QoS-Constrained Multi-path Routing for High-End Network Applications

Xiaomin Chen, Mohit Chamania, Admela Jukan
Technische Universität Carolo-Wilhelmina zu Braunschweig
Email: chen, chamania, jukan@ida.ing.tu-bs.de

André C. Drummond, Nelson L. S. da Fonseca
Institute of Computing, State University of Campinas
Email: andred, nfonseca@ic.unicamp.br

***Abstract*—We present a multi-path computation algorithm to find a set of paths for the given demand and use *Integer Linear Programming* (ILP) approach to derive an optimal solution to maximize the achievable bandwidth and minimize the required memory size. We discuss the applicability of the proposed methods in multi-domain settings and present three application schemes in a multi-homing environment. Numerical and simulation results show that the proposed multi-path routing algorithm has better performance in bandwidth as compared to traditional single path routing. Especially the multi-path multi-domain application schemes can provision more bandwidth even with partial visibility. Therefore, it can be used as a viable solution to accommodate the emerging high-end applications with extremely high bandwidth requirements in current networks.**

I. Introduction

High-end applications have become dependent on global networks, with supercomputing and data-storage facilities widely distributed. Whereas only a few of transport networks based on optical transmission technologies are dedicated to scientific research, the nature of the most shared public transport networks makes it imperative to design innovative and efficient data transfer scheme, which can maximize the achievable bandwidth while ensuring end-to-end QoS performance. It is known that multi-path routing can provide more aggregate bandwidth [1], and thus allow applications with high bandwidth requirements to be accommodated over multiple paths, which otherwise would be hard or even impossible to do over single paths. Although bandwidth should be abundant in the current networks, it is still expensive. This makes multi-path routing in the existing connection oriented transport networks, such as carrier-grade Ethernet [2] a viable candidate for the high-end applications with end-to-end QoS requirements.

Although it is expected that high-end applications can benefit from multi-path routing, with more available bandwidth and lower blocking [1], the challenges arise they have some certain QoS requirements. For instance, multiple paths may present differences in the end-to-end delay of each path, which causes jitter at the destination, a phenomena commonly referred to as the differential delay problem [3]. Such scenarios require that data coming from different flows needs to be buffered till the flow from the path with the highest delay arrives for re-ordering the data correctly. This solution poses another problem, as high speed memory (working at the line rate of Gbps) is extremely expensive, and therefore cannot be over-provisioned. To illustrate the tradeoff, take the case

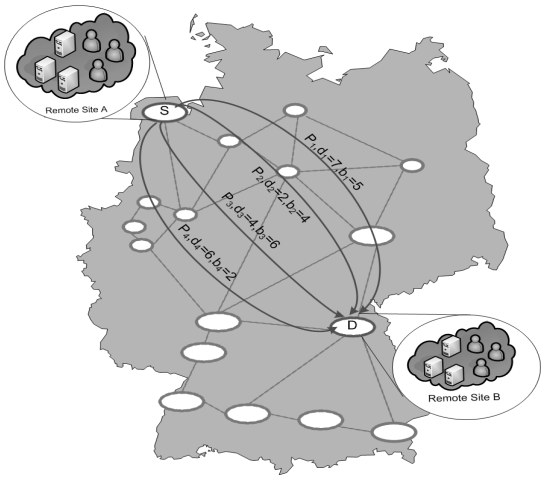

Fig. 1. An Example of Data Transmission over Multiple Paths.

of the multi-path flow as shown in Fig. 1. Between source (S) and destination (D), there are four paths P_1,P_2,P_3, and P_4, each with a different available capacity b and path delay d. Requested is a dataset transmission from S to D with size of 40 units. In the single shortest path routing, P_2 is chosen, which results in the end-to-end transmission time of $40/b_2 + d_2 = 12$ units. In the case of data transmission over two paths, P_1 and P_3, e.g. to maximize the achievable bandwidth. The resulting transmission delay is then 11, calculated by $\max\{20/b_1 + d_1, 20/b_3 + d_3\}$. The required memory size required in this case is given by $(d_1 - d_3) \cdot b_3 = 18$ units. When minimization of differential delay is considered, P_1 and P_4 will be used as the optimal solution with a memory cost $(d_1 - d_4) \cdot b_4 = 2$ units. The end-to-end transmission time in this case is $\max\{20/b_1 + d_1, 20/b_4 + d_4\} = 16$ units. This simple example illustrates that high-end applications require carefully designed multi-path routing algorithms which can optimize both the achievable bandwidth and the differential delay problem.

In this paper, we present the multi-path routing problem as a profit maximization problem with the objective of maximizing the achievable bandwidth and minimizing the memory size caused by the differential delay. The profit from provisioning a connection is determined as a function of the total bandwidth allocated for the problem, and the amount of memory used

978-1-4244-3968-3/09/$25.00 ©2009

to re-order the flows in the different paths. The problem is modeled as an Integer Linear Program (ILP), and the solution helps us better understand the advantage of using multi-path routing over conventional single path routing in the context of carrier Ethernet transport networks. We present an multi-path calculation algorithm which can find the first N shortest widest paths in the network. The routing algorithm can be used to find the paths with the highest bandwidths first, which is useful when only a limited number of paths are found before multi-path optimization is initiated. We then apply the proposed multi-path routing algorithm in multi-homing networks and present three multi-path multi-domain routing schemes, i.e., multi-domain routing with full visibility, multi-domain routing with partial visibility and multi-path routing a single preferred domain.

The reminder of the paper is organizes as follows. In Section II , we give an overview of the related work. Section III presents the multi-path computation and propose an ILP approach for optimizing performance such as the cost of memory and achievable bandwidth. It also identifies the applicability of multi-path multi-domain routing schemes in multi-homing environment. Section IV is the performance evaluation for the proposed algorithms and Section V draws the conclusions.

II. Related work

The use of multi-path routing for provisioning high bandwidth connections has been studied extensively while only a few works considered the differential delay issues. Cidon et al. [1] highlighted the advantages of multi-path routing over single path routing while also taking into consideration the connection establishment time. [1] analyzed the performance of multipath routing while the path computation problems were not fully considered. Banner et al. [4] proposed multipath routing algorithms for minimization of the network congestion. The constraint-based path selection problems were studied theoretically. Recently, differential delay problems have gained a lot of attention. Ahuja et al. [3] studied the problem of minimizing the differential delay in the context of Ethernet over SONET. The algorithms proposed in [3] select a path for a Virtually Concatenated Group (VCG) which had the minimum differential delay. Srivastava et al. [5] proposed two heuristic algorithms to route traffic into a group of virtual containers (VCs) which could satisfy the differential delay constraint in SONET/SDH networks. They also proposed an integer linear programming formulation with constraints of differential delay. In their following work, Srivastava et al. [6] pointed out that memory size issues caused by differential delay could be optimally solved by considering cumulative differential delay and proposed both heuristic and LP solutions for traffic distribution. Huang et al. [7] extended the K-shortest path algorithm and proposed a heuristic algorithm to calculate multiple paths with the differential delay constraint. While the aforementioned works consider the multipath optimization, Rao et al. [8] presented QoS multipath routing algorithms in connection-oriented networks where bandwidth can be reserved. The scheme proposed in [8] obtained a smaller end-to-end delay while differential delay problem was not considered. This work assumed that all the packets were optimally buffered and reordered at the receiver. This scheme has an expensive memory cost and overhead since it requires memory size big enough or can be dynamically adjusted at receiver.

Our contribution in this paper is proposing a multipath routing algorithm tailored to high-end applications, due to its capability of solving a profit maximization problem, where the revenue is the total achievable bandwidth and the cost is the amount of memory used. Our approach differs from other works of differential delay issues in which we don't fix memory size and bandwidth constraints, instead, we take them as bounds and try to find solutions which can minimize the required memory size while maximizing the achievable bandwidth.

III. Problem Formulation and the Proposed Algorithms

A. Preliminaries

Given is a directed graph $G(V, E)$, where V is the set of nodes and E is the set of links. Each link $e \in E$ is associated with two parameters: link capacity c_e and link delay d_e.The delay of a path P is defined as:

$$d_P = \sum_{e \in P} d_e$$

The bandwidth of a path P is denoted by b_P

$$b_P = \min\{c_e\}, e \in P$$

The differential delay between two paths P and P' can be defined as [3]:

$$dd(P, P') = |d_P - d_{P'}|$$

The memory size required at the destination is assumed to be M_r and the maximum bound of memory size is assumed to be M_d. The memory required to re-order the flow using multiple paths is dependent on the flow in each path, the delay of each path, and the highest delay from the set of all paths that are selected. In case of multiple flows, other flows will have to be buffered till the flow with the highest delay arrives. Therefore, the differential delay is measured as the difference in delay with the path having the highest delay in the solution set. Assuming that the highest delay path in the solution is $\tilde{P}$, and given that the traffic on a path P is t_P, the total buffer size required is given by:

$$M_r = \sum_{P \in \mathcal{P}} t_P(d_{\tilde{P}} - d_P)$$

In this work, our focus is on the applications which have large volumes of data to be transferred. Take the case where a 1 *terabyte* of data has to be transmitted from the west coast to the east coast of the United States. Assuming that the underlying link is a $100G$ carrier Ethernet, the time taken to transmit the data is $80sec$. On the other hand, the approximate distance between the two nodes is 3000 miles, within the propagation delay in the optical layer of about $25millisec$

which is non-dominant . Therefore, in this paper our focus is primarily on decreasing transmission delay by increasing the total bandwidth available between a pair of nodes, subject to a memory constraint. At the high level, the problem then can be formulated as:

Given a connection request r with source $s(r)$ and destination $d(r)$, optimize the achievable bandwidth for the requested connection while minimizing the the cost incurred by the memory required at the destination due to differential delay among the multiple paths.

B. Multipath Computation Algorithm

Before going into details of ILP formulation let us focus on the multi-path computation. For tractability, it is important to note that a number of paths allowed to create path set for flow should be bound (N), which is a parameter decided by the network service provider based on the local policies, specific applications and path computation complexity etc. For jitter-sensitive applications, N should be as small as possible due to the increased variance in the path chosen. For latency-sensitive applications, on the other hand, more paths can enable lower end-to-end latency. The multi-path set with maximum path number N is used as input of the proposed multi-path ILP.

We use a method which computes the paths in decreasing order of bandwidths, which is shown in Algorithm 1. The path computation begins from the source node, and creates path segments to all neighboring nodes. In each iteration, new path segments are formed by extending the given path segment to its neighboring nodes. These segments are sorted in descending order of available bandwidth. Care is taken to ensure that the new path segments do not contain loops. The new path segments are then inserted in the original array using insertion sort. In the case that the path segment selected is terminated at the destination, the next path segment in line is chosen. Also note that in case a path segment cannot be extended without forming a loop, it is removed from the array.

As shown in Algorithm 1, we introduce an array $pathSet$, which is initialized with all path segments beginning from the source to its neighboring nodes and sorted in the decreasing order of magnitude. In each iteration, we choose the first path in $pathSet$ which does not terminate at the destination node, and remove it from $pathSet$. Then, for each possible extension for the chosen path, we check if a loop is formed in the path segment. This is done by checking if the new vertex that the path segment is extended to exists in the path segment. If the new path segment is seen to be valid, the new path segment is then inserted in $pathSet$.

In order to calculate the first N shortest widest paths, the sorting algorithm check the delay of the paths in question in case of a tie. At the beginning of each iteration, we check the number of consecutive paths at the beginning of $pathSet$ terminating at the destination node. If the number becomes equal to N, the algorithm is terminated and the discovered paths are selected. Note that while the algorithm may find more than N paths to the destination, it only terminates when it has found the N shortest widest paths.

Algorithm 1: Multipath path computation algorithm

```
Input: G(V,E), (s,d), N
count = 0;
Initialize an Empty Array of Paths pathSet;
For all v_i ∈ V s.t. there exists a link e_j ∈ Efrom s to v_i,
create a path from s to v_i and insert in pathSet;
Sort pathSet in decreasing order of bandwidth;
while count < N do
    count = 0;
    for (i = 0 to pathSet.length) do
        if (pathSet[i].destination == d) then
            Increase count by 1 ;
            If(count == N) break;
        end
        else
            path = pathSet[i];
            remove pathSet[i] from pathSet;
            foreach (e_j ∈ E attached to path.destination)
            do
                Let the vertices at the end of e_j be v_k
                and path.endVertex;
                if (v_k ∉ path.vertices) then
                    Create a new path by extending path
                    with e_j; Insert new path in pathSet
                    using insertion sort;
                end
            end
        end
    end
    if (no more paths can be extended) then
        break;
    end
end
```

C. ILP Formulation

In the proposed ILP-based algorithm, we attempt to maximize the achievable bandwidth for a request while taking into consideration the buffer (memory) cost. We first define the linear functions for the revenue due to the total bandwidth and the cost of the buffer memory required. To do so, we define two cost factors, one for network flow (C_F) and one for memory cost (C_M). The relative scale of the two factors defines weather minimization of differential delay or maximization of flow plays a dominant role. For example, in case of a large C_F, the memory cost may become insignificant, and the problem reduces to a flow maximization problem, while in case where the memory cost is significantly high, the cost of memory required to accommodate the differential delay may well be more that the revenue due to the flow, and may result in solutions where only the widest path is chosen out of a set of paths with zero differential delay.

The ILP relies on the following variables:

- $\mathbf{x_e}$ - an integer variable for each link $e \in E$ denoting the current flow on link e.

- $\mathbf{t_P}$ - an integer variable for each path $P \in \mathcal{P}$ denoting the current traffic on the path P.
- $\tilde{P}$ - The path in the solution set with the highest delay. Note that for a path to be in the solution set, the flow through the path must be non zero.

The ILP-based traffic splitting algorithm is formulated as follows:

$$\textbf{Objective} : \text{Maximize} \sum_{P \in (P)} C_F \cdot \mathbf{t_P} - C_M \cdot M_r$$

Subject to:

$$\forall e \in E : \ \mathbf{x_e} = \sum_{P \in \mathcal{P} \wedge e \in P} \mathbf{t_P} \tag{1}$$

$$\forall e \in E : \ \mathbf{x_e} \leq c_e \tag{2}$$

$$M(r) = \sum_{P \in \mathcal{P}} \mathbf{t_P}(d_{\tilde{P}} - d_P) \tag{3}$$

$$\forall P \in \mathcal{P} : \mathbf{t_P} = 0 \, if \, d_P > d_{\tilde{P}} \tag{4}$$

$$\forall P \in \mathcal{P} : \mathbf{t_P} >= 0 \, if \, d_P <= d_{\tilde{P}} \tag{5}$$

$$\mathbf{t_{\tilde{P}}} > 0 \tag{6}$$

$$\sum_{P \in (P)} \mathbf{t_P} >= B_{min} \tag{7}$$

$$M_r \leq M_d \tag{8}$$

The variable x_e is defined as the sum of flows in the different paths that pass through the link $e \in \mathcal{E}$, as shown in Eq. 1. The constraint in Eq. 2 ensures that the total flow through a link does not exceed the total capacity of the link itself. Eq.3 calculates the buffer memory size at the destination node, with the assumption that $\tilde{P}$ is the *highest delay* path in the solution. Eq.4 and Eq.5 enforce the constraints that ensure that all flows are positive, and that no path with delay greater than $d_{\tilde{P}}$ is selected in the solution set, while Eq. 6 ensures that the path $\tilde{P}$ is in the solution set by ensuring that the flow $t_{\tilde{P}}$ is not zero. Eq. 7 implements a minimum bandwidth constraint on the solution and Eq. 8 indicates that the required memory size can not exceed the maximum memory boundary in the network. The maximum bandwidth constraint B_{min} is effectively derived from the total size of the data to be transferred F and the maximum delay constraint D_{max}. If the file size is sufficiently large, the link delay can be assumed to be negligible as compared to the time taken to transmit the data. The maximum delay constraint is

$$\sum_{P \in (P)} \mathbf{t_P} >= \frac{F}{D_{max}} \tag{9}$$

It can be seen that the requested memory size in Equation (3) depends on the choice of the highest delay path. As the calculation of the memory depends on the solution, to solve the ILP , we use a certain $P \in \mathcal{P}$ as a potential value of $\tilde{P}$. Each choice of $\tilde{P}$ implies that certain paths may not be taken into consideration as their delay is greater than the delay of $\tilde{P}$. Therefore, the ILP is run $N = |\mathcal{P}|$ times, once for each possible candidate for $\tilde{P}$, and the solution with the highest objective among the N iterations is the optimal solution.

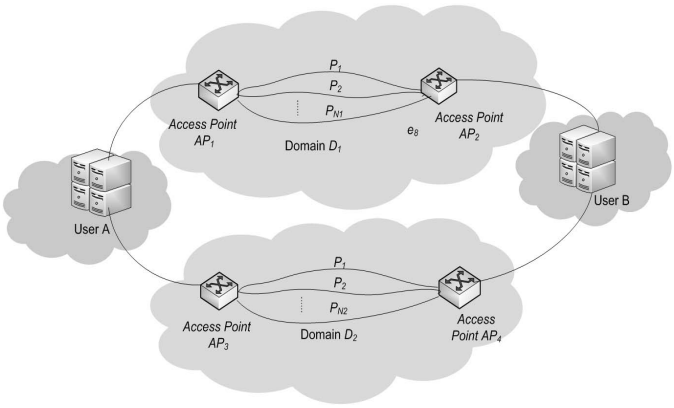

Fig. 2. Multi-path Multi-domain Routing.

The time complexity of an ILP is known to be exponential. In the worst-case scenario, the ILP would be run N times, and the i-th iteration has an input path set of size i. Therefore the time complexity of the ILP is in the order of $O(2^1 + 2^2 + .. + 2^N)$ which is equal to $O(2^{N+1})$.

D. Multi-path Multi-domain Routing

Multi-path routing can be especially useful in multi-homing networks. An illustrative example is shown in Fig. 2. Users A and B are both connected to domains D_1 and D_2. Traditionally, either D_1 or D_2 would be used to transfer data between A and B, while multi-path multi-domain routing enables data transmission via both domains simultaneously to get less transmission latency and more bandwidth. There are three possible application schemes for multi-path multi-domain routing in multi-homing networks.

1) A central entity which has visibility over the different domains in the multi-homing network can be used to optimally provision paths in the network. In the scenario shown in Fig. 2, this entity must have visibility over both domains D_1 and D_2.
2) If the domains are not visible to a central entity, paths will be calculated in each domain sequentially. In Fig. 2, D_1 calculates multiple paths based on the local policies and network utilization and represents them as an aggregated path. Information of the aggregated path such as bandwidth and delay will be used as constraints during the multi-path computation in the second domain D_2.
3) The third possibility is that the users may have their preferred provider or domain which is used as primary option. Multiple paths are optimally provisioned insided the preferred domain. Multi-path routing inside a single domain in multi-homing networks is also possible when end nodes do not have the capability to split/re-order flows , which then must be performed at the borders of transport domains.

IV. Performance Evaluation

In the performance evaluation, we study the effect of various network parameters such as load, memory cost and differential delay on the proposed multi-path routing algorithm. We first show the variation in the bandwidth achieved at different loads and memory costs in the Germany 17 reference network [9]. We simulate a dynamic network, with connection arrivals following a *Poisson* process with negatively exponentially distributed inter-arrival times, and negative exponentially distributed holding times. Each connection requests a bandwidth of 1 Gbps between a random pair of nodes. We take snapshots at different point of time, and 43 such snapshots are used in this study. Each link in the network is assumed to have a capacity of 40 Gbps and the delay of each link is estimated on the basis of the geographical distance between the two nodes. In all results, the revenue obtained per $Gbps$ (C_F) is set to $1Gbps^{-1}$. The memory cost has the unit of $ms.Gbps^{-1}$, and the delay of the links is always measured in milliseconds. We compare the performance of the single widest-shortest path and the proposed multi-path routing algorithm. Increase in network load implies that the achievable bandwidth of both single and multi-path routing decreases. Also, increase in memory cost implies that fewer paths are chosen to compensate for the increasing effect of differential delay. However, as shown in Fig. 3, the multi-path ILP can still obtain higher bandwidth from the network even when the memory cost is high. To show the effect of memory cost on the performance, we also observe the change of achievable bandwidth in a certain network load, i.e.,130 Erlang in Fig. 4. The total bandwidth of the optimal solution decreases rapidly with increasing of memory cost, and is especially significant at low memory costs.

We also study the performance of the proposed multi-path routing algorithm in the multi-homing networks. The Germany 17 and the NSFNet topologies are used as the reference topologies. As the NSFNet topology spans across a significantly larger area, the delays are extremely high as compared to the Germany 17 network which is not conducive for simulating real multi-homing networks. Therefore, we scale the distances between nodes in the NSFNet topology to 1/5 their original value such that the delay of two reference topologies is comparable. We first observe the change in performance of the three application mechanisms (multi-domain full visibility, multi-domain partial visibility, and preferred domain routing) for different loads in the network. Shown in Fig. 5 is the bandwidth distribution, when both networks are loaded identically. The cost of the memory parameter is set as $C_M = 1$.

It is seen that while the total bandwidth decreases with increase of network load, there is a significant difference in the total achievable bandwidth between both multi-domain schemes and the single domain scheme. The multi-domain full visibility scheme is seen as the benchmark of the maximum achievable bandwidth in the multi-homing networks with two domains. The multi-domain partial visibility scheme can be deployed when the full topology view is not available. This

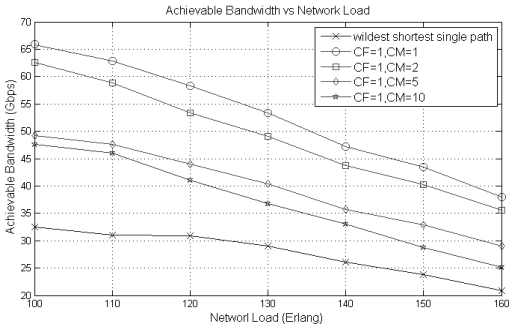

Fig. 3. Bandwidth vs Network Load

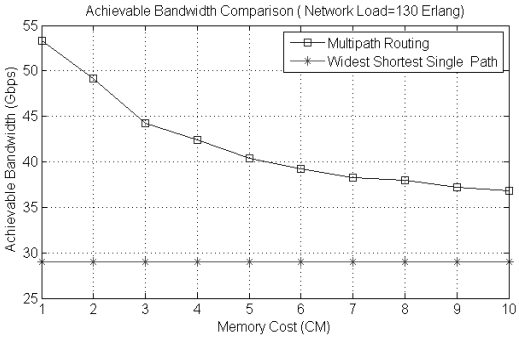

Fig. 4. Achievable Bandwidth of Different Algorithms

scheme performs extremely well at low network loads, with significantly higher bandwidth available as compared to the single preferred domain scheme. Even with skewed loads in the networks, the partial visibility scheme has a significantly higher performance as compared to the single domain scheme as seen in Fig. 6

A major factor in multi-path multi-domain routing can be the effect of the differential delay between the two domains. If the differential delay between the two domains is significant, the performance of the multi-domain schemes can be severely diminished. To study this effect, we scaled the delay of the NSFNet topology by multiplying the delay of all the links with a pre-defined scaling factor while keeping the delay of the Germany topology constant. Both networks have the same network load of 100 Erlangs and C_M is set to 1. As seen in Fig. 7, the achievable bandwidth by the single domain scheme remains relatively constant, while the achievable bandwidth of both the multi-domain schemes can vary significantly. As the scaling factor increases, the total bandwidth is seen to first increase and then decrease again. The increase suggests that the differential delay is minimized when the scaling factor is between 1.4 and 2.2. Another interesting factor observed is the gradual decrease in bandwidth at the same load with then increase in scaling factor in Fig. 7. This is observed as the net magnitude of the differential delay between a set of paths also scales with the scaling factor.

We finally study the variation in the bandwidth with the change in the memory cost. The network load of both domains is set to 100 Erlangs and C_M is varied from 0 to 10. At

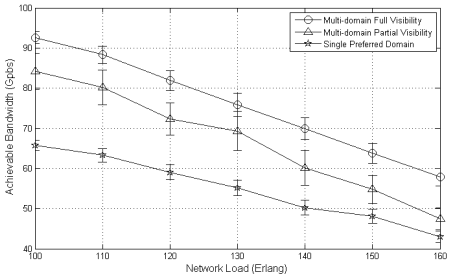

Fig. 5. Achievable Bandwidth of Different Schemes With Identical Network Loading In Both Domains.

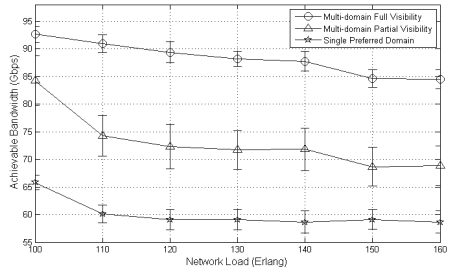

Fig. 6. Achievable Bandwidth Of Different Schemes With Skewed Network Loading. The load in the Germany 17 topology is kept constant at 100 Erlangs while the load in the NSFNet topology is varied from 100 to 160 Erlangs

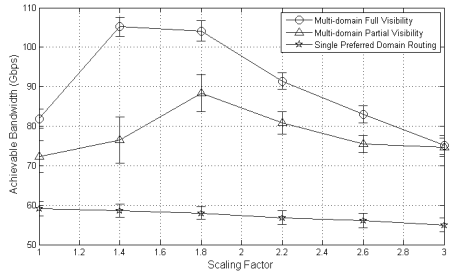

Fig. 7. Achievable Bandwidth of Different Schemes vs Delay Scaling Factor.The link delay in the NFSNet topology is scaled while that of the Germany17 is kept constant.

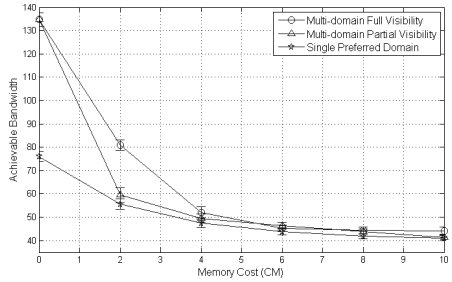

Fig. 8. Achievable Bandwidth Of Different Schemes vs Memory Cost.

$C_M = 0$, the problem is reduced to the maximum flow problem, and therefore the total bandwidth reserved by both the multi-domain schemes is the same as seen in Fig 8. The total bandwidth decreases sharply with increase in memory cost, especially at low values of C_M as also seen in Fig. 4, and is seen to reduce gradually at higher memory costs.

V. Conclusion

In this paper, we proposed the use of multi-path routing to maximize bandwidth between a pair of nodes for applications requiring high data transfer rates. We presented a multi-path computation algorithm which calculates the first N shortest-widest paths and propose an ILP approach which can maximize the achievable bandwidth while minimizing the memory cost caused by the differential delay problem in multi-path routing. We studied the applicability of the proposed multi-path routing algorithm in multi-homing networks and presented three application schemes of multi-path multi-domain routing. Our results show that multi-path routing can achieve significantly higher bandwidth as compared to traditional single path routing. We have also shown that multi-domain multi-path routing schemes can achieve significant increase in bandwidth even with partial visibility in a multi-homing environment. Algorithms proposed in this paper can therefore be used as an ideal solution for accommodating the emerging high-end applications in current networks. Our future work will focus on decreasing the complexity of multipath calculation and developing heuristic algorithms to find near-optimal solutions.

VI. Acknowledgment

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) under support code JU2757/-1/1; and FAPESP under support code 2007/54867-4.

References

[1] I.Cidon, R.Rom and Y. Shavitt,*Analysis of Multi-path Routing*, IEEE/ACM TRANSACTIONS ON NETWORKING,december 1999, vol.7(6),pp.885-896.

[2] IEEE802.1Qay, *Provider Backbone Bridge Traffic Engineering*, http://www.ieee802.org/1/pages/802.1ay.html

[3] S. Ahuja, T. Korkmaz, and M. Krunz, *Minimizing the Differential Delay for Virtually Concatenated Ethernet over SONET Systems*, in Proc. ICCCN 2004, pp. 205-210.

[4] R. Banner, A.Orda,*Multipath Routing Algorithms for Congestion Minimization*, IEEE/ACM TRANSACTIONS ON NETWORKING, April 2007,vol.15 (2), pp.413-424.

[5] A. Srivastava, S. Acharya, M. Alicherry, B. Gupta, and P. Risbood, *Differential Delay Aware Routing for Ethernet over SONET/SDH*, in Proc. IEEE INFOCOM 2005, pp.1117-1127.

[6] Anurag Srivastava and Abhinav Srivastava,*Flow Aware Differential Delay Routing for Next-Generation Ethernet over SONET/SDH* , in Proc. of IEEE ICC 2006, pp.140-145.

[7] S.Huang, B. Mukherjee and C. Martel, *Survivable Multipath Provisioning with Differential Delay Constraint in Telecom Mesh Networks*, in Proc. IEEE INFOCOM 2008,pp.191-195.

[8] N.S.V.Rao and S.G.Batsell, *QoS Routing Via Multiple Paths Using Bandwidth Reservation*, INFOCOM'99,pp.11-18.

[9] http://www.bmbf.de/de/6103.php