5. Candidate Paths Computation

5.5 Path Selection Algorithms

We have two path selection algorithms, one that checks the buffer conditions and one that does not. We show the one with buffer checking in Figure 6. Let the candidate paths be ordered in decreasing $R^{(k)}$ order. The algorithm simply checks the candidate paths one by one to verify whether there is a candidate path satisfying the delay and buffer conditions in Corollary 1. The worst case running time of the algorithm is $O(KL)$, K is the bound on the number of candidate paths, and L is their average length. The efficiency of the path selection algorithm directly results in route computation efficiency.

The path selection algorithm without buffer checking is virtually identical, with the only exception that the buffer condition tests on line 6 are omitted. The worst case running time is $O(K)$.

Path_Selection_With_Buffer(G)
1 for every distinct $R \in \{R_{i,j}\}$
2   if $R < r$, return null
3   $P \leftarrow$ the candidate path at capacity level R
4   If P does not exist, continue the loop
5   If $b/R + Cost(P) \leq d$ and
6   for each prefix P' of P ending at vertex j, $b + Cost(P')r \leq B_j$,
7   return path P

Figure 6. Path Selection with Buffer

5.6 Candidate Paths Update Algorithm

We update the set of candidate paths when some link's residual capacity changes. Observe that a change only affects the candidate paths belonging to the subgraphs which the change "crosses". A subgraph with capacity R is said to be crossed by a change of a link's capacity from R_{old} to R_{new} if $R_{old} \geq R$ but $R_{new} < R$, or vice versa. The crossed subgraphs thus have capacities between R_{old} and R_{new}. In addition, the only change in each subgraph is an addition/deletion of a single edge. The *Candidate Paths Update Algorithm* is shown in Figure 7. The worst case running time of the algorithm is $O(Q(m+n\log n))$, where Q is the number of crossed capacity planes. More efficient incremental algorithms are possible to handle addition/deletion of an edge.

Candidate_Paths_Update(G)
Find crossed capacity planes: $\{R^{(k)} | K_1 \leq k \leq K_2\}$
for every distinct $R \in \{R^{(k)} | K_1 \leq k \leq K_2\}$
  Update graph G' of capacity plane R by adding or deleting one edge
  remove the old candidate path of G'
  call Candidate_Path_In_Subgraph(G')
  if path P is found, record P as a candidate path

Figure 7. Candidate Paths Update Algorithm

6 Resource Allocation

The following corollary gives the possible resource allocations for guaranteed service given a network path.

Corollary 4 *A connection request is admissible along a network path $P = (n_0, n_1, \ldots, n_l)$, if and only if, $R^o = \max_{j=0}^{l-1} r_i \leq R^*$. R^o is the minimum feasible service rate. Any rate R, $R^o \leq R \leq R^*$ is a feasible* *service rate, with buffer allocation,* $b + (\sum_{i=0}^{j} C_i/R + \sum_{i=0}^{j} D_i)r$*, at network node* n_j*, where*

$$r_0 = (b + \sum_{i=0}^{l-1} C_i)/(d - \sum_{i=0}^{l-1} D_i)$$

$$r_j = (r \sum_{i=0}^{j} C_i)/(B_j - b - r \sum_{i=0}^{j} D_i), \ j = 1, \ldots, l-1.$$

Proof: r_0 is the minimum service rate required to achieve the delay bound condition in Lemma 1. $r_j (j > 0)$ is the minimum service rate required to satisfy the buffer bound condition on node n_j. Hence R^o is the minimum feasible service rate. The rest follows from Lemma 2 and the buffer bound in (2). □

7 Related Work

Path-finding problems with multiple cost constraints is NP-Complete [6], thus QoS routing problems with multiple independent additive or multiplicative QoS metrics is NP-Complete [17]. However, routing with *separate* bottleneck bandwidth and additive delay constraints can be solved in polynomial time. In particular, Wang *et.al.* [17] and Guérin *et.al.* [9] proposed algorithms to find the "shortest widest path". For networks with fluid-approximating switching nodes, bandwidth and delay are closely coupled. Chen *et.al.* [2] presented algorithms for a different but related "quickest path problem". Guérin *et.al.* [10] considered routing with inaccurate network information. Neither considered buffer constraints. Ma *et.al.* [11] and Pornnavalai *et.al.* [13] proposed polynomial Bellman-Ford routing algorithms in networks consisting of homogeneous WFQ-like scheduling nodes. With homogeneous scheduling, buffer constraints are easily satisfied by limiting the hop-count of path searching. We formulated the problem according to the guaranteed service specification [14], which allows each switching node to advertise its own specific scheduling error terms. Hence the results in this paper are applicable to networks with heterogeneous scheduling disciplines.

We point out that buffer constraints resulted from heterogeneous scheduling can greatly complicate the path finding problem. Recall that we expressed the max propagation delay in the error term D_i of the queueing delay. In doing so, the max propagation delay contributes to the buffer bounds at the downstream nodes, which is necessary to handle variable link delays. Variable link delays can happen in cases such as the "tunneling" of a QoS connection through a QoS-unaware subnetwork. In case that all link delays are constant, propagation delays can be separated from queueing delays and thus results in tighter buffer bounds. However, the resulting path finding problem is NP-Complete in general. Essentially, by separating propagation delays from queueing delays which determine the buffer bounds, the routing problem reduces to a path finding problem with multiple independent additive constraints. The detailed proof of the NP-Completeness is given in [21]. With respect to this NP-Complete problem, the algorithm in Figure 2 becomes an approximation algorithm that should perform reasonably well by using slightly conservative buffer bounds.

8 Summary

We addressed the problem of guaranteed quality-of-service connection routing in integrated services packet networks. We presented a polynomial time algorithm to the problem. With the buffer sufficiency assumption, we proposed an efficient routing architecture using a novel metric-separation technique. Together with resource allocation, we presented integrated solutions to the problem of QoS connection establishment.

References

[1] R. Braden, D. Clark, and S. Shenker. Integrated Services in the Internet Architecture: an Overview. *Internet RFC 1633*, June 1994.

[2] Y. Chen and Y. Chin. The Quickest Path Problem. *Computers and Operations Research*, 17(2):153–161, 1990.

[3] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. McGraw-Hill, 1990.

[4] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *Internet Draft, "draft-ietf-qosr-framework-01.txt"*, July 1997.

[5] A. Demers, S. Keshav, and S. Shenker. Analysis and Simulation of a Fair Queueing Algorithm. In *Proceedings of the ACM SIGCOMM '89*, 1989.

[6] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[7] S. Golestani. A Self-Clocked Fair Queueing Scheme for Broadband Applications. In *Proceedings of the IEEE INFOCOM '94*, 1994.

[8] P. Goyal, S. Lam, and H. Vin. Determining End-to-End Delay Bounds in Heterogeneous Networks. In *Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1995.

[9] R. Guérin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS Routing Mechanisms and OSPF Extensions. *Internet Draft, "draft-guerin-qos-routing-ospf-01.txt"*, Mar. 1997.

[10] R. Guérin and A. Orda. QoS-based Routing in Networks with Inaccurate Information. In *Proceedings of the IEEE INFOCOM '97*, 1997.

[11] Q. Ma and P. Steenkiste. Quality-of-Service Routing for Traffic with Performance Guarantees. In *Proceedings of IFIP Fifth International Workshop on Quality of Service*, 1997.

[12] A. Parekh. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks. *PhD dissertation, Massachusetts Institute of Technology*, 1992.

[13] C. Pronavalai, G. Chakraborty, and N. Shiratori. QoS Based Routing Algorithm in Integrated Services Packet Networks. In *Proceedings of the IEEE International Conference on Network Protocols (ICNP '97)*, 1997.

[14] S. Shenker, C. Partridge, and R. Guérin. Specification of Guaranteed Quality of Service. *Internet Draft, "draft-ietf-intserv-guaranteed-svc-08.txt"*, Feb. 1997.

[15] M. Shreedhar and G. Varghese. Efficient Fair Queueing using Deficit Round Robin. In *Proceedings of the ACM SIGCOMM '95*, 1995.

[16] D. Stiliadis and A. Varma. Latency-Rate Servers: A General Model for Analysis of Traffic Scheduling Algorithms. In *Proceedings of the IEEE INFOCOM '96*, 1996.

[17] Z. Wang and J. Crowcroft. Quality of Service Routing for Supporting Multimedia Applications. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, Sept. 1996.

[18] J. Wroclawski. Specification of the Controlled-Load Network Element Service. *Internet Draft, "draft-ietf-intserv-ctrl-load-svc-05.txt"*, May 1997.

[19] H. Zhang. Service Disciplines for Guaranteed Performance Service in Packet-switching Networks . In *Proceedings of the IEEE*, volume 83, pages 1374–1396, Oct. 1995.

[20] L. Zhang. Virtual Clock: A New Traffic Control Algorithm for Packet Switching. *ACM Transactions on Computer Systems*, 9(2), May 1991.

[21] W. Zhao and S. K. Tripathi. Routing Guaranteed Quality Of Service Connections In Integrated Services Packet Networks. In *Computer Science Technical Report CS-TR-3823, University of Maryland*, 1997.

A Proof of Buffer Bound of Guaranteed Service

We use the framework of *Latency-Rate Server* ($\mathcal{LR}$-Server) from [16]. Within any busy period $[0,\tau]$, the amount of traffic $W(0,t)$ serviced by a fluid-approximating server in an interval $[0,t]$ satisfies $W(0,t) \geq max\{0, R(t-\Theta)\}$, R is the service rate and Θ is the lag that the server trails behind the ideal fluid model, which is $C + D/R$.

In the single server case, the amount of backloged traffic in the server's buffer at time t within busy period $[0,\tau]$ is $A(0,t) - W(0,t)$, $A(0,t)$ is the amount of traffic arrived during $[0,t]$. We know that the incoming traffic is token bucket (b,r) constrained, $A(0,t) \leq b + rt$. Thus the backloged traffic at the server is

$$A(0,t) - W(0,t) \leq (b+rt) - max\{0, R(t-\Theta)\}.$$

Since $r \leq R$, the maximum backlog is achieved at $t = \Theta$. Therefore, we have the upper bound on the server backlog $b + r\Theta$.

Now we consider the multiple server case. In *Lemma 3* of [16], it is shown that by concatenating two $\mathcal{LR}$-Servers in series with respective latencies Θ_1 and Θ_2 we get a single $\mathcal{LR}$-Server with latency $\Theta_1 + \Theta_2$. Therefore concatenating a series of j $\mathcal{LR}$-Servers we get a $\mathcal{LR}$-Server with latency $\sum_{i=1}^{j} \Theta_i$, where Θ_i is the latency of the server i. Using the previous result on the traffic backlog on a single server, we have the upper bound on the traffic backlog in the composite server as $b + r\sum_{i=1}^{j} \Theta_i$, also an upper bound on the backlog of server j. By substituting Θ_i with the corresponding lag term $C_i + D_i/R$, we have the buffer bound in (2).

B Correctness Proof of Per-hop Constrained Shortest Path Algorithm

We show the following property: *any prefix of a shortest per-hop constrained path is itself a shortest per-hop constrained path.*

For any path P from source s to destination t, let the per-hop constraint on P's prefix p from s to u be $Cost(p) \leq C(u)$. Let P^* be a shortest per-hop constrained path from s to t. We prove the property by contradiction. Suppose a prefix P_1 of P^* from s to an intermediate node v is longer than another path P_1' from s to v satisfying the per-hop constraints, $Cost(P_1') < Cost(P_1)$. Let the rest of the path P^* from v to t be P_2. Concatenating P_1' and P_2 we get a path $P' = P_1'P_2$ which is shorter than P^*. By the fact that P_1' is a per-hop constrained path we know that any prefix p of P' that is also a prefix of P_1' satisfies the per-hop constraint. Any prefix p of P' from s to w that is not a prefix of P_1' can be broken up into two parts: P_1' from s to v and p' from v to w. By the facts that P_1' is shorter than P_1 and that P_1 is a per-hop constrained path we have $Cost(p) = Cost(P_1') + Cost(p') \leq Cost(P_1) + Cost(p') \leq C(w)$. Thus, P' satisfies the per-hop constraints and is shorter than P, contradicting the assumption that P is a shortest per-hop constrained path.

The above property is a necessary condition for the correctness of Dijkstra-like algorithms. The rest of the proof is identical to that of the Dijkstra's algorithm (pp. 529-530 of [3]).