Multipath Network Flows: Bounded Buffers and Jitter

Tricha Anjali, Alexander Fortin
Department of Electrical and Computer Engineering
Illinois Institute of Technology
Chicago, Illinois
Email: {anjali,afortin}@iit.edu

Gruia Calinescu, Sanjiv Kapoor, Nandakiran Kirubanandan
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois
Email: {calinescu,kapoor,nkiruban}@iit.edu

Sutep Tongngam
School of Applied Statistics
National Institute of Development Administration
Bangkok, Thailand
Email: sutep.tongngam@gmail.com

Abstract—In this paper we address the issue of designing multipath routing algorithms. Multi-path routing has the potential of improving the throughput but requires buffers at the destination. Our model assumes a network with capacitated edges and a delay function associated with the network links (edges). We consider the problem of establishing a specified throughput from source to destination in the network, given bounds on the buffer size available at the destination and a bound on the maximum delay paths are allowed to have. A related problem which we consider is to establish bounds on the delay variance (also called jitter) amongst the paths chosen for the multi-path routing scheme.

We show that the problems are NP-complete and present pseudo-polynomial algorithms based on linear programming. We also propose practical heuristics and present the experimental results on an existing network topology. The results are promising.

I. INTRODUCTION

Traditional algorithms for establishing network connectivity between a source-sink pair find a single path of shortest length (delay) between the source and destination. However, this path may be plagued by congestion problems since this is the path that will always be chosen to forward the total traffic between the node pair. On the other hand, splitting the traffic among multiple paths utilizes the paths simultaneously and reduces the congestion on any given path, while increasing the overall network transmission capacity. Multipath routing protocols work on the principle that higher performance can be achieved by utilizing more than one feasible path [1]. Multipath routing can be effectively used for maximum utilization of network resources by giving the node a choice of next hops for the same destination. Multipath routing has been proposed to take advantage of network redundancy, reduce congestion, and address QoS issues [2], [3]. Traffic engineering, lower delay, increased fault tolerance, and higher security are other compelling reasons that exist for discovering and utilizing multiple paths. The main disadvantage associated with the multipath routing manifests as the packet disordering at the receiver, since the traffic is split into these multiple paths with different latencies creating jitter. Solutions to this problem for the TCP protocol have been proposed in the literature such as [4], [5]. The problem of different latencies on multiple paths can be resolved via either bounding the jitter or by the use of buffers at the destination. The choice of multipaths entails that, to synchronize across various paths, buffers need to be established at the destination so that the data packets can be stored and sequenced appropriately.

In this paper, we formulate optimization problems to model the problems in multipath routing. We want to establish a specified throughput from source to destination in the network, given bounds on the buffer size available at the destination or requirements on the jitter. The paths to be established are required to be of bounded delay. This formulation is general enough and can be applied to networks with different approaches to implement multipath routing. Our model assumes a network with capacitated edges and a delay function associated with the network links (edges). We formulate the optimal Fixed Buffer MultiPath Flow (FBMPF) problem to determine the set of multipaths between the source-destination pair given a fixed buffer size at the destination. The related problem is to establish bounds on the delay variance (also called jitter) amongst the paths chosen for the multi-path routing scheme, termed the Bounded Jitter Multipath Flow (BJMPF) problem. We show that the problems are NP-complete and design a minimum cost network flow problem which models the bounded-buffer problem. This leads to a pseudo-polynomial algorithm, based on linear programming. For the bounded-jitter problem, we present a faster pseudo-polynomial algorithm which closely approximates the required throughput.

The solution of the linear program brings out an interesting feature associated with the FBMPF problem. We found that it may be beneficial to use long paths in the network in order to satisfy the buffer bounds, since paths which have similar delays would require lower buffer size. Long paths can be
obtained by accruing the delay in a cycle one or multiple times. Thus, the artificial construct of choosing non-simple paths (paths repeating nodes, or in other words, with cycles) can lead to smaller buffer size at the receiver. The explicit routing capability provided by technologies such as MultiProtocol Label Switching (MPLS) can be used to establish these non-simple paths in a network. Before the actual data transfer, a Label Switched Path (LSP) is established between the end-hosts. During the establishment of the LSP, label translation tables are created at each intermediate hop. The data is then forwarded on the LSP by performing a label translation.

Related problems have been studied, but we are not aware of any work dealing explicitly with buffer size constraints. In the seminal work [6], the authors address multipath routing issues as a non-linear optimization problem. They are concerned with delay and loss rate in an adaptive setting, and their cost per link is a function of the traffic on that link, an approach which ignores the buffers at the receiver. A fixed number (k) of actual paths are constructed in [7] to maximize the total flow (throughput). We also maximize flow, but we consider together the buffer constraint, and allow softer (dictated by total cost) constraint on the paths.

The paper is organized as follows. In Section II, we introduce the mathematical model for the FBMPF problem, as well as the related Bounded Jitter MultiPath Flow (BJMPF) problem. Unfortunately, both problems are NP-Complete as we prove in Section III. In Section IV, we provide an approximate solution for BJMPF followed by the approximate FBMPF solution in Section V. The solutions we present use related ideas. In Section VI, we discuss the heuristic implementation followed by the experimental results in Section VII and conclusions in Section VIII.

II. Multi-Path Problems

In this section we define related network routing problems that are of interest:

We first define the bounded-latency network flow problem: Consider a digraph $G = (V, E)$, with $n$ nodes and $m$ edges (links), a capacity function $c(u, v) : E \rightarrow R^+$, a length (delay) function $l(u, v) : E \rightarrow R^+$, a source $s$, and a sink $t$. Note that we do not assume that delays or capacities across the edges to be equal.

For a flow path $P$ from $s$ to $t$, define the total length (delay) path, $L(P)$, as:

$$L(P) = \sum_{(u, v) \in P} l(u, v)$$

that is, the total length (delay) of the path is equal to the sum of the lengths of the edges on the path. The multipath flow problem (MPF), is to generate a set of $s-t$ paths $P_1, P_2, ..., P_k$ with corresponding flow values $f_1, f_2, ..., f_k$ such that the following conditions are satisfied

$$\sum_{i=1}^{k} f_i = \gamma \quad \text{(demand constraint)}$$

$$\sum_{(u, v) \in P_i} f_i \leq c(u, v); \forall (u, v) \in E \quad \text{(capacity constraints)}$$

$$L(P_i) \leq L; \forall P_i \quad \text{(path length (delay) constraint)}$$

where $\gamma \in Z^+$. An instance of the multipath flow problem is denoted as $MPF(N, \gamma, L)$, where $N$ is the input network characterized by $N = (G, c, l, s, t)$.

Additional constraints for jitter and buffer size can be imposed on the length (delay) of the paths selected as outlined below.

- **Bounded Jitter MultiPath Flow (BJMPF):** The delay variance (jitter) on the paths selected, significantly affects the buffer size at the sink. Thus, it is advisable to bound the maximum allowed delay variance ($\delta$) for the paths chosen, that is

$$L(P_i) - L(P_j) \leq \delta; \forall P_i, P_j$$

- **Fixed Buffer MultiPath Flow (FBMPF):** Suppose paths $P_1, P_2, ..., P_k$ carry flow $f_1, f_2, ..., f_k$, where $P_1$ is the path of maximum length (or delay), in order to achieve the desired throughput. The schedule to transmit the flow would require that a unit of flow pushed on $P_i$ at time $t$ would arrive at the destination at time $t + L(P_i)$. Data packets transmitted at the rate $f_i$ on paths $P_i$, $1 \leq i \leq k$ would accumulate at the destination within this time frame. The number of packets arriving between times $t + L(P_i)$ and $t + L(P_1)$ on path $P_i$ is given by $f_i(L(P_1) - L(P_i))$. These packets would require to be buffered at the destination till the packet on the path $P_1$ arrives at the destination, since these packets have arrived out of sequence. Assuming that the buffer size at the sink is given as a constant, we achieve the following constraints:

$$\sum_i f_i \ast (L(P_1) - L(P_i)) \leq B.$$  \hspace{1cm} (1)

Here $B$ is the buffer size at sink, and $P_1$ is the path with maximum length (delay).

Due to the nature of the problem, we can get solutions involving cycles. In the following example, we show that how cycles can significantly improve the performance.

**Construction**

![Fig. 1. Why cycles are “good”](image)
In the figure we have a network with all edges capacities 1, and length (delay) as noted on each edge. The source is $s$, the sink is $t$, and the flow demand is 2. Consider the BJMPF problem with jitter 0. A flow of value 2 is achieved by using the paths $s, x, y, z, x, t$ and $s, u, t$, each of length (delay) 5 and carrying one unit of flow. No solution with jitter 0 exists with only simple paths (or, in other words, without cycles). Similarly, in the same example, if the buffer size is 0, the only solution for FBMPF with flow demand 2 is the one given above with non-simple paths.

Also, as opposed to classical max-flow problems, for neither BJMPF nor FBMPF it is true that a solution carrying integral flows always exists.

Construction

Such a counterexample (see Figure 2) is given below: the network $N = (G, c, l, s, t)$ with $G = (V, E)$ has $V = \{s, t, v, x, y, z, v', x', y', z'\}$, $E = \{sx, sy, sx', sy', xy, x'y', yz, y'z', xv, x'v', vt, zt, z't\}$, all edges have $c(e) = 1$, and all edges $e$ have $l(e) = 1$ except $l(xv) = l(x'y') = 3$ and $l(yz) = l(y'z') = l(xy) = l(x'y') = 2$. If $\gamma = 3$, and either $\delta = 0$ (for BJMPF) or $B = 0$ (for FBMPF), then the only solution uses each of the following six paths carrying 0.5 units of flow: $P_1 = s, x, v, t$, $P_2 = s, x', v', t$, $P_3 = s, x, y, z, t$, $P_4 = s, x', y', z', t$, $P_5 = s, y, z, v, t$, and $P_6 = s, y', z', v', t$. One can check this by inspection, noting that the paths $s, x, v, t$ and $s, x', y', v', t$ - of which again at least one is needed in any solution with demand $\gamma = 3$, have length 4, shorter than the length of the paths $s, x, v, t$ and $s, x', y', v', t$ - of which at least one is needed in any solution with demand $\gamma = 3$.

In the counterexample above all capacities and lengths are non-negative integers bounded by 3. As we will see later (Sections IV and V) we can solve solve such instances in polynomial time.

III. NP-COMPLETENESS

In this section, we show that the problem of finding multi-paths with bounded jitter (or fixed buffer) is NP-Complete.

Theorem 1: BJMPF and FBMPF are NP-Complete.
Proof: For both problems we use the same reduction, from the well known Partition problem [8]: given a multiset $X = \{x_1, x_2, \ldots, x_n\}$ of integers, the decision problem asks is there is a partition of $X$ into $A$ and $D$ such that $\sum_{i \in A} s_i = \sum_{i \in D} s_i$.

We construct our network as follows: $V = \{v_0, v_1, \ldots, v_n\}$, $s = v_0$, $t = v_n$, and for all $i = 1, 2, \ldots, n$, we put two edges $e_i$ and $e'_i$, both with tail $v_{i-1}$, head $v_i$, and capacity 1. We set $l(e_i) = 1 + x_i$ and $l(e'_i) = 1$. We set $\gamma = 2$, and for BJMPF we set $\delta = 0$, while for FBMPF $B = 0$. See Figure 3 for an example.

![Fig. 2. Example without integral solutions](image)

Fig. 3. Example of the NP reductions. If the instance of Partition is \{2, 3, 4, 5, 10\}, then we get the graph above with $s = v_0$ and $t = v_n$. This instance can be partitioned into $A = \{3, 4, 5\}$ and $D = \{2, 10\}$, and the multipath solution uses the path which uses the first and last upper edges (with the others lower), and the path with first and last lower edges (with the others higher) - making both jitter and buffer size 0.

If there is a partition of $X$ such that $\sum_{i \in A} s_i = \sum_{i \in D} s_i$, we use two paths $P_1$ and $P_2$ defined as follows: if $i \in A$, we put $e_i$ in $P_1$ and $e'_i$ in $P_2$, otherwise ($i \in D$), we put $e_i$ in $P_2$ and $e'_i$ in $P_1$. The $l(P_i) = n + \sum_{i \in A} x_i$, while $l(P_2) = n + \sum_{i \in D} x_i$, satisfying both the jitter and the buffer constraint.

Conversely, any solution that satisfies the demand constraint, and either the jitter or the buffer constraint, must consist of a set of paths $P_i$, for $i \in \{1, 2, \ldots, q\}$ for some $q \geq 2$, of the same length (delay) $L$. Every edge in the network must be used at full capacity as it appears in some min-cut. Then

$$\sum_{i=1}^{q} f_i l(P_i) = \sum_{e \in E} l(e) = 2n + \sum_{i=1}^{n} x_i.$$ 

Since $\sum_{i=1}^{n} f_i = 2$ and all the paths have the same length (delay), we obtain $l(P_i) = n + (1/2) \sum_{i=1}^{n} x_i$. Now if we let $A = \{i | 1 \leq i \leq n$ and $P_i$ uses $e_i\}$, it is immediate that $\sum_{i \in A} x_i = (1/2) \sum_{i=1}^{n} x_i$. Defining $S = A \cup A$ gives us $\sum_{i \in A} x_i = \sum_{i \in D} x_i$; thus the Partition instance is feasible.

Note: If the network is undirected, or symmetric (each edge has an opposite edge with the same delay and capacity), the construction above (or the symmetric variant) still works as one cannot ship two units of flow from $s$ to $t$ while using positive flow from some $v_j$ to $v_{j-1}$.

It turns out that NP-Completeness is indeed a consequence of having large values of the delay function $l(e)$ for $e \in E$. However, as we show below, the problems do admit pseudo-polynomial algorithms, precisely polynomial time algorithms when the delay function can only assume values from a set of small integers.
IV. SOLUTION TO BOUNDED JITTER MULTIPATH FLOW

By rounding we assume that the delay function is given as \( l : E \to \{1, 2, \ldots, k\} \). The rounding can cause small errors when computing the jitter (or, in the next section, buffer size), but is necessary to obtain efficient solutions. We obtain an algorithm polynomial in \( n, m, k, \) and \( U \), where \( U = \max_{e \in E} c(e) \).

First thing we do, is fix a value \( L \) and insist that all paths \( P_i \) used by the solution have length (delay) between \( L - \delta \) and \( L \). In fact we do not know \( L \) and must try many values from a range - we delay for the moment the calculation of this range.

We denote by \( \mathcal{P}(L, \delta) \) the sets of such paths. With the length (delay) restriction above, the problem becomes: Given a network \( N = (V, E, c, l, s, t) \) and integers \( L, \delta, \gamma \), find an integer \( q \) and for each \( i \in \{1, 2, \ldots, q\} \), a path \( P_i \in \mathcal{P}(L, \delta) \) and positive flow value \( f_i \), such that:

\[
\sum_{i=1}^{q} f_i = \gamma
\]

\[
\sum_{i=1}^{q} f_i m(i, e) \leq c(e), \quad \forall e \in E
\]

where \( m(i, e) \) is the number of times path \( P_i \) uses edge \( e \). It is beneficial to rephrase this problem as the following "packing" linear program, with exponentially many variables:

\[
\text{maximize } \sum_{P \in \mathcal{P}(L, \delta)} f_P
\]

subject to

\[
\sum_{P \in \mathcal{P}(L, \delta)} f_P \cdot m(P, e) \leq c(e), \quad \forall e \in E
\]

\[
f_P \geq 0 \quad \forall P \in \mathcal{P}(L, \delta)
\]

where \( m(P, e) \) is the number of times path \( P \) uses edge \( e \). If the objective function is at least \( \gamma \), the paths \( P \) with \( f_P > 0 \) give a feasible solution to BJMPF.

This linear program can be solved in time polynomial in \( n, m, L, \log U \) by doing a layered construction as in [9] and in [7]. While this program resembles the classical maximum flow problem, we do not have a combinatorial method for solving it exactly. Among other differences, as opposed to maximum flow, it is not true that if all the input capacities are integers the optimum objective is an integer.

If we are willing to settle for an \( 1 - \varepsilon \) approximation for the objective function (so instead of \( \gamma \) units of flow, we only get \( \gamma(1 - \varepsilon) \), then we can apply the Garg-Könemann algorithm [10] as explained next.

The linear program above is a packing LP. In general, a packing LP is defined as

\[
\max \left\{ c^T x \mid Ax \leq b, \ x \geq 0 \right\}
\]

where \( A, b, \) and \( c \) have positive entries; we denote the dimensions of \( A \) as \( m \times n \). In our case the number of columns of \( A \) is prohibitively large (exponential in number of edges).

The algorithm of [10] assumes that the LP is implicitly given by a vector \( b \in \mathbb{R}^m \) and there exists an oracle which finds the column of \( A \) minimizing a so-called length function. The length of column \( j \) with respect to LP in Equation (4) is defined as

\[
\text{length}_P(j) = \frac{\sum_{e \in P} \hat{c}(j)}{c(j)}
\]

for any positive vector \( y \).

This means, for our particular LP, that we must find, for a vector \( y(e) : e \in E \) a path \( P \) minimizing \( y(P) := \sum_{e \in P} y(e) \) where we make the convention that \( P \) is a multiset and each edge \( e \) is counted every time \( P \) uses it. Finding such a \( P \) can be accomplished by a shortest path algorithm in the following layered graph. See Figure 4 for an example. For each \( i = 0, 1, 2, \ldots, L \), make \( V_i \) a copy of \( V \). Construct a new directed graph \( \tilde{N} \) with \( V = V(N) = \bigcup_{i=0}^{L} V_i \). Call \( u_i \) the copy of node \( u \) from \( V_i \). For an edge \( e \in E \) with tail \( u \) and head \( v \) put in \( \tilde{E} = E(N) \), for each \( i = 0, 1, \ldots, L - l(e) \), an edge \( e_i \) with tail \( u_i \) and head \( v_{i+l(e)} \). Set \( y(e_i) = y(e) \) or else we could simple use max-flow ignoring the delay constraints, and then decompose the flow into simple paths of lengths (delays) between 1 and \( nk \). Now find a shortest path, with respect to \( y \), from \( s_0 \) to \( t_L \) in \( \tilde{N} \).

Since \( \tilde{N} \) is acyclic, the running time of the shortest path algorithm is \( O(mL) \). The running time of the Garg-Könemann algorithm [10] is \( O((1/\varepsilon)^2 m T_{orc}) \), since \( m \) is the number of constraints (other than non-negativity) and \( T_{orc} \) is the time required to compute the minimum length column - in our case \( O(mL) \). Thus the total running time, for one value of \( L \), is \( O((1/\varepsilon)^2 m^2 L) \).

It remains now to bound the range of possible values of \( L \). First, we can assume \( \gamma > \min_{e \in E} c(e) \), or else we can use one single \( s - t \) path. Second, we can assume \( \delta \leq nk \), or else we could simple use max-flow ignoring the delay constraints, and then decompose the flow into simple paths of lengths (delays) between 1 and \( nk \). Now, every path \( P_i \) in a feasible solution has length (delay) at least \( L - \delta \) and uses \( f_i(L - \delta) \) length-capacity. The total length-capacity of the graph is \( \sum_{e \in E} \hat{l}(e)c(e) \leq mk \sum_{e \in E} c(e) \). As \( \sum_i f_i = \gamma \), we obtain \( (L - \delta)\gamma \leq mk \sum_{e \in E} c(e) \). We conclude \( L \leq nk + mk \min_{e \in E} c(e) \).

Note that \( \frac{\sum_{e \in E} \hat{l}(e)c(e)}{\min_{e \in E} c(e)} \leq mU \). Using \( \sum_{L=1}^{m-k^2} L = O((m-k^2)^2) \), we obtain an algorithm producing \( (1 - \varepsilon)\gamma \) flow with total running time \( O((1/\varepsilon)^2 m^2 (m-k^2)^2 U^2) = O((1/\varepsilon)^2 m^2 k^2 U^2) \).
V. Solution to Fixed-Buffer-MultiPath-Flow

Again we assume that the delay function is given as \( l : E \to \{1, 2, \ldots, k\} \), i.e. it only takes as value small integers.

First, we fix a value \( L \) and guess that the longest path used has length (delay) exactly \( L \). In fact we do not know \( L \) and must try many values from a range - we delay for the moment the calculation of this range.

We denote by \( \mathcal{P}(L) \) the sets of such paths. With the length (delay) restriction above, the problem becomes: Given a network \( N = (V, E, c, l, s, t) \) and integers \( L, \gamma, B \), find an integer \( q \) and for each \( i \in \{1, 2, \ldots, q\} \), a path \( P_i \in \mathcal{P}(L) \) and positive flow value \( f_i \) such that:

\[
\begin{align*}
    l(P_i) &= L \quad (5) \\
    \sum_{i=1}^{q} f_i &= \gamma \quad (6) \\
    \sum_{i=1}^{q} f_i (L - l(P_i)) &\leq B \quad (7) \\
    \sum_{i=1}^{q} f_i m(i,e) &\leq c(e), \quad \forall e \in E \quad (8)
\end{align*}
\]

for all \( e \in E \), where \( m(i,e) \) is the number of times path \( P_i \) uses edge \( e \).

We solve the above problem by constructing a linear program which resembles the minimum cost flow problem. Similar constructions appeared in [9], [7] for variations of max-flow.

To describe and construct the linear program, we first construct from \( N \) a network \( \hat{N} \) as follows. \( V = V(\hat{N}) = \cup_{i=0}^{L} V_i \). We call \( u_i \) the copy of node \( u \) from \( V_i \). For an edge \( e \in E \) with tail \( u \) and head \( v \) put in \( \hat{E} = E(\hat{N}) \), for each \( i = 0, 1, \ldots, L - l(e) \), an edge \( e_i \) with tail \( u_i \) and head \( v_{i+l(e)} \). Also add edges \( g_j \), for \( j = 0, 1, 2, L - 1 \), with tail \( t_j \) and head \( t_L \); these are the only edges with cost: \( \text{cost}(g_j) = L - j \). The goal is to ship at minimum cost \( \gamma \) units of flow from \( s_0 \) to \( t_L \) in \( \hat{N} \), subject to joint capacity constraints as follows. For each edge \( e \in E \), and each \( i = 0, 1, 2, \ldots, L - l(e) \), we have \( f(e_i) \) as the flow on edge \( e_i \) of \( \hat{N} \). Also, \( f(g_j) \) is defined for the \( g_j \) edges above. We use the following notation: for a node \( u, \delta^+(u) \) is the set of edges with tail \( u \), and \( \delta^-(u) \) is the set of edges with head \( u \). The linear program is:

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=0}^{L-1} (L - j) f(g_j) \\
\text{subject to} & \quad \\
\sum_{e \in \delta^-(\hat{e})} f(\hat{e}) &= \sum_{e \in \delta^+(\hat{e})} f(\hat{e}) \quad \forall \hat{e} \in \hat{E} \setminus \{s_0, t_L\} \quad (9) \\
\sum_{e \in \delta^+(s_0)} f(\hat{e}) &= \gamma \quad (10) \\
\sum_{j=0}^{L-l(e)} f(e_i) &\leq c(e) \quad \forall e \in E \\
f(\hat{e}) &\geq 0 \quad \forall \hat{e} \in \hat{E} \quad (12)
\end{align*}
\]

Constraints 9, 10, and 11 are flow conservation, respectively total flow out (replacing constraint 6 ) and joint capacity constraint (replacing constraint 8). More precisely:

**Claim 1:** Assuming the longest path has length (delay) exactly \( L \), the above linear program has objective value at most \( B \) if and only if the FBMPF instance has a feasible solution.

**Proof:** Assume first that the FBMPF has a feasible solution: there is an integer \( q \) and for each \( i \in \{1, 2, \ldots, q\} \), a path \( P_i \in \mathcal{P}(L) \) and positive flow value \( f_i \) satisfying Equations 5, 6, 7 and 8.

For a \( P \in \mathcal{P}(L) \), let \( c(P,j) \) be the \( j \)-th edge of \( P \). We use the convention that summation over an empty set is 0. In the linear program, for each \( e \in E \) and \( j = 0, 1, \ldots, L - l(e) \), define \( Q(e,j) = \{ P \in \mathcal{P}(L) \mid \exists r > 0 \ (c(P,r) = e \text{ and } \sum_{j=0}^{r-1} l(e(P,r)) = j \} \) and \( f(e_j) = \sum_{P \in Q(e,j)} f_i \). In the feasible solution, \( Q(e,j) \) represents the set of paths in \( \mathcal{P}(L) \) that use edge \( e \) after a delay of \( j \) time units. In the linear program, \( f(e_j) \) is set to be the value of the flow carried by these paths through edge \( e \) at the \( j \)-th time step. For each \( j = 0, 1, \ldots, L - 1 \), define \( f(g_j) = \sum_{P \mid \hat{P}_1 = f_i} f_i \). Then one can easily check that all the constraints of the linear program are satisfied, and that the objective value is at most \( B \).

Now assume that the linear program has a feasible solution with objective function at most \( B \). Since the flow conservation constraints are satisfied, this solution can be decomposed in \( \hat{N} \) into a set of path \( P_i \). Each \( P_i \) directly correspond to paths of \( N \) after, if necessary, we remove the last edge of \( P_i \) if that edge is of type \( g_j \), for \( j \in \{0, 1, \ldots, L - 1\} \). Simple calculations show that Equations 5, 6, 7 and 8 are satisfied by these values \( P_i \).

It remains now to bound the range of possible values of \( L \). First, we can assume \( \gamma > \min_{e \in E} c(e) \), or else we can use one single \( s-t \) path. Second, we can assume \( B \leq nk\gamma \), as any max-flow using simple paths has length (delay) at most \( nk \).

Thus:

\[
\sum_{i} f_i (L - l(P_i)) \leq B \leq nk\gamma. \quad (13)
\]

Assume now we have a feasible solution with paths \( P_i \) and flows \( f_i \). We have that \( \sum_i f_i(l(P_i)) \) cannot exceed the total length-capacity of the graph, \( \sum_{e \in E} l(e) c(e) \), which is upper-bounded by \( mk \sum_{e \in E} c(e) \). Thus:

\[
\sum_{i} f_i(l(P_i)) \leq mk \sum_{e \in E} c(e). \quad (14)
\]

From Equation 13 and the previous equation we obtain:

\[
\sum_{i} L f_i \leq nk\gamma + mnk \sum_{e \in E} c(e). \quad (15)
\]

With \( \sum_i f_i = \gamma \), we conclude that for a feasible solution

\[
L \leq nk + mk \sum_{e \in E} c(e)/\gamma = O(m^2 k U), \quad (16)
\]

where as defined before, \( U = \max_{e \in E} c(e) \). Thus FBMPF has an algorithm polynomial in \( k, m, U \).
VI. HEURISTIC SOLUTION

Because of the NP-complete nature of the BJMF and FBMPF problems, we propose a heuristic algorithm to find feasible paths that satisfy the constraints. The interesting aspect of heuristics is that they allow us to get faster, real-time solutions with results close to the optimal, besides the fact that we can work with larger network topologies and larger parameters than the optimal program.

Our heuristics aim to maximize the routed flow subject to either jitter or buffer constraint - in fact it can handle both constraints if needed.

The proposed heuristics are greedy in nature. Given the network topology, we use Shier’s algorithm [11] to find the k shortest paths in the network. Repeated nodes are allowed in each of the paths. This algorithm sorts the paths in the increasing order of their costs. The link delays are used as the cost for the links. So, the paths obtained by this algorithm are in increasing order of their delays. For the first heuristic, proceeding in a greedy manner, we start selecting the paths from the first one. We push the maximum flow on this path and then consider the next path in the list. The paths will be selected such that the jitter and buffer constraints are respected. Hence, the most important constraint of the heuristic is to keep the delay as low as possible by considering the shortest paths first, and then the capacity constraint comes into account and allows the flow to be sent to the target node. By proceeding in this way, if the sum of the capacities of the paths is lower than the total demand γ, then we will not be able to send the amount of flow desired and the demand constraint will be approximately respected.

For the second heuristic, we reorder these paths in order of their capacity. In other words, we select paths with higher capacities first. This is done with the objective of selecting high capacity paths first.

The pseudo-code for the first heuristic is given next. It can be minimally modified for the second heuristic. The jitter constraint applies in lines 5-7, where Length_Constraint is the length of the shortest path among our collection of paths plus the δ, the maximum delay variance allowed. The buffer constraint as given in the description of FBMPF is enforced in lines 8-10. Note that P_1 from the FBMPF description, being the longest path, must be updated after each added path. If we want to enforce only one of the jitter or buffer constraints, we make buffer size or Length_Constraint very large.

Algorithm 1: FBMPF
1: Find the k shortest paths in the network using length (delay) as cost
2: counter=1
3: while Paths are available and not all flow has been routed do
4: Select the path in the list at location counter
5: if Length of this path > Length_Constraint then
6: break
7: end if
8: if Buffer constraint is violated then
9: break
10: end if
11: Add this path to list of chosen paths
12: Reduce the demand to be routed by the amount send on this path
13: Decrease every link’s capacity by this amount as well
14: Increase counter
15: end while

VII. SIMULATION

We have implemented the optimal solution and the heuristics on a Java platform. We have used two topologies to demonstrate the results. The first one is a simple 6 node topology for easy verification of the results and second one is a real topology, obtained from [12]. This network called GEANT is a pan-European backbone which connects Europe’s national research and education networks. The network has 33 nodes and 94 links. The link delays are estimated based on the length of the links and are assigned between 1 and 30 msec.

In the figures 5 and 6, we show the total flow that can be sent between source (node number 1) and destination (node number 6 for the first topology and 33 for the second) as a function of the buffer size at the receiver, obtained with the two heuristics, for the two topologies respectively. The first data point where buffer is 0 corresponds to the least cost single path flow, currently being used in TCP/IP protocols. As can be seen, increasing the buffer size has substantial impact on the flow in the network. For the GEANT topology, using a buffer of approximately 0.22Gb, we are able to achieve throughput of 20Gb/s which is about twice as compared to the single shortest path throughput of 10Gb/s. The optimal program, on the other hand, is able to achieve a higher throughput with no buffer. This is because the optimal program allows cycles in the paths which can adjust the delay of the paths such that all paths have similar delay. The heuristic program typically runs in less than 1 sec whereas the optimal program takes about 1 hour to execute.

VIII. IMPLEMENTATION ISSUES

We expect that the problems mentioned in this paper will be applied to the core of the network. Typically, the access nodes have only single connection to the network and multipaths are not possible. Also, typically the access links are the bottlenecks in the end-to-end paths. So, these algorithms are only applied to the network core where multiple paths
are included for redundancy and efficiency, and there is not bottleneck link.

Currently, the algorithms are designed to be executed by a central entity responsible for the network. If there is a distributed way of propagating information about the network state to all the nodes, then the proposed algorithms can be deployed in a distributed manner.

Also, one assumption is that the network delays are not dynamic. In other words, during the calculation of the paths, we assume that the delay values do not change. We can incorporate slight variability depending on the update frequency. As new requests arrive in the network, updated link delay values can be used to calculate the current paths for the flows.

**IX. Conclusions**

In this paper, we have motivated the multipath routing problem with a buffer constraint at the destination. This problem is based on the requirement on the destination to buffer the packets received along paths with lower delay until the flows on other paths have arrived. The larger the difference in delays of the paths, larger the buffer required at the destination.

We have shown optimal theoretical results to obtain optimal buffer size for a desired throughput. We have also presented a greedy heuristic which has been experimentally shown to substantially improve the throughput as compared to the current TCP/IP protocol.

Our theoretical solutions give polynomial-time approximations for the variations of BJMF and FBMPF when we have a fixed number of source-sink pairs. In the future, we will develop optimal and heuristic solutions for the case when multiple users are requesting service from the network.

**REFERENCES**