Efficient Multipath Flow Monitoring

Samuel Micka, Sean Yaw, Brittany Terese Fasy, Brendan Mumey, Mike Wittie
Gianforte School of Computing, Montana State University
Bozeman, Montana, USA
{samuel.micka, sean.yaw}@msu.montana.edu {brittany.fasy, brendan.mumey, mike.wittie}@montana.edu

***Abstract*—Network administrators and traffic management tools need up-to-date traffic metrics to monitor and balance traffic load. Recording and reporting flow information is costly in terms of control plane traffic and router memory. Prior work mitigates these costs by limiting the number of monitoring devices, sampling, or only reporting constraint violations. However, these techniques are limited to single-path routing, do not address multiple network flows, and do not guarantee network coverage. We propose strategies to minimize the cost of placing a sufficient number of logical monitors on edges (called *turnstiles*) so as to monitor all (possibly multipath) flows. Our main result is an $(\ln m+1)(\ln k+1)$-approximation algorithm for the general *Turnstile Placement* problem, for a network with m edges and k flows (commodities). We also show achieving an $o(\ln k)$ approximation ratio is NP-hard. We examine a simple heuristic algorithm for the problem as well as a method to adapt existing single path solutions. Simulations show that our approximation algorithm achieves near optimal performance and outperforms existing approaches. The proposed methods reduce monitoring costs in multipath networks, enabling more agile and more accurate load balancing of network flows.**

I. Introduction

With the increasing prevalence of Internet-connected devices, delivering data promptly continues to be an important challenge. Software defined networks (SDNs) reduce network management complexity through a simple, yet powerful control model. A logically centralized controller with a global view of the network state installs forwarding rules on routers, which match and apply them to arriving packets. Flow volumes, collected by router hardware counters, are a crucial source of network data for the controller. Based on this information, the controller alters forwarding rules to balance network load.

As SDN controllers become integrated with application delivery controllers (ADCs) to meet application and even flow-level quality of service (QoS) requirements, they require flow volume information of increasing spatial and temporal resolution [1], [12]. These flow-specific metrics can be collected from routers configured to monitor flow traffic, which we simply refer to as *monitors* in this paper. Such detailed network state, if recorded and reported naïvely, creates control plane congestion and limits the scalability of the centralized control model in dynamic network scenarios.

Optimizing the placement of monitors is necessary to observe all flows in a network, and will help reduce the amount of control plane reporting overhead. This cost reduction will allow for up-to-date, low-cost, and complete reporting of flow volumes to the SDN centralized controller. Accurate reporting of flow specific metrics will help traffic management tools make load balancing decisions to improve network performance through congestion reduction. Perhaps most importantly, less costly monitoring enables SDNs to take on more load balancing functions and assures the scalability of centralized network control in dynamic networks.

ISBN 978-3-901882-94-4 © 2017 IFIP

We update existing theoretical work on minimum monitor placement to include the multipath flows present in modern networks. We call the monitors we use *turnstiles*, and place them logically on edges in the network. Physically, these monitors are not additional hardware, but rather traffic recording on specific router interfaces. The ***Turnstile Placement (TP)*** Problem looks for minimum cost turnstile placements to monitor multiple multipath flows in a network. The TP Problem formulation supports general turnstile cost functions for customized performance metric optimization (e.g., minimize the number of turnstiles, minimize the distance to controller). We establish a polynomial time solution to the TP Problem when there is only a single commodity by relating it to the cycle-transversal problem [18], [19]. We show the general TP Problem is NP-hard to approximate within a bound of $o(\ln k)$, and introduce a novel $(\ln m + 1)(\ln k + 1)$-approximation algorithm for the TP Problem, where m is the number of edges in the graph and k is the number of commodities. We also present a spanning tree based algorithm for the general TP Problem that leverages the relationship between the single commodity TP Problem and the cycle-transversal problem. Finally, we develop a method for adapting existing single path flow monitoring solutions to multipath TP instances.

To evaluate the efficacy of our approaches, we compare the monitor placement costs of our algorithms on real Internet topologies with realistic traffic patterns. The simulations show that the approximation algorithm placements are within 1% of optimal, on scenarios for which the optimal solution could be computed. On larger examples, the approximation algorithm outperforms the other methods proposed.

The rest of this paper is organized as follows. Section II discusses related work. Section III formalizes the TP Problem, details its relationship to other fundamental graph problems, and discusses its complexity. Section IV presents an approximation algorithm, a maximum spanning tree approach, and a method to apply existing single path solutions to TP instances. Section V presents experimental results, and we conclude in Section VI. The Appendix includes a proof of the complexity results reported in Section III.

II. Related Work

We classify previous efforts to reduce the overhead of control traffic in network monitoring into three categories: (1) sampling or stopping monitor placement after covering a portion of the network, (2) utilizing heuristics, and (3) considering a simplified problem formulation in which flows have a single path.

The first category of related work deals with monitor placement using sampling techniques [10], [17], [20], [22]. Jackson et al. focus on capturing a high percentage of network traffic data by using a minimum number of monitors on autonomous system boundaries [10]. However, considering different autonomous systems as single entities results in a loss of flow monitoring precision at the router level. Another common approach to reduce monitoring load is to sample only a fraction of packets at a router [17], [20], [22]. Suh et al., in one problem formulation, consider the problem of monitoring flows by maximizing the coverage without violating the deployment and operating costs of monitors, while optimizing sampling rates [17]. Cantieni et al. also optimize monitor placement and sampling strategies to achieve high monitoring coverage [2]. However, sampling approaches result in a probabilistic coverage of the network flows, which can lead to lower accuracy of flow volume estimation and increase the likelihood of missing smaller flows. Probabilistic and redundant coverage of sampling is also a drawback of monitoring tools available on routers such as NetFlow, or CMON [5], [8].

The second type of solution uses heuristics to place monitors [3], [7], [14], [20], [21]. Zang et al. propose greedy heuristics that monitors flows on each edge [20]. Huang et al. optimize monitor placement in dynamic routing scenarios using one Mixed Integer Linear Program (MILP) and two heuristics [7]. The authors later introduce a framework, LEISURE, that balances monitoring tasks equally between different monitors in the network [3]. Heuristic solutions, however, do not offer performance guarantees.

Third, previous work considers single path routing for each flow [2], [4], [17]. Chaudet et al. consider minimizing monitoring overhead with passive monitors by minimizing the number of monitors needed to observe the network edges [4]. Assuming fixed path routing allows the authors to show the problem's equivalence to Set Cover as well as allowing for a reduction from their problem to the Minimum Edge Cost Flow problem. In one formulation of the problem, Chaudet et al. consider the multipath version of the problem but only introduce an ILP with no additional complexity results. Suh et al. define several variations of network monitoring problems, without sampling, show them to be NP-Hard, and propose approximation algorithms and Integer Program solutions [17]. The shortcoming of single path routing is that they are not compatible with multipath forwarding of equal-cost multipath routing (ECMP), or multipath TCP (MPTCP).

Our work takes two approaches to addressing these gaps. First, we provide an approximation algorithm for multipath flow instances that achieves a performance guarantee and a novel heuristic. Second, we provide a method to adapt existing single path flow monitoring solutions in a multipath context. Our solutions also use a flexible cost function able to minimize the overhead of monitoring control traffic.

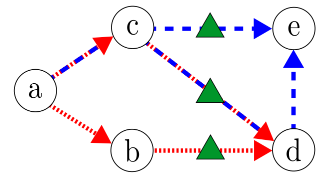


Fig. 1. Sample TP instance including a directed network and two multipath commodities. Three turnstiles (shown as green triangles) suffice to monitor all commodity flows.

One final related work category monitors vertices in the network, rather then edges [6], [13], [14], [21]. Gupta et al. develop techniques to reduce overall monitoring overhead in wireless mesh networks [6]. One of the techniques that they utilize is minimizing the number of routers used as monitors with a modified vertex cover approximation algorithm that prioritizes vertices with a high degree. Park and Lee also maximize network coverage while reducing the number of monitors, placed on vertices, to detect DDoS attacks in networks [13]. Zeng et al. and Qin et al. focus on minimizing the number of vertex monitors in a network using heuristics [14], [21]. These approaches have the cost equivalent to monitoring traffic on all the edges at a vertex – a solution subsumed by our approach and avoided in practice due to its high cost.

III. Problem Formulation

We consider a network, comprised of a set of vertices V and directed edges E. Each of the k flows in the network consists of a source s_k and destination t_k vertex, along with a set of edges, $E_k \subset E$, hosting the flow. Flow instances (s_k, t_k, E_k) are known, but traffic volumes on each edge are unknown. Additionally, a weight $c(e)$ is associated with each edge e in E, reflecting the cost of monitoring the traffic on that edge. This monitoring cost function enables optimization of various performance metrics (e.g., minimize the number of monitors placed, minimize distance between monitors and controllers).

By monitoring traffic patterns on a select set of edges, we seek to determine the flow volumes on all edges of the network. The network is subject to the normal conservation of flow assumption: The amount of flow into each vertex equals the amount out, except at sources and destinations. We aim to identify the amount of traffic traversing each edge in the network by utilizing *turnstile* monitors to disambiguate traffic volume on each edge. A turnstile on an edge e logs all traffic through e and is able to determine which flow the traffic belongs to. The placement of a turnstile incurs the cost of the edge weight associated with monitoring that edge. The goal is to place a minimum cost set of turnstiles so as to determine the traffic volume on every edge in the network. The Turnstile Placement Problem is formally defined below, and a sample instance is presented in Figure 1.

Definition 1 (Turnstile Placement Problem)**.** *Given a weighted, directed graph, $G = (V, E, c)$, where $c\colon E \to \mathbb{R}$ is a turnstile placement cost function, and a set of commodity flows $\{(s_k, t_k, E_k)\}_k$, where s_k is the source, t_k is the destination, and E_k are the edges used for the k^{th} commodity, the* ***Turnstile Placement (TP)*** *Problem seeks a subset $M \subseteq E$ of minimum total cost, such that knowing the commodity flow volumes on the edges in M uniquely determines all commodity flows on all edges.*

A. Turnstile Placement for a Single Commodity

The special case of the TP Problem with a single commodity (one multipath flow) is called the ***Single Commodity TP (SC-TP)*** Problem. Given an undirected graph, $G = (V, E)$, a *cycle transversal* of the graph is a set of edges, $S \subset E$, such that every cycle in the graph includes at least one edge from S [18], [19]. Establishing the equivalence between the SC-TP and the cycle transversal problem requires a slight modification to the SC-TP input. We add a back-edge from the destination to the source vertex to complete a source-destination cycle. This modified graph is represented as $G = (V, E \cup \{(t_k, s_k)\}_k, c)$, where (t_k, s_k) ranges over all destination-source pairs in V. With the back-edge in place, we insist on conservation-of-flow at each each vertex in G, as the back-edge carries the total flow from s_k to t_k, and back to s_k.

Lemma 1. *Given an instance to the SC-TP Problem, $G = (V, E \cup \{(t_k, s_k)\}_k, c)$ and a single flow, the set $T \subset E$ is a feasible solution to the SC-TP instance if and only if T is a cycle transversal for G with undirected edges.*

Proof. Suppose that $T \subset E$ is a feasible solution to an SC-TP Problem instance. Then, the traffic values for each edge in E can be determined from the turnstiles in T, but the cost of T need not be optimal. Consider, for the sake of a contradiction, a cycle in the undirected version of G that did not have an edge in T. If such a cycle exists, then an arbitrary amount of traffic r could be added to each edge e of the cycle, where the flow on e is increased by r if the edge points in the direction of the cycle and decreased by r otherwise. Hence, edges can host unrestricted traffic values, and contradicts T being a solution to the SC-TP Problem. Thus, T must provide a cycle transversal of the undirected version of G.

Suppose that $S \subset E$ is a cycle transversal for the undirected version of G. For the sake of a contradiction, further suppose that S is not a valid solution to the SC-TP Problem. This means that there exists some edge, (u, v), whose flow value cannot be determined from knowing the flows on the edges in S. Since conservation of flow holds at each vertex, there must be at least one other edge incident to v whose flow is also not determined. Without loss of generality, say this is the edge from v to w. Again, we can find an edge starting at w and going to some vertex other than v on which the flow is also undetermined. Note that this edge is necessarily not in S. Since G is finite, continuing to follow a path of undetermined edges in this manner must ultimately create a cycle without any edges in S. Existence of this cycle contradicts the assumption that S provides a cycle transversal. Thus, S must be a valid solution to the SC-TP Problem. $\square$

As a consequence to this lemma, a minimum cost cycle transversal solution equates to a minimum cost SC-TP solution, and vice versa. The equivalence of the SC-TP and cycle transversal problems provides a solution to SC-TP instances, as the minimum cycle transversal problem can be solved by determining a maximum cost spanning tree (easily computed using a standard minimum-weight spanning tree algorithm and negating all edge costs).

Lemma 2. *Given an undirected graph with edge costs, $G = (V, E, c)$, and a spanning tree, $T \subset E$, of maximum cost, the remaining edges, $E \setminus T$, form a minimum cost cycle transversal of G.*

Proof. Suppose that T is a maximum cost spanning tree of G. Since T is a tree, any cycle in G must have at least one edge in $E \setminus T$; otherwise, the entire cycle would be contained in T. Therefore, the set $E \setminus T$ forms a cycle transversal of G.

Since T is a spanning tree of maximum cost, the remaining edges in $E \setminus T$ form a minimum cost spanning tree complement, and therefore, a minimum cost cycle transversal of G. $\square$

We note here that placing turnstiles on the artificially inserted back-edges is undesirable, as those edges are not physically present in the network. To avoid this, we set the cost of turnstile placement on these edges to be much greater than the cost of any 'real' edge in the SC-TP instance. By doing so, the back-edges will be included in the maximum cost spanning tree, thereby ensuring they will not be selected for turnstile placement.

B. Turnstile Problem Complexity

Lemma 1 and Lemma 2 show how the SC-TP Problem can be solved optimally in polynomial time. However, The general TP Problem is inapproximable within a bound of $w \ln k$ for some $w > 0$, where k is the number of commodities in the TP instance. This result is obtained via a reduction from the SET-COVER problem and details are included in the Appendix as part of Lemma 5. Due to the complexity of the TP Problem, in Section IV we introduce an approximate solution in lieu of pursuing optimal approaches.

IV. Algorithms

In this section, we provide three algorithms for the TP Problem. The first is an approximation algorithm that greedily selects turnstile locations to minimize total solution cost. The second algorithm leverages the relationship between the SC-TP Problem and the complement of maximum spanning trees detailed in Lemma 1 and Lemma 2. The third is a method that enables using existing single path solutions for a multipath TP instance. In Section V, performance of these solutions are demonstrated with a constant valued turnstile placement cost function, resulting in the objective of minimizing the number of turnstiles placed. However, the algorithms detailed in this section are able to handle any generalized turnstile placement cost function.

A. Approximation Algorithm

In this section, we introduce an approximation algorithm, TP-Approx, for the TP Problem with a general turnstile placement cost function, which achieves an approximation ratio of $(\ln m + 1)(\ln k + 1)$, where k is the number of commodities and m is the number of edges in the network. (Theorem 1 below actually proves a slightly tighter bound.)

Each edge in a TP instance hosts some number of flows. A turnstile placed on an edge is said to *help* a commodity if that turnstile lies on at least one cycle in that commodity not already containing a turnstile. In this way, a proposed turnstile location only helps a commodity if that turnstile will reduce the number of additional turnstiles required for that commodity.

TP-Approx makes iterative turnstile placements, where the turnstile placed in each iteration is on the edge that minimizes the cost per commodity helped. This process continues until all cycles in commodity flows are monitored. The turnstile cost function determines the cost of a candidate turnstile. The number of commodities helped by an edge (u, v) is calculated by checking, for each commodity, if any undirected path exists from u to v besides the direct edge (u, v) in each commodity's subgraph. If a such a path exists, then (u, v) belongs to an uncovered cycle, and thus placing a turnstile on (u, v) helps that commodity. Once edge e is determined to be the best turnstile location, it is added to the set of selected turnstiles τ, and removed from all commodity graphs. This ensures that cycles discovered in subsequent iterations are not covered by any turnstile selected up to that point. An outline of TP-Approx is presented in Algorithm 1 and its running time is analyzed in Lemma 3.

Algorithm 1 TP-Approx

```
τ = ∅
c(e) = cost of placing turnstile on edge e
helps(e) = set of commodities helped by a turnstile on e
while ∃e ∈ E : helps(e) ≠ ∅ do
    Determine e′ = arg min_e c(e)/|helps(e)|
    τ = τ ∪ {e′}
    Remove e′ from G
    Update helps(e) for remaining edges e ∈ G
end while
return τ
```

Lemma 3. *TP-Approx returns a set of turnstiles that resolves all traffic flows on each edge in the network and has a running time of $O(km^3)$, where k is the number of commodities and m is the number of edges in the network.*

Proof. Lemma 1 shows that a cycle transversal provides a valid turnstile solution for a single commodity. Since the algorithm runs until all intra-commodity cycles have at least one turnstile on them, all traffic for each commodity is monitored, and thus the turnstiles selected provide a valid solution for the entire network.

In each iteration of the algorithm, the number of commodities helped must be calculated for each edge. This amounts to, for each edge, searching for a cycle in each commodity's subgraph, which can be done with depth first search in a total of $O(km^2)$. The number of iterations needed is maximized when the algorithm places a turnstile on each edge. Therefore, the running time of Algorithm 1 is in $O(km^3)$. □

Theorem 1. *TP-Approx achieves a $(\ln R_{\max} + 1)(\ln k + 1)$-approximation ratio, where k is the number of commodities and $R_{\max}$ is the maximum number of turnstiles needed by any single commodity, considered in isolation as an SC-TP instance. $R_{\max} = \max_i(m_i - n_i + 1)$ where n_i is the number of vertices and m_i is the number of edges in the subgraph hosting commodity i.*

Proof. Let R_i^t be the number of remaining turnstiles needed for commodity K_i, just after TP-Approx places the t-th turnstile, T_t. Note that for each commodity K_i, we have $R_i^0 = m_i - n_i + 1$ since the spanning tree has exactly $n_i - 1$ edges and the complement of the spanning tree are the edges that break all cycles, as detailed in Lemma 2. We say that commodity K_i is *finished* when R_i^t reaches zero. We define the function *helps* on a turnstile T at iteration t as,

$$\text{helps}_t(T) = \{K_i : R_i^t = R_i^{t-1} - 1, \text{ if } T \text{ is chosen at } t\} \quad (1)$$

Let OPT $= \{T_1^*, \ldots, T_p^*\}$ be an optimal placement of turnstiles. Suppose the TP-Approx algorithm chooses turnstiles, ALG $= \{T_1, \ldots, T_a\}$, in this order. Consider $K_i \in \text{helps}_t(T_t)$. Just prior to TP-Approx selecting T_t, recall that R_i^{t-1} more turnstiles are needed for commodity K_i. Moreover, we can select these turnstiles from OPT (but note that this subset of turnstiles need not be unique). Let OPT(t, i) be one such subset of R_i^{t-1} turnstiles from OPT.

Note that the per-commodity cost of T_t is at most the per-commodity cost of the aforementioned OPT turnstiles (since TP-Approx makes a greedy choice). Observe that

$$\frac{c(T_t)}{|\text{helps}_t(T_t)|} \leq \frac{1}{R_i^{t-1}} \sum_{T^* \in \text{OPT}(t,i)} \frac{c(T^*)}{|\text{helps}_t(T^*)|}.$$

We have

$$\begin{aligned}
c(\text{ALG}) &= \sum_{t=1}^{a} c(T_t) \\
&= \sum_{t=1}^{a} \sum_{K_i \in \text{helps}_t(T_t)} \frac{c(T_t)}{|\text{helps}_t(T_t)|} \\
&\leq \sum_{t=1}^{a} \sum_{K_i \in \text{helps}_t(T_t)} \frac{1}{R_i^{t-1}} \sum_{T^* \in \text{OPT}(t,i)} \frac{c(T^*)}{|\text{helps}_t(T^*)|} \\
&= \sum_{T^* \in \text{OPT}} c(T^*) \sum_{t=1}^{a} \sum_{\substack{K_i \in \text{helps}_t(T_t) \\ T^* \in \text{OPT}(t,i)}} \frac{1}{R_i^{t-1} |\text{helps}_t(T^*)|} \quad (2)
\end{aligned}$$

Notice that $R_i^t = R_i^{t-1} - 1$ for each commodities helped by T_t (and $R_i^t = R_i^{t-1}$ otherwise). If $R_i^{t-1} > R_i^t = 0$ and $K_i \in \text{helps}_1(T^*)$, then $|\text{helps}_t(T^*)| < |\text{helps}_{t-1}(T^*)|$. Order the commodities in $\text{helps}_1(T^*)$ by increasing R_i^0 values; say, the list is $K_{1^*}, \ldots K_{k_0^*}$. The above sum over t is maximized if TP-Approx helps the commodities in this order (so that the first commodity, K_{1^*}, is finished before any others are helped, etc.). In this case, we have:

$$\begin{aligned}
&\sum_{t=1}^{a} \sum_{\substack{K_i \in \text{helps}_t(T_t) \\ T^* \in \text{OPT}(t,i)}} \frac{1}{R_i^{t-1} |\text{helps}_t(T^*)|} \\
&= \sum_{k=1}^{k_0} \frac{1}{k_0 - k + 1} \sum_{h=R^0_{k^*}}^{1} \frac{1}{h} \\
&\leq \sum_{k=1}^{k_0} \frac{1}{k_0 - k + 1} (\ln R^0_{k^*} + 1) \\
&\leq \sum_{k=1}^{k_0} \frac{1}{k_0 - k + 1} (\ln R_{\max} + 1) \\
&= (\ln R_{\max} + 1) \sum_{k=1}^{k_0} \frac{1}{k_0 - k + 1} \\
&\leq (\ln R_{\max} + 1)(\ln k + 1), \qquad (3)
\end{aligned}$$

where $R_{\max} = \max_i R_i^0$. Substituting (3) into (2) yields

$$\begin{aligned}
c(\text{ALG}) &\leq \sum_{T^* \in \text{OPT}} c(T^*)(\ln R_{\max} + 1)(\ln k + 1) \\
&= (\ln R_{\max} + 1)(\ln k + 1) c(\text{OPT}),
\end{aligned}$$

as was to be shown. □

Corollary 1. *Since $R_{\max} \leq m$, the TP-Approx algorithm trivially achieves an $(\ln m + 1)(\ln k + 1)$ approximation ratio.*

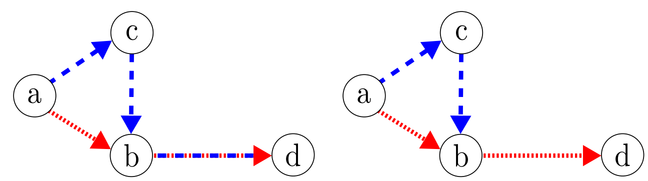
B. Maximum Spanning Tree Approach

In this section, we introduce another approach to solve the TP problem based on the relationship between the turnstile problem and the cycle transversal problem detailed in Section III-A. Lemma 2 showed that placing turnstiles on the complement of a maximum spanning tree (MaxST) for a single commodity instance, SC-TP, provides an optimal turnstile placement for that instance.

Given an instance of the TP problem, the weight of each edge is set to be its cost, $w(e) = c(e)$. The commodities are considered in an arbitrary order and for each commodity k, a MaxST is found for the subgraph containing k. The edges not included in the MaxST are added to the set of turnstile locations, τ, so ensuring that commodity k is monitored. After an edge is added to τ its weight is set to 0 so that it will be reused by subsequent commodities, if possible. TP-MaxST is detailed in Algorithm 2.

Algorithm 2 TP-MaxST

$\tau = \emptyset$
for $e \in E$ **do**
  $w(e) = c(e)$
end for
for commodities k in TP instance **do**
  Find a MaxST T_k that spans E_k
  $\tau = \tau \cup (E_k \setminus T_k)$
  for $e \in E_k \setminus T_k$ **do**
    $w(e) = 0$
  end for
end for
return τ



(a) $s - t$ path decomposition (b) Pathlet decomposition

Fig. 2. The multipath flow on all edges shown in 2(a) from a to d requires at least two turnstiles. However, if this multipath flow were to be decomposed into $s - t$ paths, blue and red, a turnstile placed on edge (b, d) would cover the two single path flows, but not the original multipath flow with undirected cycle (a, b, c). The pathlet decomposition shown in 2(b) avoids this issue.

C. Multipath to Single Path Translation

In this section, we detail a process for adapting existing multicommodity, single path, flow monitoring approaches to multipath TP instances. One naïve approach would be, given an instance with a multipath commodity, decompose its flow into a collection of single path *s*-*t* flows. Aggregate solutions for the single path flows could then be used as a solution for the multipath flow. This approach can lead to situations where all cycles in the single path flows are monitored, but where cycles remain in the underlying multipath flow, and is illustrated in Figure 2(a).

To successfully decompose a multipath TP instance, we segment each multipath flow into a set of sub-paths (pathlets), such that a turnstile anywhere on each pathlet will provide a valid turnstile placement for the original multipath flow. These pathlets can then be treated as their own single path flows, and the same monitors placed by existing multicommodity single path monitoring approaches for the set of pathlets will suffice to monitor the original TP instance.

The translation from multipath TP instances into single path commodities involves decomposing each multipath flow into a set of pathlets. For each multipath flow from the TP instance, the decomposition begins by marking the source and sink vertices in the flow. Unmarked pathlets are then iteratively added to the set. An *unmarked* pathlet is a simple path constructed between marked vertices such that only unmarked edges and vertices are traversed. One way to generate this is by using breadth first search from some marked vertex over unmarked edges until the first marked vertex is reached. The pathlet is added to the set of pathlets, and each vertex and edge in the pathlet is marked. This process is continued until all edges in the flow are marked. Once a flow is decomposed into pathlets, all marks are disregarded and the next flow is decomposed. The set of pathlets resulting from the decomposition of all multipath flows in the TP instance is returned as a set of single path flows, ready for existing single path sensor placement algorithms. This translation is detailed in Algorithm 3 and shown in Figure 2(b).

Algorithm 3 Multipath Flows to Single Path Flows

```
P = set of pathlets = ∅
for commodities (s_k, t_k, E_k) in TP instance do
    Mark s_k and t_k
    while unmarked edges remain in E_k do
        Find unmarked path between marked vertices
        Add path to P and mark all its vertices and edges
    end while
end for
return Set of pathlets, P
```

Lemma 4. *A turnstile placed anywhere on each pathlet in the set of pathlets returned by Algorithm 3 will form a valid turnstile placement for the multipath TP input.*

Proof. By Lemma 1, a placement of turnstiles is valid if each cycle in the graph has a turnstile on at least one edge. We aim to show that each cycle in the graph must have one pathlet from $\mathcal{P}$ totally contained within that cycle. If that were the case, then any turnstile placement along that pathlet would sufficiently cover that cycle, thereby providing a valid placement as each cycle has such a pathlet.

Consider a cycle in the graph. Because each pathlet in $\mathcal{P}$ is simple, this cycle must consist of multiple pathlets. Select some pathlet on the cycle that also contains edges off the cycle (if no such pathlet exists, then the pathlets are fully contained in the cycle and any turnstile placement will suffice). The vertices of that pathlet that lie on the cycle cannot be intermediate vertices of any other pathlet, since pathlet building ends as soon as a marked vertex is encountered. This means that those vertices must be endpoints for one or more pathlets. Thus, the remaining pathlets composing the rest of the cycle cannot all include edges off the cycle, as otherwise, some pathlet must include a vertex from another pathlet as an intermediate vertex. □

V. Evaluation

In this section, we detail the process, and present the results, of evaluating the algorithms presented in Section IV. We apply the multipath to single path translation described in Section IV-C and then employ a greedy Set-Cover motivated by the solution introduced in [4] to compare our solutions to existing single path solutions. We label this hybrid algorithm the Greedy Single Path (GSP) algorithm. To minimize the total number of turnstiles placed, a constant valued turnstile placement cost function is employed.

To conduct our experiments, we use five network topologies from the Internet Topology Zoo [11]: GEANT, a European backbone network, UUNET, a United States backbone network, DFN, a German backbone network, Viatel, a European backbone network, and Tinet, a global backbone network. We choose these topologies as representative examples of large real-world networks that support multipath flows.

Network traffic is generated on a per-commodity, per-instance, basis. We sample from a set of vertices in each topology to serve as sources and destinations. The Fast Network Simulation Setup (FNSS) is used to generate flow volumes [16]. Two methods are used for generating commodity edge sets. The first process uses a standard approach for determining multipath routing by having flows, volumes, and edge sets generated by solving an instance of the multicommodity flow problem with the linear program solver CPLEX. In the second method, edge sets are generated for flows by choosing edges in the k-shortest paths from source to destination where k varied between one, three, and five. This method was used to reduce instance variability favoring one algorithm over another by explicitly controlling the number of paths accessible to each flow. Both methods result in a network with a defined topology and a set of commodities with multipath routes from their sources to destinations. The algorithms are tested on topologies containing four commodities and 100 commodities. The instances containing 100 commodities are the result of merging 25 instances of four commodities on the same topology. The volumes of these commodities make these instances representative of a topology with a 100 commodities that has the capacity to support four of them transmitting at full volume at any given time. We look at the number of turnstiles placed using different topologies and different algorithms, noting here that the standard deviation remained less than three across most experiments.

A. Number of Turnstiles Placed

The first simulation aims to determine the number of turnstiles placed by the algorithms on instances small enough for optimal turnstile placements to be computed. Figure 3 shows the results of these simulations. Four commodities are generated, per instance, using the technique described above on the five Internet Topology Zoo topologies using the CPLEX LP-Solver to generate the commodity edge sets. The graph shows the average number of turnstiles needed to monitor each instance over 50 simulations, on each topology. OPT and TP-Approx average around four turnstiles for each topology with TP-Approx placing, on average, fewer than 1% more turnstiles than OPT. The number of turnstiles chosen by the optimal solution to be necessary for the topologies averages at 3.91. GSP and TP-MaxST place about 2% and 20% more turnstiles than OPT respectively. We suspect that the Viatel topology requires fewer turnstiles than the other topologies evaluated because there are fewer distinct paths through the Viatel network than the others.

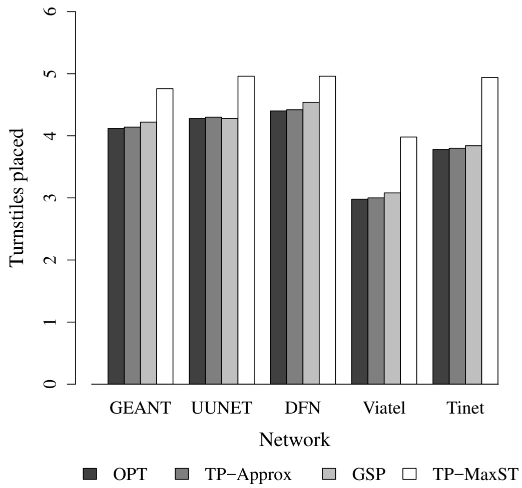


Fig. 3. The average number of turnstiles placed by each algorithm over 50 instances, on each topology, with four commodities per instance.

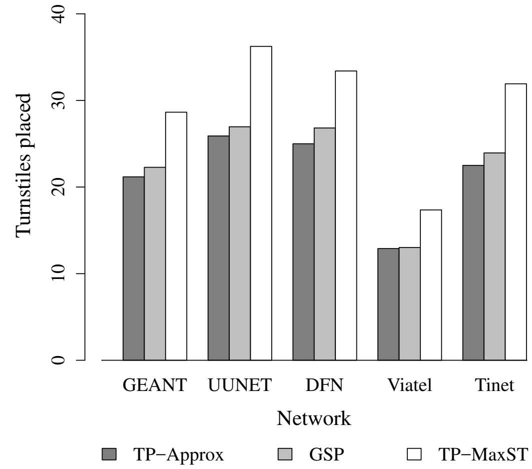


Fig. 4. The average number of turnstiles placed by each algorithm over 50 instances with 100 commodities each, on each topology.

The second simulation aims to determine the number of turnstiles placed by the algorithms on the five Internet Topology Zoo topologies, with larger traffic patterns which contain too many flows to solve optimally. To generate an instance with 100 commodities, the instances are generated, four commodities at a time using the CPLEX LP-Solver to generate commodity edge sets, and merged into a single graph. Figure 4 shows the average number of turnstiles required by each algorithm over 50 iterations on each topology. TP-Approx places, on average across the five topologies, 21.5 turnstiles in each simulation, whereas GSP places 22.6, and TP-MaxST places 29.512. As a result, TP-Approx places about 5% fewer turnstiles than GSP and about 27% fewer turnstiles than TP-MaxST. Increasing the number of commodities results in more of the underlying network being utilized, leading to a larger number of turnstiles being placed by all solutions. The relative number of turnstiles required between the various algorithms also increases due to TP-Approx and, to a lesser extent, GSP making better use of inter-commodity collaborative placements.

The third simulation aims to determine the number of turnstiles placed by the algorithms on the five Internet Topology Zoo topologies, with commodity edge sets determined using the k-shortest path algorithm on each source destination pairing. Like the previous simulation, 100 commodities are generated on each instance. To determine the commodity edge sets, we use k-shortest paths with k equal to one, three, and five, to explore the effectiveness of the different algorithms as the commodities contain more paths. Figure 5 shows the average number of turnstiles required by each algorithm over 50 iterations on the Tinet topology. TP-Approx places fewer turnstiles in every instance, even when k is set to one. This result confirms that TP-Approx is a viable solution for the single path version of the problem, as well as the multipath formulation. Increasing the value of k highlights the reduced number of turnstiles placed by TP-Approx compared to the other solutions. TP-Approx places 34.3 turnstiles on average while GSP and TP-MaxST place 39.78 and 46.76 respectively when k is set to three. In this case, the GSP solution places about 16% more turnstiles than TP-Approx. TP-Approx places 38.84 turnstiles on average and GSP and TP-MaxST place 47.56 and 52.12 when k is set to five. This results in GSP placing about 22% more turnstiles than TP-Approx.

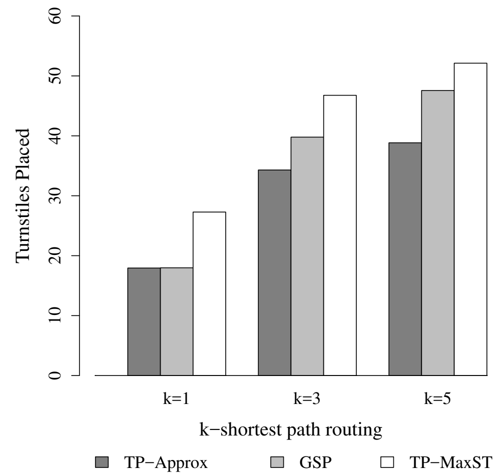


Fig. 5. The average number of turnstiles placed by each algorithm over 50 instances with 100 commodities on the Tinet topology.

B. Placement Effectiveness

In certain scenarios (e.g. large topologies, restricted monitoring budget, loose monitoring requirements), placing fewer turnstiles than necessary makes sense, accepting that the entire network may not be covered. In this scenario, it is advantageous to employ an algorithm that provides the highest rate of return, in terms of network coverage, per turnstile placed. The fourth simulation explores this scenario by investigating the rate at which the algorithms cover the network. Specifically, the Monitoring Benefit for placing a turnstile on an edge is

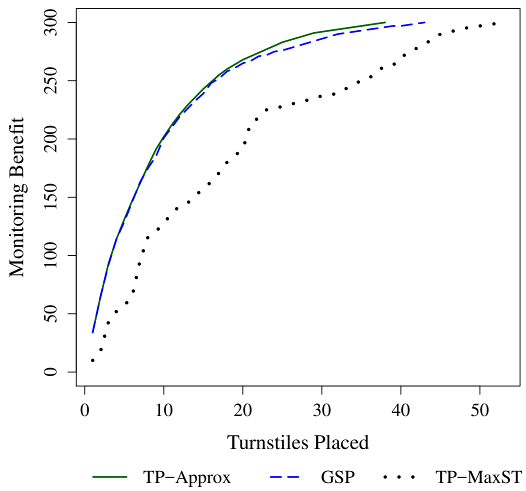


Fig. 6. The monitoring benefit provided by the placement of each sequential turnstile for each algorithm. Shown, is the Tinet topology with 100 commodities with commodity edge sets generated using 3-shortest paths between each s,t pairing. Monitoring benefit is defined as the number of commodities helped (defined in equation 1) as each turnstile is placed.

the number of flows that have a cycle covered as a result of the placement.

Figure 6 shows the total number of turnstiles placed, in the order that they are placed, versus the number of commodity-required turnstiles covered. Note that the number of commodity-required turnstiles is just the sum of the number of turnstiles each individual commodity requires, $\sum_{i=0}^{k} m_i - n_i + 1$, where k is the number of commodities as detailed in the beginning of the proof to Theorem 1. Using the Tinet topology, with 100 commodities, we obtain the results shown in Figure 6. The commodity edge sets are generated using k-shortest paths with k set to three.

TP-Approx achieves 95% of the total number of commodity-required turnstiles after placing 26 turnstiles, whereas GSP requires 30 and TP-MaxST requires 44 to cover 95% of the commodity-required turnstiles. In this scenario, TP-Approx requires almost 14% fewer turnstiles than GSP and about 40% fewer turnstiles than TP-MaxST for cover 95% of the commodity-required turnstiles. TP-Approx is not only able to place a fewer number of total turnstiles for full coverage, but is capable of placing turnstiles that benefit a large number of flows from the beginning. Figure 6 shows that this trend holds steady for any percent of the total number of commodity-required turnstiles. Interestingly, initial turnstiles placed by GSP are close to benefiting the same as TP-Approx, but the effectiveness of the placements diverge slightly around 90% coverage.

VI. Conclusions

The turnstile placement techniques introduced in this paper provide a new and efficient way to monitor multipath traffic flows, in settings such as software defined networks, to improve load balancing and network efficiency. Minimizing the number of turnstiles reduces the cost of monitoring; we have developed several efficient algorithms that minimize the number of turnstiles needed to monitor traffic flows. In particular, the TP-Approx algorithm achieves near optimal performance in simulations on real Internet topologies. Several interesting open questions regarding flow monitoring with turnstiles remain: First, besides the cost of turnstile placement, one may consider be additional costs related to the volume of the flows monitored. Although flow volumes are initially unknown, once turnstiles are placed and edge-flow volumes calculated, the turnstile costs could be updated to reflect current flow volumes. Second, the network is dynamic and new flows will be added while old flows are removed. If the existing turnstiles are insufficient (or redundant), then a new turnstile solution should be found, that accrues the least changeover cost. Finally, turnstile monitoring may be of use for flow monitoring in other types of networks, e.g., transportation networks, social networks, etc.

Appendix

Lemma 5. *Consider the TP Problem with k commodities. There exists a constant, $w > 0$, such that no polynomial time $w \ln k$-approximation algorithm exists, unless $P = NP$.*

Proof. This result is via an approximation-preserving reduction from the SET-COVER Problem: Given a collection S of subsets of a finite sized universe, U, find a minimum sized $S' \subset S$ such that for all $u \in U$, u is contained in some set in S'. Let $S = \{S_1, ..., S_n\}$, where $S_i \subset U$ for all i, be an instance of SET-COVER. Reduce this instance to a TP instance as follows:

Create a graph component with a line topology having $n+1$ vertices and n edges, where vertex j shares a directed edge to vertex $j+1$. These n edges will each be sequentially associated with a unique member of S (i.e., the edge from j to $j+1$ corresponds to the set S_j). Let the turnstile placement cost function be unit valued for all edges in the graph, thereby making the objective to minimize the total number of turnstiles placed.

Each element of the universe, U, will be represented by a single-path commodity flow. Do this by first creating, for each element in U, two new vertices to serve as that commodity's source and destination. For each commodity, i, consider the ordered list of sets, $S^i \subset S$ that its corresponding element of the universe is a member of. Create a source-to-destination path for each commodity by beginning at that commodity's source and making a directed edge to the vertex corresponding to the source vertex of its first element of S^i. So, if S_j is the first element of S^i, an edge will be made from commodity i's source to vertex j. Now, take the existing edge to that set's destination vertex and create a new directed edge to the source vertex of the next element of S^i, unless the destination of the current element and source of the next element is the same. Continue this process until there are no remaining elements in S^i, and then terminate at that commodity's destination vertex. See Figure 7 for an example reduction.

A solution to the SET-COVER instance provides a solution to the TP instance of the same size since a subset of S that

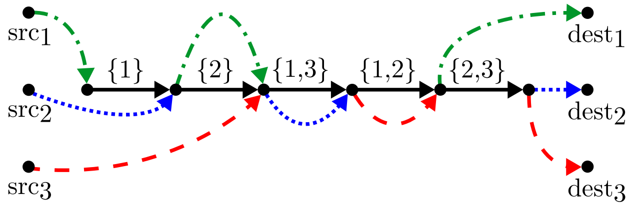


Fig. 7. Sample reduction of SET-COVER instance to a TP instance. The SET-COVER instance shown has universe, $U = \{1, 2, 3\}$, and sets, $S = \{\{1\}, \{2\}, \{1, 3\}, \{1, 2\}, \{2, 3\}\}$.

contains all elements of the universe will correspond to a set of edges that are cumulatively traversed by each commodity. Since each commodity is a single path, requiring a single turnstile to monitor its traffic, these edges provide a valid solution to the TP instance.

Given any solution to the TP instance, this solution is equivalent to one with turnstiles placed only on edges generated by elements of S, as opposed to the commodity specific edges. This solution corresponds to an equally sized solution to the SET-COVER instance, as a subset of S is selected that must cover all elements of the universe, since the turnstiles cover all commodities.

Since the turnstiles placed and the set cover elements are in one-to-one correspondence, the number of turnstiles placed for the TP instance is the same as the size of the set cover, thereby making the costs equal and the reduction approximation-preserving. Since there exists a constant, $w > 0$, such that SET-COVER cannot be approximated within a factor of $w \ln n$, where n is the size of the universe U, unless $P = NP$ [15]. Therefore, TP cannot be approximated within a factor of $w \ln k$, where k is the number of commodities, unless $P = NP$. □

ACKNOWLEDGEMENT

We thank Schloss Dagstuhl, as this line of research stems from problems posed during a working group at Dagstuhl Seminar 16022 [9]. We also thank National Science Foundation for supporting this work via grants CNS-1555591, CNS-1527097, CCF-1618605, and ABI-0435060.

REFERENCES

[1] KEMP SDN Adaptive, powered by the HP VAN SDN Controller. https://kemptechnologies.com/sdn-adaptive-load-balancing/, Accessed Jul 2016.

[2] G. R. Cantieni, G. Iannaccone, C. Barakat, C. Diot, and P. Thiran. Reformulating the monitor placement problem: Optimal network-wide sampling. In *ACM CoNEXT*, Dec 2006.

[3] C.-W. Chang, G. Huang, B. Lin, and C.-N. Chuah. LEISURE: Load-balanced network-wide traffic measurement and monitor placement. *Parallel and Distributed Systems*, 26(4):1059–1070, Apr 2015.

[4] C. Chaudet, E. Fleury, I. G. Lassous, H. Rivano, and M.-E. Voge. Optimal positioning of active and passive monitoring devices. In *ACM conference on Emerging network experiment and technology*, Oct 2005.

[5] B. Claise. Cisco systems netflow services export version 9. 2004.

[6] D. Gupta, P. Mohapatra, and C.-N. Chuah. Efficient monitoring in wireless mesh networks: Overheads and accuracy trade-offs. In *IEEE Mobile Ad Hoc and Sensor Systems (MASS)*, Sep 2008.

[7] G. Huang, C.-W. Chang, C.-N. Chuah, and B. Lin. Measurement-aware monitor placement and routing: a joint optimization approach for network-wide measurements. *IEEE Transactions on Network and Service Management*, 9(1):48–59, Mar 2012.

[8] G. Iannaccone, S. Bhattacharrya, N. Taft, and C. Diot. Always-on monitoring of IP backbones: requirements and design challenges. Technical report, Sprint ATL, July 2013.

[9] G. F. Italiano, M. van Kreveld, B. Speckmann, and G. Theraulaz. Geometric and Graph-based Approaches to Collective Motion (Dagstuhl Seminar 16022). *Dagstuhl Reports*, 6(1):55–68, Jan 2016.

[10] A. W. Jackson, W. Milliken, C. A. Santivánez, M. Condell, and W. T. Strayer. A topological analysis of monitor placement. In *IEEE Network Computing and Applications*, Jul 2007.

[11] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *IEEE Journal on Selected Areas in Communications*, 29(9):1765–1775, Oct 2011.

[12] R. G. Little. F5 SDN: Why ADCs are relevant in a programmable world. http://searchsdn.techtarget.com/news/2240226557/F5-SDN-Why-ADCs-are-relevant-in-a-programmable-world, Aug. 2014.

[13] K. Park and H. Lee. On the effectiveness of route-based packet filtering for distributed dos attack prevention in power-law internets. In *SIGCOMM CCR*, volume 31, pages 15–26, Aug 2001.

[14] Y. Qin, D. G. Feng, K. Chen, and Y. F. Lian. Research on monitor position in network situation assessment. In *IEEE Industrial Control and Electronics Engineering (ICICEE)*, Aug 2012.

[15] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. In *ACM Symposium on Theory of Computing (STOC)*, May 1997.

[16] L. Saino, C. Cocora, and G. Pavlou. A toolchain for simplifying network simulation setup. In *International Conference on Simulation Tools and Techniques (ICST), SIMUTOOLS*, Mar 2013.

[17] K. Suh, Y. Guo, J. Kurose, and D. Towsley. Locating network monitors: Complexity, heuristics, and coverage. *Computer Communications*, 29(10):1564–1577, Jun 2006.

[18] G. Xia and Y. Zhang. On the small cycle transversal of planar graphs. *Theoretical Computer Science*, 412(29):3501 – 3509, Jul 2011.

[19] G. Xia and Y. Zhang. Kernelization for cycle transversal problems. *Discrete Applied Mathematics*, 160(7):1224–1231, May 2012.

[20] H. Zang and A. Nucci. Traffic monitor deployment in IP networks. *Computer Networks*, 53(14):2491–2501, Sep 2009.

[21] Y. Zeng, D. Wang, W. Liu, and A. Xiong. An approximation algorithm for weak vertex cover problem in IP network traffic measurement. In *IEEE Network Infrastructure and Digital Content (IC-NIDC)*, Nov 2009.

[22] Y. Zhang. An adaptive flow counting method for anomaly detection in SDN. In *ACM CoNEXT*, Dec 2013.