ORNL/CP-95289

CONF-9703134--

QOS Routing Via Multiple Paths Using Bandwidth Reservation[†]

Nageswara S. V. Rao and Stephen G. Batsell
Oak Ridge National Laboratory [‡]
Oak Ridge, Tennessee 37831-6364
{raons,sgb}@ornl.gov

Submitted to *IEEE INFOCOM98: The Conference on Computer Communications*, March 29 - April 2, 1998, San Francisco, USA.

[†]Research sponsored by the LDRD project at Oak Ridge National Laboratory.

[‡]Managed by Lockheed Martin Energy Research Corp. for the U.S. Department of Energy under Contract No. DE-AC05-96OR22464.

QoS Routing Via Multiple Paths Using Bandwidth Reservation[†]

Nageswara S. V. Rao
Mailstop 6364
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6364
Email: raons@ornl.gov
Phone: (423) 574-7517
Fax: (423) 574-7860

Stephen G. Batsell
Mailstop 6367
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6367
Email: sgb@ornl.gov
Phone: (423) 574-7401
Fax: (423) 574-0680

Corresponding Author: Nageswara S. V. Rao

Abstract

We address the problem of computing a multipath, consisting of possibly overlapping paths, to transmit data from the source node s to the destination node d over a computer network while ensuring deterministic bounds on end-to-end delay or delivery rate. We consider two generic routing problems within the framework wherein bandwidth can be reserved, and guaranteed, once reserved, on various links of the communication network. The first problem requires that a message of finite length be transmitted from s to d within τ units of time. The second problem requires that a sequential message of r units be transmitted at a rate of η such that maximum time difference between two units that are received out of order is no more than q. We propose a polynomial-time algorithm to the first problem based on an adaptation of the classical Ford-Fulkerson's method. We present simulation results to illustrate the applicability of the proposed algorithm. We show the second problem to be NP-complete, and propose a polynomial-time approximate solution.

Keywords and Phrases: Routing algorithms, quality of service, maximum flow methods, multiple paths, bandwidth reservation.

†Research sponsored by the Laboratory Director's Research Project at the Oak Ridge National Laboratory.

1 Introduction

The ability to provide user- or application-level guarantees that a transmission task will be performed under strict Quality of Service (QoS) requirements is vital to the development of next generation of network services. For example, a medical image or a robot control packet may be required to be transmitted over a network with the minimum end-to-end delay. Another example is the transmission of video over a computer network without undesirable delays and jitter. To ensure the performances typified in these examples, it is necessary to employ methods that guarantee strict delay and/or rate. The expected rapid proliferation of services, such as multimedia, remote medicine, and remote robot laboratories, over the wide area networks, would require performances unprecedented in the currently available best-effort routing methods. In particular, the present Internet routing mechanisms (based on the best-effort paradigm) are unlikely to provide satisfactory end-to-end performance for services required in these future applications [3, 29]. Thus, there is a definite need for architectures and algorithms that provide QoS guarantees beyond those of currently available ones.

In QoS mechanisms for computer networks, in general, the parameters may be optimized either at the network-level or at the user-level, and here we choose the latter. In particular, we consider source-based routing algorithms [35] for two generic user-level transmission tasks. In our framework, bandwidth can be reserved on the communications links, and, once reserved, is guaranteed for the required time period. While such requirements call for additional mechanisms, they in turn enable us to provide *deterministic* delay and/or rate guarantees. This type of bandwidth guarantees can be naturally supported in ATM networks [24] which are being increasingly employed (see Section 3.4). In other scenarios, for example the Internet, the bandwidth reservations may require additional mechanisms, such as RSVP [42] and specific queuing implementations at the routers [7]. Our framework is different from the dynamic frameworks which utilize feedback mechanisms [22, 10, 4] to provide only "soft" guarantees. Our potential applications include the transmission of: (a) files of varied sizes, for example, ranging from small robot control packets to large image files, and (b) data streams such as video-on-demand or robot vision data.

We discuss algorithms that plan *multipaths*, consisting of possibly overlapping paths, based on the available bandwidths on various links of the network. We formulate the following two generic transmission tasks, and discuss their solutions and computational complexity in a rigorous manner:

(a) The first problem deals with transmitting a message of r units from the source s to the destination d over a network within τ units of time. The message units can be received at d in any order. This problem abstracts services such as file and image transfers, where a volume of data needs to be transmitted over the network.

(b) The second problem deals with transmitting a sequence of r units, denoted by $\{u_1, u_2, \ldots, u_r\}$, from the source node s to the destination node d. Let t_i, $i = 1, 2, \ldots, r$, be the time u_i is received at d. The problem is to ensure that $t_i - t_j \leq q$ for $i < j$, and, in addition, the message units must be received at a rate of η. This problem abstracts services such as video-on-demand, video calls, and vision feedback to a teleoperator. Consider that s sends video or vision data to be played at d at a rate of η. Since different packets can travel via different paths, they can be delayed by different extents (note that this problem does not arise if single paths are employed). Under the condition $t_i - t_j \leq q$, for $i < j$, the destination can start playing the video within q time units after receiving the first message unit, while buffering no more than $q\eta$ message units at any time. In practical applications involving vision or video data q must be a small number.

We consider routing algorithms to compute multipaths from s to d for both these problems. It is assumed that the information about the available bandwidths of all links is centrally available or can be gathered when needed using a distributed algorithm. Once the routing algorithm is initiated, the bandwidths are "held available" until the paths are computed and the requests for bandwidths are received. Thus, it is critical that the routing algorithms be computationally efficient in order not to tie-up bandwidth during their execution. We present a polynomial-time algorithm for solving the first problem. We show the second problem to be computationally intractable (NP-complete [17]), thereby motivating our approximation algorithm to ensure reasonable execution times.

1.1 Relation to Prior Work

Utilization of multiple paths to provide improved performance compared to single paths has been explored extensively in the past for various network problems. To name a few, some of the early works are due to [16, 19], and some of the recent works are due to [4, 10]. In particular, the two problems described in the introduction bare resemblances to a number of network flow problems studied extensively in the eighties, and also to QoS problems of more recent origin. These two problems can be formulated as (static) special cases of the well-known optimal routing problem [16, 6, 33, 19] using possibly non-differentiable cost functions, and can be solved by general methods. A comprehensive treatment on the optimal routing problem and its solutions, and also on other similar problems can be found in [5]. Similarly, our problems can also be posed as special cases of the classical minimum cost flow problem [1] studied in transportation and operations research. These general solutions, however, do not guarantee low polynomial-time computational complexity nor indicate the computational complexity of the present problems.

On the other hand, our two problems are strictly more difficult than the maximum flow problem without flow costs (see Section 3.5), for which polynomial-time algorithms are known [1]. In addition, our first problem is also similar to several other problems studied in computer networks, but none of them provide an algorithm for the former with polynomial time complexity. For example, the disjoint path methods [37, 36, 32] are inadequate here since the minimum end-to-end delay could be achieved by a set of non-disjoint paths. Also, the solutions based on computing k shortest paths [38] do not yield the minimum end-to-end delay; despite a superficial resemblance of our algorithm to this approach, the solutions to these problems could be quite different depending on the bandwidth values.

A majority of recently proposed QoS routing algorithms that provide bounds on end-to-end delay and/or transmission rates are limited to single paths [41, 40] with the possible exception of [25, 10, 4]. Optimization of network-level parameters is discussed in [25], and their QoS parameters are not deterministic. In spirit, our first problem is a special case of the one studied in [14], where the transmission task is specified by several parameters. Our work differs from [14] in the following ways: (a) we employ multiple paths thereby achieving lower end-to-end delays, (b) we show the optimality and polynomial-time computability of our algorithm, and (c) our algorithm is source-based while their algorithm is distributed. Problems similar to our second one are discussed in [10, 3, 22, 4], but their bandwidth guarantees are "soft" due to the dynamic nature of the formulation. In summary, despite the intuitive nature and potential applicability, we are unaware of systematic algorithmic treatments of our two problems.

1.2 Contribution and Organization of the Paper

The deployment of the multiple paths for transmission problems seems intuitively obvious in that more the number of paths the larger will be the resultant bandwidth [35, 10]. Surprisingly, such increase in the bandwidth *does not* necessarily result in a lower end-to-end delay. Moreover, the addition of newer paths to existing set of paths, in fact, increases the end-to-end delay under certain conditions (see Example 2 in Section 3). This curious phenomenon is due to the finiteness of r and non-zero values for the delays. As a result, the traditional maximum flow methods based on augmenting existing set of paths with additional paths [15, 26] are not applicable here. We identify the necessary and sufficient conditions under which the additional paths that increase the bandwidth also reduce the end-to-end delay (Section 3.2, Lemma 3.1). A careful application of this result to an adaptation of the classical maximum flow method (due to [13]) yields a polynomial time algorithm for the first problem. We also show that the classical maximum flow problem [15, 1] is a special case of the first problem when message size is sufficiently large or all link delays are zero. Although both our problems are abstract compared to real-life applications, a concrete analysis of these problems enables us to understand the underlying complexities of more complex tasks. In particular, the intractability of the second problem motivates us to search only for approximate solutions if real-time response is required (unless the $P = NP$ question is affirmatively settled [17]). We also discuss simulation results that illustrate the applicability of the proposed algorithms in the context of existing networks, namely ESnet and NSFNET. The total benefits of our method, however, would require more specialized networks with bandwidth guarantees on all links.

The organization of this paper is as follows: In Section 2, the two generic problems are formulated precisely, and preliminaries are discussed. The message transmission problem is discussed in Section 3. The sequence transmission problem is discussed in Section 4.

2 Problem Formulation

We consider a network represented by a graph $G = (V, E)$ with n nodes and m edges or links. Each edge $e = (i, j) \in E$ enables the transmission of messages at a *bandwidth* of $B(e) \geq 0$ units per second. There is a *link delay* $D(e) \geq 0$ for each message unit such that a message unit sent from node i at time t on link e will arrive at node j at time $t + D(e)$. The link delay includes the preparation and propagation time of the link. By pipelining the message units, a message of r units can be sent along the edge e in $r/B(e) + D(e)$ time. An edge of bandwidth B and delay D can be visualized as two parallel edges with bandwidths B_1 and B_2, $B_1 + B_2 = B$, and each with delay D.

Consider a *path* from i_0 to i_k given by $(i_0, i_1), (i_1, i_2), \ldots, (i_{k-1}, i_k)$, where $(i_j, i_{j+1}) \in E$, for $j = 0, 1, \ldots (k-1)$. The path is *simple* if all $i_0, i_1, \ldots, i_k$ are distinct. The *delay* of this path P, denoted by $D(P)$, is given by $\sum_{j=0}^{k-1} D(e_j)$, where $e_j = (i_j, i_{j+1})$. The *bandwidth* of this path, denoted by $B(P)$, is given by $\min_{j=0}^{k-1} B(e_j)$.

A *multipath* from s to d, denoted by MP, is set of (possibly overlapping) simple paths from s to d. When MP consists of a single path P, we often denote $MP = \{P\}$ by P to simplify notation. MP can be visualized as a subgraph of G consisting of s and d such that every edge of this subgraph is contained in a path of MP from s to d. Let $P_1, P_2, \ldots, P_k$ denote the paths that constitute MP. A *cut* C of the multipath MP is a set of its edges whose removal disconnects s and d [15]. The bandwidth of a cut C of MP is the sum of bandwidths of edges of the cut. Among all cuts of MP, the one with minimum bandwidth is called the *minimum cut.* The *bandwidth* of MP, denoted by $B(MP)$, is defined as the bandwidth of its minimum cut. For $MP = \{P_1, P_2, \ldots, P_k\}$ when all P_i's are edge disjoint, we have $B(MP) = \sum_{i=1}^{k} B(P_i)$; in general, however, we can only say $B(MP) \geq \max_i B(P_i)$, with equality achieved when all paths contain the same edge with minimum bandwidth. The *end-to-end delay*, denoted by $t(MP)$, of a multipath MP from s to d is defined as the time required to send a massage of r units from s to d.

We call a subset $\widehat{MP}_1$ of MP a *sub-multipath* of MP. Note that set difference $MP - \widehat{MP}$ is also a sub-multipath of MP. Consider the subgraph of G corresponding to a multipath MP. Then add an edge e to MP to connect two of its vertices to form a new subgraph $\overline{MP}$ of G. Since each edge e of MP can be visualized as two parallel edges, with bandwidths $B(e)$ and 0, the new edge can be viewed as constituting a zero bandwidth path from s to d. Thus $\overline{MP}$ is a multipath and MP is its sub-multipath.

We now provide formal definitions of the two transmission problems outlined in the introduction. The first problem is defined as follows.

> **Message Transmission Problem:** (MTP)
> Given are the computer network $G = (V, E)$, the delays $D(e)$ for all $e \in E$, and the bandwidths $B(e)$, for all $e \in E$. The task is to compute a multipath [1] MP to transmit a message of r units from source s to destination d over the network $G = (V, E)$ such that the time elapsed since the first unit was sent from s until the last unit is received at d is no more than τ.

Note that the message units can be received at the destination in any order, and it is only required that $t(MP) \leq \tau$. The above problem is the *decision version*, and its *optimization version* requires the end-to-end delay to be minimized over all multipaths between s and d. The second problem is stated as follows:

> **Sequence Transmission Problem:** (STP)
> Given are the computer network $G = (V, E)$, the delays $D(e)$ for all $e \in E$, the

[1]To avoid degenerate cases in the transmission problems, it is assumed that every edge of a solution multipath is utilized in transmitting at least one message unit.

bandwidths $B(e)$, for all $e \in E$, and a message consisting of r units, denoted by $\{u_1, u_2, \ldots, u_r\}$. The task is to compute a multipath to transmit the message from the source s to the destination d such that (a) units arrive at d at a rate of η, and (b) $t_i - t_j \leq q$ for $i < j$ where t_i is the time u_i is received at d.

An *optimization version* of STP requires that the bandwidth of the multipath between s and d be maximized for any given value of q. Note that if a single path is employed for transmission, then all message units will arrive at d in sequence, i.e. $q = 0$. Since multipaths are allowed we can potentially benefit from added bandwidth due to additional paths, but, at the cost of units arriving out of order at d. By maintaining a buffer of size $q\eta$, the sequence can be reconstructed and displayed at d at the rate η starting no later than q from the time first segment arrives at d. Note that the entire sequence will be reconstructed in time $r/\eta + q$ at d. A variation of STP that requires a multipath MP such that: (a) $t(MP) \leq \tau$, and (b) $t_i - t_j \leq q$, for $i < j$, is considered in Section 4 (STP will be shown to be a special case of this problem).

Both MTP and STP require the computation of a suitable multipath in a source-based paradigm. It is assumed that all information about reservable bandwidth is centrally available. Also, once bandwidths are reserved, the corresponding nodes are equipped with suitable servicing mechanisms to ensure that the bandwidths are suitably guaranteed. Although the main motivation of MTP and STP is to provide strict QoS guarantees, the overall framework can also be used (in some cases) to obtain an "average" end-to-end delay by using "average" estimates for the bandwidth and delay.

3 Message Transmission Problem

First let us consider the applicability of two currently available methods to solving MTP. The shortest-widest paths [41] have been employed as a mechanism for QoS routing, where a *shortest-widest* path is a path with the shortest delay among all paths with the largest bandwidth from s to d. This method, however, does not exploit multipaths to decrease the end-to-end delay. Furthermore, even when single paths are employed, for certain ranges of message sizes and delays, smaller end-to-end delays may *not* be achieved by the shortest-widest paths (Example 1) [31]. The flow-methods are often proposed for various multiple path routing problems [8, 5]. One of the widely used maximum flow method, the Ford-Fulkerson's method [15], computes the flow by repeated path augmentation. Such an augmentation method does not always result in lower delays, and in fact, can increase the end-to-end delay (Example 2). By carefully controlling the augmentation process, we will derive an algorithm for MTP in Section 3.3; this algorithm requires a detailed examination of the properties of multipaths, which is presented in Section 3.2.

3.1 Shortest-Widest Paths

We first show that a solution to MTP based on a multipath can result in a better performance compared to the shortest-widest path [41].

Example 1: Consider the network shown in Fig. 1 that has two disjoint paths from s to d with 2 and 3 edges respectively. The delay of each edge is D units. The bandwidth of

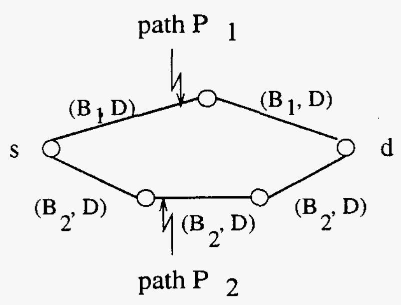

Figure 1: Widest-Shortest paths versus multipaths.

each edge of path P_i is B_i, for $i = 1, 2$. Assume $B_2 > B_1$, and thus P_2 is the shortest-widest path. Now consider the transmission of a message of $r = 1500$ units via P_2 with $D = 10$ sec, $B_1 = 100$ units/sec, and $B_2 = 200$ units/sec. The end-to-end delay due to path P_2 is $1500/200 + 30 = 37.5$ sec. On the other hand, if path P_1 is used, the delay is given by $1500/100 + 20 = 35$ sec. For this network, the condition for P_1 resulting in a smaller delay is given by $r/B_2 + 3D \geq r/B_1 + 2D$ or equivalently by $r \leq \frac{DB_1B_2}{B_2-B_1}$. Thus, for smaller values of $r \leq 2000$, it is advantageous *not* to use the shortest-widest path.

Now consider that the message of $r = 1500$ units is split into two messages of 1164 and 334 units and sent on P_1 and P_2, respectively. Then, the submessages are received via P_1 and P_2 in $1166/100 + 20 = 31.66$ sec and $334/200 + 30 = 31.67$ sec, respectively. Thus, the entire message is transmitted in 31.67 sec which is smaller compared to using individual paths. However, as will be shown in Example 2, the use of multipaths does not always result in a smaller delay. □

In general, consider a network consisting of two non-intersecting paths P_1 and P_2 between s and d such that $D_i = D(P_i)$ and $B_i = B(P_i)$, $i = 1, 2$. Consider that $B_2 > B_1$ and $D_2 < D_1$, and thus P_2 is the shortest-widest path of [41]. If $r < \frac{(D_2-D_1)B_1B_2}{B_2-B_1}$ we have $t(P_2) > t(P_1)$, i. e. the delay of the shortest-widest path is larger. In general if $D_2 > D_1 + \frac{(B_2-B_1)r}{B_1B_2}$, the total delay of P_1 is smaller. In general, the end-to-end delay can be minimized by a shortest delay path for very small values of r, and by a shortest-widest path for large values of r. For any given value of r, the Dijkstra's shortest path algorithm can be adapted to solve this problem [32].

3.2 Properties of Multipaths

We first present an illustrative example to show that augmenting existing paths with additional paths does not necessarily reduce the end-to-end delay. Then, we derive the necessary and sufficient conditions under which such augmentation reduces the end-to-end delay.

Example 2: Consider a network consisting of two edges consisting of paths P_1 and P_2 as shown in Fig. 2. Let $B_1 = 10$ units/sec, $B_2 = 20$ units/sec, $D_1 = 2$ sec, and $D_2 = 12$ sec. Consider a message of size $r = 100$ units. Using path P_1 alone for the transmission will result in a delay of $100/10 + 2 = 12$ seconds. The delay of path P_2 is $100/20 + 12 = 17$ sec. If single path is to be used, P_1 will be chosen for this message size. On the other hand, let us say that 99 units are sent on P_1 and 1 unit is sent on P_2; the corresponding delays are given by $99/10 + 2 = 11.9$ sec and $1/20 + 12 = 12.05$ sec respectively, resulting in an end-to-end delay of 12.05 seconds. Clearly, it is advantageous *not* to use the multipath $\{P_1, P_2\}$ for this message size. In other words, although the bandwidth of $\{P_1, P_2\}$ is larger than that of P_1 and P_2, it did not result in a smaller delay.

Now consider a message of size $r = 1000$ units. The end-to-end delays of individual paths P_1 and P_2 are 102 sec and 62 sec, respectively. Thus if a single path is to be used, then P_2 will be chosen. In summary, it can be shown that: for $r > 200$, P_2 is the choice; for $r < 200$, P_1 is the choice; and for $r = 200$ either can be chosen (see Fig. 3).

Consider using a multipath $\{P_1, P_2\}$ for $r = 1000$ such that 400 and 600 units are sent via P_1 and P_2 respectively, resulting in the individual delays of 42 sec for each path; hence the resultant end-to-end delay is 42 sec which is smaller than that of P_1 or P_2 (given by 102 and 62 sec, respectively).

In general, the message size determines if multipath $\{P_1, P_2\}$ achieves a lower delay. The allocation for the least end-to-end delay is given by: if $r \leq 102$ use P_1, and use the multipath $\{P_1, P_2\}$ otherwise. To see this consider that message is split into two parts of size x and $r - x$, and sent on P_1 and P_2, respectively. Then the delay due to the multipath is $\max\left(\frac{r-x}{20} + 12, \frac{x}{10} + 2\right)$, which is minimized at $x = (r + 200)/3$. Thus the delay due to multipath is $r/30 + 8.66$. The delays of P_1, P_2 and $\{P_1, P_2\}$ are shown in Fig. 3. Note that delay of $\{P_1, P_2\}$ is smaller than that of path P_1, for $r > 102$. The lower envelop of the various delay lines in Fig. 3 corresponds to the optimal strategy for this case. $\square$

It is convenient to visualize a multipath MP as *line segment* in the interval $[1, r]$ as shown in Fig. 4. Here when message is of unit length the total delay of the path is $D(MP) = D$, and as large messages are considered the delay increases with a slope of $1/B(MP)$. Subsequently, we denote the segment of MP increasing from *left-to-right* or *right-to-left* depending on the context.

Now consider that we employ the multipath $MP = \{P_1, P_2\}$ to solve the message transmission problem, where P_1 and P_2 are two edge-disjoint simple paths. Recall that an edge of bandwidth B and delay D, can be visualized as two parallel *disjoint* edges with bandwidths B_1 and B_2, $B_1 + B_2 = B$, and each with delay D. The following lemma identifies conditions under which it is advantageous to employ multipath $\{P_1, P_2\}$ over a single path.

Lemma 3.1 *Consider two simple edge-disjoint paths P_1 and P_2 from s to d. Consider the multipath from s to d given by $MP = \{P_1, P_2\}$. We have $t(MP) \geq \min\{t(P_1), t(P_2)\}$ if and only if $D(P_1) + r/B(P_1) \leq D(P_2)$ and $D(P_2) + r/B(P_2) \geq D(P_1)$.*

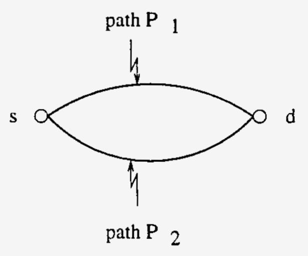

Figure 2: Single paths versus multipaths.

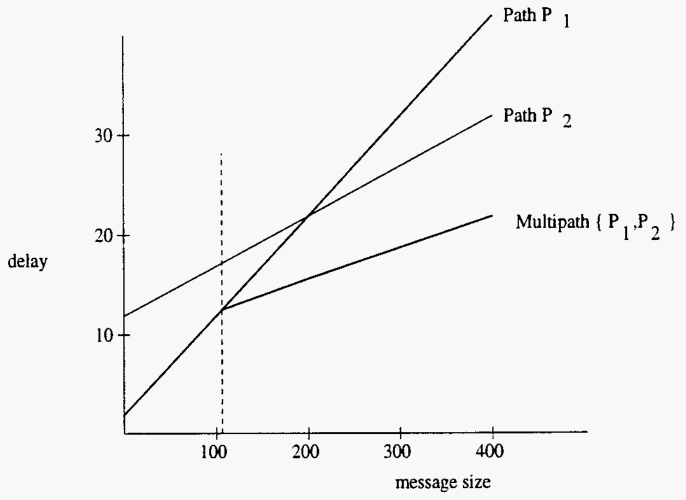

Figure 3: Delays of single paths and multipaths.

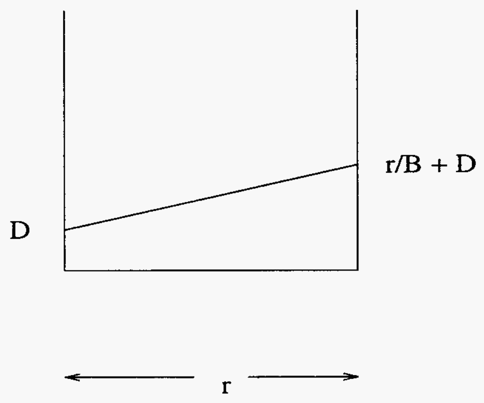

Figure 4: Visualization of segment of a multipath.

Proof: First consider the if part. Let r_i denote the number of units transmitted along P_i, $i = 1, 2$. Thus $r_1 + r_2 = r$ Then the total delay is given by $\max\{r_1/B_1 + D_1, r_2/B_2 + D_2\}$. Under the condition $D(P_1) + r/B(P_1) \leq D(P_2)$ and $D(P_2) + r/B(P_2) \geq D(P_1)$, the delay is minimized when total delays of both paths are the same. To see this refer to Fig. 5, where $t(P_i)$ is linearly decreasing function of r_i starting with $r_i = r$. By visualizing (r_1, r_2) as a point on the interval $[0, r]$ with r_1 represented as segment $[0, r_1]$ and r_2 represented as segment $[r - r_1, r]$, the minimum end-to-end delay is achieved under the condition

$$D_2 + \frac{r - r_1}{B_2} = D_1 + \frac{r_1}{B_1}.$$

This condition in turn yields

$$r_1 = \frac{B_1 B_2}{B_1 + B_2}(D_2 - D_1 + r/B_2)$$

which yields the total delay of

$$t(MP) = \frac{r}{B_1 + B_2} + \frac{B_2 D_2 + B_1 D_1}{B_1 + B_2},$$

which is no more than $\min\{t(P_1), t(P_2)\}$.

For the only if part, consider that $D(P_1) + r/B(P_1) \geq D(P_2)$ (the other case is identical). This condition means that entire message can be sent via P_1 in time less than $D(P_2)$. Thus is a single unit is sent via P_2, then more total delay is encountered. □.

The minimum end-to-end delay of $\{P_1, P_2\}$ is achieved by dividing the message into two parts of sizes r_1 and r_2, $r_1 + r_2 = r$, to be sent via P_1 and P_2 respectively. The sizes of the parts are given by

$$r_1 = \frac{B_1 r}{B_1 + B_2} + \frac{B_1 B_2 (D_2 - D_1)}{B_1 + B_2} \quad \text{and} \quad r_2 = \frac{B_2 r}{B_1 + B_2} - \frac{B_1 B_2 (D_2 - D_1)}{B_1 + B_2}.$$

An example when the condition of Lemma 3.1 is violated is shown in Fig. 6. Based on the proof of Lemma 3.1, the multipath $\{P_1, P_2\}$, can be visualized as a single path with

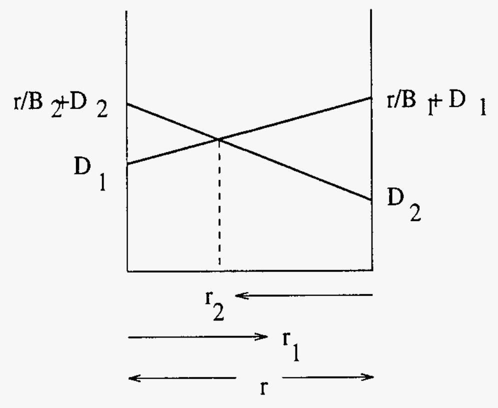

Figure 5: Illustration of conditions of Lemma 1.

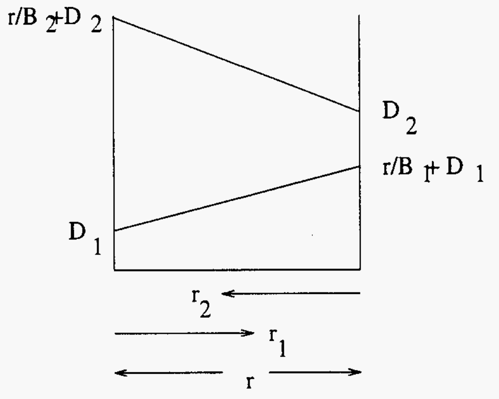

Figure 6: Example of conditions of Lemma 3.1 not satisfied.

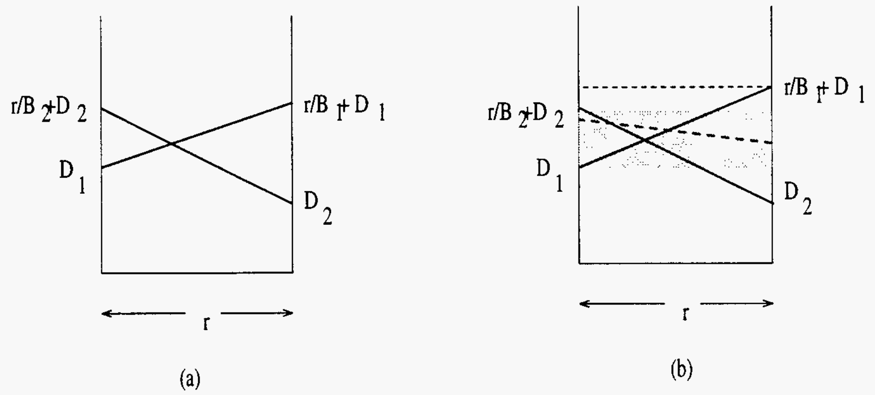

Figure 7: Region for segment of MP when $C(MP_1, MP_2)$ is true.

bandwidth $B_1 + B_2$ and delay $\frac{B_2D_2+B_1D_1}{B_1+B_2}$, when the condition of Lemma 3.1 is satisfied. Using this argument, the proof of Lemma 3.1 can be generalized in an obvious way to the case where we consider two multipaths MP_1 and MP_2. For ease of reference, we denote by $C(MP_1, MP_1)$ the condition:

$$D(MP_1) + r/B(MP_1) \geq D(MP_2) \quad \text{and} \quad D(MP_2) + r/B(MP_2) \geq D(MP_1).$$

Lemma 3.2 *Consider two multipaths MP_1 and MP_2 from s to d. Consider the multipath from s to d given by $MP = \{MP_1, MP_2\}$. We have $t(MP) \leq \min\{t(MP_1), t(MP_2)\}$ if and only if $C(MP_1, MP_2)$ is true.*

The effect of forming multipaths can be schematically visualized as follows. Consider the multipath $MP = \{MP_1, MP_2\}$, and $D_i = D(MP_i)$, $B_i = B(MP_i)$, for $i = 1, 2$. Now it is direct that (a) $D(MP) \geq \max\{D_1, D_2\}$, and (b) $B(MP) \geq \max\{B_1, B_2\}$. The latter implies that $r/B(MP) \leq \min\{r/B_1, r/B_2\}$. Thus MP has higher delay and lower slope compared to either of MP_1 and MP_2. First consider that $C(MP_1, MP_2)$ is true as shown in Fig. 7(a). Take the segment of MP_2 that increases from right-to-left to identify the shaded region that contains the segment of MP; a typical example of segment of MP is shown in dotted lines in Fig. 7(b). Now consider that $C(MP_1, MP_2)$ is false, and assume that $D_2 > r/B_1 + D_1$ as shown in Fig. 8. The shaded region shows the range in which the segment of MP lies with a typical example shown in dotted lines.

Lemma 3.2 identifies the critical condition when an existing multipath can be augmented with an additional path or multipath. By systematically scheduling such augmentation steps, we show that MTP can be solved in polynomial time in the next section.

3.3 Routing Algorithm

The routing algorithm assumes that entire graph with the bandwidths and delays is available at a centralized location. This algorithm is obtained by combining the classical Ford-Fulkerson's maximum flow algorithm with the conditions stated in Lemma 3.2. We present an algorithm that computes a multipath with minimum delay to solve the optimization version of MTP, which can be used directly to solve the decision version.

Familiarity with the basic ideas of Ford-Fulkerson method and its implementations is assumed in this section (a complete discussion can be found in [11, 1]). We define *capacity* for $u, v \in V$ as follows: $c(u, v)$ is the bandwidth of the edge (u, v) if such edge exists, and is 0 otherwise. Let *flow* in G be a real-valued function $f : V \times V \mapsto \Re$ such that: (i) for all $u, v \in V$, we have $f(u, v) \leq c(u, v)$; (ii) for all $u, v \in V$, we have $f(u, v) = -f(v, u)$; and (iii) for all $u \in V - \{s, d\}$, we have $\sum_{v \in V} f(u, v) = 0$. The *residual capacity* of (u, v) is defined by $c_f(u, v) = c(u, v) - f(u, v)$. Given $G = (V, E)$ and the flow f, the *residual network* of G induced by f is $G_f = (V, E_f)$, where

$$E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}.$$

The routing algorithm MTA is described in Algorithm 1, which is similar in spirit to the classical Ford-Fulkerson method. The algorithm returns the multipath MP and the flow function f which specifies the rate at which units are to be transmitted on various links (or equivalently the bandwidth requirements on various edges). The algorithm initializes MP with the shortest delay path P_0 in lines 1-6. Then a new path P_f which is the shortest delay path on G_f is repeatedly added to MP in each iteration in lines 7-12, as long as the condition $C(MP, P_f)$ is true. With each additional path the flow function is suitably adjusted in lines

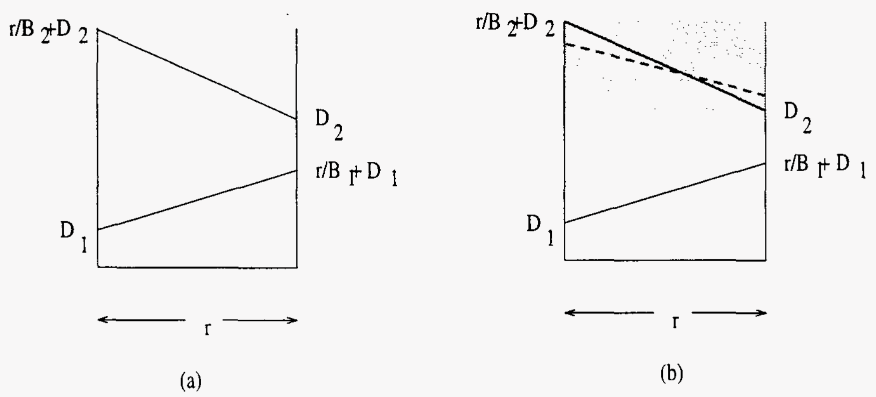

Figure 8: Region for segment of MP when $C(MP_1, MP_2)$ is false.

```
algorithm MTA
Initialization
1. compute shortest delay path $P_0$ on $G$;
2. $MP \leftarrow \{P_0\}$;
3. $c_f(P_0) \leftarrow \min\{c_f(u,v) : (u,v)$ is in $P_0\}$;
4. for each edge $(u,v)$ in $P_0$ do
5.     $f[u,v] = c_f(P_0)$;
6.     $f[v,u] = -f[u,v]$;
Repetitive flow augmentation
7. while for shortest delay path $P_f$ on $G_f$, the condition $C(MP, P_f)$ is true do
8.     $MP \leftarrow MP \cup \{P_f\}$;
9.     $c_f(P_f) \leftarrow \min\{c_f(u,v) : (u,v)$ is in $P_f\}$;
10.    for each edge $(u,v)$ in $P_f$ do
11.        $f[u,v] = f[u,v] + c_f(P_f)$;
12.        $f[v,u] = -f[u,v]$;
13. return $MP$ and $f$;
```

Algorithm 1. Algorithm for solving MTP.

10-13; in particular, each edge on path P_f receives a flow increase corresponding to the bandwidth of P_f in G_f. The algorithm terminates the first time $C(MP, P_f)$ becomes false.

We now establish the correctness of the algorithm MTA. The time complexity estimation closely follows that of Edmonds-Karp algorithm [13] for the maximum flow problem. First notice that the sequence of path computed in step 2 of the algorithm are of non-decreasing delay. Thus when the condition of Lemma 3.2 fails, it is because $D(P_f) > r/B(MP) + D(MP)$; if the shortest path computation were to be continued in step 7, then all the newer paths will fail the same condition.

Lemma 3.3 *The multipath MP computed by the algorithm MTA satisfies the properties:*

(i) No sub-multipath of MP has a lower delay, i.e. for every sub-multipath $\widehat{MP}$ of M, we have $t(MP) \leq t(\widehat{MP})$, and

(ii) No other multipath of G has lower delay than MP, i.e. for any multipath $\widehat{MP}$ of G from s to d we have $t(MP) \leq t(\widehat{MP})$.

Proof: Assume the contrary to Part (i) that the sub-multipath $\widehat{MP}_1$ of MP satisfies the condition $t(\widehat{MP}_1) < t(MP)$. Let $\widehat{MP}_2$ be the sub-multipath of MP such that $\widehat{MP}_1 \cup \widehat{MP}_2 = MP$. Now $t(\widehat{MP}_2) \geq t(MP)$, for otherwise we have union $\widehat{MP}_1 \cup \widehat{MP}_2$ will have strictly smaller delay than MP. We have $t(\widehat{MP}_1) < t(MP) \leq t(\widehat{MP}_2)$ which implies $r/B(\widehat{MP}_2) + D(\widehat{MP}_2) > D(\widehat{MP}_1)$. Since $\widehat{MP}_2$ is a sub-multipath of MP, we have $D(MP) > D(\widehat{MP}_2)$, which implies $t(\widehat{MP}_1) \geq D(MP) > D(\widehat{MP}_2)$, where the first inequality is because $\widehat{MP}_1$ is a sub multipath of MP. Thus $C(\widehat{MP}_1 \cup \widehat{MP}_2)$ is true. By Lemma 3.2, $\widehat{MP}_1 \cup \widehat{MP}_2 = MP$ has smaller delay than $\widehat{MP}_1$, which is a contradiction.

Now consider Part (ii). Assume the contrary that $\widehat{MP} \neq MP$ is a multipath such that $r/B(\widehat{MP}) + D(\widehat{MP}) < r/B(MP) + D(MP)$ which implies $D(\widehat{MP}) < r/B(MP) + D(MP)$. Now there exists at path $\hat{P}$ with non-zero bandwidth in $\widehat{MP}$ not contained in MP. Otherwise $\hat{P}$ is properly contained in MP which is not possible by Part (i). Then the condition $r/B(\widehat{MP}) + D(\widehat{MP}) < D(MP)$ must be satisfied. Otherwise, $C(\widehat{MP}, MP)$ is true and hence the delay of MP can be further reduced by adding $\widehat{MP}$ to it, which is a contradiction (since the algorithm MTA stops only when no more additional paths can be added to MP). Let $\{P_1, P_2, \ldots, P_q\}$ be the list of paths added to MP by algorithm MTA in sequence; thus P_1 is shortest delay path in G, and $D(P_1) \leq D(P_2) \leq \ldots \leq D(P_q)$. Let MP_i consist of P_1, $P_2, \ldots P_i$, and thus $t(MP_q) < t(MP_{q-1}) < \ldots < t(MP_2) < t(MP_1)$.

Since $D(\hat{P}) \geq D(P_1)$, there must exist k' such that $D(MP_{k'}) < D(\widehat{MP})$. Thus there exists $k \leq k'$, such that $D(MP_k) \leq D(\hat{P})$. Now we have $t(\hat{P}) > t(\widehat{MP}) > D(\widehat{MP}) \geq D(MP_k)$. Furthermore, $t(MP_k) > D(\hat{P})$, for otherwise, we have $t(MP_k) \leq D(\hat{P}) < t(\widehat{MP})$ which is a contradiction. Then $C(MP_k, \hat{P})$ is satisfied, hence $\hat{P}$ must be added to MP_k by algorithm MTA, which is a contradiction. □

We first note that the number of path augmentations in the algorithm MTA is upperbounded by those in the case when the algorithm is executed with a sufficiently large r – in such case this algorithm reduces to the maximum flow algorithm, as will be shown in Lemma 3.4. The runtime of algorithm MTA is estimated by using the result of Edmonds and Karp [13] who showed that the number of flow augmentations is upperbounded by $S \sum_{(u,v) \in E} 1/[a(u,v) + a(v,u)] + m$, where S is the maximum weight of a path from s to d, and $a : V \times V \mapsto \Re$ such that $a(u,v) + a(v,u) > 0$. By using the delay of edges, we conclude that number of augmentations is upperbounded by $\frac{nmd_{\max}}{d_{\min}} + m$, where $d_{\max} = \max_{e \in E} D(e)$ and $d_{\min} = \min_{e \in E} D(e)$. The time complexity of each augmentation requires shortest path computation which can be achieved by Dijkstra's algorithm in $O(n^2)$ time or by using Fibonacci heaps in $O(n \log n + m)$ time [11]. Thus the time complexity of this algorithm is given by $O(\frac{n^3 m d_{max}}{d_{min}} + n^2 m)$ using the former.

Summarizing the above discussion, we have the following theorem.

Theorem 3.1 *The message transmission problem for a network $G = (V, E)$ of n nodes and m edges can be solved in $O(\frac{n^3 m d_{max}}{d_{min}} + n^2 m)$ time, where $d_{\max} = \max_{e \in E} D(e)$ and $d_{\min} = \min_{e \in E} D(e)$.*

3.4 Maximum Flow Algorithm

We now discuss the relationship between the algorithm MTA and the maximum flow algorithm by showing that in two special cases the former reduces to the latter. When all link delays are zero, the end-to-end delay is constrained by the bandwidth only, i.e., $\tau = r/B^*$, where B^* is the bandwidth of minimum-cut of s and d in G. In this case Lemma 3.2 is satisfied in each augmentation, and the algorithm MTA reduces to a variation of Edmonds-Karp algorithm [13] (which is an implementation of Ford-Fulkerson's method). Another case is when r is large enough that the condition of Lemma 3.2 is satisfied in each execution of step 2. For example $r > \sum_{e \in E} D(e) / \min_{e \in E} B(e)$ is a sufficient condition. Intuitively, if the length of the message is sufficiently long, then link delay becomes an insignificant contributor to the end-to-end delay, which will be controlled entirely by the bandwidth (given by the minimum cut). In general, in the augmentation step in the algorithm of [15], any path[2] P can be used in flow augmentation. In the algorithm MTA, once the condition of Lemma 3.2 is violated, no further augmentation takes place. A sufficient condition under which the maximum flow algorithm solves MTP is given in the following lemma.

Lemma 3.4 *Let $D_{\max}$ and $D_{\min}$ denote the delays of the longest and shortest paths in G, respectively. A sufficient condition for a solution of the maximum flow problem to be a solution of the message transmission problem is that length of the message be at least $B_{\min}(D_{\max} - D_{\min})$ where $B_{\min}$ is the bandwidth of the minimum cut that separates s and d.*

Proof: Consider the hypothetical path $\tilde{P}$ with delay $D_{\min}$ and bandwidth $B_{\min}$. Let $P_{\max}$ denote the path with longest delay in G. Let for a particular message size r, condition $C(\tilde{P}, P_{\max})$ be false, i.e. $D_{\min} + r/B_{\min} < D_{\max}$. By increasing r to $r + \Delta r$, condition $C(\tilde{P}, P_{\max})$ can be fulfilled if $\Delta r/B_{\min} \geq D_{\max} - (D_{\min} + r/B_{\min})$ or equivalently $r + \Delta r \geq B_{\min}(D_{\max} - D_{\min})$. Thus for a message of size at least $r' = r + \Delta r$, condition $C(\tilde{P}, P_{\max})$ is true. Now for any multipath MP with bandwidth B and delay D we have $B \geq B_{\min}$ and $D \leq D_{\min}$, which implies $D_{\min} + r'/B_{\min} \leq D + r'/B$. Hence condition $C(MP, P_{\max})$ is true for message of size r'. Thus in the execution of the algorithm MTA, condition in Step 7 is satisfied in every iteration. $\square$

As a result of this lemma, the condition on the end-to-end delay can be converted to a condition on the bandwidth for sufficiently large message sizes. This condition, however, is not computationally conducive since the problem of finding $D_{\max}$ is NP-complete [17].

The algorithm MTA can be adapted to generate a *path table* that gives for each message size r a multipath with minimum end-to-end delay. Such table is generated by executing the algorithm MTA for the message size given in Lemma 3.4 with a modification each time a new path P_i is added to the present multipath MP_i. Let r_i be the largest message size for which MP_i minimizes the end-to-end delay. Then for the interval $[r_{i-1} + 1, r_i]$, the table entry is MP_i. Thus the entire path table has no more than $\frac{nmd_{\max}}{d_{\min}} + m$ entries and the complexity of generation of the entire path table is same as that of algorithm MTA. An algorithm to compute the path table when only single paths are allowed is significantly different and is presented in [31].

3.5 Simulation Results

We present three simulation examples to illustrate the performance and relevance of the proposed multipath algorithm. The first example is entirely illustrative, and the other two examples are based on existing networks, namely ESnet and NSFNET [3]. Although the latter two networks are not specifically designed to provide bandwidth guarantees, the ATM portions of these networks can be utilized to support the proposed multipath algorithm.

[2]This path need not be shortest in G_f, and does not need to satisfy the condition of Lemma 3.2.

[3]The topologies used here for both the networks are virtual in that they do not necessarily reflect all specific details of the actual physical connections. This level of abstraction is chosen here mainly as a means of illustrating the salient features of the proposed algorithm. More details about various connections can be incorporated into the proposed algorithms without changing the overall nature of our conclusions.

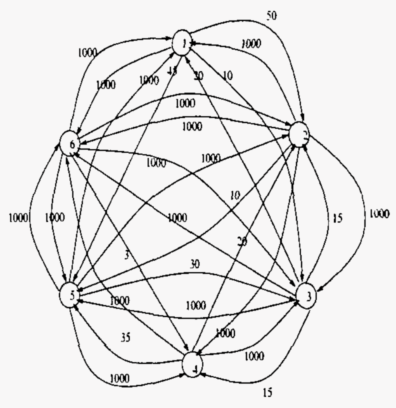

Figure 9: Illustrative example.

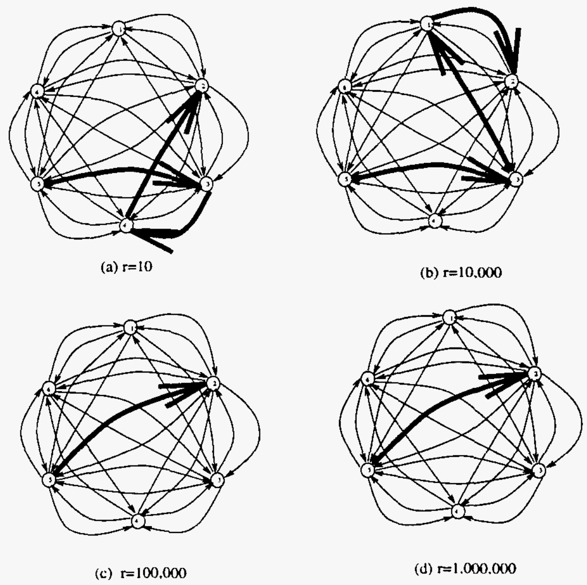

Figure 10: Migration of minimum end-to-end delay single paths.

In all three examples a comparison with the single path algorithm of [31] that guarantees minimum end-to-end delay is also provided. The multipath algorithm is implemented in C on a SPARC workstation. The execution times of the algorithm is no more than a few seconds in all the examples.

Example 3: Consider the network shown in Figure 9. The main purpose of this example is to illustrate (perhaps in an artificially enhanced manner): (a) effects of bandwidths and link delays on the end-to-end delay, and (b) benefits of multipath routing. The link delay and the bandwidth for each link are represented by the same number indicated in Figure 9; thus, in an overall sense, paths with higher delays also have higher bandwidths. Consider that $s = 5$ and $d = 2$. The shortest delay path is $5 - 3 - 4 - 2$ with a delay of 65 and bandwidth of 15 (Figure 10(a)). The shortest-widest path is $5 - 2$ with a delay of 1000 and bandwidth of 1000 (Figure 10(c)). As r is varied from 10 to 10^6, the corresponding single paths with minimum end-to-end delay migrate from low bandwidth to higher bandwidth paths as shown

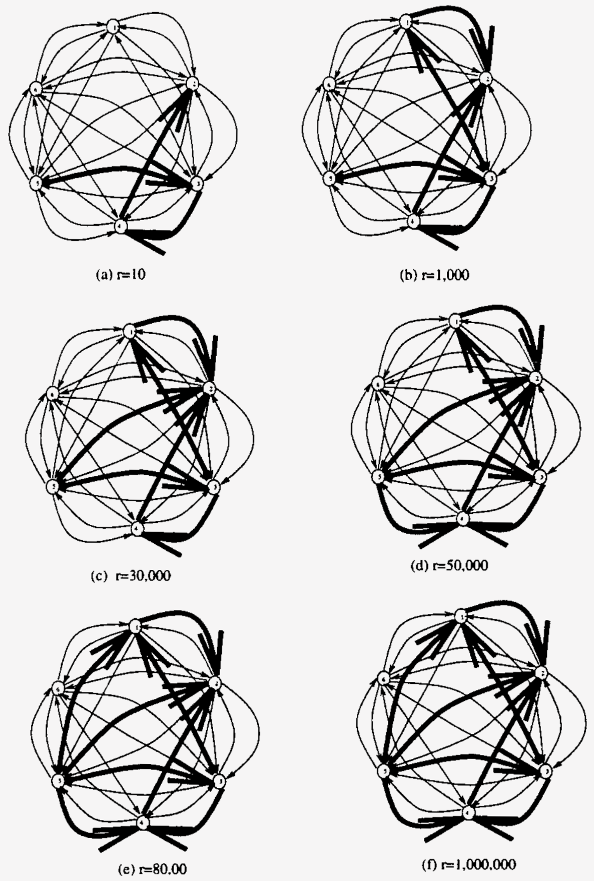

Figure 11: Illustration of minimum end-to-end delay multipaths.

in Figure 10. The multipaths that minimize end-to-end delay are shown in Figure 11. For small values of r (=100) the multipath coincides with the above single path, but for higher values more paths are added. For $r = 100$ the multipath consists of $5-3-4-2$ only, and the path $5-3-1-2$ is added for $r = 1,000$. Then the shortest-widest path $5-2$ is added when $r = 30,000$. For $r = 50,000$ and $r = 80,000$, the paths $5-4-2$ and $5-1-2$ are added, respectively. Note that although the number of paths of the multipath increases with the message size, for any fixed message size, *only* a certain subset of paths must be chosen. Furthermore, in general, multipaths are clearly advantageous over single paths as illustrated in Table 1. □

Example 4: Consider the topology of ESnet shown in Figure 12. The link delays are computed by using an approximate length of the link in miles and suitably adjusted velocity of light. The full bandwidth of T1 and OC3 connections are assumed to be available for this illustration. When a single path is employed, the message transmission between Oak Ridge National Laboratory (ORNL) and Argonne National Laboratory (ANL) is via direct T1 link at 1.54 mbits/see for $r = 10,000$. For larger message sizes, for example $r = 10^5$, the end-to-end delay is minimized by the OC3 connection at 155 Mbits/sec via Chicago. Note that the minimum end-to-end delay is not achieved by the OC3 connection for $r = 10,000$, in spite of its larger bandwidth. When multipaths are employed, for low values of r the

message size	end-to-end delay	
	single path	multipath
10	65.66	65.66 (one path)
1,000	131.66	115.83 (two paths)
10,000	600.00	415.83 (two paths)
20,000	1020.00	749.16 (two paths)
30,000	1030.00	1002.40 (three paths)
40,000	1040.00	1012.11 (three paths)
50,000	1050.00	1021.81 (four paths)
60,000	1060.00	1031.47 (four paths)
70,000	1070.00	1041.13 (four paths)
80,000	1080.00	1050.77 (five paths)
90,000	1090.00	1060.11 (five paths)
100,000	1100.00	1069.46 (five paths)
1,000,000	2000.00	1910.58 (five paths)

Table 1: End-to-end delays for Example 3.

multipath consists of the direct T1 link. For $r = 10^5$, the end-to-end delay is minimized by the multipath consisting of the direct T1 link and OC3 connection via Chicago. For $r = 10^6$, the optimal delay for a single path is 25.017ms and that of a multipath is 24.815ms.

Now consider a more complicated scenario of transmitting from Lawrence Berkeley National Laboratory (LBNL) to ORNL. For large message sizes the single paths realize a bandwidth of 155Mb/s corresponding to the OC3 connection. The multipath realizes a bandwidth of 158Mb/s corresponding to the seven paths shown in Table 2. The initial path is via OC3 connection with the largest bandwidth which is sequentially augmented by a number of rather diverse paths for larger message sizes. For example, the third path is via OC3 connection between PNNL and ANL but using only the bandwidth of T1 connection because of the bottleneck connection between LBNL and PNNL. Although the advantages of the multipath are more pronounced at large message sizes, reduction of end-to-end delay by few ms observed at low message sizes could be vital in applications such as remote robot control. □

Example 5: Consider the topology of NSFNET shown in Figure 13. The link delays are computed as in the previous example. The available bandwidths are assumed to be around the general values for a link but are suitably chosen to illustrate the effect of message size; more precisely, only a portion of the bandwidth is assumed as follows: Each link to regional nodes has a bandwidth of 1 Mbits/sec, each link to a supercomputer has a bandwidth of 10 Mbits/sec, and links to Chicago, Washington DC have a bandwidth of 20 Mbits/sec. The available bandwidths between Palo Alto and NSP#2, NSP#1 and VBNS are taken to be 20, 5 and 1 Mbits/sec. For the transmission of a message from Palo Alto to Washington DC, the single and multipaths that achieve minimum end-to-end delay are given in Table 3. Note that the single paths with minimum end-to-end delay migrate via VBNS, NSP#1

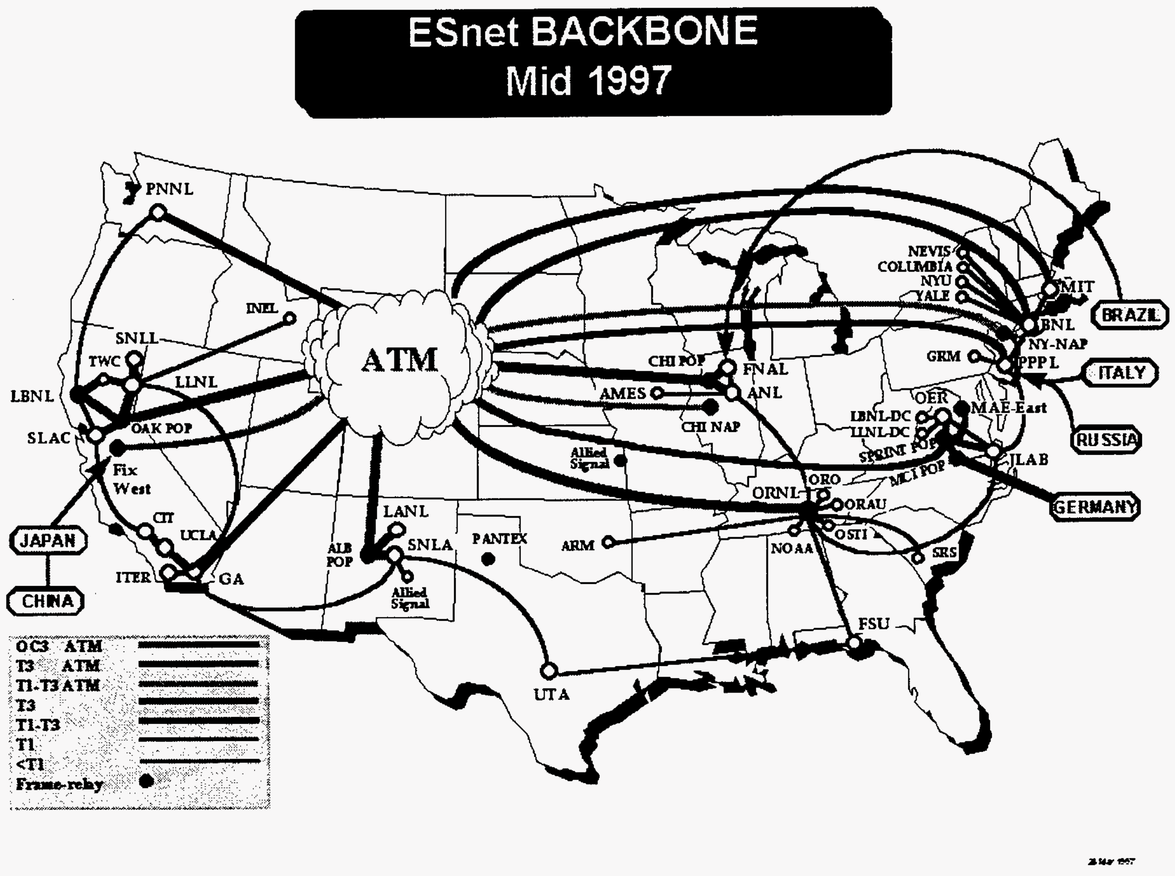

Figure 12: Topology of ESnet.

message size	end-to-end delay (ms)	
	single path	multipath
1.0M	20.940	20.940 (one path) Initial path: LBNL - OAK POP - ORNL
1.2M	22.232	22.230 (two paths) Newly added path: LBNL - SLAC - CIT - UCLA - GA - CHI POP - ANL - ORNL
1.5M	24.167	24.157 (two paths)
2.0M	27.393	27.351 (three paths) Newly added path: LBNL - PNNL - CHI POP - ANL - ORNL
2.3M	29.329	29.264 (six paths) Newly added paths: LBNL - PNNL - SPRINT POP - JLAB - ORNL LBNL - TWC - LLNL - GA - SPRINT POP - JLAB - ORNL LBNL - TWC - LLNL - GA - SNLA - UTA - FSU - ORNL
2.5M	30.619	30.519 (seven paths) Newly added path: LBNL - TWC - LLNL - OAK PROP - SLAC - CIT - UCLA - - GA - SNLA - UTA - FSU - ORNL
10.0M	79.006	77.502 (seven paths)

Table 2: End-to-end delays for connection between LBNL and ORNL.

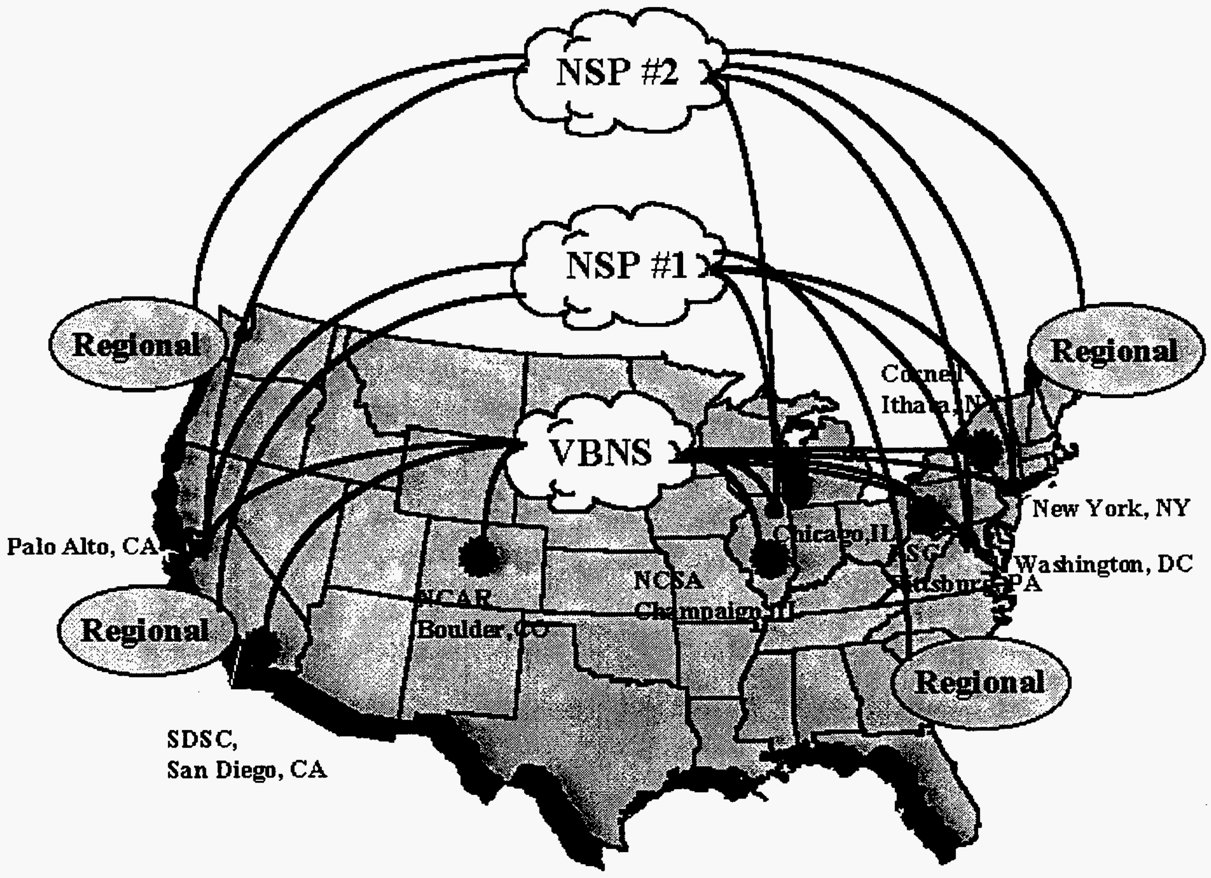

Figure 13: Topology of NSFNET.

message size	single path	end-to-end delay (ms)
1,000	P_1: Palo Alto - VBNS - Washington DC	19.11
10,000	P_2: Palo Alto - NSP#1 - Washington DC	24.95
100,000	P_3: Palo Alto - NSP#2 - Chicago - VBNS - Washington DC	29.45
1,000,000	P_3: Palo Alto - NSP#2 - Chicago - VBNS - Washington DC	74.45

message size	multipath	end-to-end delay (ms)
1,000	P_1	19.11
10,000	P_1 and P_2	23.21
10,000	P_1, P_2 and P_3	27.81
100,000	P_1, P_2 and P_2	62.42

Table 3: End-to-end delays for connection between Palo Alto and Washington DC.

and NSP#2 as the message size is increased. Also, notice that the multipaths achieve lower end-to-end delays for message sizes of the order of 1000 and higher. □

3.6 Delay-Bandwidth Product

For large message sizes, the bandwidth is the main contributing factor to the end-to-end delay, and hence MTP can be solved by classical flow augmentation methods. Such algorithm is not guaranteed to work if delay is also a significant factor in the end-to-end delay, which happens when the two terms of the end-to-end delay are approximately equal, namely $r/B \approx D$ or equivalently $r \approx DB$. Thus, the *Delay-Bandwidth Product* (DBP) is a "first-level" indicator of the range of message size for which the both the link-delay and bandwidths are dominant contributors to the end-to-end delay. The message size must be order of magnitude larger than DBP for the bandwidth to be the dominant factor, and must be order of magnitude smaller than DBP for the delay to be the dominant factor.

We now consider two illustrative cases to compute typical values for DBP, and relate them to some file sizes that arise in certain real-life applications. The first example is a gigabit network (such as MAGIC or ATDNET) that uses OC48 switch with a bandwidth of 2.4 gigabits/sec. Consider such connection across the United States which is about 3000 miles long. The delay of this connection is approximately 32 milliseconds [28]. Then the DBP for this case is about 10^6 bytes, which is typical of high resolution medical or satellite images. The second example is ESnet that currently supports and is expected to support connections over OC3 and OC12 ATM switches, respectively, at the rates of 155 and 622 Mbits/sec. For a 3000 mile connection using this network DBP is of the order of 8×10^4 and 0.25×10^6 bytes, respectively. These sizes are typical of medium to low resolution medical images, images used in face recognition applications, and ccd vision images used in remote robotic applications.

4 Sequence Transmission Problem

We first show that STP is NP-complete, and then solve a special case where the paths are disjoint using a polynomial-time algorithm. Then we present a polynomial-time algorithm that yields an approximate solution to the optimization version of STP.

4.1 Intractability Results

Theorem 4.1 *The sequence transmission problem is NP-complete.*

Proof: First, the problem is NP since it can be solved by checking the condition for each multipath i.e. subset of paths from s to d (each such checking can be done in polynomial time). We now show the theorem by reducing the knapsack problem to the above problem. The knapsack problem is stated as follows [17]: We are given n items indexed by $i = 1, 2, \ldots, n$ such that v_i and c_i denote the *value* and *cost* of ith item, respectively. The problem is to decide if there exist a subset of items indexed by $i_1, i_2, \ldots, i_k$ such that $\sum_{j=1}^{k} v_{i_j} \geq A$ and $\sum_{j=1}^{k} c_{i_j} \leq C$ for two given real numbers A and C. We generate an instance of STP so that its solution exists if and only if the solution to the instance of the knapsack problem exists.

Let $S = \sum_{i=1}^{n} v_i$. We generate a network of $n+3$ nodes denoted by $\{s, 1, 2, \ldots, n, n+1, d\}$ arranged linearly as shown in Fig. 14. Each edge is represented by a parameter pair (v, c) where v and c are bandwidth and delay, respectively. The ith item, $i = 1, \ldots, n$, of the knapsack problem is represented by the pair of nodes i and $i+1$. There are four types of edges in the graph. There is one *outer edge* from s to d with the parameter pair $\left(n \sum_{i=1}^{n} v_i, 0\right) = (nS, 0)$. There are n *lower edges*, one each between node s and node j, $j = 1, 2, \ldots, n$ with the parameter pair $\left(\sum_{i=1}^{n} v_i + v_j, 0\right) = (s + v_j, 0)$ There n *upper edges*, one each between node $j+1$, $j = 1, 2, \ldots, n$ and node d with the parameter pair $\left(\sum_{i=1}^{n} v_i + v_j, 0\right) = (s + v_j, 0)$. There are $2n$ *middle edges* generated as follows. There are two parallel edges between the nodes i and $i+1$, for $i = 1, 2, \ldots, n$ with parameter pairs $\left(k \sum_{i=1}^{n} v_i, t_j\right)$ and $(0, 0)$, which are called the *upper-middle* and *lower-middle* edges, respectively.

Given an instance of the knapsack problem, we generate an instance of STP which requires finding a multipath $MP = \{P_1, P_2, \ldots, P_l\}$, for some l, such that $B(MP) \geq (n+k) \sum_{i=1}^{n} v_i + A$ and $\max_{i,j} |D(P_i) - D(P_j)| \leq C$. Due to presence of the outer edge from s to d with 0 delay, the latter condition can be reduced to $\max_{i} D(P_i) \leq C$, because the outer edge must be included in the solution to realize the required bandwidth term $n \sum_{i=1}^{n} v_i$. Note that the instance of STP can be generated in polynomial time.

Now we show that a solution to the multipath problem exists if and only if the solution to the knapsack problem exists. First, given the solution to the knapsack problem, we generate the solution to the STP as follows. For each item j in the solution of the knapsack problem

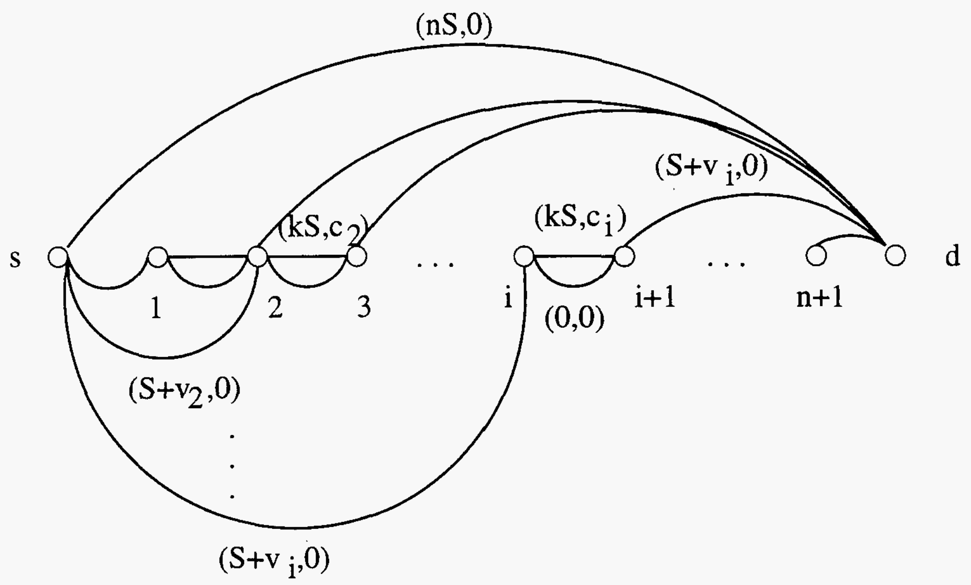

Figure 14: Reduction of knapsack problem to STP.

we generate one path from s to d with bandwidth $S + v_j$ by choosing the path consisting of the lower edge (s, j), the upper-middle edge $(j, j+1)$ and the upper edge from $j+1$ to d. Note that this path has a bandwidth of $S + v_j$ and the delay of c_j. By combining these paths with the outer path, we achieve a total bandwidth of $nS + kS + \sum_{j=1}^{k} v_{i_j} \geq (n+k)S + A$. For every item not in the solution of the knapsack problem, we choose the lower-middle edge with delay 0. Let a be the lowest index of the chosen items in the knapsack solution. Then the longest path in the multipath consists of the lower edge (a, k) followed by the sequence of upper-middle and lower-middle edges corresponding to the items included and not included, respectively, in the solution of the knapsack problem. Clearly, the delay of this path is upper bounded by $\sum_{j=1}^{k} c_{i_j}$. Thus the resultant multipath satisfies both the required conditions.

Consider that a solution to the multipath problem is given. The lower edges contained in the multipath form a cutset that separates s and d, hence they must have been chosen to yield a total bandwidth of at least $(n+k)\sum_{i=1}^{n} v_i + A$. Consider all the upper-middle edges $(j, j+1)$ in the multipath; such j is called a *non-zero nodes.* The items corresponding to to all non-zero nodes yield the solution to the knapsack problem as follows. The delay of the longest path is upperbounded by the sum of c_i's of the edges, which is upperbounded by C, thereby satisfying the first condition of the knapsack problem. We now show that the second condition is also satisfied. First add lower edges of the form (s, j) to each non-zero node j, and such addition still satisfies the bandwidth condition of the multipath, since bandwidth only increases as a result. Now consider the lower edges (s, j), where j in not a non-zero node, i.e. only lower-middle edge $(j, j+1)$ is present in the multipath. The paths containing the sequence of these two edges do not contribute to the bandwidth, since they have zero bandwidth. Thus the sum of bandwidths of all lower edges between s and non-zero nodes is at least $(n+k)\sum_{i=1}^{n} v_i + A$, which implies the sum of v_j's of the non-zero nodes is at least as large as A. Thus the second condition of the knapsack problem is satisfied. □

A number of problems dealing with the computation of single paths with multiple constraints such as delay, jitter, bandwidth, etc., have been shown to be NP-complete in [21, 41, 34]. These results do not imply NP-completeness of STP, since multipaths are allowed here. For example, the problem studied in [41] dealing with minimizing jitter while ensuring bandwidth over a single path cannot be reduced to STP. Since the existence of a single path implies multipath but not vice versa, the former is not subsumed by the latter by the restriction.

Consider a combination of STP and MTP that requires a multipath MP such that: (i) the total delay of MP is no more than τ, and (ii) $t_i - t_j \leq q$ for all $i < j$. As shown in the last section in Lemma 3.4, this problem reduces to MTP under sufficiently large message size r, when the delay condition can be expressed as a condition on the bandwidth. Thus this problem is NP-complete by restriction (note that the problem is in NP since a solution can be verified in polynomial time).

4.2 Approximation Algorithm

We first consider a variation of STP stated as follows:

Disjoint path STP:
We are given the set of *vertex-disjoint* paths, $\{P_1, P_2, \ldots, P_s\}$ and two real numbers B and T. Does there exist a subset of paths $P_{i_1}, P_{i_2}, \ldots, P_{i_k}$ such that (i) $\sum_{j=1}^{k} B(P_{i_j}) \geq B$, and (ii) $\max_{j,l} |D(P_{i_j}) - D(P_{i_l})| \leq T$?.

Since the paths do not intersect, the bandwidth of the multipath $\{P_{i_1}, P_{i_2}\}$, $i_1 \neq i_2$, is $B(P_{i_1}) + B(P_{i_2})$. Assume that $D(P_{i_1}) \leq D(P_{i_1})$ and $D(P_{i_2}) - D(P_{i_1}) \leq T$. Now every path P_{i_3} such that $D(P_{i_1}) \leq D(P_{i_3}) \leq D(P_{i_2})$, ($i_3 \neq i_1$ and $i_3 \neq i_2$), can be added to $\{P_{i_1}, P_{i_2}\}$ to increase the bandwidth, while still guaranteeing the condition on the difference between longest and shortest paths. This idea leads to the algorithm Disjoint-STP. Note that the disjointness of the paths is important in ensuring the condition on the delay: if paths intersect, the length of the longest path can change, since the resultant multipath can be any subgraph (the problem of obtaining an upperbound on the length of longest path is NP-complete [17]).

The algorithm Disjoint-STP solves the disjoint path STP with a time complexity of $O(s^2)$. When the paths are constrained to be disjoint, some of the bandwidth that could otherwise be obtained by combining the paths is not utilized. However, there could be other advantages such as higher fault tolerance, which might provide justification for giving up part of the bandwidth. Algorithms specifically designed for such problems have been proposed in [39].

We now present a heuristic algorithm, Approx-STP, to approximately solve the STP by combining the above algorithm with the algorithm MTA of the last section. The outline of the algorithm is as follows. Starting with the shortest-widest delay path, new paths are added to current MP such that (a) all edges of the added path are removed from G_f, and (b) shortest-widest path in residual G_f is computed. The addition is continued until just before the step that results in exceeding q. Then the multipath is returned as a candidate. Then the initial path of the multipath is removed from current MP, and the the addition of the paths

```
algorithm Disjoint-STP
1. let $P_{(1)}, P_{(2)}, \ldots, P_{(s)}$ be sorted list of paths according to increasing delay $D(.)$
2. for $j = 1$ to $s$ do
3.         compute largest $k$ such that $D(P_{(j+k)}) \leq T + D(P_{(j)})$;
3.         $BW_i \leftarrow \sum_{i=0}^{k} D(P_{(j+i)})$;
4. $BW^* \leftarrow \min_i BW_i$;
5. if $BW^* \geq B$ then return yes
6. else return no
```

Algorithm. 2. Algorithm disjoint path STP.

is continued until the next candidate is found. This process is continued until no more paths are available to add to current MP. Then, from among all candidate multipaths, the one with largest bandwidth is returned. Thus this algorithm provides an approximate solution to the optimization version of the STP. The time complexity of this algorithm is $O(n^2m)$, since there are no more than m paths considered, and in each iteration the computation of shortest-widest path can be computed in $O(n^2)$ time [41].

Consider a sorted list of bandwidths of edges given by $B_{(1)}, B_{(2)}, \ldots, B_{(m)}$. Let q be the size of minimum cut of G, and p be the number of paths in the multipath returned by Approx-STP. The optimal bandwidth is upperbounded by $\sum_{j=0}^{q-1} B_{(m-j)}$, and is lowerbounded by $\sum_{j=1}^{q} B_{(j)}$. When ith path is added to MP, let B_i^P be the bandwidth of this path and B_i^M be the bandwidth of the edge of this path with largest bandwidth. The bandwidth realized by Approx-STP is $\sum_{j=1}^{p} B_i^P \geq \sum_{j=1}^{p} B_{(i)}$. Thus the ratio of the optimal bandwidth to that realized by Approx-STP is upperbounded by $\frac{\sum_{i=0}^{q-1} B_{(m-i)}}{\sum_{i=1}^{p} B_{(i)}}$. Also the total "unutilized bandwidth" by Approx-STP is upperbounded by $\sum_{i=1}^{p}(B_i^M - B_i^P) \leq \sum_{i=0}^{p-1} B_{(m-j)} - \sum_{i=1}^{p} B_{(i)}$, which is small when the variation in bandwidths of edges is small.

5 Conclusions

We formulated two generic routing problems within the framework wherein the bandwidth can be reserved (and guaranteed once reserved) on various links of a communication network. The first problem requires that a message of finite length be transmitted from s to d within τ units of time. The second problem requires that a sequential message of r units be transmitted at a rate of η such that maximum time difference between two units that are received out of order is no more than q. We showed that the first problem cannot be adequately solved by existing methods based on flow algorithms or shortest-widest paths. We proposed a polynomial-time solution by leveraging several analysis and algorithmic methods developed for the classical maximum flow problems. We showed the second problem to be NP-complete, and proposed a polynomial-time approximation algorithm.

This paper constitutes only a first step towards formulating and solving QoS routing problems in a rigorous computational framework from a user or application perspective (as opposed to optimizing network-level parameters). Our main contribution is to exploit multiple paths to provide deterministic bounds for the QoS parameters. Several future research directions can be pursued. It would be interesting to see if the current best complexity of $O(n^3/\log n)$ for the maximum flow problem [9] can be achieved in solving MTP. Also of interest is an algorithm for MTP whose complexity does not depend on edge delays; such an algorithm is called *strongly polynomial* in the literature on flow algorithms. The proposed algorithm is based on the classical Ford-Fulkerson's method; several other flow methods with improved time complexity (e.g. preflow methods [23]) have been extensively studied [2, 18]. It would be interesting to see if these methods can be used to design algorithms for MTP with a lower complexity. More work is needed in designing polynomial-time approximation algorithms for STP and also in investigating performance guarantees of the algorithm proposed in this paper. The applicability of the algorithm MTA to more difficult tasks, such as multicasting and multiple source-destination transmission is of interest.

Concrete computational formulations of other transmission tasks, such as multicasting, and video conferencing, can be attempted to better understand the complexity of these tasks. Also, the proposed framework is based on guaranteeing bandwidth, while there could be several other frameworks that guarantee parameters such as delay upper bounds, and upper bounds on queue lengths, etc.; QoS routing algorithms for such frameworks will be very useful in a number of applications. In particular, the methods of [12, 27] enable us to estimate parameters such as worst-case delay, for networks based on more general frameworks. It would be interesting to obtain a generalization of Lemma 3.2 for multipaths in these general networks, which could be very useful for designing QoS multipath routing algorithms. An integration of QoS services and the best-effort services used in the Internet [20, 30] within a single framework to identify various algorithmic and complexity issues would also be of future interest.

References

[1] R. K. Ahuja, T. L. Magnati, and J. B. Orlin. *Network Flows.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] R. K. Ahuja and J. B. Orlin. A fast and simple algorithm for the maximum flow problem. *Operations Research*, 37(5):748–759, 1989.

[3] C. Aurrecocechea, A. T. Campbell, and L. Hauw. A survey of QoS architectures. *Multimedia Systems Journal*, 1996.

[4] S. Bahk and W. El-Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *Proc. of ACM SIGCOM*, 1992.

[5] D. Bertsekas and R. Gallager. *Data Networks.* Prentice-Hall, 1992.

[6] D. P. Bertsekas, E. M. Gafni, and R. G. Gallager. Second derivative algorithms for minimum delay distributed routing in networks. *IEEE Transactions on Communications*, COM-32:911–919, 1984.

[7] R. Braden, D. Clark, and S. Shenker. Integrated services in the internet architecture, 1994. IETF RFC 1633.

[8] D. G. Cantor and M. Gerla. Optimal routing in a packet-switched computer network. *IEEE Transactions on Computers*, 23(10):1062–1069, 1974.

[9] J. Cherian, T. Hagerup, and K. Mehlhorn. An $o(n^3)$-time maximum-flow algorithm. *SIAM Journal on Computing*, 25(6):1144–1170, 1996.

[10] H. Chu and K. Nahstedt. Dynamic multi-path communication for video traffic. In *Proc. of Hawian Int. Conf. on Systems Sci.*, 1997.

[11] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* McGraw-Hill Book Co., New York, 1990.

[12] R. L. Cruz. A calculus for network delay, Part II: Network analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[13] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the Association of Computing Machinery*, 19(2):248–264, 1972.

[14] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, 1990.

[15] L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[16] R. G. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, 1977.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Co., San Francisco, 1979.

[18] A. V. Goldberg. A new approach to the maximum-flow problem. *Journal of the Association of Computing Machinery*, 35(4):921–940, 1988.

[19] B. Hajek and R. G. Ogier. Optimal dynamic routing in communication networks with continous traffic. *Networks*, 14:457–487, 1984.

[20] C. Huitema. *Routing in the Internet.* Prentice Hall, 1989.

[21] J. M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[22] H. Kanakia, P. P. Mishra, and A. R. Reibman. An adaptive congestion control scheme for real time packet video transport. *IEEE/ACM Transactions on Networking*, 3(6):671–682, 1995.

[23] A. V. Karzanov. Determining the maximal flow in a network by the method of preflows. *Soviet Math. Dokl.*, 15:434–437, 1974.

[24] O. Kyas. *ATM Networks.* International Thomson Computer Press, 1997. Second Edition.

[25] S. Murthy and J. J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *Proc. of IEEE INFOCOM'96*, 1996.

[26] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

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

[28] C. Partridge. *Gigabit Networking.* Addison-Wesley Pub. Co., 1994.

[29] V. Paxson. End-to-end behavior in the internet. In *Proc. of ACM SIGCOM'96*, 1996.

[30] R. Perlman. *Interconnections: Bridges and Routers.* Addison-Wesley, 1992.

[31] N. S. V. Rao and S. G. Batsell. Algorithm for minimum end-to-end delay paths. *IEEE Communications Letters*, September 1997.

[32] N. S. V. Rao, S. G. Batsell, and V. V. Fedorov. On end-to-end delay guarantees for message transmission, 1997. submitted to International Conference on Network Protocols.

[33] A. Segall. The modeliing of adpative routing in data-communication networks. *IEEE Transations on Communications*, COM-25:85–95, 1977.

[34] R. Simha and B. Narahari. Single path routing with delay considerations. *Computer Networks and ISDN Systems*, 24:405–419, 1992.

[35] M. E. Steenstrup, editor. *Routing in Communications Networks.* Prentice Hall, 1995.

[36] J. W. Suurballe. Disjoint paths in a network. *Networks*, 4:125–145, 1974.

[37] J. W. Suurballe and R. E. Tarjan. A quick method for finding shortest pairs of disjoint paths. *Networks*, 14:325–336, 1984.

[38] D. M. Topkis. A k shortest path algorithm for adaptive routing in communications network. *IEEE Transactions on Communications*, COM-36(7):855–859, 1988.

[39] D. Torrieri. Algorithms for finding an optimal set of short disjoint paths in a communications network. *IEEE Transactions on Communications*, 40(11):1698–1702, 1992.

[40] R. Vogel, R. G. Herrtwich, W. Kalfa, H. Wittig, and L. C. Wolf. QoS-based routing of multimedia streams in computer networks. *IEEE Journal on Selected Areas in Communications*, 14(7):1235–1244, 1996.

[41] Z. Wang and J. Crowcroft. QOS routing for supporting resource reservation. *IEEE Journal on Selected Areas in Communications*, 14(7):1228–1234, 1996.

[42] L. Zhang, S. E. Deering, D. Estrin, S. Shankar, and D. Zappala. RSVP: A new resource reservation protocol communications network. Technical Report 95-607, ISI, Univeristy of Southern California, 1995.

M98001424

Report Number (14)	ORNL/CP-95289
	CONF-9703134--
Publ. Date (11)	199711
Sponsor Code (18)	DOE, XF
UC Category (19)	UC-900, DOE/ER

DOE