On the Trade-Off between Control Plane Load and Data Plane Efficiency in Software Defined Networks

Gabi Nakibly Reuven Cohen Liran Katzir

***Abstract*—Software Defined Networking (SDN) is a new emerging network architecture, which allows, among other things, an efficient convergence between circuit switching and packet switching. However, this convergence imposes a trade-off between control plane load and data plane efficiency: while the SDN controller needs to establish one or more paths (circuits) that can deliver the required bandwidth with minimum data plane cost, this is made difficult by the load imposed on the control plane due to setup and OAM demands. The present paper defines and studies this trade-off from both theoretical and practical perspectives.**

I. Introduction

Software Defined Networking (SDN) is a new emerging network architecture built on the following main concepts [17]: (1) separation between data and control planes; (2) flow based datapath, where flows (not packets) are the fundamental unit of control; (3) a control protocol, such as OpenFlow [18], between a logically centralized controller and the switch flow tables for programming and controlling the datapath; and (4) a means to provide network virtualization by slicing the network and isolating the slices. Using these concepts, SDN allows service providers and enterprises to create infrastructure-as-a-service models by enabling on-demand procurement, provisioning and configuration of these services. New instances of network services can be established very quickly, and capacity can be scaled up or down in a real time.

One of the main promises of SDN is that it allows an efficient convergence between circuit switching and packet switching [8]. This is because the network operator can easily establish, modify and take down circuits, based on the requirements of the packet switching layer. However, this convergence imposes a trade-off between the control plane load and the data plane efficiency. Suppose that there is a demand for a certain bandwidth between two nodes in order to deliver a traffic aggregate ("traffic flow"). Thus, the SDN controller needs to establish one or more paths (circuits) that can deliver the required bandwidth with minimum data plane cost. However, this incurs a load imposed on the control plane by path setup and OAM (operations and maintenance) demands.

- Setup: The setup of each path, using a control protocol such as OpenFlow [18], requires message exchange between the controller and each of the nodes along the path. This is translated to the consumption of memory and CPU resources, which might be scarce especially at the controller.
- OAM: Carrier-class services require very fast detection of data plane failures between the ingress and egress of each path. This is translated into expensive OAM protocols that have to be continuously executed along each path [6], [16].

We study the above trade-off in two different cases. In the first case, the controller is given a network flow that satisfies the bandwidth demand between the source and destination nodes. This network flow is pre-determined according to data plane considerations (e.g., minimum cost). The problem faced by the controller is how to break this flow into a set of paths (circuits) while enforcing a minimum control plane load. In the second case, the controller is not given a network flow, but only a bandwidth demand between the source and destination. Thus, it needs to find a set of paths over which this demand can be delivered. This version is suitable when the operator is more concerned with control plane load rather than with data plane efficiency.

We translate these two cases into two optimization problems:

1) The *Decomposition with Minimum Control Load* (DMCL) problem: given a network and a feasible flow that satisfies the bandwidth demand for a pair of ingress and egress nodes, decompose this flow into a set of paths that imposes minimum control load.
2) The *Routing with Minimum Control Load* (RMCL) problem: given a network and a bandwidth demand B for a pair of ingress and egress nodes, find a feasible set of paths that satisfies this demand and imposes minimum control load.

For both problems we measure the control load using either the number of paths or the number of nodes traversed by the paths. Thus, we actually solve two pairs of problems:

G. Nakibly and R. Cohen are with the Department of Computer Science, Technion, Haifa, Israel.

L. Katzir is with Yahoo! Israel Labs, Haifa, Israel.

(a) DMCL(p) and RMCL(p) for minimizing the number of paths; (b) DMCL(n) and RMCL(n) for minimizing the number of nodes.

Throughout the paper we focus on the case where flows are admitted one by one. While there are scenarios where the operator can admit many flows at the same time, we believe that the "one flow at a time" scenario is more important for the following three reasons. First, in many SDN applications flows are admitted for a pre-specified duration. The starting times and due dates of the flows are usually independent. Thus, when a new flow has to be admitted, previous flows already use their own paths. Second, an operator may decide to set up a new set of paths between two nodes in order to respond better to periodic congestion. This is an on-line decision, which is captured by the "one flow at a time" approach. Third, when a link or a node fails, each path that crosses this link or node has to be re-routed, which is again an on-line problem.

It is important to note that the results of this paper are applicable not only to Software Defined Networks (SDNs) with a centralized controller, but also to more traditional virtual circuit technologies, such as MPLS and GMPLS, in which paths are set up in a distributed manner. Using routing protocols such as OSPF-TE [13] and MPLS-TE [7], each router constructs a map of the network topology and the bandwidth available on each link. Then, each ingress router finds a route with sufficient bandwidth to an egress router for each traffic aggregate. The ingress router then establishes an MPLS LSP (Label Switched Path) over this route using a signaling protocol, such as RSVP-TE [10]. Each router along the LSP must keep a state of this LSP. This state must be periodically refreshed using the signaling protocol and maintained using the OAM protocols. Therefore, it is desirable to minimize the number of LSPs or the number of routers crossed by these LSPs in order to reduce the control load.

The rest of this paper is organized as follows. In Section II we illustrate in more detail the DMCL and RMCL problems. In Section III we discuss related work. In Section IV we formally define the DMCL(p) problem, discuss its computational complexity and present approximation algorithms. In Section V we do the same for the RMCL problem. In Section VI we address DMCL and RMCL while minimizing the number of nodes rather than the number of paths. The actual performance of the proposed algorithms is examined through simulations in Section VII. Finally, Section VIII concludes the paper.

II. DMCL vs. RMCL

The optimization problems above are illustrated in Figure 1. In this example, we show the trade-off between control load and bandwidth cost. The cost of a bandwidth flow on a link is the link cost times the volume of flow it carries. For the sake of simplicity, let the cost of each link be 1. Figure 1(a) shows a network with the bandwidth available on each link. First, suppose that the operator needs a bandwidth of 1Gb/s for a traffic aggregate from node a to c. The most efficient routing solution is to use the shortest path $a-b-c$. The cost of this solution is 2. It is cheap and can be delivered using a single short path. However, if the required bandwidth is 8Gb/s, the shortest path cannot carry it. When the main optimization criterion is to minimize the control load (RMCL), the operator should set up two paths of 4Gb/s: one on the path $a-d-e-f-g-c$ and another on the path $a-h-i-j-k-l-m-c$, as illustrated in Figure 1(b). In this example, this solution minimizes both the number of paths (2) and the number of nodes that carry them: $6+8=14$ (we count nodes a and c twice to account for the control load accrued by both path), but this is not always the case. The cost of this solution is $4*5+4*7=48$.

Now, suppose that the operator's main optimization criterion is to minimize the cost of carrying 8Gb/s from a to c. The operator can use a standard algorithm for finding a minimum-cost flow [1] whose output is the feasible minimum-cost network flow illustrated in Figure 1(c). The cost of this network flow is 36. The operator can use this network flow as an input to DMCL in order to decompose it into a set of paths (sub-flows) that minimizes the control load. Figure 1(d) shows a decomposition of the network flow in Figure 1(c), which imposes minimum control load under both criteria: number of paths (4) and number of nodes that carry them (22).

As already indicated, we are the first to in-depth investigate the problem of control load minimization and its trade-off with data plane efficiency. Specifically, we make two theoretical and one practical contributions:

1) We are the first to define and solve the RMCL(n) and DMCL(n) problems. We present for these problems approximation algorithms with performance guarantees.
2) We show that simple greedy decomposition algorithms for DMCL have an approximation ratio that is independent of the size of the network.
3) We compare the performance of the RMCL and DMCL algorithms. The purpose of this comparison is to better understand the trade-off between data plane efficiency and control plane load. This comparison allows us to identify an algorithm that yields an excellent trade-off between these two objectives.

Table I summarizes our main results from a computational complexity perspective. In this table, B denotes the flow bandwidth demand, b denotes the quantum of the edge capacities, *opt* denotes the value of the optimal solution and α is a tuning parameter. For each problem the table indicates a lower bound on its approximation ratio and the approximation ratio achieved of algorithms we present for it. Throughout the paper we consider the minimization of the bandwidth cost as the data plane efficiency criterion. However, our results are applicable to any other data plane criterion, such as throughput maximization or maximal load

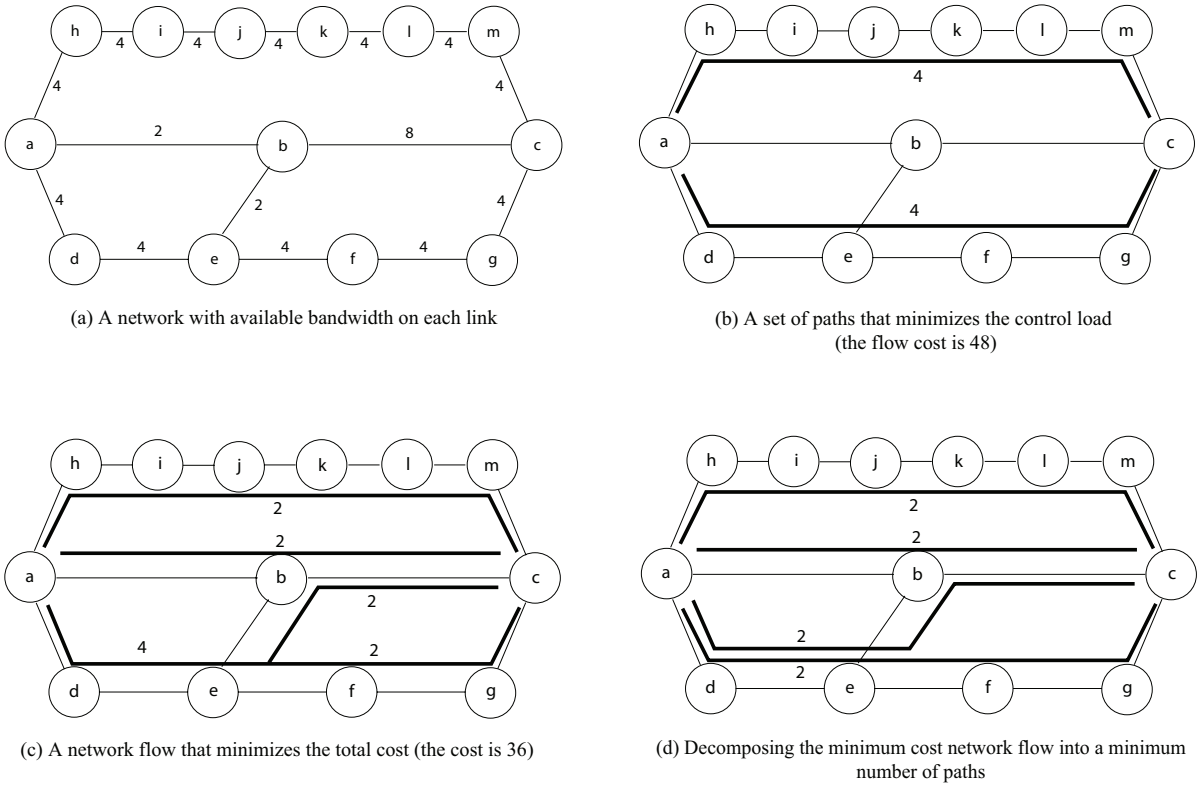


Fig. 1. An example for the two optimization problems considered in this paper

problem	minimum bound	approximation ratio
DMCL(p)	-	$O(\log(B/b))$
DMCL(n)	-	$O(\log(B/b))$
RMCL(p)	$3/2$	$O(\frac{B}{\text{opt}\cdot\alpha})$
RMCL(n)	$3/2-\epsilon$	$O(\frac{B}{\alpha})$

TABLE I
OUR MAIN COMPUTATIONAL COMPLEXITY RESULTS

minimization.

Figure 2 gives an overview of the scope of this paper.

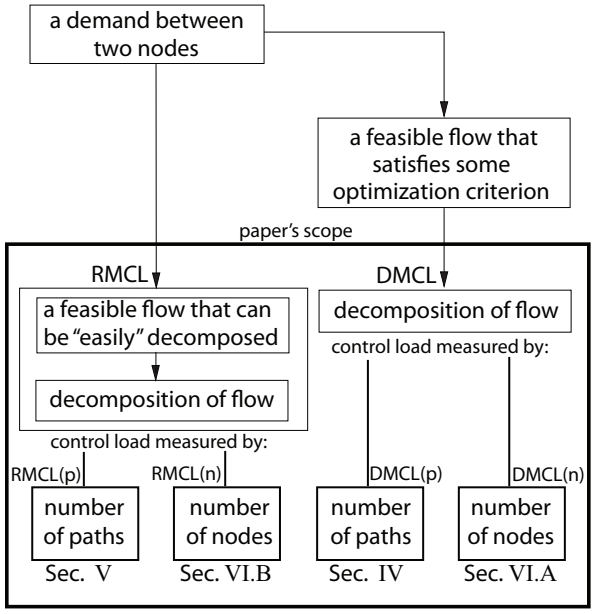


Fig. 2. The scope of this paper

III. Related Work

To the best of our knowledge, no prior work deals with minimizing the number of nodes traversed by paths that satisfy a given bandwidth demand (RMCL(n)). Moreover, no prior work deals with the decomposition of a given flow while minimizing the number of nodes traversed by the paths (DMCL(n)). There are, however, a few works that address the DMCL(p) and RMCL(p) problems. We note that if minimizing the number of paths is not important, it is easy to decompose a given flow with at most $O(|E|)$ paths [1].

In [4], the RMCL(p) problem is addressed. In this work, the number of paths that satisfy a given bandwidth demand is minimized while guaranteeing an upper bound on the load imposed on the network links. This work presents an algorithm that may violate the maximum load bound. The violation gets smaller as the number of paths increases. The

actual performance of the proposed algorithm is not studied.

The objective of [24] is to decompose a given (maximum) flow into a minimum number of paths. The authors prove that the problem is NP-hard, present several heuristics, and evaluate their performance using simulations. A greedy algorithm that iteratively decomposes the maximum flow path is shown to achieve the best performance. The problem of decomposing a given flow into a minimum number of paths is also studied in [23]. That paper is mainly concerned with decompositions that produce independent paths. Such paths are iteratively produced by reducing to 0 the flow on at least one edge during each step. Such decompositions are shown to have an approximation ratio of $n-1-\frac{n^2-3n+1}{m}$, where n is the number of vertices and m is the number of edges in the graph. Two decomposition algorithms are evaluated: one is the greedy algorithm, and the other chooses the shortest path during each step. As in [24], the greedy algorithm is shown to have the best performance.

Another relevant branch of work deals with the k-splittable flow problem. This problem can be viewed as the reverse version of our RMCL(p) problem. An upper bound k on the number of paths is given and the objective is to maximize the satisfied bandwidth demand [3], [14], [15]. In [3], the directed graph version of this problem is proven to be NP-hard. Moreover, it is proven that it cannot be approximated within a factor of $3/2$. A 2-approximation algorithm is also presented. In [14], an optimal polynomial solution is given for the cases where k is constant and the graph has a special property called bounded treewidth [21]. In addition, a polynomial time approximation is presented for the case where k is part of the input. In [15], a comprehensive study of the k-splittable flow problem is presented and proven to be NP-hard for undirected graphs. Moreover, it is proven that for a constant k the problem cannot be approximated within a factor of 5/6. Finally, it is also proven that the problem is NP-hard for $2 \leq k \leq m-n+1$, and that it is polynomially solvable for any other k.

Several papers address the problem of minimizing the maximum load while bounding the number of paths. In [20], the splittable version of this problem is optimally solved. The number of paths is kept below $m+d$, where m is the number of edges and d is the number of demands.

IV. DMCL WITH PATH MINIMIZATION (DMCL(P))

In this section we define the DMCL(p) problem, discuss its computational complexity and propose approximation algorithms. Throughout the paper, a network flow that does not violate the capacity constraints is referred to as a *feasible network flow*. In addition, we refer to a flow carried by a path as a *single-path flow*.

Problem 1 [DMCL(p)]:

Instance: Let $G=(V,E)$ be a directed graph. Let $s,t \in V$ be the source and target nodes. Let f be a feasible network flow from s to t and $f(e)$ be the bandwidth of f carried on edge $e \in E$.

Objective: Find a minimum path decomposition of f. A decomposition of f is a set $p_1, p_2, \ldots, p_k$ of simple directed paths from s to t, where path p_i carries a single-path flow of bandwidth w_i, and on each edge e the sum of bandwidths carried by the paths traversing the edge equals $f(e)$.

Using a reduction from the partition problem, the authors of [24] prove that DMCL(p) is NP-hard in the strong sense. Thus, a pseudo-polynomial algorithm that finds an optimal solution for it is unlikely to exist.

Consider a greedy algorithm for the DMCL(p), which iteratively decomposes the remaining network flow at each step into the widest feasible single-path flow. This intuitive algorithm was previously studied in [23] and [24]. In [23], this algorithm is proven to have an approximation ratio of $|V|-1-\frac{|V|^2-3|V|+2}{|E|}$. Our contribution here is to prove that this algorithm also yields an approximation ratio that does not depend on the size of the network.

In the following discussion we assume that the edge capacities are b-integral, where b is an integer greater than 0. For instance, b might be 1Kb/s or 1Mb/s. We show that the approximation ratio for the greedy algorithm is $\log(B/b)$, where B is the total bandwidth of the network flow. For dense networks, this approximation ratio is tighter than the one proposed in [23], since $|V|-1-\frac{|V|^2-3|V|+2}{|E|}$ might be in the order of several hundreds while $\log(B/b)$ is in the order of 10.

As indicated above, the input network flow for DMCL(p) is f, while $f(e)$ is the value of f carried over edge e. Let $f(p)$ denote the value of a single-path flow carried over path p, i.e., $f(p)=\min_{e \in p}\{f(e)\}$. Let $f \backslash p$ be the network flow whose value $\forall e \in p$ is $f(e)-f(p)$ and its value $\forall e \notin p$ is $f(e)$.

The greedy algorithm iteratively finds a single path whose bandwidth is maximal until it reaches a total bandwidth of B or more.

Algorithm 1: (A greedy algorithm for DMCL(p))

1) $B^0 \leftarrow B$, $f^0 \leftarrow f$, $P \leftarrow \phi$, $i \leftarrow 0$.
2) Repeat until $B^i = 0$:
   a) Choose the path p that can provide the largest portion of f^i from the source to the destination. This can be found using the extended Dijkstra algorithm [1] in time $O\left(|E|\log(|V|)\right)$.
   b) $B^{i+1} \leftarrow B^i - f^i(p)$, $f^{i+1} \leftarrow f^i \backslash p$, $P \leftarrow P \cup p$, $i \leftarrow i+1$.
3) Return P. □

It is easy to see that the algorithm returns a feasible solution that carries a total bandwidth of B, because on each path $p \in P$ we can route a bandwidth of $f^i(p)$, where i is the step during which p is selected.

During each step of the algorithm, the flow carried on at least one edge in f^i is reduced to 0. Thus, the number of steps is bounded by $|E|$. Since each step can be performed

in time $O(|E|)$, the total running time of Algorithm 1 is $O(|E|^2 \log(|V|))$.

Theorem 1: The approximation ratio of Algorithm 1 is $O(\log(B/b))$.

Proof: The proof is similar in spirit to the one used in [12] for the Minimum Set Cover problem. Let G' be the same as G, but with edge capacities scaled down by a factor of b. Denote the number of paths in the optimal solution by OPT and each path in the optimal solution by p_j^*, where $1 \leq j \leq OPT$. Let p_i be the path chosen by the algorithm in the i-th iteration. Since at each step the chosen path is the widest one, then for every j

$$f^i(p_i) \geq f^i(p_j^*).$$

Hence,

$$OPT \cdot f^i(p_i) \geq \sum_{j=1}^{OPT} f^i(p_j^*) \geq B^i/b.$$

The right inequation is due to the fact that the entire set of optimal paths can decompose the network flow f and hence any network flow f^i of value B^i. This leads to

$$\frac{1}{f^i(p_i)} \leq \frac{OPT}{B^i/b}.$$

Each iteration reduces B^i by at least b units of bandwidth. Thus,

$$\frac{OPT}{B^i/b} \leq \frac{OPT}{B/b-(i+1)}.$$

The left side of this inequation can be viewed as the "cost" of each b bandwidth units routed by p_i. Summing up the cost for all the sets of b bandwidth units in B results in the number of paths chosen by the algorithm, denoted as ALG. We now order the sets of b bandwidth units according to the order of the algorithm steps during which they are routed. Multiple sets that are routed at the same step are arbitrarily ordered. For each set k of b bandwidth units routed at step i, $\frac{OPT}{B/b-(i+1)} \leq \frac{OPT}{B/b-(k+1)}$ holds. Hence, we get

$$\begin{aligned} ALG &= \textstyle\sum_{k=1}^{B/b} \frac{OPT}{B/b-(k+1)} = OPT \cdot (1 + \frac{1}{2} + \cdots + \frac{1}{B/b}) \\ &\leq OPT \cdot \log B/b, \end{aligned}$$

which concludes the proof. ■

V. RMCL WITH PATH MINIMIZATION (RMCL(P))

In this section we formally define the RMCL(p) problem. Unlike DMCL, the network flow for RMCL is not given in advance but only the volume of the bandwidth demand. We discuss the computational complexity of RMCL and propose approximation algorithms with bounded performance guarantees.

Problem 2 [RMCL(p)]:

Instance: Let $G = (V, E)$ be a directed graph. Let $c(e)$ be the bandwidth capacity of edge $e \in E$. Let $s, t \in V$ be the source and target nodes and $B \in \mathcal{R}^+$ be the bandwidth demand from s to t.

Objective: Find a minimum set of simple directed paths from s to t, that carry together a feasible network flow of B from s to t.

To solve RMCL, we first construct a feasible network flow that satisfies the bandwidth demand. We strive to make the constructed flow "easily decomposable." After the flow is constructed, it is decomposed into paths.

Theorem 2: RMCL(p) is NP-complete.

Proof: We prove this by a reduction from DMCL(p). Consider an instance of DMCL(p) that consists of a directed graph G and a network flow f from s to t. We transform this instance into an instance of RMCL(p) in the following way. We take the same graph G and set its edge capacities such that $\forall e \in E,\ c(e) = f(e)$. We take the bandwidth demand B of RMCL(p) to be equal to the value of flow f of DMCL(p) and consider the same source and destination nodes. By construction, in the resulting graph there is only one possible network flow of value B from s to t. This flow is exactly f of DMCL(p). Hence, an optimal solution for the constructed RMCL(p) instance is also an optimal solution for the original DMCL(p) instance. ■

Theorem 3: RMCL(p) cannot be approximated within a factor of 3/2.

Proof: In [3], a reduction from SAT to the 2-splittable flow problem is shown. In the 2-splittable flow problem, the objective is to find a maximum flow that can be decomposed into at most 2 paths. The reduction constructs a graph with source and destination nodes such that a satisfiable SAT instance, for which there is a truth assignment that satisfies all its clauses, yields a feasible flow with 2 paths that carry together 3 flow units. In contrast, an unsatisfiable SAT instance, for which there is no truth assignment that satisfies all its clauses, yields a feasible flow with 2 paths that carry together only 2 flow units. In the latter case, the flow can be augmented by a third path that carries 1 flow unit. Consequently, an unsatisfiable SAT instance yields a feasible flow of 3 paths that carry together only 3 flow units. In both cases we have flows of 3 units delivered by either 2 or 3 paths, which implies that even for $B = 3$ it is NP-hard to determine whether 2 or 3 paths are needed to accommodate the demand. Therefore, it is NP-hard to approximate RMCL(p) with a ratio of $3/2$. ■

We now present an approximation algorithm for RMCL(p), which uses the following observation:

Observation 1: A network flow of value B in a network with integral capacities can be decomposed into $\lceil B \rceil$ paths. □

At first glance, this observation does not seem to be very helpful, because B may be larger than the number $|E|$ of edges in the network, which is a straightforward upper bound on an optimal solution. However, we can scale down the edge capacities by a significant factor such that each unit of flow will be larger in relation to the total network flow.

This scaling process reduces the original demand B into a small number that makes a solution of unit-flow paths more attractive.

Algorithm 2 below uses a parameter α for the scaling process. The algorithm finds a network flow whose value is slightly less than B using no more than $\lceil \frac{B}{\alpha} \rceil$ paths. Choosing a larger α would yield fewer paths whose total bandwidth is smaller.

Algorithm 2: (A basic scaling algorithm for RMCL(p))

1) Scale the capacities by α, i.e., $\forall e \in E\ c'(e) \leftarrow \left\lfloor \frac{c(e)}{\alpha} \right\rfloor$.
2) Find a network flow f whose value is not larger than $\lceil \frac{B}{\alpha} \rceil$ in the scaled network.
3) Find any decomposition of f into paths. Let the resulting set of paths be $P = p_1, ..., p_k$, where path p_i carries a single-path flow of f_i.
4) Use every path $p_i \in P$ to carry a single-path flow of αf_i in the original graph. □

The network flow in Step 2 and its decomposition in Step 3 can be arbitrary. Furthermore, We denote the computational complexity of this step by $O(\text{Flow-Alg})$. The total computational complexity of Algorithm 2 is

$$O(\text{Flow-Alg} + |E| \cdot \frac{B}{\alpha}),$$

because the time complexity of the scaling process in Step 1 is linear in the size of the network and the time complexity of the decomposition process in Step 3 is $O(|E| \cdot \frac{B}{\alpha})$. Note that the above time complexity is only pseudo-polynomial because it depends on B. Algorithm 3 presented later refines this and yields a polynomial running time complexity.

Theorem 4: Algorithm 2 returns a set of at most $\lceil \frac{B}{\alpha} \rceil$ paths whose total bandwidth is at least $B - k^* \cdot \alpha$, where k^* is the number of paths in an optimal solution.

Proof: The scaled network has integral capacities. From Observation 1 follows that the decomposition step produces no more than $\lceil \frac{B}{\alpha} \rceil$ paths, which is the value of the flow found in Step 2. We now prove the lower bound on the bandwidth. Let $p_1, p_2, \ldots, p_{k^*}$ be the set of paths in an optimal solution. Each of them is a simple path from s to t. Each path p_i carries a single-path flow of w_i, where $\sum_i w_i = B$. Consider the same set of paths in the scaled network, and let each path p_i carry a single-path flow of $w'_i = \lfloor \frac{w_i}{\alpha} \rfloor$. This scaled solution is a feasible solution in the scaled network due to the following set of equations, which holds $\forall e \in E$:

$$\begin{aligned}
c(e) &\geq \sum_{e \in p_i} w_i \\
c(e) &= \Delta + \sum_{e \in p_i} w_i,\ \Delta \geq 0 \\
\frac{c(e)}{\alpha} &= \frac{\Delta}{\alpha} + \frac{\sum_{e \in p_i} w_i}{\alpha},\ \Delta \geq 0 \\
\left\lfloor \frac{c(e)}{\alpha} \right\rfloor &\geq \left\lfloor \frac{\Delta}{\alpha} \right\rfloor + \left\lfloor \frac{\sum_{e \in p_i} w_i}{\alpha} \right\rfloor,\ \Delta \geq 0 \\
\left\lfloor \frac{c(e)}{\alpha} \right\rfloor &\geq \left\lfloor \frac{\sum_{e \in p_i} w_i}{\alpha} \right\rfloor.
\end{aligned}$$

The first equation holds because the optimal solution must be feasible in the original graph. The second and third equations follow from the first one. The fourth equation holds because $\lfloor \sum x_i \rfloor \geq \sum \lfloor x_i \rfloor$, and the last one follows from the fourth.

We also note that:

$$\begin{aligned}
\sum_{i=0}^{k^*} w_i &= B \\
\sum_{i=0}^{k^*} \frac{w_i}{\alpha} &= \frac{B}{\alpha} \\
\sum_{i=0}^{k^*} \left\lfloor \frac{w_i}{\alpha} \right\rfloor &\leq \left\lceil \frac{B}{\alpha} \right\rceil \\
\sum_{i=0}^{k^*} \alpha \left\lfloor \frac{w_i}{\alpha} \right\rfloor &\leq \alpha \left\lceil \frac{B}{\alpha} \right\rceil,
\end{aligned}$$

where $\alpha \lceil \frac{B}{\alpha} \rceil$ is the value (bandwidth) of the flow returned by Algorithm 2. This can be lower bounded as follows:

$$\alpha \left\lceil \frac{B}{\alpha} \right\rceil \geq \alpha \cdot \sum_{i=1}^{k^*} \left\lfloor \frac{w_i}{\alpha} \right\rfloor \geq \alpha \cdot \sum_{i=1}^{k^*} \left(\frac{w_i}{\alpha} - 1 \right) = \sum_{i=1}^{k^*} (w_i - \alpha)$$
$$= \sum_{i=1}^{k^*} w_i - k^* \cdot \alpha = B - k^* \cdot \alpha.$$

■

Corollary 1: Let k^* be the number of paths in an optimal solution. For $\alpha = \frac{B}{k^* \cdot \beta}$, Algorithm 2 produces a solution with at most $k^* \cdot \beta + 1$ paths whose value is no less than $B \cdot \left(1 - \frac{1}{\beta}\right)$.

The parameter β can be considered as a tuning parameter. As β increases, the value of the output flow of Algorithm 2 approaches the original demand B, but the number of paths increases. Since k^* is not known in advance, it is not easy to find the value of α. One can try all values of k^*, and find the minimum one that yields a network flow whose total bandwidth is larger than $B \cdot \left(1 - \frac{1}{\beta}\right)$. This requires running Algorithm 2 on all possible values of k^*, which is $O(|E|)$. To improve the total time complexity, Algorithm 3 below

uses the output returned by Algorithm 2 for a given k as the initial network flow when running Algorithm 2 with $k+1$. This is possible because the scaling parameter α decreases as k increases. Thus, the capacities of the scaled network increase.

Algorithm 3: (A scaling approximation algorithm for RMCL(p) using a tuning parameter β)

1) $k \leftarrow 1$.
2) Let f be an initial network flow such that $f(e) \leftarrow 0$ for every $e \in E$.
3) While $k \leq |E|$ and the total value of f is smaller than $B \cdot \left(1 - \frac{1}{\beta}\right)$ do
   a) Run Algorithm 2 with a scaling factor $\alpha = \frac{B}{k \cdot \beta}$ and use f as the initial network flow for Step 2 in Algorithm 2.
   b) Set f as the flow returned by Algorithm 2.
   c) $k \leftarrow k + 1$.
4) Return the set paths output by the last execution of Algorithm 2. □

Assuming that the capacities are integral, if Algorithm 3 is invoked with $\beta = B$, the resulting value of the network flow is guaranteed to be at least B. However, there is no guarantee on the number of paths it uses.

The running time complexity of each iteration of Step 3 is the time complexity of Algorithm 2. Since Algorithm 2 does not need to construct a flow from scratch, its running time is $O(|E| \cdot \frac{B}{\alpha})$. Since the number of iterations does not exceed $|E|$, the running time complexity of Algorithm 3 is $O(|E|^2 \cdot \frac{B}{\alpha})$, i.e., $O(|E|^2 \cdot k^* \cdot \beta)$.

Algorithm 2 and Algorithm 3 have theoretical value because they have worst case performance guarantees. However, simulation results indicate that their actual average performance is not good. Specifically, when the bandwidth provided by the flow is close to B, the number of paths increases very rapidly. We therefore present another algorithm for RMCL(p). While this algorithm has no worst case performance guarantee, its actual performance is shown later to be very good.

The main idea behind the new algorithm is to break the RMCL(p) solution into two stages. First, a network flow that provides a bandwidth of at least B is found. Then, this flow is decomposed using Algorithm 1.

Algorithm 4: (A 2-phase algorithm for RMCL(p))

1) Find an initial feasible network flow of bandwidth B or more from s to t.
2) Use Algorithm 1 for decomposing the flow into a minimum number of paths that provide bandwidth B.
3) Return the set of paths produced by Algorithm 1. □

We now present several procedures for finding an initial network flow. In Section VII we compare the performance of Algorithms 4 using each of these procedures. All of the procedures produce a *maximum* network flow between s and t although the algorithm only requires that the bandwidth of the initial network flow will $\geq B$. We found that starting with a maximum flow gives the algorithm greater flexibility in minimizing the number of paths. When we evaluated similar procedures that limit the bandwidth of the initial flow to B, the number of decomposed paths were larger.

The procedures for finding an initial network flow are as follows.

- The *Maximum Widest Path Flow (WIDE)* procedure: Here, to find an initial feasible network flow, the procedure iteratively augments the *widest path* available from s to t until the maximum flow is reached. If there are multiple paths, one of them is selected arbitrarily. The running time of this procedure is $O(|E|^2 \log(|V|) \log(C_{max}))$ [1], where C_{max} is the maximal capacity of an edge in the network.
- The *Maximum Shortest Path Flow (SHORT)* procedure: Here, to find an initial feasible network flow, the procedure iteratively augments the *shortest path* available from s to t until the maximum flow is reached. If there are multiple paths, one of them is selected arbitrarily. This is the well-known Edmonds-Karp algorithm [9] for finding a maximum flow. The running time of this procedure is $O(|V||E|^2)$.
- The *Maximum Shortest Widest Path Flow (S-WIDE)* procedure: This procedure is similar to WIDE, except that when there is more than one path of maximum width in any iteration, the shortest is chosen. This procedure can be implemented using a simple dynamic programming algorithm with a running time of $O(|V||E|^2 \log(C_{max}))$.
- The *Maximum Widest Shortest Path Flow (W-SHORT)* procedure: This procedure is similar to SHORT, except that when there is more than one path of minimum length in any iteration, the widest of them is chosen. This procedure can be implemented using a simple dynamic programming algorithm with a running time of $O(|V||E|^2)$.
- The *Maximum Width/Length Path Flow (WID/LEN)* procedure: This procedure iteratively chooses the path with the largest width-length ratio. This procedure can be implemented using a dynamic programming algorithm with a running time of $O(|V|^2|E|^3 \log^2(|V|))$.

VI. DMCL AND RMCL WITH NODE MINIMIZATION

In many cases, a network operator seeks to minimize the number of nodes that carry the paths rather than the number of paths. In such a case, it may be better to set up many short paths rather than fewer long ones. To address such cases, we now change our optimization problem and view the control load as the number of nodes that carry the paths. More formally, given a set Π of simple directed paths from s to t, the control load imposed by Π is measured by $\sum_{p \in \Pi} |p|$, where $|p|$ is the number of nodes along the path p.

A. DMCL with Node Minimization (DMCL(n))

We now show that DMCL(n) can not be solved in polynomial time. Then, we propose an approximation algorithm for it.

Theorem 5: DMCL(n) is NP-complete.

Proof: We show this using a reduction from DMCL(p) to DMCL(n). Given an instance of the former problem, we construct an instance of the latter. We set the source of DMCL(n) to be a new node, s', which is connected to s using a chain of $|V||E|$ links whose capacity is B. The network flow of DMCL(n) is carried over the new chain from s' to s, and then to t as the network flow in the DMCL(p) instance. We now show that the minimum-node decomposition of the DMCL(n) flow, P_n^*, has the same number of paths as the number of paths in the minimum-path decomposition of the DMCL(p) flow, P_p^*. First, $|P_n^*| \geq |P_p^*|$ must hold, because otherwise P_p^* is not a minimum-path decomposition in DMCL(p). Second, if $|P_n^*| > |P_p^*|$ then P_p^* induces a decomposition for the DMCL(n) flow with a smaller number of nodes than that imposed by P_n^* (because each additional path in the DMCL(n) decomposition increases the number of nodes by $|V||E|$, which is greater than the number of nodes of any decomposition in DMCL(p)).

Therefore, an optimal solution for the DMCL(p) instance can be derived from an optimal solution for the constructed DMCL(n) instance. ■

Algorithm 1 can be modified to approximate DMCL(n) with the same approximation ratio $O(\log(B/b))$. The idea is to choose in each iteration the path with the greatest ratio between the bandwidth it carries and the number of nodes it traverses:

Algorithm 5: (A greedy algorithm for DMCL(n))

1) $B^0 \leftarrow B$, $f^0 \leftarrow f$, $P \leftarrow \phi$, $i \leftarrow 0$.
2) Repeat until $B^i = 0$:
   a) Choose the path p from the source to the destination for which $f^i(p)/n_p$ is maximum, where $f^i(p)$ is the bandwidth of p in f^i and n_p is the number of nodes p traverses.
   b) $B^{i+1} \leftarrow B^i - f^i(p)$, $f^{i+1} \leftarrow f^i \backslash p$, $P \leftarrow P \cup p$, $i \leftarrow i + 1$.
3) Return P. □

Algorithm 5 has the same computational complexity as Algorithm 1.

Theorem 6: The approximation ratio of Algorithm 5 is $O(\log(B/b))$.

Proof: The proof is similar to that of Theorem 1. Denote the number of paths in the optimal solution by OPT and each path in the optimal solution by p_j^*, where $1 \leq j \leq OPT$. Let p_i be the path chosen by the algorithm in step i, and u_p^i be the ratio $f^i(p)/n_p$. The path chosen in each step is the one with the greatest ratio u. Thus, for every j $u_{p_i}^i \geq u_{p_j^*}^i$ and $u_{p_i}^i \cdot n_{p_j^*} \geq u_{p_j^*}^i \cdot n_{p_j^*}$ hold. Therefore,

$$OPT \cdot u_{p_i}^i \geq \sum_{j=1}^{OPT} u_{p_j^*}^i \cdot n_{p_j^*} \geq B^i/b. \quad (1)$$

The second inequality holds because $u_{p_j^*}^i \cdot n_{p_j^*} = f^i(p_j^*)$ and because the entire set of optimal paths is a decomposition of the network flow f and, therefore, of any network flow f^i. This leads to

$$\frac{1}{u_{p_i}^i} \leq \frac{OPT}{B^i/b} \leq \frac{OPT}{B/b - (i+1)}.$$

The rest of the proof is identical to the proof of Theorem 1. ■

B. RMCL with Node Minimization (RMCL(n))

Using a proof similar to that of Theorem 5 for DMCL(n), it can be shown that

Theorem 7: RMCL(n) is NP-complete.

Theorem 8: An α-approximation algorithm for RMCL(n) yields an $(\alpha + \epsilon)$-approximation algorithm for RMCL(p), where $\epsilon > 0$ is arbitrarily small.

Proof: The reduction used in the proof of Theorem 5 can be used again, but this time the length of the added chain is $M = |V||E| \cdot \alpha \cdot (1/\epsilon)$. If we have an α-approximation algorithm for RMCL(n), we can apply it to the new flow constructed by the reduction. Let ALG_p and ALG_n be the number of paths and the number of nodes in the solutions found by the two approximation algorithms. Let OPT_p and OPT_n be the number of paths and the number of nodes in the corresponding optimal solutions. From the reduction it is obvious that

$$ALG_n \geq ALG_p \cdot M.$$

On the other hand, we have:

$$ALG_n \leq \alpha \cdot OPT_n \leq \alpha \cdot (OPT_p \cdot M + X),$$

where X is the number of nodes in the minimum-node decomposition on the original graph, which is obviously smaller than $|V||E|$. The right inequality holds because in the proof of Theorem 5 we showed that the number of paths of the minimum-node decomposition must be equal to the number of paths in the minimum-path decomposition. Hence we have

$$\alpha(OPT_p + \frac{X}{M}) \geq ALG_p$$
$$OPT_p(\alpha + \epsilon) \geq ALG_p.$$

■

From Theorems 3 and 8 we derive the following corollary:

Corollary 2: RMCL(n) cannot be approximated within a factor of $3/2 - \epsilon$.

We now present an approximation algorithm for RMCL(n). The algorithm is based on Algorithm 2 for RMCL(p). In Algorithm 2 each flow unit is transformed

into a path. Therefore, if the network nodes were assigned a cost of 1, the number of nodes for the decomposed paths returned by Algorithm 2 would be equal to the cost of their total network flow. Hence, we shall modify Algorithm 2 to find a minimum cost network flow before it is decomposed. Clearly, Corollary 1 still holds for this minimum cost flow version of the algorithm.

Since an algorithm for finding a minimum cost network flow addresses the case where the edges, rather than the nodes, have a cost, we will consider the following simple reduction. Consider a network where every unit of flow on a node v incurs a cost of 1. Every node v is transformed into two nodes, v_i and v_o, connected by an edge $v_i \rightarrow v_o$ with infinite capacity and a cost of 1. All the other edges have zero cost. All edges going into v will go into v_i and all edges from v will go out from v_o.

Algorithm 6: (A scaling algorithm for RMCL(n))

1) Assign to each node in the network a cost of 1.
2) Transform the network to one with costs on the edges (as described above).
3) Add a source node s' and an edge $s \rightarrow s'$ with capacity B.
4) For $k = 1 \ldots |E|$.
   a) Run a minimum cost network flow version of Algorithm 2 with a scaling factor $\alpha = \frac{B}{k \cdot \beta}$.
   b) Store the result as f_k.
5) Return f_i with minimum cost whose value is at least $B\left(1 - \frac{1}{\beta}\right)$. □

Theorem 9: The solution returned by Algorithm 6 has a value greater than $B\left(1 - \frac{1}{\beta}\right)$ and a cost smaller than $\beta k^* N^*$, where k^* and N^* are the number of paths and the number of nodes in the optimal solution.

Proof

Corollary 1 holds during the iteration where $k = k^*$. Hence, the value of f_{k^*} is $\geq B\left(1 - \frac{1}{\beta}\right)$. Let $p_1, p_2, \ldots, p_{k^*}$ be the set of paths in an optimal solution. The optimal scaled solution is feasible. The value of this solution is the number of nodes in $p_1, p_2, \ldots, p_{k^*}$. Since every path p_i carries a single-path flow of w_i, where $\sum_i w_i = B$, this value is $\sum_{i=1}^{k^*} |p_i| > \max_i |p_i|$. In addition, the cost of the scaled optimal solution is

$$\sum_{i=1}^{k^*} \left\lfloor \frac{w_i}{\alpha} \right\rfloor |p_i| < \left\lfloor \frac{B}{\alpha} \right\rfloor \max_i |p_i|.$$

Clearly, the cost of f_{k^*} is less than that of the scaled optimal solution. Hence,

$$\frac{f_{k^*}}{N^*} < \frac{k^* \cdot \beta \cdot \max_i |p_i|}{\max_i |p_i|} = k^* \cdot \beta.$$

□

As in RMCL(p), the above algorithm for RMCL(n) has theoretical value. However, our simulation results indicate that its actual average performance is not good enough. Therefore, we present an additional algorithm that has no worst case performance guarantee, but a very good actual performance.

The algorithm is similar to Algorithm 4 presented for RMCL(p). Its main idea is to break the RMCL(n) solution into two stages. First, a network flow that provides a bandwidth of at least B is found. Then, this flow is decomposed using Algorithm 5.

Algorithm 7: (A 2-phase algorithm for RMCL(n))

1) Find an initial feasible network flow of bandwidth B or more from s to t.
2) Use Algorithm 5 for decomposing the network flow into a set of paths that traverse a minimum number of nodes and deliver together a bandwidth of B.
3) Return the set of paths produced by Algorithm 5. □

The initial network flow can be found by one of the four procedures (WIDE, SHORT, S-WIDE and W-SHORT) described in Section V.

VII. Simulation Study

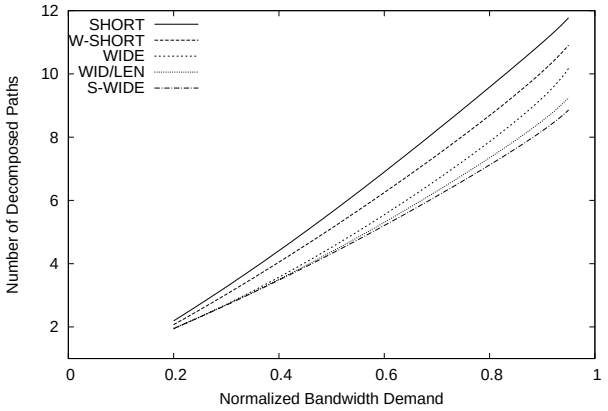
In this section we evaluate the performance of the algorithms for RMCL and DMCL. We first examine the performance of the two variants of the RMCL algorithms. Then we evaluate the trade-off between the bandwidth cost of a network flow and the load imposed on the control plane by comparing the performance of the RMCL algorithms to the performance of the DMCL algorithms as they apply to a network flow of minimum bandwidth cost.

We use the BRITE simulator [19] to simulate network domain topologies according to the "preferential attachment model" of Barabasi et al. [5]. This model captures two important characteristics of network topologies: incremental growth and preferential connectivity of a new node to well-connected existing nodes. These characteristics yield a power-law degree distribution of the nodes. In addition, we also run our algorithms on actual ISP topologies, as inferred from the RocketFuel project [22]. These topologies reflect better the model presented in [2]. For each synthetic or real topology, we generate a bandwidth demand between a source and a destination. The characteristics of the simulated topologies and the methods for choosing the bandwidth demands are described for each setting. A network topology together with a bandwidth demand are considered as one simulation instance. We apply the various algorithms for each such instance.

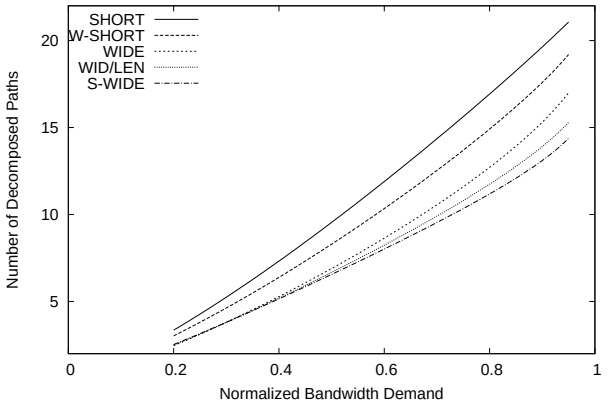
A. Minimizing the Control Load

Figure 3 depicts the number of paths over which the required bandwidth can be delivered as a function of the bandwidth demand for three types of network domains:

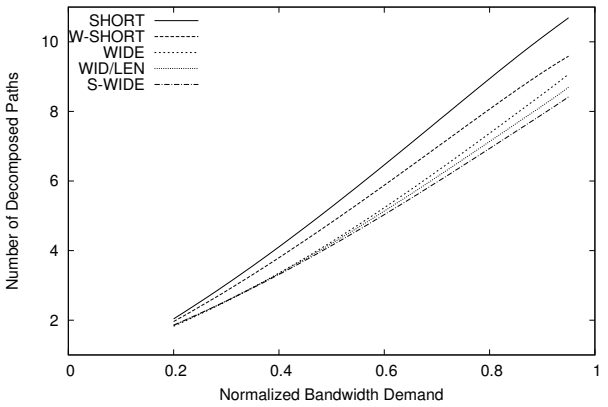
(a) Networks with 100 nodes whose average degree is 5 links (Figure 3(a)).



(a) 100 nodes with average degree = 5



(b) 100 nodes with average degree = 10



(c) 50 nodes with average degree = 5

Fig. 3. The number of paths found by Algorithm 4 with the various procedures as a function of the normalized bandwidth demand for various sizes of network domains

(b) Networks with 100 nodes whose average degree is 10 (Figure 3(b)).

(c) Networks with 50 nodes whose average degree is 5 (Figure 3(c)).

For each such network, the edge capacities are uniformly distributed in $[0.5C, 1.5C]$. C is a normalizing factor for the edge capacities and the volume of bandwidth demands. The y-axis of all the graphs in Figure 3 represents the number of decomposed paths produced by the various algorithms. The x-axis represents the normalized bandwidth demand from s to t, i.e., the bandwidth demand divided by the value of the largest maximum network flow that exists between any pair of nodes in the network. For every network size, we generate 100 instances. For each network instance and for each average bandwidth value, we generate 100 instances of bandwidth demands that are uniformly distributed with a variation of $\pm 10\%$. For each demand, the source and destination nodes are uniformly selected from among the network nodes, and the five variants of Algorithm 4 are executed.

As clearly indicated by all the graphs in Figure 3, Algorithm 4 minimizes the number of paths needed for delivering the requested bandwidth when it uses S-WIDE in Step 1. W-SHORT performs better than SHORT because it produces network flows with larger average bandwidth on each edge. This allows Algorithm 1 (in Step 2 of Algorithm 4) to choose wider, and consequently fewer, paths. S-WIDE and WID/LEN perform better than WIDE because they take into account the length of the paths. Hence, less bandwidth is consumed during each iteration, and more bandwidth is left for latter iterations. Consequently, wider paths are found. S-WIDE is still slightly better than WID/LEN since it chooses wider paths.

The number of decomposed paths increases linearly with the bandwidth demand. For a network with 100 nodes and an average degree of 10 (Figure 3(b)), the number of decomposed paths as well as the slope of the curves are almost doubled compared to that of networks with an average degree of 5 (Figure 3(a)). This is because we use larger bandwidth demands (recall that the bandwidth demand shown in the graphs is normalized to the largest maximum network flow in the network, which increases with the node degree). In addition, the relative difference between the number of paths using S-WIDE and the number of paths using SHORT increases: it is now roughly 50% compared to 25% for a network with an average degree of 5 (Figure 3(a)). This is because the number of possible paths between any two network nodes significantly increases. This allows S-WIDE to find wider augmenting paths. Hence, the network flow is constructed with fewer iterations, which is translated into a smaller number of decomposed paths. For a network with 50 nodes and an average degree of 5 (Figure 3(c)), the results are very similar to those achieved for 100 nodes and the same average degree. However, the number of decomposed paths is slightly smaller, because the

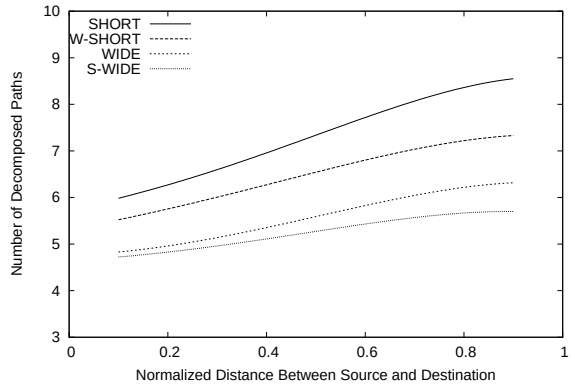


Fig. 4. The number of paths found by Algorithm 4 with the various procedures as a function of the distance between source-destination pairs

average path length is smaller when the network is smaller. Consequently, the average path capacity increases.

In Figure 4 we examine how the distance between the source and destination influences the number of decomposed paths. As in Figure 3, the y-axis represents the number of decomposed paths for each procedure. The x-axis represents the distance between the source and destination divided by the diameter of the network. The network has 100 nodes and an average degree of 5. We generate 100 network instances of this size. For each instance we generate 100 bandwidth demands. For each demand, the distance between the source and the destination is assigned a given distance value with $\pm 10\%$ variation. The average normalized bandwidth for each demand is 0.6. For all of the procedures, the number of decomposed paths increases with the distance. Consequently, the capacities of the paths between the source and destination decrease. Thus, each decomposed path can carry less bandwidth on the average. The number of decomposed paths increases more moderately for WIDE and S-WIDE. This is because the source-destination distance has a smaller effect on the length of the widest path between them than on the length of the shortest path between them.

To validate the results from the synthetic graphs, we present in Figure 5 results for real AS topologies, as inferred from the RocketFuel project [22]. These topologies reflect the model presented in [2], which may better represent a router-level ISP topology. We used the following network topologies:

1) Exodus ISP, which consists of 80 routers with average degree of 1.8,
2) Telstra ISP, which consists of 115 routers with average degree of 1.3,
3) Abovenet ISP, which consists of 145 routers with average degree of 2.6,
4) Ebone ISP, which consists of 88 routers with average degree of 2.

The bandwidth demands are generated as described for Fig. 3. Figure 5 shows the performance of Algorithms 4 with the various procedures. We can see that the relative performance rank is the same as for the synthetic graphs (Figure 3) that reflect the preferential attachment model. In the Telstra topology (Figure 5(b)), the performance differences are smaller because of its lower link degree, which substantially reduces the path diversity in the network.

B. Minimizing the Number of Nodes

We now examine the performance of Algorithm 7, the goal of which is to minimize the number of nodes. Figure 6 depicts the number of nodes traversed by all of the paths that deliver the required bandwidth as a function of the bandwidth demand for the network domains considered in Figure 3. The network instances and bandwidth demands are generated as described for Fig. 3.

It is clear that Algorithm 7 gives the best performance when it uses WID/LEN for finding an initial network flow. S-WIDE, which was shown to yield the smallest number of paths, produces solutions with roughly 20% more nodes than WID/LEN. This result is consistent for all three routing domain sizes.

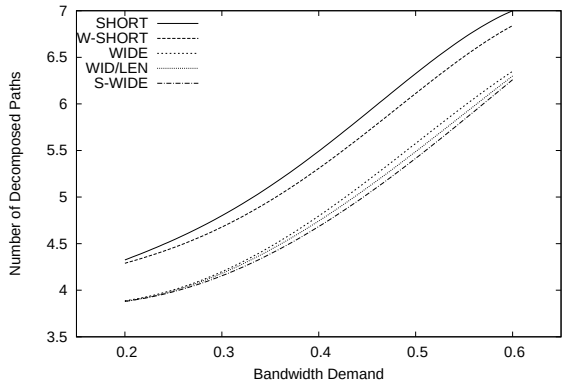
The advantage of WID/LEN over S-WIDE indicates that it is better to choose more shorter paths rather than fewer wide ones in order to minimize the number of nodes. This insight is supported by the results of WIDE, which yields the worst performance. This is because WIDE is the only procedure that does not take into account the length of the chosen paths. Despite of this shortcoming, S-WIDE is slightly better than SHORT. This indicates that the number of paths still influences the number of nodes.

C. The Trade-Off Between Bandwidth Cost and Control Plane Load

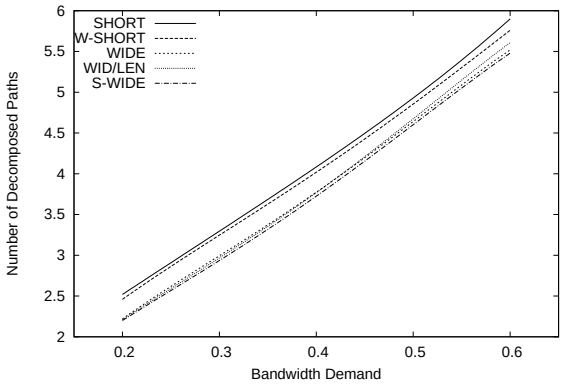
We now study the trade-off between the bandwidth cost of a network flow and the load imposed on the control plane for setting up and maintaining the paths into which a network flow is decomposed. To this end, we focus on the following three questions:

- What is the extra burden imposed on the control plane when the main target is minimizing the bandwidth cost?
- What is the extra bandwidth cost imposed on the data plane when the main target is minimizing the load on the control plane?
- How do the various procedures perform with respect to this trade-off?

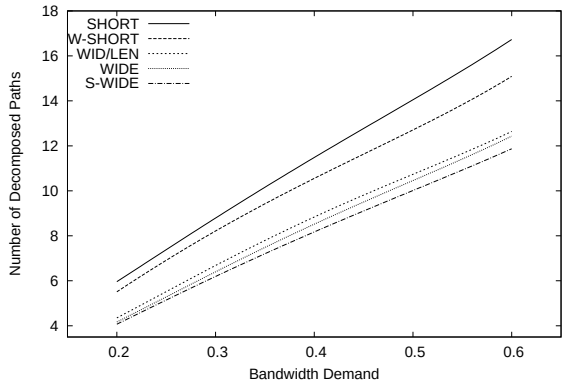
Finding the minimum-cost network flow in general networks is a well-studied problem [1], [9], [11]. In what follows we use the well-known Edmonds-Karp algorithm [9]. This algorithm iteratively adds to the constructed network flow the least cost path until the bandwidth demand is satisfied. In the following, we refer to this procedure as COST. We decompose the network flow using Algorithms 1



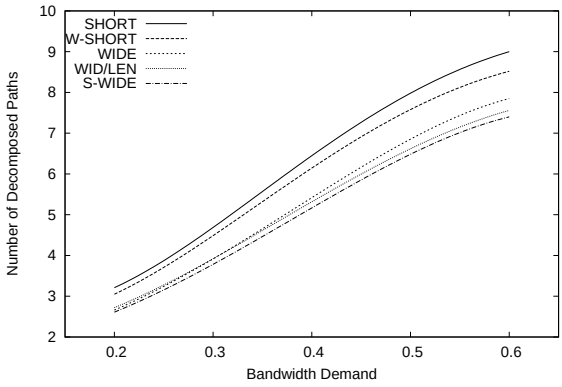
(a) Exodus ISP, 80 routers with average degree of 1.8



(b) Telstra ISP, 115 routers with average degree of 1.3



(c) Abovenet ISP, 145 routers with average degree of 2.6



(d) Ebone ISP, 88 routers with average degree of 2

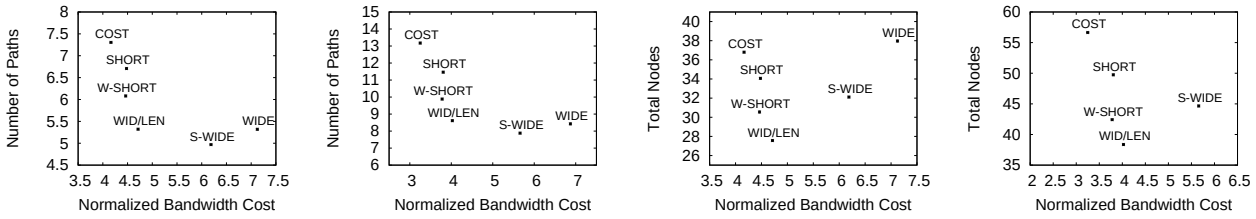
Fig. 5. The number of paths found by Algorithm 4 with the various procedures for real ISP topologies

and 5 to find the paths that impose the minimal control load. Note that the difference between COST and SHORT is that SHORT constructs a maximal network flow while COST returns a network flow of bandwidth B. Consequently, the decomposition algorithm has less flexibility in the latter case.

We use the same simulation setting as described earlier, and assign an equal cost '1' to each flow unit on every link. Figures 7(a) and 7(b) show the trade-off between the bandwidth cost and the number of decomposed paths for each of the six procedures for Step 1 of Algorithm 4. The results are shown for networks with 100 nodes whose average node degree is 5 or 10. The x-axis represents the bandwidth cost normalized by the value of the actual bandwidth demand, while the y-axis shows the number of decomposed paths. The results are shown for a normalized bandwidth demand of 0.6.

As expected, for both network sizes, the bandwidth cost is minimized using COST, but yields the largest number of paths. S-WIDE minimizes the number of paths, but its bandwidth cost is 50% more than COST. From Figures 7(a) and 7(b) we conclude that WID/LEN yields a very good trade-off between these two extemes. Its bandwidth cost is only 10% more than that of COST while it has only 5% more paths than S-WIDE.

Figures 7(c) and 7(d) show the trade-off between the bandwidth cost of a network flow and the aggregated number of nodes that participate in the setup and maintenance of all paths that carry this flow for each of the six procedures for Step 1 of Algorithm 7. The results are shown again for networks with 100 nodes whose average degree is 5 or 10 and a normalized bandwidth demand of 0.6. As noted above, WID/LEN is the best procedure in terms of the aggregated load it imposes on the network nodes. Moreover, its bandwidth cost is almost as small as that of COST. Therefore, WID/LEN gives the best trade-off between bandwidth cost and control load imposed on the network nodes.



(a) 100 nodes, avg. degree = 5 (b) 100 nodes, avg. degree = 10 (c) 100 nodes, avg. degree = 5 (d) 100 nodes, avg. degree = 10

Fig. 7. The trade-off between the bandwidth cost and the number of paths/nodes that carry this bandwidth

Our conclusion from these simulations is that WID/LEN is the procedure of choice for Step 1 of both Algorithm 4 and Algorithm 7 because it gives the best trade-off between control overhead and bandwidth cost.

VIII. Conclusions

This paper is the first to in-depth investigate the problem of minimizing the control load while delivering a given amount of bandwidth between pairs of ingress and egress nodes. This paper is also the first to study the trade-off between minimizing the cost of a network flow and minimizing the control load. To this end, we defined two optimization problems: the Decomposition with Minimum Control Load (DMCL) problem and the Routing with Minimum Control Load (RMCL) problem. We measure the control load by either the number of paths that carry the bandwidth or by the number of nodes they traverse. Both problems are NP-hard for both control load measures. However, we presented approximation algorithms for all four problem variants. Furthermore, we presented efficient practical algorithms for RMCL. These algorithms first find an initial network flow and then decompose it while trying to minimize the number of paths or the number of nodes. The procedure for selecting the initial network flow was shown to have a critical impact on the performance of the algorithm. While S-WIDE is preferable when the main optimization criterion is to minimize the number of paths, WID/LEN gives the best trade-off between bandwidth cost and control load.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, February 1993.

[2] D. Alderson, L. Li, W. Willinger, and J. C. Doyle. Understanding internet topology: principles, models, and validation. *IEEE/ACM Transactions on Networking*, 13(6):1205–1218, 2005.

[3] G. Baier, E. Köhler, and M. Skutella. The k-splittable flow problem. *Algorithmica*, 42(3-4):231–248, 2005.

[4] R. Banner and A. Orda. Efficient multipath-routing schemes for congestion minimization. Technical report, Technion – Israel Institute of Technology, 2004.

[5] A. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: the topology of the World Wide Web. In *Physica A: Statistical Mechanics and Its Applications*, volume 281, pages 69–77, June 2006.

[6] R. Bonica, D. Gan, D. Tappan, and C. Pignataro. ICMP extensions for multiprotocol label switching. IETF RFC 4950, August 2007.

[7] J. Boyle and et. al. Applicability statement for traffic engineering with MPLS. IETF RFC 3346, August 2002.

[8] S. Das, G. Parulkar, N. McKeown, P. Singh, D. Getachew, and L. Ong. Packet and circuit network convergence with openflow. In *Optical Fiber Communication (OFC), collocated National Fiber Optic Engineers Conference*, 2010.

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

[10] D. A. et al. RSVP-TE: Extensions to RSVP for LSP tunnels. IETF RFC 3209, December 2001.

[11] A. Goldberg and R. Tarjan. Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM*, 36(4):873–886, 1989.

[12] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[13] D. Katz et al. Traffic engineering (TE) extensions to OSPF version 2. IETF RFC 3630, September 2003.

[14] R. Koch, M. Skutella, and I. Spenke. Maximum k-splittable s,t-flows. *Theory of Computing Systems*, 43(1):56–66, 2008.

[15] R. Koch and I. Spenke. Complexity and approximability of k-splittable flows. *Theoretical Computer Science*, 369(1):338–347, 2006.

[16] K. Kompella and G. Swallow. Detecting multi-protocol label switched (MPLS) data plane failures. IETF RFC 4379, February 2006.

[17] N. McKeown. Software-defined networking. In *INFOCOM (keynote talk)*, 2009.

[18] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.*, 38, March 2008.

[19] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: An approach to universal topology generation. In *Proceedings of MASCOTS*, 2001.

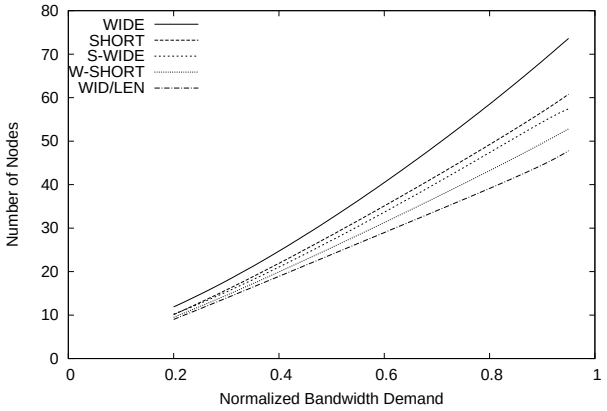
[20] V. S. Mirrokni, M. Thottan, H. Uzunalioglu, and S. Paul. A simple polynomial time framework to reduced path decomposition in multi-path routing. In *IEEE INFOCOM*, 2004.

[21] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of treewidth. *Journal of Algorithms*, 7:309–322, 1986.

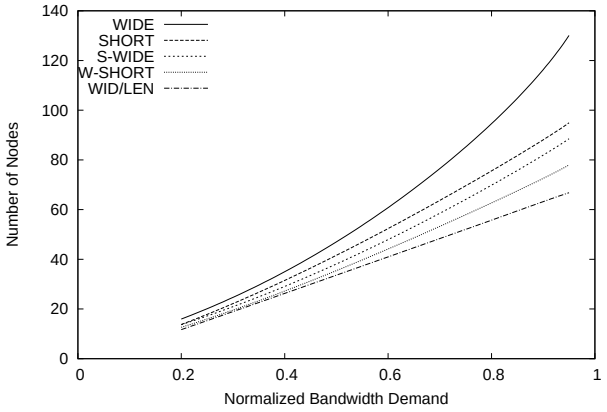
[22] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with RocketFuel. In *Proceedings of the ACM SIGCOMM*, August 2002.

[23] B. Vatinlen, F. Chauvet, P. Chrétienne, and P. Mahey. Simple bounds and greedy algorithms for decomposing a flow into a minimal set of paths. *European Journal of Operational Research*, 185(3):1390–1401, 2008.

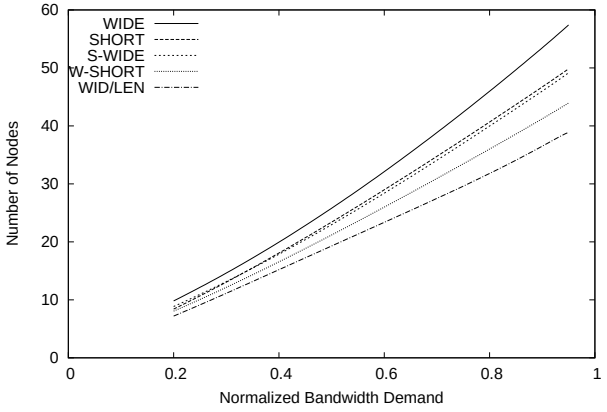
[24] H. Wang, J. Lou, Y. Chen, Y. Sun, and X. Shen. Achieving maximum throughput with a minimum number of label switched paths in MPLS networks. In *Proceedings of ICCCN*, pages 187–192, 2005.



(a) Num. nodes = 100, average degree = 5



(b) Num. nodes = 100, average degree = 10



(c) Num. nodes = 50, average degree = 5

Fig. 6. The number of nodes found by Algorithm 7 with the various procedures as a function of the normalized bandwidth demand for various sizes of network domains

gabi.jpg

Gabi Nakibly received the B. Sc. in Information Systems engineering (summa cum laude) and PhD in Computer Science from the Technion - Israel Institute of Technology, Haifa, Israel, in 1999 and 2008, respectively. Gabi is a professional fellow in the National EW Research & Simulation Center at Rafael Advanced Defense Systems. He also serves as an adjunct researcher and lecturer in the Computer Science department at the Technion. Gabi received his Ph.D. in computer Science in 2008 from the Technion, and he is a recipient of the Katzir Fellowship. His main research interests include network security and traffic engineering.

reuven.jpg

Reuven Cohen received the B.Sc., M.Sc. and Ph.D. degrees in Computer Science from the Technion - Israel Institute of Technology, completing his Ph.D. studies in 1991. From 1991 to 1993, he was with the IBM T.J. Watson Research Center, working on protocols for high speed networks. Since 1993, he has been a professor in the Department of Computer Science at the Technion. He has also been a consultant for numerous companies, mainly in the context of protocols and architectures for broadband access networks. Reuven Cohen has served as an editor of the IEEE/ACM Transactions on Networking and the ACM/Kluwer Journal on Wireless Networks (WINET). He was the co-chair of the technical program committee of Infocom 2010 and headed the Israeli chapter of the IEEE Communications Society from 2002 to 2010.

liran.jpg

Liran Katzir received his B.A., M.A., and Ph.D. degrees in computer science from the Technion Israel Institute of Technology, Haifa, Israel, in 2001, 2005, and 2008 respectively. His PhD. thesis is about scheduling in advanced wireless networks. Dr. Katzir is now a member of the networking research group in the Technion's CS department, working on technologies for the fourth generation cellular network.