Single Path Routing with Delay Considerations [1]

RAHUL SIMHA

Department of Computer Science

College of William & Mary

Williamsburg, VA 23185

email: `simha@cs.wm.edu`

BHAGIRATH NARAHARI

Department of EE & CS

George Washington University

Washington, DC 20052

email: `narahari@gwusun.gwu.edu`

Abstract: The importance of optimal routing arises in high traffic networks where the average delay across a link is sensitive to the assignment of routes to source-destination pairs. The current single path routing algorithms do not take into account the effects of routing multiple source-destination pairs over common links and, consequently, may not lead to a minimum delay allocation of routes. In this paper we formulate and study the problem of minimizing queueing delays by explicitly considering the impact of overlapping routes on links. The problem is shown to be $\mathcal{NP}$-Complete. Specifically, we prove that the problem of finding a feasible solution is $\mathcal{NP}$-Complete and thus no polynomial time approximation algorithms exist that provide solutions within a bounded distance from the optimum. Similar complexity results are shown when we consider the special case of a ring network and where it is desirable to minimize packet losses. We provide a polynomial time algorithm for constructing a minimum spanning tree within our routing model for the purpose of broadcasting. A new heuristic to minimize delays, the Sequential Minimum Costs algorithm is proposed for the case of a single source and multiple destinations.

Keywords: Routing, single path routing, shortest paths, problem complexity, spanning trees.

[1] This work was partially supported by the Office of Naval Research under grant N00014-87-K-0304

1 Introduction

In this paper we examine the routing problem for the case in which only one path is used between every source and destination pair in the network. Such single path routing allocations are appropriate in networks where a *virtual circuit* [3] is desired between the source and the destination and when the nodes do not possess the capability to resequence messages received on different paths or when such resequencing is expensive. We thus do not consider bifurcated routing [3] in which a node maintains several paths between itself and any destination and sends a certain fraction of the traffic over each path.

There is a large class of single path routing problem formulations based on computing the 'shortest' path between every pair of nodes [4, 14]. Some of these algorithms are centralized whereas others are designed to execute in a decentralized manner. In most formulations of this problem, 'shortest' refers to the sum of lengths or *costs* of individual links in a path. In this case, the cost of a link is assumed to remain *constant* with respect to the assignment of routes throughout the execution of the algorithm. Stated differently, the cost of a link is fixed prior to the execution of the algorithm and does not account for the delays caused by overlapping routes on a link. We will refer to shortest path algorithms of this type by the generic term *shortest path method* since we will be mainly concerned with the results produced by shortest path routing rather than the efficiency of the individual algorithms.

We examine single path routing problems with the primary intention of minimizing packet delays. In our formulation of the problem, the cost of a link has a different interpretation; specifically, the cost does *not* stay fixed because of delays caused by overlapping routes on a link. As in the bifurcated routing problem of [5] and the single path formulation in [2], we employ a queueing model to assess the delay produced across a link by a given routing assignment. Thus, the cost of a link depends on the routing assignment itself. For example, a routing assignment that assigns all message traffic to a single route will result in high queueing delays along that route. We model the stream of messages between a source and a destination as a stationary stochastic process. For example, this arrival process of messages may be taken to be a Poisson process [7]. Depending on the traffic routed over a given link, one can associate a similar stochastic process with the stream of messages arriving to that link from all contributing sources. Then, this arrival process together with the transmission characteristics of the link define a *queueing* process [7] on that link. We consider the problem of minimizing some functional defined over the queueing processes of all links in the network, such as the mean packet delay for the network [5]. In our formulation of single path routing, we do not focus on identifying the cost of a link in isolation but rather, we associate a cost with all the links in the network. However, we will loosely associate a 'cost' with an individual link while computing the overall cost for a routing assignment. Our cost embodies the increased average message delay caused by overlapping routes on a link.

Our main result is that the general problem of minimizing queueing delay costs over all links is $\mathcal{NP}$-Complete. We show that the problem of simply finding a *feasible* solution is $\mathcal{NP}$-Complete, where a feasible solution is one that satisfies link capacity constraints. One consequence of the latter result is that no approximation algorithms exist for the minimization problem unless $\mathcal{P} = \mathcal{NP}$. Similar complexity results are given for ring networks and the related problem of minimizing packet loss. We give a polynomial time algorithm for finding a spanning tree to minimize delays and describe a a heuristic algorithm for the minimum delay problem. Based on the intractability of the simple case when the network structure is restricted to a ring, we anticipate that it is unlikely that useful tractable instances can be found.

This paper is organized as follows. Section 2 describes our model of the network and defines the Single Path Routing problem. We do not attempt to provide a survey of routing techniques in this paper and, instead, refer the reader to [3, 4, 14]. In order to motivate our investigation of the complexity of the problem, in Section 3 we discuss the performance of the commonly used shortest path algorithm. We observe that the reason behind the potentially poor performance obtained using this algorithm arises from ignoring the consequences of overlapping routes to multiple destinations. Section 4 contains a formal investigation of the complexity of the problem, and Section 5 addresses the complexity of two related problems - the maximum link packet loss problem and the spanning tree problem. Finally, in Section 6 we outline a heuristic algorithm and Section 7 contains our conclusions.

2 System Model and Problem Formulation

We now describe our model of the routing problem and introduce the notation used throughout the remainder of the paper. We model a computer network as a graph $G = (V, E)$ of undirected edges (i.e. we assume bidirectional links) and define the following:

- $V = \{1, 2, ..., n\}$ - the set of nodes in the network.
- $E \subseteq V \times V$ - the set of links in the network.

- λ_{ij} - the traffic rate from node i to j, i.e. the rate of message traffic *initially entering the network* at node i which is destined for node j. One may interpret λ_{ij} as the sum total rate of traffic carried by all sessions from i to j.

- μ_l - service rate of link l, $l \in E$. We will follow the usual convention in which links are taken to be bidirectional with individual service rates; thus, for example, we might have $l_1 = (i,j)$, $l_2 = (j,i)$ and $\mu_{l_1} \neq \mu_{l_2}$. We will use the term link to denote a particular direction; above, l_1 and l_2 are different links. It is interesting to note that most of the complexity results (the results for a ring network being the exception) hold for other interpretations of graph edges as well: they hold for directed graphs and also when a common service rate is assumed for both directions.

- δ_{lij} - an integer valued variable, $\delta_{lij} \in \{0,1\}$, $i,j \in V$ such that $\delta_{lij} = 1$ if and only if λ_{ij} is routed through link l. Thus, an assignment of 0,1 values to each δ_{lij} is a routing assignment. We will use the notation δ to refer to a routing assignment.

- $L(i)$ - the set of links incident to node i i.e. $L(i) = \{(i,k)|(i,k) \in E\} \cup \{(k,i)|(k,i) \in E\}$. Thus $E = \bigcup_{i \in V} L(i)$.

- D_l - the steady state *expected* delay across link l computed using a queueing formula. For example, when the links are modeled as $M/M/1$ queues this expected value can be expressed as [7]:

$$D_l = \frac{1}{\mu_l - \lambda_l}$$

  where $\lambda_l = \sum_{i,j \in V} \delta_{lij}\lambda_{ij}$ is the rate of the total traffic carried by link l. As will be seen, our complexity results are independent of the queueing distributions.

- w_l - the weighted traffic on a link l (ratio of traffic on link l to total traffic):

$$w_l = \frac{\sum_{i,j \in V} \delta_{lij}\lambda_{ij}}{\sum_{i,j \in V} \lambda_{i,j}}$$

- R - The collection of relevant parameters defined above, such as μ_l, λ_{ij}, and the queueing model is denoted by R.

Furthermore, we define a set of constraints, $C = \{C1, C2, C3\}$ that will allow only single path routing assignments. Informally, the constraints $C1$ and $C2$ ensure that there is only one loop-free path assigned between every pair of nodes and $C3$ ensures that every link along that path can carry the message traffic assigned to it.

$C1$. There is only one path used to carry traffic between a given pair of nodes i and j, i.e. exactly one sequence of links $l_1, l_2, .., l_m, .., l_k$, where $l_1 = (i, i_1), .., l_m = (i_{m-1}, i_m), ..., l_k = (i_{k-1}, i_k)$, such that $\delta_{l_m ij} = 1$, $l_m = (i_{m-1}, i_m)$, for $1 \leq m \leq k$. Let p_{ij} denote the set containing these links, i.e. $p_{ij} = \{l_m | 1 \leq m \leq k\}$. Furthermore, $\delta_{lij} = 0$ unless defined as above.

$C2$. The single path carrying traffic from a source to destination is loop-free: nodes may not be repeated in a path. Note that loop-freedom is an important property required of any routing mechanism [3]. In single path routing, we observe that a loop may be 'short-circuited' at the repeated node with no loss of delay and hence, paths containing loops need not be considered.

$C3$. For each link l, the queue *stability* condition [7], $\mu_l > \sum_{ij \in V} \delta_{lij} \lambda_{ij}$, is satisfied. Note that this stability condition must be satisfied for the general $(G/G/1)$ single server queue [7].

We now formulate the problem of routing to minimize delays. We consider two performance metrics: (1) the weighted average of the expected delay across all links and (2) the maximum of the expected path delays over all source-destination pairs. In the case that the first metric [5] is used we denote the problem as the Minimum Average Delay problem; it can be stated more precisely as follows:

The Minimum Average Delay problem: Given a graph $G = (V, E)$ and a routing parameter set R, compute a routing assignment satisfying constraints C that minimizes $\sum_{l \in E} w_l D_l$.

Note that the weighted average above is the average message delay for the entire system [5]. We next consider the second performance metric: the maximum expected path delay over all source-destination pairs. This metric is based on the observation that minimizing the expected delay over all links (as above) could potentially lead to a situation where the expected delay for a particular source-destination pair is greatly compromised in order to decrease the expected delay over all messages. In order to ensure fairness to all destination nodes it is desirable to minimize the worst expected delay over all source-destination paths. Additionally, we note that in networks carrying real-time traffic (e.g. voice packets) it may be necessary to provide an upper bound on the expected path delays between every source and destination. In this case it is of interest to generate an allocation where all expected path delays are less than a given bound. We denote the routing problem with the latter metric as the Min-Max Delay problem. It can be stated more precisely as:

The Min-Max Delay problem: Given a graph $G = (V, E)$ and a routing parameter set R, compute a routing assignment satisfying constraints C that minimizes $\max_{i,j \in V} \left(\sum_{l \in p_{ij}} D_l \right)$.

As we will see later, the expected path delay to any given destination is affected by the assignment of paths made to every destination that shares a common link with the given path. Thus it is not possible to minimize the expected path delay to a particular destination without affecting the expected path delays to other destinations. This observation motivates the definition of the performance metrics above which explicitly consider the effects of overlapping routes. Note that we assume some means of obtaining the mean service rate of each link is available. For example, given the current traffic patterns and the delays across each link it is possible, using the $M/M/1$ queueing formula, to arrive at a reasonable estimate of the service rate of a link (a related idea is used in the Codex network [3]). In the section that follows we briefly examine the widely used shortest path algorithm, in the context of the model given above.

3 Shortest Path Routing

It is well-known that shortest path routing sometimes results in oscillations and unstable behavior [2]. In this section we briefly discuss a generic shortest path routing algorithm (such as the one described in [9]) in the context of our goal to minimize the previously defined network delay characteristics. We note that what is of interest here is not any particular implementation of the algorithm, such as Dijkstra's Algorithm or the Floyd Warshall algorithm [3, 4, 15] but rather the resulting allocation of routes computed by any shortest path algorithm. The algorithm may briefly be described as a two step procedure. An initial *feasible* set of routes is chosen and its delay characteristics are observed in operation. The steps indicated below are repeated by each node in the network:

1. Each node maintains an estimate of the average delay (called the *length* of the link) incurred by messages for every incident link. This might be, for example, the average delay over all messages sent over a link in a certain time period with the current assignment of routes. When it has been determined that the delay has changed significantly (see [9]), then all the nodes in the network are informed of the new estimate of delay across that link.

2. When a node receives update information about a change in link delay it executes a modified shortest path algorithm (such as Dijkstra's algorithm) to determine the new set of routes

between itself (source) and the other nodes in the network (destinations).

We note that the above scheme possesses several desirable features. The algorithm is simple and efficient (in the worst case an $O(n^2)$ algorithm) at each node. Various methods have been proposed [10, 16] that efficiently utilize pre-computed information to generate new shortest paths in a relatively short time.

Now consider the use of the shortest path method in the context of our routing problem, using our model. In step 1, of the procedure outlined above, the delays of each link are computed based on some initial routing assignment. This assignment defines a queueing process on each link, leading to a steady-state mean delay for each link. Thus, packets traversing a particular link experience an average delay predicted by the queueing formula. This corresponds to (and, if the queueing-theoretic formula is accurate, closely approximates) the average delay that would be measured across the given link in a real network. Since we are using a model to evaluate the shortest path algorithm, we can use this predicted delay as the estimate for the delay across a link in the above algorithm. Thus, given these predicted delays (and only these predicted delays) one executes the shortest path algorithm to obtain the next routing assignment, which in turn will result in new link delays and the next iteration of the algorithm. The allocation generated by the shortest path method is a result of iterating this procedure.

During each iteration, it is seen (from the definition of the shortest path algorithms) that the delay resulting from the *cumulative* placement of routes does not manifest itself in the shortest path algorithm. In other words, the delays of each link is assumed to be constant (more precisely, it is the delay computed *before* we invoke the shortest path algorithm at each iteration) and hence does not consider the actual delays arising from overlapping routes. Due to this feature, the shortest path method does not perform well in the context of routing to minimize delays. We conjecture that previously observed oscillations when shortest path routing is used is due to ignoring the effects of superimposing routes within an iteration. The routing problem defined in Section 2 seeks to minimize delay by *explicitly considering the effects of overlapping routes.* In the section that follows we study the complexity of this problem.

4 Optimal Single-Path Routing

In the previous section we demonstrated that the shortest path formulation is liable to result in poor routes that are caused by ignoring the consequences of overlapping paths. To consider these effects we adopt the routing formulation of Section 2 in which single paths are required from source to destination and the cost of a link reflects the cumulative effect of superimposing routes. In a network, several or all of the nodes operate as sources and transmit messages to multiple destinations. The delays measured on a link in the formulation of the previous section should include the delay caused not only by the overlap of routes from a single source but also from the multiplicity of paths used by several sources. A network wide optimal assignment of routes must consider the combined effects of such superimpositions. In this section we discuss the complexity of the general problem. This problem will be shown to be $\mathcal{NP}$-Complete even if there is only one source and thus considerable computation might be required to locate an optimal assignment of routes. In fact, we prove that no polynomial time approximation algorithm can be found unless $\mathcal{P} = \mathcal{NP}$.

4.1 Complexity of Single Path Routing

The decision versions of the Minimum Average Delay problem and the Min-Max Delay problems can be stated as follows:

The Minimum Average Delay (Single Source) Problem: Given a graph $G = (V, E)$, a real number D, a specified source s and a routing parameter set R, is there a routing assignment that satisfies $C1, C2$ and $C3$ such that

$$\sum_{l \in E} w_l D_l < D?$$

The Min-Max Delay (Single Source) Problem: Given a graph $G = (V, E)$, a real number D, a specified source s and a routing parameter set R, is there a routing assignment that satisfies $C1, C2$ and $C3$ such that

$$\max_{i,j \in V} \left(\sum_{l \in p_{ij}} D_l \right) < D?$$

Any feasible routing assignment for the Minimum Average Delay or the Min-Max Delay problems must satisfy the constraints $C1, C2$ and $C3$. Therefore, we first address the complexity of an algorithm that would find an assignment satisfying the constraints C. The feasible routing assignment problem can be stated as follows:

The Feasible Routing Assignment Problem: Given a graph $G = (V, E)$ and a routing parameter set R is there a routing assignment δ which satisfies C?

In what follows, this problem will be shown to be $\mathcal{NP}$-Complete by a reduction involving the well-known Bin Packing problem [6]. This result implies that the problem of finding an approximation algorithm is also very difficult and polynomial time approximation algorithms are unlikely to exist.

THEOREM 1: *The Feasible Routing Assignment problem is $\mathcal{NP}$-Complete even if there is only a single source.*

PROOF: We examine the single source version of the problem. Note that the requirement of a single source, with the source specified as s, implies that for all nodes $i \neq s$ and $j \in V$, $\lambda_{ij} = 0$.

We first note that the problem is in $\mathcal{NP}$ because there exists a polynomially bounded certificate [12] function. Given a routing assignment, we can check for feasibility in $O(n^3)$ steps where each step consists of evaluating condition $C3$ for each link, an $O(n)$ computation of the sum $\sum_{j \in V} \delta_{lsj} \lambda_{sj}$ and the number of links is $O(n^2)$. Hence the checking can be done in polynomial time. We prove it is $\mathcal{NP}$-Complete by showing a reduction from the Bin Packing Problem, which is known to be $\mathcal{NP}$-Complete [6]. The bin packing problem is defined below.

The Bin Packing Problem: Given a finite set U of items, $U = \{u_1, ..., u_M\}$, and sizes $s(u_i) \in \mathbb{Z}^+$ for each $u_i \in U$, a positive bin capacity $B \in \mathbb{Z}^+$ and a positive integer K, is there a partition of U into disjoint sets $U_1, ..., U_K$ such that the sum of the sizes of the items in each U_i is B or less?

Given an instance of Bin Packing with $U = \{u_1, ..., u_M\}$ and parameters K and B, we construct a graph $G_1 = (V_1, E_1)$ as shown in figure 1 where

$$\begin{aligned} V_1 &= \{0, 1, ..., K, K+1, ..., K+M\} \\ E_1 &= \cup_{i,j} \{\{0, i\} \cup \{i, j\}\}, \qquad 1 \leq i \leq K,\ K+1 \leq j \leq K+M \end{aligned}$$

The node 0 is defined to be the source s in our routing problem, and we add edges between the source $s = 0$ and the vertex set $\{1, 2, ..., K\}$. In addition, we define a complete bipartite graph between the vertex sets $\{1, 2, ..., K\}$ and $\{K+1, K+2, ..., K+M\}$. From the construction

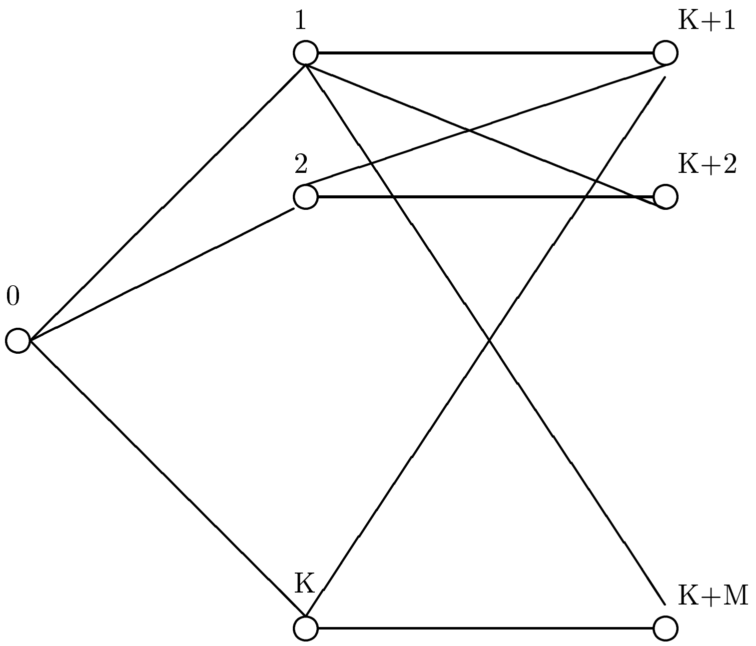


Figure 1: Construction of G_1

of graph G_1 it can be seen that any simple acyclic path from node 0 to any node in $\{K+1, ..., K+M\}$ must use exactly one edge in the set $\{(0,i) \mid 1 \leq i \leq K\}$. We now define the parameter set R_1 as follows:

$$\begin{aligned}
\lambda_{0i} &= 0, \quad 1 \leq i \leq K \\
\lambda_{0j} &= s(u_{j-K}), \quad K+1 \leq j \leq K+M \\
\mu_{0i} &= B + \frac{1}{2}, \quad 1 \leq i \leq K \\
\mu_{ij} &= 1 + \sum_{m=1}^{M} s(u_m) \quad \text{for all other links}
\end{aligned}$$

From the parameter set, we observe that the messages are to be sent between the source node 0 and destinations i, where $K+1 \leq i \leq K+M$. In addition, we set the service rate of each link in the set $\{(i,j) \mid 1 \leq i \leq K, \mid K+1 \leq j \leq K+M\}$ to be at least the sum of all traffic rates.

We now argue that the Bin Packing instance has a solution if and only if G_1 with parameter set R_1 has a feasible routing assignment. To obtain the reduction from the bin packing problem, we identify the total traffic on edge $(0,i)$, where $1 \leq i \leq K$, with the total size of the items placed in bin i in the bin packing problem.

We first show that for every feasible routing assignment δ there corresponds a solution to the bin packing problem. Given a feasible routing assignment δ, define the partition of U into K sets, $U_1, ..., U_K$, such that $u_j \in U_m$ if $\delta_{l_m 0j} = 1$, (i.e., if link $l_m = (0,m)$, $1 \leq m \leq K$, carries λ_{0j} where $K+1 \leq j \leq K+M$, then place item u_j in bin m). Note that from the bounds for j and m there are K partitions and M items. Any feasible routing assignment δ must satisfy the constraints C. If this is the case then from constraints $C1$ and $C2$ it follows that each λ_{0j}, $K+1 \leq j \leq K+M$, is sent on exactly one of the links in $(0,i)$, $1 \leq i \leq K$, and on exactly one link (i,j), $K+1 \leq j \leq K+M$. In addition, the assignment must route all messages λ_{0j}. Thus each U_i is a partition of the set U. From the definition of each U_i, the sum of the sizes of items in partition U_i, i.e., $\sum_{u_j \in U_i} s(u_j)$, is the sum total traffic on link $l = (0,i)$. From constraint $C3$, the sum total traffic on each $l = (0,i)$ link, $1 \leq i \leq K$, is less than the service rate $\mu_l = B + \frac{1}{2}$ on that link, and this can be stated as:

$$\sum_{K+1 \leq j \leq K+M} \lambda_{0j} \delta_{l0j} < \mu_l = B + \frac{1}{2}, \qquad \text{for } l = (0,i),\ 1 \leq i \leq K$$

Therefore in the Bin Packing problem, the sum of sizes of items in a bin U_i is less than $B + \frac{1}{2}$ and since sizes are integers the sum is less than or equal to B, the common bin capacity. It follows that a feasible routing assignment δ results in a solution for the Bin Packing problem. Since the Bin Packing problem requires that the sum of item sizes only be B *or less*, we use the fact that the item sizes are integers and hence, if the sum of item sizes is strictly less than $B + \frac{1}{2}$, it is less than or equal to B.

To complete the proof, we show that every solution for the Bin Packing instance results in a feasible routing assignment. Let l_m denote link $(0,m)$ and let $j = K+i$. If $U_1, ..., U_K$ is the partition of the M items in U then, define $\delta_{l_m 0j} = 1$ if and only if $u_j \in U_m$ where $1 \leq m \leq K$, $1 \leq j \leq M$. Since the sum of the sizes of items in each partition is less than or equal to B, it follows that the total traffic on each link $(0,m)$, $1 \leq m \leq K$ is strictly less than $\mu_{l_m} = B + \frac{1}{2}$. To route the messages to the final destinations set $\delta_{(i,j)0j} = 1$ if $\delta_{(0,i)0j} = 1$ for $1 \leq i \leq K$, and $K+1 \leq j \leq M+K$, i.e., if link $(0,i)$ is on the path, for sending messages from 0 to j, then place (i,j) on the path. This will satisfy constraint $C3$ since the service rates of all links (i,j), $1 \leq i \leq K$, $K+1 \leq j \leq K+M$, are greater than the total traffic rate. Since $U_1, ..., U_K$

is a partition of U, the routing assignment δ satisfies constraints $C1$ and $C2$. Since the total traffic on each link in G_1 is at most the service rate, constraint $C3$ is satisfied. Thus the assignment δ corresponding to a solution of the bin packing problem is a feasible assignment. Thus, if there is no routing assignment to satisfy C, there is no partition to fit the bins $\square$.

We have shown that the Feasible Routing problem is $\mathcal{NP}$-Complete. There is, consequently, every indication that searching for a feasible solution can be a computationally difficult task. We note that the multiple-source extension of the problem follows from Theorem 1. Similar arguments show that the multiple source version is in $\mathcal{NP}$: a polynomial certificate function can be easily constructed from the certificate function of the single source problem - again, an $O(n^3)$ computation. To see that it is $\mathcal{NP}$-Complete, observe that we may trivially reduce the single-source problem by setting message rates for any $(n-1)$ nodes (in n node network) to zero.

We now turn to the routing problems studied in this paper, that of minimizing average packet delay and the min-max delay. We first investigate the possibility of a polynomial time ϵ-approximation algorithm [12] producing solutions within a bounded distance from the optimum. If A^* is the cost of an optimum solution to a minimization problem, an ϵ-approximation algorithm is an algorithm that produces a feasible solution of cost A such that $A \leq (1+\epsilon)A^*$. Since every approximate assignment must at least be feasible, an approximation algorithm would be able to decide feasibility. Therefore the existence of a polynomial time approximation algorithm implies the existence of a polynomial algorithm for the feasible routing problem. Note that once the approximation algorithm returns a solution, the assignment can be checked for feasibility in polynomial-time using the certificate function in Theorem 1. Since both routing problems must satisfy the constraints, the following theorems follow from the above discussion.

THEOREM 2: *The existence of a polynomial-time approximation algorithm for the Minimum Average Delay problem implies $\mathcal{P} = \mathcal{NP}$ even when there is only a single source.*

THEOREM 3: *The existence of a polynomial-time approximation algorithm for the Min-Max Delay Problem implies $\mathcal{P} = \mathcal{NP}$ even when there is only a single source.*

From Theorems 2 and 3, the decision versions [12] of the Minimum Average Delay and the Min-Max Delay problems are seen to be $\mathcal{NP}$-Complete. Given the appropriate queueing formula for each link, it is easy to compute the delay in an $O(n)$ computation. Thus, an $O(n^3)$ certificate function may be constructed to check whether a given assignment is feasible. This proves that the problems are in $\mathcal{NP}$. We observe that results for multiple source versions follow directly from the single source problems above. It must be noted that our complexity results do not depend on the specific distributions associated with the queueing process on each link, since $C3$ must be satisfied for $G/G/1$ queues [7].

5 Related Routing Problems

In this section we focus on the complexity of certain routing problems related to single path routing. In particular, we address the special case of ring networks, a problem involving the minimization of packet loss and one that addresses the construction of a spanning tree. We first examine tree and ring networks.

5.1 Routing on Trees and Rings

Tree topologies are found in star networks [17] and rings are the basis for several important local area networks [8]. We present results in the context of a single source with the understanding that the results may be extended in a straightforward manner to multiple sources.

When the network graph $G = (V, E)$ is a tree the problem is seen to be trivially solvable. Recall that in a tree there is a only single acyclic path between any pair of nodes. Thus, there is no choice of routes and one need only check for feasibility which can be done $O(n)$ time (since there are $O(n)$ edges in the tree). We have a similar degenerate case for unidirectional rings, in which the ring direction (messages travel only along this direction) must be fixed. In this case, there is only one single path between any pair of nodes in a unidirectional ring once the direction is fixed. Therefore we can compute respective costs for each direction and pick the direction with the lower cost (if it is feasible).

We now turn to bidirectional rings (see [11] for an example) and, despite the simplicity of the network, the problem is shown to be $\mathcal{NP}$-complete even if there is only a single source.

THEOREM 4: *The Feasible Routing Assignment problem is $\mathcal{NP}$-Complete for bidirectional rings even when there is only a single source.*

PROOF: Recall that the requirement of a single source, with the source specified as s, implies that for all nodes $i \neq s$ and $j \in V$, $\lambda_{ij} = 0$. From the arguments in Theorem 1 the problem is in $\mathcal{NP}$. We now show that it is $\mathcal{NP}$-complete by showing a reduction from the set partition

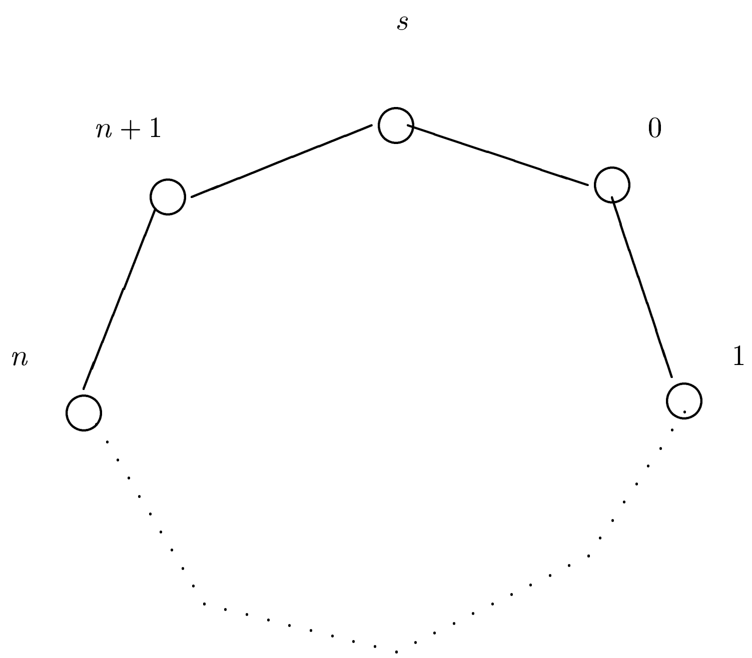


Figure 2: Ring constructed from partition problem

problem [6], following an argument similar to one for multiprocessor scheduling (page 65, [6]). The partition problem is defined below.

Partition: Given a finite set A and sizes $s(a) \in \mathbb{Z}^+$ for each $a \in A$, is there a subset $B \subset A$ such that

$$\sum_{a \in B} s(a) = \sum_{a \in A-B} s(a) \ ?$$

Given an instance of partition, with set $A = \{1, \ldots, n\}$ and sizes $s(1), \ldots, s(n)$, we construct a bidirectional ring, shown above in Figure 2, with additional nodes $s, 0$ and $n+1$. The node s is the single source, and the parameter set R_1 is defined as follows:

$$\begin{aligned}
\lambda_{si} &= s(i), \quad 1 \leq i \leq n \\
\lambda_{s0} &= \lambda_{s(n+1)} = 0 \\
\mu_{s0} &= \mu_{s(n+1)} = \frac{1}{2} \sum_{i \in A} s(i) = \mu. \\
\mu_{ij} &= 1 + 2\mu_{s0} \quad \text{for all other links}
\end{aligned}$$

We now argue that the partition instance has a solution if and only if the bidirectional ring with parameter set R_1 has a feasible routing assignment.

We first show that for every feasible routing assignment there is a corresponding solution to the partition problem. Observe that in the ring all traffic from source s must use links $(s,0)$ or $(s,n+1)$ or both. Let B denote the set of destinations whose traffic is routed across $(s,0)$. Then, because of acylicity, traffic to destinations in $A-B$ is sent across the other outgoing link, $(s,n+1)$. Now, for a feasible routing assignment, the traffic must be less than the capacity on each of the two links $(s,0)$ and $(s,n+1)$. Since their capacities are the same we may write this as:

$$\max\left(\sum_{i\in B}\lambda_{si}, \sum_{i\in A-B}\lambda_{si}\right) \leq \mu.$$

Or equivalently,

$$\max\left(\sum_{i\in B}s(i), \sum_{i\in A-B}s(i)\right) \leq \frac{1}{2}\sum_{i\in A}s(i).$$

Since the total size of the items is 2μ, this implies that

$$\sum_{i\in B}s(i) = \sum_{i\in A-B}s(i),$$

In other words B is the required partition. So a feasible routing assignment results in a partition.

Now assume that there is a subset B which is a solution to the partition problem. Given the set B we can construct an acyclic feasible routing assignment. If $a_i \in B$ then route traffic λ_{si} across link $(s,0)$; If $a_i \in A-B$ then route traffic λ_{si} across link $(s,n+1)$. Since the total size of items in B and $A-B$ are equal, it follows that the above assignment is feasible. To complete the routing assignment for each $a_i \in B$, we assign link (k,m) where $0 \leq k \leq i-1$ and $1 \leq m \leq i$ to carry the traffic λ_{si}. Since all links in this path have a capacity greater than the the total traffic the assignment is feasible. Thus we have a feasible routing assignment corresponding to the solution for the partition problem $\square$.

Note that, as in Theorem 1, we have taken the capacity constraints to be '$\leq$' rather than '$<$'. This is easily fixed as before by choosing a small enough real number and adding it to the capacity. Finally, the $\mathcal{NP}$-completeness of the problem for bidirectional rings implies that the routing problem remains $\mathcal{NP}$-complete for *planar* graphs.

5.2 Link Packet Loss Problem

For networks allowing packet loss, we may model each link as a $G/G/1/K$ queue (K is the size of the finite buffer [7]) and require that the steady state packet loss on each link be bounded. The objective is to find single path routing assignments that bound the packet loss on each link. Assuming that a formula for packet loss F_l is available in terms of the routing parameters, we may formulate the decision version of this problem as follows.

The Maximum Link Packet Loss (Single Source) Problem: Given a graph $G = (V, E)$, a specified source s, a routing parameter set R and a packet loss limit B_l for each link l, is there a routing assignment satisfying constraints $C1$,$C2$ and $\forall l \in E : F_l < B_l$?

We now show that for queueing processes (such as the $M/M/1/K$ queue [7]) in which F_l is an increasing and continuous function of λ_l, the problem is $\mathcal{NP}$-Complete. An extension to a multiple source version follows easily.

THEOREM 5: *The Maximum Link Packet Loss (Single Source) Problem is $\mathcal{NP}$-Complete.*

PROOF: The constraints $C1$ and $C2$ place the single path restriction, and the constraint $\forall l \in E : F_l < B_l$ states that packet loss is bounded on each link by the loss limit B_l. Note that the parameter set R is the same as for the delay problem and the only difference is that constraint $C3$ is replaced by a packet loss bound.

That the problem is in $\mathcal{NP}$ is argued in much the same manner as in the problems of the previous section - checking to see if the packet loss condition at each link is satisfied can be done in polynomial time.

To show that it is $\mathcal{NP}$-Complete we show a straightforward reduction to the general single path routing problem with all three constraints. To achieve this we will define a new parameter set R' and the same set of constraints $C1, C2$ and $C3$.

Since $F_l(\lambda_l)$ is increasing in λ_l, we define the inverse of F_l as

$$F^{-1}(a) = \inf_x \{x : F(x) > a\}$$

For each link $l \in E$, define $\mu'_l = F^{-1}(B_l)$.

From the definition of μ'_l and the continuity of F_l, it follows that $F_l(\lambda_l) < B_l$ if and only if $\lambda_l < \mu'_l$.

To complete the proof, we construct a new parameter set R' with arrival rates λ_{ij} from R above and the fictitious services rates μ'_l. The constraint $F_l(\lambda_l) < B_l$ is satisfied if and only if the constraint $\lambda_l < \mu'_l$ is satisfied. But $\lambda_l < \mu'_l$ is the same as constraint $C3$ (recall that $\lambda_l = \sum_{i,j \in V} \delta_{lij} \lambda_{ij}$). Therefore, a routing assignment for the Maximum Link Packet Loss problem is a feasible assignment if and only the assignment is a feasible assignment for the single-path routing problem (in Theorem 1) with $\mu_l = \mu'_l$. From the result of Theorem 2 we conclude that the Maximum Link Packet loss problem is $\mathcal{NP}$-Complete $\square$.

5.3 The Spanning Tree Problem

We now turn to a spanning tree problem, one that is the analog of the Minimum Spanning Tree problem [12] when delays due to the assignment of the spanning tree are considered in evaluating the tree. As with the shortest path formulation, the traditional approach to constructing spanning trees takes the cost of each link to be fixed prior to the execution of the algorithm. In our formulation we consider the effects on queueing delays caused by constraining message traffic to flow within the spanning tree.

A spanning tree is often cited as a suitable structure to implement broadcasts [3, 14]. We consider the case when we need to construct a spanning tree to implement multiple broadcasts. We will assume that each node i broadcasts messages to all other nodes at a rate λ_i. Thus, in the formulation of the routing problem in previous sections, $\lambda_{ij} = \lambda_i$. Then, a spanning tree is simply a single path routing assignment such that the links in the routing assignment form a spanning tree. It is seen that this additional restriction makes the problem easier to solve and we give a polynomial time algorithm to find a spanning tree if it exists. We first note that the model undergoes some subtle changes due to the fact that we are broadcasting messages. Since each node sends a given message to *all* other nodes, we assume that a node i sends only one copy of its message (with rate λ_i), and that intermediate nodes work under a "copy and forward" model. For example, if node A is connected to B and B is connected to C then the link (A, B) only carries one copy of A's message (at a rate λ_A) and after B receives the message it forwards it to node C. Thus, the traffic across the link (A, B) will be

λ_A and not $2\lambda_A$ (when distinct messages were sent the traffic on link (A, B) was defined as $\lambda_{AB} + \lambda_{AC}$). The cost of a spanning tree is defined as the Average Delay, i.e., $\sum_{l \in E} w_l D_l$. Our objective is to find the Minimum Average Delay spanning tree satisfying the constraints C. An assignment δ is called a spanning tree assignment if the set of nodes V_T and edges E_T used in the assignment form a spanning tree of the graph G.

The Minimum Average Delay Spanning Tree Problem: Given a graph $G = (V, E)$ and a parameter set R, compute a spanning tree assignment δ which satisfies C and minimizes $\sum_{l \in E_T} w_l D_l$.

We now show that for delay functions (such as the one for the $M/M/1$ queue in section 2) which are continuous and decreasing in μ_l that a minimum spanning tree assignment can be found in polynomial time.

THEOREM 6: *The Minimum Delay Spanning Tree Problem can be solved in polynomial time.*

PROOF: We shall give a polynomial time algorithm to construct the minimum spanning tree for the network if a spanning tree exists. If a spanning tree does not exist then the algorithm indicates that an assignment is not feasible. The algorithm consists of three steps and returns the minimum delay spanning tree when one exists or returns FALSE when there is no spanning tree assignment.

Input: Graph $G = (V, E)$ with parameter set R

Output: Spanning Tree $T = (V, E_T)$ which minimizes $\sum_{l \in E_T} w_l D_l$

Step 1: Construct the subgraph $G_1 = (V, E_1)$ where $E_1 \subseteq E$ such that $l \in E_1$ *if and only if* $\mu_l > \sum_{i \in V} \lambda_i$.

Step 2: Determine if graph G_1 is connected. If G_1 is connected then go to step 3 else return FALSE.

Step 3: For each $l \in E_1$ define the cost of the link as $c(l) = -\mu_l$. Find the Minimum Spanning Tree $T = (V, E_T)$ of $G_1 = (V, E_1)$ using any Minimum Spanning Tree algorithm [12] (with cost of each link taken to be $c(l)$). Return T.

We prove the correctness of the above algorithm. Any spanning tree for G must have exactly one path between any pair of nodes and furthermore the removal of any link must disconnect the graph. Next, for any link l in a spanning tree let the removal of l disconnect the graph into two components C_1 and C_2. Since every node in C_1 broadcasts a message that must be sent to all nodes in C_2, the total traffic rate from C_1 to C_2 on link l is equal to $\sum_{i \in C_1} \lambda_i$. Similarly, the total traffic rate from C_2 to C_1 on link l is $\sum_{i \in C_2} \lambda_i$. Therefore the total traffic rate on any link l is $\sum_{i \in C_1} \lambda_i + \sum_{j \in C_2} \lambda_j = \sum_{i \in V} \lambda_i$. In other words, all messages must pass through link l, for all links in the tree. To support the traffic rate each link l must have its service rate at least as large as the total traffic rate, i.e., $\mu_l > \sum_{i \in V} \lambda_i$. All links with service rates less than the total traffic rate cannot be assigned in a spanning tree assignment, and thus we remove these edges from the graph to get the resultant graph G_1. This process is accomplished by step 1. Since a spanning tree must connect all pairs of nodes, it follows that there exists a spanning tree assignment for G if and only if the graph G_1 is connected. If G_1 is not connected then there is no spanning tree assignment and the algorithm, in step 2, return false. If the graph is connected we can always find a spanning tree and thus there is a spanning tree assignment.

We now prove that the spanning tree constructed in Step 3 is the minimum cost spanning tree, where the cost is defined to be the Minimum Average Delay. Since the total traffic rate on any link is equal to $\lambda_{sum} = \sum_{i \in V} \lambda_i$, the steady state delay on a link is $D_l = \frac{1}{\mu_l - \lambda_{sum}}$. The weighted traffic on each link is now $w_l = \frac{\lambda_{sum}}{\sum_{i \in V} \lambda_i} = 1$. The Minimum Average Delay of a spanning tree assignment is $\sum_{l \in E_T} w_l D_l = \sum_{l \in E_T} D_l$. Since λ_{sum} is the same for all links, it follows (since $D_l = \frac{1}{\mu_l - \lambda_{sum}}$) that for any two links k and l, $D_k < D_l$ if and only if $\mu_k > \mu_l$. Consequently, we have a Minimum Average Delay spanning tree when $\sum_{l \in E_T} \mu_l$ is maximized. To complete the proof note that maximizing $\sum_{l \in E_T} \mu_l$ is equivalent to minimizing $\sum_{l \in E_T} (-\mu_l)$. By defining cost of each link as $c(l) = -\mu_l$ and finding the minimum spanning tree, as is done in Step 3, the algorithm finds the spanning tree with Minimum Average Delay. This completes the proof of correctness. We note that algorithm computes the Minimum Average Delay for all delay functions which satisfy the relation $D_k < D_l$ if and only if $\mu_k > \mu_l$, i.e., for delay functions which are continuous and decreasing in μ_l $\quad \square$.

The complexity of the above algorithm is seen to be $O(\mid V \mid^2)$. The first step takes $O(E)$ time since all links must be examined. In practice steps 2 and 3 can be combined into one step. To do this we find the minimum spanning tree $T = (V_T, E_T)$ of G_1 (i.e., do step 3) and check if it connects all vertices, i.e., check if $V_T = V$. If it is not connected than we return False, otherwise, T

provides the minimum cost spanning tree assignment. The minimum spanning tree can be found in $O(|V|^2)$ steps [1], and hence the complexity of our algorithm is $O(|V|^2)$.

6 Heuristic Algorithms

In the previous section we have shown that our problem of optimally allocating routes, for either of the two objectives, is $\mathcal{NP}$-Complete and that it is very unlikely that even approximation algorithms will be found. We now focus our attention on heuristic algorithms, which while not guaranteed to produce even a feasible solution, execute in polynomial-time and at least attempt to consider the effects of overlapping routes. We present a heuristic method that can be applied for both performance metrics (Minimum Average Delay and Min-Max Delay) in the case of a single source and multiple destinations, and discuss its extension to multiple sources.

6.1 Sequential Minimum Cost Paths: Single Source

Let s denote the single source and A denote an ordered sequence of destinations $(a_1, ..., a_{n-1})$ (later in this section we describe various means of computing this sequence). Furthermore let $\lambda_l^{(i)}$ be the traffic assigned to link l at iteration i of the algorithm, based on paths allocated to destinations $a_1, ..., a_{i-1}$ i.e., $\lambda_l^{(i)} = \sum_{k=1}^{i-1} \delta_{lsk}\lambda_{sk}$. Similarly, let $w_l^{(i)}$ be the weighted traffic on link l at iteration i of the algorithm, i.e., $w_l^{(i)} = \frac{\lambda_l^{(i)}}{\sum_{i,j\in V} \lambda_{i,j}}$. Let F denote the objective function.

When the objective (i.e., performance metric) is the Min-Max delay, F is defined (at each iteration i) as: $F = D_l = \frac{1}{\mu_l - \left(\lambda_l^{(i)} + \lambda_{si}\right)}$.

When the objective is the Minimum Average Delay, F is defined (at each iteration i) as: $F = \sum_{l\in E} w_l D_l$ where $D_l = \frac{w_l^{(i)}}{\mu_l - \left(\lambda_l^{(i)} + \lambda_{si}\right)}$.

Intuitively, the algorithm employs a minimum cost first search based on the additional delay introduced by links that are candidates for the path to a given destination. Each destination is considered in the order determined by the sequence A. At the i^{th} iteration the route for the i^{th} destination is determined based on the routes fixed for the previous $(i-1)$ nodes computed in the earlier iterations. Depending on which definition of F is selected (note that only one definition of F is used for all iterations of the algorithm), every iteration uses the search algorithm to locate the

(1) minimum maximum delay path or (2) minimum average delay path for a particular destination in the sequence A. The algorithm shown below works for both objectives.

The Sequential Minimum Costs algorithm:

Compute the sequence A using one of the methods
  outlined below in this section;
for each $i \in A$
  begin
    $V_1 := \{s\}$; $V_2 := V - \{s\}$;
    repeat
      Pick a $v \in V_2$ such that $l = (u, v) \in E$ for some
        $u \in V_1$ and such that F is
        minimized over all $v \in V_2$;
      $V_1 := V_1 \cup \{v\}$;
      $\delta_{lsi} := 1$;
    until $v = a_i$
  end

The sequence A may be computed in a number of different manners with each sequence representing a different heuristic. Below we suggest some orderings which, while maintaining the efficiency of the above Sequential Minimum Cost Algorithm, may serve to improve (i.e. minimize) the overall maximum path delay:

- *Minimum Hop Ordering.* Sort the destinations in order of increasing hop-length. Thus, a_1 is a destination one hop away from the source s, whereas a_{n-1} is the destination that is farthest away from s in terms of hop-length.
- *Maximum Hop Ordering.* Sort the destinations in order of decreasing hop-lengths.
- *Maximum Traffic Ordering.* Sort the destinations in order of decreasing message traffic rates.

The Hop Ordering heuristics sort the destinations based on the distances measured as the number of links, thus ignoring source-destination traffic rate information. When the algorithm uses the Minimum Hop Ordering heuristic, destinations that are closer are assigned links (and paths) earlier than those that are further away. The converse is true when the algorithm uses the

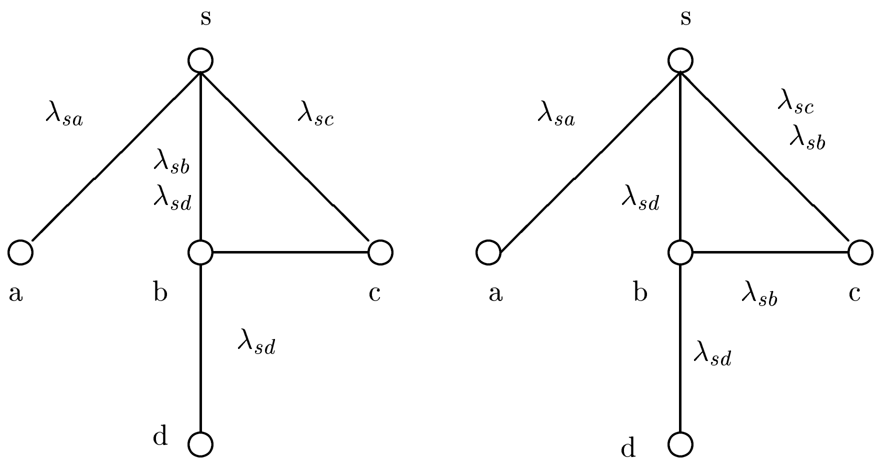


Figure 3: Minimum Hop Ordering - an example

Maximum Hop ordering heuristic. The Maximum Traffic Ordering, on the other hand, sorts the destinations based on message traffic rates. It attempts to first assign links (and paths) to the destinations with heavy message traffic, following which it attempts to spread the low message traffic destinations among the available links. It attempts to place heavy traffic on disjoint paths thus achieving a 'load-balanced' [13] system leading to a lower average (or maximum path) delay. In a situation when the message traffic rates, for all destinations, are similar (or identical) then an ordering based on hops (i.e., distances) may be a suitable heuristic. On the other hand when there is a large variance in the message traffic rates, using the Maximum Traffic Ordering heuristic may be more appropriate due to the load balancing effect.

We give an example in which the optimal allocation is not generated by a *Minimum Hop* ordering. The service rate of each link is taken to be $\mu_l = 2.0$ and the message rates from the source s are taken to be $\lambda_{sa} = 0.5, \lambda_{sb} = 0.5, \lambda_{sc} = 0.5$ and $\lambda_{sd} = 1$. The maximum mean path delay is computed to be 3.0. The optimum assignment of paths is shown in figure 3(b) for which maximum mean path delay is 2.0. Thus, the algorithm is not guaranteed to produce the optimal allocation of routes to destinations.

We note that the worst case complexity of the above heuristic is $O(n^3)$. For each node in any ordering, in the worst case each edge may have to be evaluated. There are at most $O(n^2)$ edges in a graph and since there are $(n-1)$ nodes in the ordering A, the heuristic generates a solution in $O(n^3)$ steps.

For future work, it would be important to experimentally quantify the relative performance of the proposed heuristics. Evaluation of the heuristic is likely to be a difficult task, involving a resort to large simulations over random graphs. We demonstrate this difficulty by showing how a relatively weak bound is affected by graph structure. Let

$$\mu_{min} = \min_{l \in E} \mu_l$$

and

$$\lambda_{sum} = \sum_{i,j \in V} \lambda_{ij}.$$

Then, a simple upper bound on the cost of any assignment is provided by assuming the worst: each link carries all the traffic. Denoting this bound by B,

$$B = \frac{|E|}{\mu_{min} - \lambda_{sum}}$$

Clearly, one may construct networks for which this upper bound grossly overestimates the optimal solution. What is interesting is that the bound is *achieved* by other networks. For example, a simple one-link network with a single source and destination trivially achieves the upper bound. These observations suggest that the effectiveness of a bound on the optimal solution depends on its ability to account for the structure of the underlying graph. At the same time, there will be a dependence on the parameter set R for a given instance. Thus, it is not clear whether averages taken across random graphs is meaningful unless one addresses the questions raised by averaging results across various parameter sets.

6.2 Multiple Sources

To use the above heuristic for multiple sources, we suggest the following possibilities, both requiring synchronization between the multiple sources. Initially, a particular heuristic among the single source versions is chosen, for example, the *Maximum Traffic Ordering.* As in the single source method, each source-destination route is assigned one at a time. A particular multiple source heuristic results from the choice made in serializing the routes from different sources. We suggest the following possibilities:

- *Rotation.* Each source assigns one route in turn at a time using the minimum sequential paths heuristic. In this case, each source may use any ordering it chooses. In order for a proper distributed execution of this algorithm, a token may be rotated among the nodes to ensure that only one route is assigned at a time. Of course, once a route is assigned, all other nodes must be informed of the choice.

- *Global Ordering.* In this variation, source destination pairs are ordered globally according to traffic, regardless of source, and the minimum sequential paths heuristic is used to assign routes in global order. Thus, the second route assigned would be the one with the second largest traffic even if it was from the same source as the first assigned route.

One may anticipate that the latter heuristic will result in better performance since the routes are globally ordered, whereas the former would find a more natural distributed implementation.

7 Conclusions and Future Work

In this paper we have formulated a single path routing problem with two performance metrics, the average message delay and the maximum mean path delay. We discussed the shortcomings of the standard shortest path algorithms when applied to this problem. We showed that the problems are $\mathcal{NP}$-Complete and that polynomial time approximation algorithms are unlikely to exist by proving that the problem of finding a feasible solution is $\mathcal{NP}$-Complete. This suggested the need to devise efficient heuristic algorithms. We proposed fast, polynomial time, heuristic algorithms for the case of a single source and multiple destinations and discussed possible extensions to multiple sources. The Minimum Sequential Cost heuristic took into account the effects of overlapping routes, which was ignored by the shortest path algorithms. Of course, these heuristic algorithms could generate potentially sub-optimal routing assignments or possibly might not even generate a feasible solution even if one did exist. The problem of single path routing with minimum packet loss was also seen to be $\mathcal{NP}$-Complete. We also examined the problem of constructing spanning trees and provided a polynomial time algorithm for broadcasting.

In an effort to recognize tractable instances, we considered the general Single Path Routing problem on bidirectional rings and determined that it remained intractable. Based on the intractability of this simple case we anticipate that it is unlikely that useful tractable instances can be found. For future endeavors, we note that bifurcated routing does consider the delay due to overlapping routes, although the resequencing overhead introduced could degrade the performance of this type of routing. Thus, in terms of the effects of overlapping routes, it would be interesting to compare single-path and bifurcated routing.

Acknowledgements:

The authors would like to thank Professors Jim Kurose and Don Towsley of the Computer and Information Sciences Department, University of Massachusetts, Amherst, for several valuable suggestions made towards an earlier version of this paper.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[2] D.Bertsekas, "Dynamic Behavior of Shortest Path Routing Algorithms for Communication Networks", *IEEE Trans. on Comm.*, Feb. 1982, pp. 60-74.

[3] D.Bertsekas and R.Gallager, *Data Networks*, Prentice-Hall, 1987.

[4] N.Deo and C-Y.Pang, "Shortest Path Algorithms: Taxonomy and Annotation", *Networks*, Vol. 14, 1984, pp. 275-323.

[5] R.Gallager, "A Minimum Delay Routing Algorithm Using Distributed Computation", *IEEE Trans. on Comm.*, Jan. 1977, pp. 71-85.

[6] M.R.Garey and D.S.Johnson, *Computers and Intractability - A Guide to the Theory of $\mathcal{NP}$-Completeness*, W.H.Freeman, 1979.

[7] L.Kleinrock, *Queueing Systems: Vol 1: Theory*, Wiley, 1975.

[8] M.T.Liu and D.M.Rouse, "A Study of Ring Networks", in I.N.Dallas and E.B.Spratt, editors, *Ring Technology Local Area Networks*, Elsevier Science 1984.

[9] J.M.McQuillan, I.Richer and E.C.Rosen, "The New Routing Algorithm for the ARPANET", *IEEE Trans. on Comm.*, Vol. COM-28, pp. 711-719, May 1980.

[10] P.M.Merlin and A.Segall, "A Failsafe Distributed Routing Protocol", *IEEE Trans. on Comm.*, vol COM-27, no.9, pp. 1280-1288, 1979.

[11] Y.Ofek and I.Cidon, "Metaring: A Ring with Fairness and Spatial Reuse", *IBM Research Report RC 14961*, also in *Proc. INFOCOM*, 1990.

[12] C.H.Papadimitriou and K.Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, 1982.

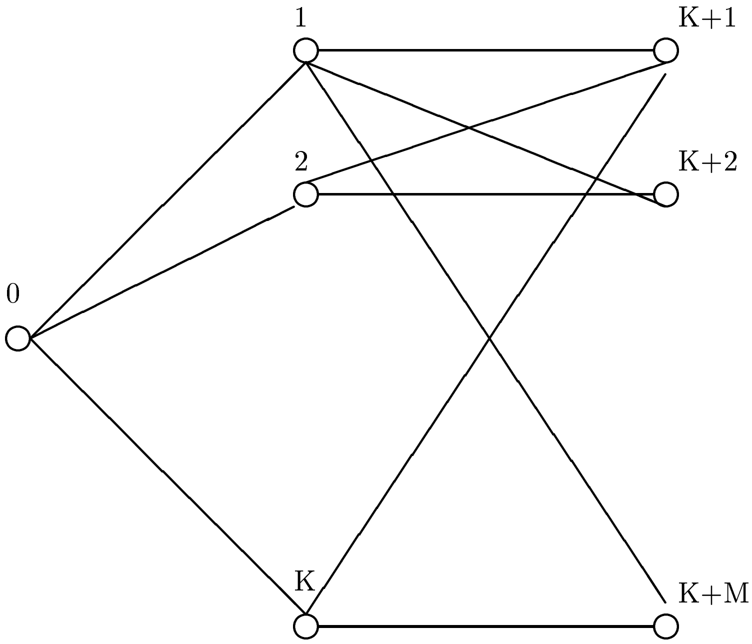
[13] A.N.Tantawi and D.Towsley, "Optimal Static Load Balancing in Distributed Computer Systems", *Journal of the ACM*, Vol. 32, pp. 168-175, 1985.

[14] M.Schwartz and T.E.Stern, "Routing Techniques used in Computer Communication Networks", *IEEE Trans. on Comm.*, vol COM-28, pp. 539-552, 1980.

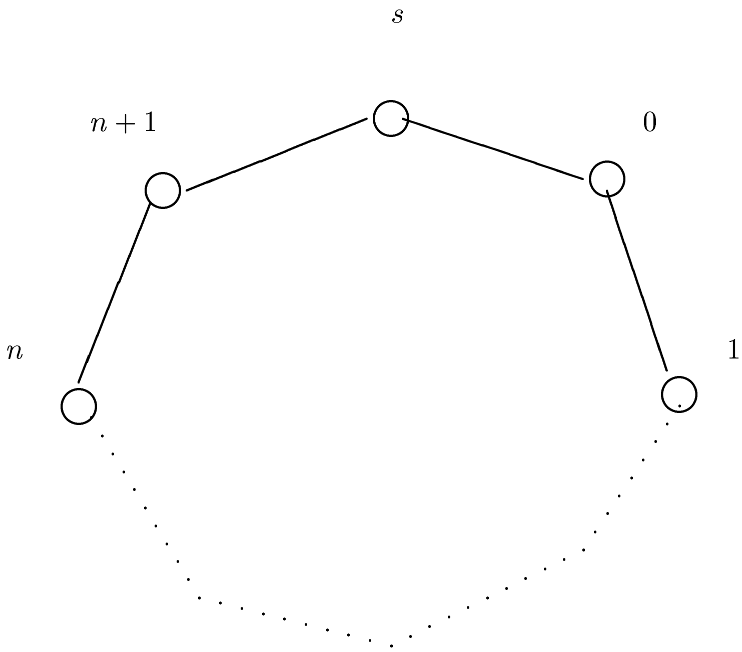
[15] M.Schwarz, *Telecommunication Networks: Protocols, Modeling and Analysis*, Addison-Wesley, 1987.

[16] A.Segall, "Distributed Network Protocols", *IEEE Trans. on Info. Theory*, vol IT-29, no.1, pp23-35, 1983.

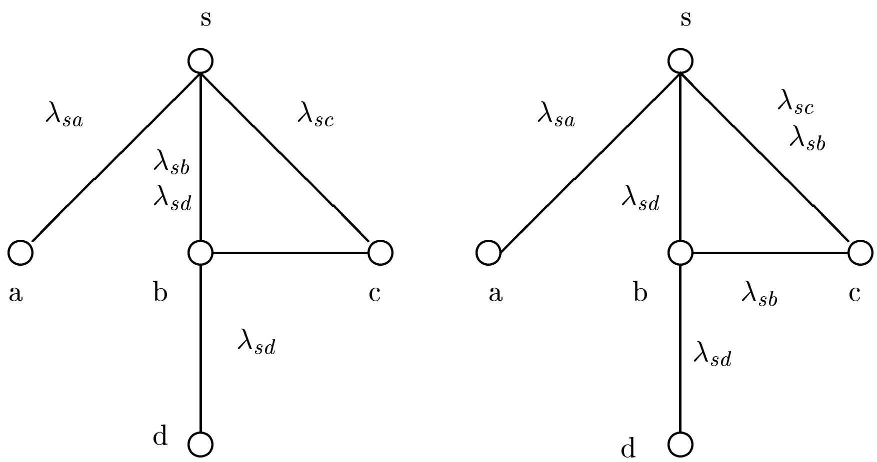
[17] W.Stallings, *Local Networks*, Chapter 3, Macmillan, 1990.



1
K+1
2
K+2
0
K
K+M



s
$n+1$
0
n
1



s
λ_{sa}
λ_{sc}
λ_{sb}
λ_{sd}
a
b
c
λ_{sd}
d
s
λ_{sa}
λ_{sc}
λ_{sb}
λ_{sd}
a
b
λ_{sb}
c
λ_{sd}
d