Coordinated Sampling sans Origin-Destination Identifiers: Algorithms and Analysis

Vyas Sekar*, Anupam Gupta*, Michael K. Reiter†, Hui Zhang*
*Carnegie Mellon University, †UNC Chapel Hill

***Abstract*—Flow monitoring is used for a wide range of network management applications. Many such applications require that the monitoring infrastructure provide high flow coverage and support fine-grained network-wide objectives. Coordinated Sampling (cSamp) is a recent proposal that improves the monitoring capabilities of ISPs to address these demands.**

In this paper, we address a key deployment impediment for cSamp-like solutions–the need for routers to determine the Origin-Destination (OD) pair of each packet. In practice, however, this information is not available without expensive changes. We present a new framework called cSamp-T, in which each router uses only local information, instead of the OD-pair identifiers. Leveraging results from the theory of maximizing submodular set functions, cSamp-T provides near-ideal performance in maximizing the total flow coverage in the network. Further, with a small amount of targeted upgrades to a few routers, cSamp-T nearly optimally maximizes the minimum fractional coverage across all OD-pairs. We demonstrate these results on a range of real topologies.

I. Introduction

Flow monitoring supports critical network management tasks in ISPs such as anomaly detection [1], identifying unwanted application traffic [2], understanding traffic structure [3], botnet analysis [4], and forensic analysis [5], in addition to traditional traffic engineering and accounting [6]. These applications require that monitoring infrastructures provide high flow coverage (number of flows logged) and the ability to achieve network-wide flow measurement goals.

Our previous work presented a system called cSamp [7] to address these requirements. cSamp delivers the optimal possible flow coverage and can achieve fine-grained network-wide flow coverage goals. It avoids redundant monitoring to efficiently use the available capacity and naturally load balances responsibilities to avoid monitoring hotspots.

The key to these benefits is that cSamp coordinates the sampling actions across routers. To achieve this coordination, cSamp assumes that each router can determine the Origin-Destination (OD) pair (i.e., the ingress and egress router) for each packet it sees. However, due to practical issues such as multi-exit peers and prefix-aggregation, interior routers cannot identify the OD-pair given just the source and destination IP addresses. Thus, cSamp requires: (i) upgrades to border routers to compute the OD-pair identifiers and (ii) modifications to packet headers to carry OD-pair identifiers. These are expensive changes that present significant deployment barriers for ISPs. Thus, while cSamp has the potential to substantially improve flow monitoring, it lacks a practical deployment path.

In this paper, we address the challenge of providing flow monitoring capabilities comparable to cSamp, without relying on OD-pair identifiers. We present cSamp-T,[1] an approach in which each router makes sampling decisions using only *locally available* information. These local decisions are based on packet headers and local routing tables rather than the global OD-pair identifiers. Since cSamp-T does not rely on OD-pairs, it is more immediately deployable than cSamp.

However, in this new framework, computing optimal sampling strategies to maximize network-wide objectives such as the total flow coverage (i.e., the number of unique flows logged) and the minimum fractional coverage across all OD-pairs is NP-hard [8]. Our key challenge is to develop efficient techniques for computing sampling strategies to (approximately) optimize these objectives.

To maximize the total flow coverage, we use the insight that the objective function is *submodular*. We obtain near-optimal performance by extending results from the theory of optimizing submodular functions subject to budget constraints [9, 10]. Specifically, we implement efficient greedy algorithms with good theoretical approximation guarantees and near-ideal performance in practice. Our evaluations on several real ISP topologies show that cSamp-T achieves more than 85% of the optimal total flow coverage provided by cSamp.

The minimum fractional coverage objective — the minimum across all OD-pairs of the fraction of flows logged per OD-pair — is not submodular. The greedy algorithm performs poorly in theory [11] and practice in this case. We present two strategies to improve the performance: (a) augmenting targeted routers with more resources and (b) incrementally upgrading border routers with the capability to add OD-pair identifiers to packet headers. We show that only a *few* such upgrades are necessary; augmenting the total memory budget by 20% or upgrading 6% of the ingresses suffices to achieve more than 80% of the ideal cSamp performance.

By relaxing the dependence on OD-pair identifiers, cSamp-T makes the benefits of solutions like cSamp immediately available to ISPs. It also provides an incremental deployment path and provides near-ideal performance even with limited deployment. We also believe that the algorithms and heuristics we develop (e.g., extending results from the theory of submodular set maximization, intelligent resource provisioning, hybrid cSamp and cSamp-T deployment) can be more broadly applied to other network management problems (e.g., [12]).

[1] cSamp-T denotes cSamp minus Tags for OD-pairs

978-1-4244-5489-1/10/ $26.00 ©2010 IEEE

II. Background and Motivation

In this section, we provide a brief overview of cSamp and also explain a key challenge that makes it impractical for ISPs to deploy cSamp-like solutions today.

A. Why cSamp?

Flow monitoring is crucial for several network management functions including several anomaly detection and security applications (e.g., [1–5, 13]), and this set of applications continues to grow. Synthesizing arguments from previous work [14–20], we identify four key requirements:

- Provide high flow coverage, i.e., log as many flows as possible, to support security applications which need a fine-grained understanding of "who-talked-to-whom".
- Work efficiently within (heterogeneous) router resource constraints and minimize redundant reports.
- Satisfy network-wide flow monitoring goals where a network operator can specify some subsets of traffic as more important than others or require guaranteed minimum coverage for all ingress-egress pairs.
- Support a broad spectrum of monitoring applications.

Based on the insights from previous work, these lead to three natural design choices in cSamp [7]: (1) avoiding the bias of packet sampling against small flows by using flow sampling [14]; (2) coordinating routers to avoid redundant sampling and use router resources efficiently [19]; and (3) a network-wide optimization framework for assigning monitoring responsibilities to meet the ISP's objectives [18].

B. Overview of cSamp

cSamp assigns sampling responsibilities to routers in a coordinated manner to optimize network-wide flow monitoring goals. Network operators typically express such goals in terms of *Origin-Destination (OD) pairs*, identified by the ingress and egress routers. Thus, the objectives are expressed as functions of the fraction of flows logged (i.e., the coverage) for each OD-pair.

cSamp assigns sampling responsibilities in terms of *hash-ranges per OD-pair per router*. These configurations are called *sampling manifests*. The manifest for a router is a set of entries of the form $\langle OD, [start, end] \rangle$, where $[start, end] \subseteq [0, 1]$ denotes a hash range, and OD is a specific OD-pair. Each router's sampling algorithm is as follows. For each packet, the router determines the OD-pair (assuming that this is feasible). Next, it computes a HASH of the flow 5-tuple $\langle$*srcIP, dstIP, srcport, dstport, proto*$\rangle$, which returns a value in $[0, 1]$, and checks if the hash value lies in the range assigned to it for the OD-pair. If the packet is selected, the router updates the packet and byte counters for this flow.

The key idea is that all routers are configured with the same hash function but are assigned *non-overlapping* hash ranges for each OD-pair. This ensures that the sets of flows sampled by different routers do not overlap. Next, we discuss how the sampling manifests are generated using a network-wide optimization framework.

Optimization Framework: The inputs to the optimization are the flow-level traffic matrix (number of flows per OD-pair), router-level path(s) for each OD-pair, the resource constraints of routers, and the ISP's flow monitoring objective expressed as a function of the fractional flow coverage per OD-pair. Each OD-Pair OD_i $(i = 1, \ldots, M)$ is characterized by its router-level path P_i and the approximate number T_i of distinct IP-level flows on that path in a measurement interval (e.g., five minutes).[2] Each router R_j $(j = 1, \ldots, N)$ is constrained by the available SRAM for keeping per-flow counters [21]; L_j denotes the number of flows R_j can record in a given measurement interval.

d_{ij} denotes the fraction of flows of OD_i that router R_j logs. (If R_j does not lie on path P_i, then the variable d_{ij} will not appear in the formulation.) For $i = 1, \ldots, M$, let C_i denote the fraction of flows on OD_i that is logged.

The specific goal in cSamp [7] has two parts. First, we find the largest possible minimum fractional coverage per OD-pair $\min_i\{C_i\}$ subject to the resource constraints. Next, we use this as θ in Eq 3 in the linear program shown below and maximize the total flow coverage $\sum_i (T_i \times C_i)$. This provides good network-wide visibility by maximizing the minimum fractional coverage and high flow coverage by maximizing the total number of flows logged.

$$\begin{array}{|ll|} \hline \text{Maximize} \sum_i (T_i \times C_i), \text{ subject to} & \\ \forall j, & \sum_{i:R_j \in P_i} (d_{ij} \times T_i) \leq L_j \quad (1) \\ \forall i, & C_i = \sum_{j:R_j \in P_i} d_{ij} \quad (2) \\ \forall i, & \theta \leq C_i \leq 1 \quad (3) \\ \forall i, \ \forall j, & d_{ij} \geq 0 \\ \hline \end{array}$$

The solution $d^* = \{d^*_{ij}\}$ to this two-step procedure is then translated into sampling manifests specifying the flow monitoring responsibility for each router.

C. Assumptions in cSamp

There are three main assumptions: (i) the presence of a centralized optimization module with access to routing and traffic matrices, (ii) routers implement hash-based flow sampling, and (iii) routers can obtain OD-pair information from packet headers.

The first two assumptions are feasible within current operational realities. First, centralization is viable if the configurations are generated reasonably quickly (within 5-10 minutes). Recent trends show that ISPs favor centralized management [22, 23] and that routing and traffic information are already available [24, 25].[3] Second, the hash functions required for flow sampling are simple and amenable to fast hardware implementations [20, 26]. Flow sampling requires

[2] We assume that each OD-pair has a single routing path. It is easy to extend the framework to accommodate multi-path routing or route changes [7].

[3] One possible concern is that ISPs only have packet- or byte-level traffic matrices. cSamp does not need exact flow-level matrices; approximate estimates suffice and the optimization is robust to estimation errors. Further, this is only needed for bootstrapping cSamp's operation. The flow reports generated by cSamp can be used to generate flow-level matrices subsequently.

lookups for each packet and is feasible if the flow counters are in fast SRAM [21, 27].

The assumption that routers can obtain OD-pair identifiers is crucial to cSamp. Specifically, Eq 2 assumes that the hash-ranges assigned to different routers for a given OD-pair are non-overlapping. In fact, this step is critical to model the optimization problem as a linear program (which can be solved efficiently) since it allows us to express the coverage for an OD-pair as the *sum* across routers on its path. If routers cannot obtain OD-pair information, this would no longer hold. As we discuss next, this assumption is not practical for ISPs today.

D. Challenges in OD-pair identification

A router needs to determine the OD-pair (i.e., the ingress and egress routers) for a packet based on the source and destination IP addresses and its routing table. The feasibility of doing this depends on whether the ISP uses IP-based or MPLS-based forwarding. While IP forwarding is destination-based, MPLS can also take into account source information. However, we are unaware of deployments configured in this way [28]. As such, here we restrict our attention to destination-based MPLS forwarding, which we believe to be the norm.

Information to Resolve	Routing/Forwarding	
	IP (dest-based)	MPLS (dest-based)
Ingress	Difficult	Difficult
Egress	With some ambiguity	Possible

TABLE I
FEASIBILITY OF RESOLVING INGRESS AND EGRESS INFORMATION USING PACKET HEADERS AND LOCAL ROUTING TABLES.

Table I summarizes the feasibility of resolving the ingress and egress in these two scenarios. In both cases, resolving the ingress is nearly impossible. For example, for traffic entering from a multi-exit peer (i.e., a neighboring AS to which an ISP is connected at multiple peering points), source IP address and routing information cannot determine the ingress on which the packet entered. With MPLS, the egress can be resolved exactly; with IP the egress can be resolved within some ambiguity. Further, in IP forwarding, ingress and egress resolution may be additionally difficult due to prefix aggregation.

Due to the above challenges, cSamp assumes that ingress routers explicitly add OD-pair identifiers to packet headers. However, this leads to a practical deployment bottleneck. It requires additional processing on ingress routers to resolve and add the egress information and requires modifications to packet headers to carry OD-pair identifiers.

III. Problem Statement

The above challenges in OD-pair identification bring us to the motivating question for this work:
Can we provide the flow coverage benefits of cSamp without requiring OD-pair identifiers?

That is, we want each router to make sampling decisions using only *locally available information* instead of OD-pair identifiers, but still get performance comparable to cSamp. Here, local refers to information that a router can immediately obtain from packet headers and its local routing and forwarding state. We refer to this new approach as cSamp-T.

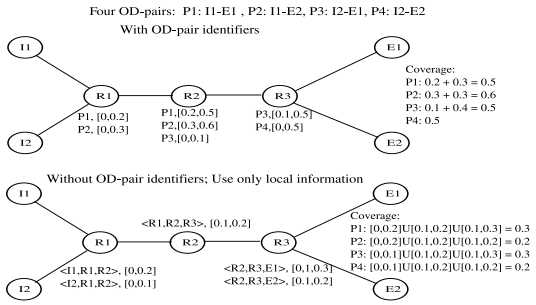


Fig. 1. Example showing sampling decisions made using local information in cSamp-T and contrasting it with cSamp. Each router's cSamp-T sampling decision depends only on the previous and next hop for each packet.

As an example, consider the network in Figure 1 with 2 ingresses and egresses and 4 OD-pairs P1–P4 . The top half shows a cSamp configuration. Each router's responsibilities are hash ranges *per OD-pair* and for each OD-pair the ranges on the routers on its path are non-overlapping.

The bottom half shows a scenario where routers cannot obtain OD-pairs. Suppose each router is assigned a hash range *per router 3-tuple* specified by the previous hop, current router, and the next hop. The router uses this range to decide whether or not to sample the flow/packet. Note that a router can determine the 3-tuple using only local information: the interface the packet arrived on, the destination IP, and its forwarding table. The coverage of an OD-pair is obtained by "stitching" together the coverage provided by each router on the path. That is, the coverage for OD-pair OD_i is the *union*: $\bigcup_{R_j \in P_i} Coverage(R_j, P_i)$, where $Coverage(R_j, P_i)$ is the hash range corresponding to the 3-tuple comprising R_j and its previous/next hops for P_i. For example, the coverage for path P1 which passes through R1, R2, and R3 is $Coverage(R1, P1) \cup Coverage(R2, P1) \cup Coverage(R3, P1) = [0, 0.2] \cup [0.1, 0.2] \cup [0.1, 0.3] = [0, 0.3]$.

The example highlights two differences between cSamp-T and cSamp. First, the sampling responsibilities are specified using local information rather than global OD-pair identifiers. Second, the coverage for each OD-pair is no longer the sum across the routers on the path; it is the union of the ranges assigned to the routers.

Now, how do we assign sampling responsibilities in cSamp-T to maximize network-wide flow coverage objectives while respecting each router's resource constraints? The following sections present a formal framework to address this.

A. Problem Formulation

We retain two assumptions from cSamp: (a) a centralized module for assigning responsibilities with access to routing and traffic matrices and (b) routers implement hash-based flow sampling using SRAM counters and SRAM size constrains the number of flows a router can log. As discussed earlier, both are feasible. Next, we discuss how a centralized module can assign sampling responsibilities without OD-pair identifiers.

We introduce the notion of a *SamplingSpec* to capture the granularity at which a router makes sampling decisions. For

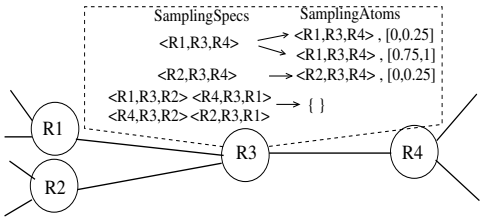


Fig. 2. Example illustrating SamplingSpecs and SamplingAtoms

the current discussion, the SamplingSpecs are *router three-tuples* $\langle R_{j_1}, R_{j_2}, R_{j_3}\rangle$ that appear contiguously on a network path; i.e., R_{j_1} and R_{j_3} are neighbors of R_{j_2}. Let a_k denote a generic SamplingSpec.

$a_k \in P_i$ denotes that the SamplingSpec a_k lies on the path P_i for OD_i.[4] For example, if the path P_i uses routers $\cdots, R_{j_1}, R_{j_2}, R_{j_3}, \cdots$ in that order, then the SamplingSpec $a = \langle R_{j_1}, R_{j_2}, R_{j_3}\rangle \in P_i$. This is a natural extension of the notion that a router R_j lies on a path P_i. We use $t_k = \sum_{i:a_k \in P_i} T_i$ to denote the total traffic that traverses a_k. SamplingSpecs are mapped to routers in a many-to-one fashion; we denote the set of SamplingSpecs mapped to R_j by R_j.specs. That is, R_j can be assigned sampling responsibilities corresponding to $a_k \in R_j$.specs. In the 3-tuple case, if $a_k = \langle R_{j_1}, R_{j_2}, R_{j_3}\rangle$, then $a_k \in R_{j_2}$.specs.

If R_j.specs $\ni a_k$, then R_j can log some of the traffic on paths $P_i \ni a_k$. But what fraction should it log? We formalize this using *SamplingAtom*s. Suppose the traffic traversing a_k is mapped to the unit interval $[0,1]$ by hashing and that the interval $[0,1]$ is divided into $\frac{1}{\delta}$ equal-sized intervals $h_l = [(l-1)\delta, l\delta]$, of length δ. A SamplingAtom is a pair $\langle a_k, h_l\rangle$. If a SamplingAtom, $g_{kl} = \langle a_k, h_l\rangle, a_k \in R_j$.specs, is *assigned*, then router R_j logs flows that traverse a_k such that the hash of the flow falls in h_l. We use $h(g_{kl})$ as a synonym for h_l.

EXAMPLE: Figure 2 illustrates these definitions with an example, where $\delta = 0.25$. $R3$ has three SamplingSpecs in the forward direction (and three similar SamplingSpecs in the reverse direction): $\langle R1, R3, R4\rangle$, $\langle R1, R3, R2\rangle$ and $\langle R2, R3, R4\rangle$. $R3$ is assigned three SamplingAtoms, two for $\langle R1, R3, R4\rangle$, one for $\langle R2, R3, R4\rangle$, and none for $\langle R1, R3, R2\rangle$. Consider paths of the form $\{.., R1, R3, R4, ..\}$. (There may be many such paths.) $R3$ will log all flows along these paths whose hashes fall either in the range $[0, 0.25]$ or $[0.75, 1]$, and flows on paths of the form $\{.., R2, R3, R4, ..\}$ such that the hash of the flow falls in the range $[0, 0.25]$.

Measures of Goodness: Given a set of assigned SamplingAtoms, $\{\widehat{g_{kl}}\}$, we can compute the *fractional coverage* for each OD_i. The coverage due to one particular SamplingSpec $a_k \in P_i$ is $\cup_l\ h(\widehat{g_{kl}}) \subseteq [0,1]$, and hence

$$\text{coverage } C_i = \left|\bigcup_{a_k \in P_i} \bigcup_l\ h(\widehat{g_{kl}})\right| \tag{4}$$

Here, given an interval $S \subseteq [0,1]$, $|S|$ denotes the fraction of the unit interval covered by S. Note that the coverage for a path is the *union* of the assigned hash-ranges across its constituent SamplingSpecs — if the *same* hash-range is assigned to several SamplingSpecs along a path, then the same set of flows gets sampled and we do not get any extra coverage.

The *monitoring load* on a router is given by adding, over all SamplingSpecs $a_k \in R_j$.specs, the portion of the traffic through a_k that R_j logs:

$$Load_j = \sum_{a_k \in R_j.\text{specs}} t_k \times \left|\bigcup_l\ h(\widehat{g_{kl}})\right| \tag{5}$$

Given the C_is, the specific functions we are interested in optimizing are the *total traffic coverage* $f_{tot} = \sum_i T_i C_i$, and the *minimum fractional coverage* $f_{min} = \min_i C_i$. Formally, our goal is to obtain a set of assigned SamplingAtoms $\{\widehat{g_{kl}}\}$ to maximize f_{tot} or f_{min}, while operating within the router constraints (i.e., $Load_j \leq L_j$ for all j). We choose these specific objective functions because of their use in cSamp [7]; our framework can accommodate a wider range of objective functions expressed as combinations of the C_i values.

The maximization problem: We can rewrite the above maximization problem as follows. Consider a ground set $\mathcal{V}$ which contains as its elements all possible SamplingAtoms: i.e., $\mathcal{V} = \{\langle a_k, h_l\rangle$ *for all possible SamplingSpecs* a_k *and all* $\frac{1}{\delta}$ *hash-ranges* $h_l\}$. Suppose a subset $S \subseteq \mathcal{V}$ of these SamplingAtoms are chosen and assigned to their corresponding routers. These give us the fractional coverages defined by Eq 4 and router loads given by Eq 5. Now, f_{tot} or f_{min} can be viewed as functions from subsets of $\mathcal{V}$ to the reals. The problem is to select an *optimal* $S^* \subseteq \mathcal{V}$, i.e., that maximizes f_{tot} or f_{min}, subject to $Load_j \leq L_j$.

Notation	Explanation
M	Number of OD-pairs
N	Number of routers
OD_i	OD-pair i
C_i	Fraction of flows on OD-pair i covered
R_j	Router j
L_j	Available resources on R_j
$Load_j$	Total monitoring load on R_j
SamplingSpec	Info. used for making sampling decisions
a_k	SamplingSpec k
R_j.specs	Set of SamplingSpecs on R_j
t_k	Total traffic traversing SamplingSpec a_k
SamplingAtom	SamplingSpec along with a specific hash range
g_{kl}	SamplingAtom l on a_k
$\widehat{g_{kl}}$	An assigned/selected SamplingAtom
$h(g_{kl})$	Hash-range $\subseteq [0,1]$ for SamplingAtom g_{kl}

TABLE II
NOTATION IN THE PROBLEM STATEMENT

B. Exact Solutions are Hard

Finding an optimal S^* to maximize f_{tot} or f_{min} subject to the load constraints on routers is NP-hard. (We prove hardness via a reduction from 3-SAT; we omit the proof for brevity and refer readers to our technical report [8]). Moreover, it is infeasible for practical system sizes. For example, consider the problem as an integer linear programming formulation using $\{0,1\}$ indicator variables for each g_{kl} to denote whether it is assigned or not. Even on the Internet2 topology with just 11 routers, the commercial solver `CPLEX` did not converge after a day. Because of the intractability of solving the problem exactly, we use approximation algorithms. However, as we

[4]Since this notion of on-path-ness is quite general, our approach works even in the case of multi-path routing.

will see, the performance of our algorithms is comparable to the ideal performance of cSamp.

IV. Submodularity and Algorithms

In this section, we show that there are practical approximation algorithms to obtain the sampling strategies. The key insight is that the coverage functions have a natural submodularity property which allows us to extend results from the theory of maximizing submodular set functions.

A. Submodularity

Definition: A function $F : 2^{\mathcal{V}} \to \Re$, mapping subsets of a ground set $\mathcal{V}$ to the reals, is *submodular* if for all sets $S \subseteq S' \subseteq \mathcal{V}$, and for all elements $s \in \mathcal{V}$,

$$F(S \cup \{s\}) - F(S) \geq F(S' \cup \{s\}) - F(S')$$

i.e., *the marginal benefit obtained from adding s to a larger set is smaller* [9]. This captures the intuitive property of diminishing returns. A function F is *monotone (nondecreasing)* if $\forall S \subseteq S', F(S) \leq F(S')$.

Submodular set maximization: The goal is to pick a subset $S \subset \mathcal{V}$ maximizing $F(S)$ subject to a budget constraint of the form $c(S) \leq B$; i.e., given costs $c(s)$ for all $s \in \mathcal{V}$, the total cost $c(S) := \sum_{s \in S} c(s)$ of elements picked in set S cannot exceed the budget B. This general problem is NP-hard [9], but good approximation guarantees are known. In particular, the algorithm in Figure 3 greedily picks feasible elements that give the greatest marginal benefit or give the maximum marginal benefit per unit element-cost. The better of these two settings gives a constant factor $(1 - e^{-1})$ approximation [29].

B. Application to cSamp-T

It is easy to check the coverages C_i viewed as functions from $2^{\mathcal{V}} \to \Re$ where $\mathcal{V} = SamplingAtoms$ are monotone submodular, and hence so is their weighted sum $f_{tot} = \sum_i T_i C_i$. Each C_i is monotone because adding a SamplingAtom g_{kl} to a set can only increase its value. To see why it is submodular, consider adding a SamplingAtom $g_{kl} = \langle a_k, h_l \rangle$ to sets A and B, where $A \subseteq B$. If we look at the impact on some C_i such that $a_k \in P_i$,[5] three cases arise: (i) h_l is covered in both A and B, (ii) h_l is not covered in both A and B, and (iii) h_l is covered in B, but not A. (Since $A \subseteq B$, the fourth case cannot occur.) In all cases, the marginal benefit of adding g_{kl} to A is at least as high as that of B.

Budget constraints in cSamp-T: The budget constraints in cSamp-T come from the bounds on router load. To model router load, we need a knapsack constraint $Load_j \leq L_j$ for each router R_j. A naive approach is to consider the cSamp-T problem as a submodular set maximization problem with multiple knapsack constraints. This naive approach yields a $O(N)$ approximation, where N is the number of routers. This is clearly undesirable, especially for large networks. However, these budget constraints have a special structure.

[5]If $a_k \notin P_i$, then g_{kl} does not contribute to C_i.

SubmodularGreedy$(F, \mathcal{V}, cbflag, B)$
// $F : 2^{\mathcal{V}} \to \Re$ submodular, B is total budget
// if *cbflag* is `true` use benefit/cost instead of benefit
1 $S \leftarrow \emptyset$
2 **while** $(\exists s \in \mathcal{V} \setminus S : c(S \cup \{s\}) \leq B)$ **do**
3 **for** $s \in \mathcal{V} \setminus S$ **do**
4 $norm \leftarrow ((cbflag = \mathbf{true})\ ?\ c(s)\ :\ 1)$
5 $\psi_s \leftarrow \frac{F(S \cup \{s\}) - F(S)}{norm}$
6 $s^* \leftarrow \mathbf{argmax}_{s \in \mathcal{V} \setminus S} \psi_s$
7 $S \leftarrow S \cup \{s^*\}$
8 return $\langle S, F(S) \rangle$

Fig. 3. Basic greedy algorithm

GreedyMaxMin$(F_1, \ldots, F_M, \epsilon, \mathcal{V}, B, \gamma)$
// Maximize $\mathbf{min}_i \{F_i\}$
// $\forall i,\ F_i : 2^{\mathcal{V}} \to [0, 1]$ is submodular
1 $\tau_{lower} \leftarrow 0$; $\tau_{upper} \leftarrow 1$
2 **while** $(\tau_{upper} - \tau_{lower} > \epsilon)$ **do**
3 $\tau_{current} \leftarrow \frac{\tau_{upper} + \tau_{lower}}{2}$
// Define the modified objective function
4 $\forall i, \hat{F}_i \leftarrow \mathbf{min}(F_i, \tau_{current})$; $\hat{F} \leftarrow \sum_i \hat{F}_i$
// Run greedy without budget constraints
5 $B_{used} \leftarrow$ SubmodularGreedy$(\hat{F}, \mathcal{V}, \mathbf{true}, \infty)$
6 **if** MaxUsage$(B_{used}, B) > \gamma$
7 **then** $\tau_{upper} \leftarrow \tau_{current}$
8 **else** $\tau_{lower} \leftarrow \tau_{current}$
9 Return τ_{lower}

Fig. 4. Maximizing the minimum of a set of submodular functions with resource augmentation

Specifically, since each SamplingAtom contributes to the load on exactly one router, this results in a collection of *non-overlapping* knapsack constraints. We call the resulting problem *submodular function maximization subject to partition-knapsack constraints* – each partition corresponds to a different router and the load constraint on each router is a knapsack constraint. A simple extension of Figure 3 gives a constant-factor $(\frac{e-1}{3e-1})$ approximation.[6] (We omit the proof due to space constraints and refer readers to our technical report [8].)

Maximizing f_{tot}: To match the theoretical guarantees [29], we run two instances of the greedy algorithm—with and without the benefit-cost flag set to true, and return the solution with better performance.

Maximizing f_{min}: To maximize f_{min}, we need to go from one submodular function F to many submodular functions $F_1, F_2, \ldots, F_M$ corresponding to the fractional coverages $C_1, \ldots, C_M$. The problem is now to pick $S \subseteq \mathcal{V}$ to maximize $F^{\min}(S) = \min_i F_i(S)$, the *minimum* across these different functions. This new function $F^{\min}$ is not submodular. In fact, obtaining any non-trivial approximation guarantee for this max-min problem is NP-hard [11]. However, we can maximize $F^{\min}$ if we are allowed to exceed the budget constraint by

[6]Note that these are *worst-case* approximation guarantees; the greedy algorithms for submodular maximization perform much better in practice.

some factor [11]. Formally, if S^* is an optimal set satisfying the budget constraints, the algorithm in Figure 4 finds a set S with $F^{\min}(S) \geq F^{\min}(S^*) - \epsilon$, but exceeds the budget constraints by a factor of γ, where $\gamma = O\big(\log(\frac{1}{\epsilon}\sum_{v\in\mathcal{V}} F_i(v))\big)$.

The key idea is that the modified objective function $\hat{F}_\tau = \sum_{i=1}^{M} \min(F_i, \tau)$ is submodular. For any τ, $\hat{F}_\tau$ has the property that its maximum value is $M \times \tau$ and at this maximum value $\forall i, F_i \geq \tau$. Running the greedy algorithm assuming no resource constraints always gives a set such that the actual resource usage at router R_j is at most $\gamma \times Load_j$. This holds for all τ, and in particular, for the optimal $\tau^* = F^{\min}(S^*)$. Since τ^* is unknown, we use binary search over τ.

Router algorithm: Given a solution to the problem of maximizing f_{tot} or f_{min}, each router is assigned a set of non-contiguous hash ranges for each SamplingSpec. For each packet, the router determines the SamplingSpec using the packet header and other local information (e.g., routing table, which interface does the packet arrive on and leave from). It selects the packet if the hash of the flow 5-tuple falls in one of the hash ranges assigned for this SamplingSpec.

C. Practical Issues

Reducing computation time: The computation time of the algorithm in Figure 3 can be reduced using the insight that for each element $s \in \mathcal{V}$, the marginal benefit ψ_s obtained by picking s decreases monotonically across iterations of the greedy algorithm. Thus, we can use *lazy evaluation* [10]. The intuition behind lazy evaluation is that not all ψ_s values need to be recomputed in Step 5 of Figure 3; only those likely to affect the choice of s^* in Step 6 need to be computed. Section VI-B shows that this reduces the computation time by more than an order of magnitude.

For very large topologies (>200 nodes), we use two additional optimizations: (1) In each greedy iteration, we evaluate the next k best choices in *parallel* (using the OpenMP library). (2) We use the cSamp solution for the minimum fractional coverage as the starting upper bound and avoid unnecessary iterations for the binary search in Figure 4.

Generalizing SamplingSpecs: We assumed that the SamplingSpecs are defined in terms of router three-tuples. Note, however, that our algorithms are generic and do not depend on SamplingSpecs being router three-tuples. Thus, we can generalize our results to any definition of SamplingSpecs – the SamplingSpecs can be made coarser (e.g., ignore previous and next hop information), or more fine-grained (e.g., add egress information if available).

Effects of Discretization: Section III defined a discretization interval δ such that $g_{kl} = \langle a_k, [(l-1)\delta, l\delta]\rangle$, $l \in \{1, \ldots, \frac{1}{\delta}\}$. There are two practical issues here. First, we can make the width δ arbitrarily small; there is a tradeoff between potentially better coverage vs. the time to compute the solution. In our evaluations, we fix $\delta = 0.02$ since we find that it works well in practice. Second, instead of $\frac{1}{\delta}$ disjoint intervals, we can also consider the $\frac{1}{\delta}^2$ hash-ranges of the form $[m\delta, (m+n)\delta]$ to make assignments as contiguous as possible. This increases the computation time without giving any coverage benefits. We avoid this overhead and instead run a simple merge procedure (Section VI-C) to compress the sampling manifests.

V. Heuristic Extensions

While the theoretical guarantee for f_{tot} is encouraging, the result for f_{min} requires fairly high resource augmentation factors (γ) to get non-trivial guarantees.

In this section, we describe two practical extensions to improve the performance for f_{min}.

1. Targeted provisioning to use fewer additional resources.
2. Incremental deployment where some ingress routers are upgraded to add OD-pair identifiers.

We present these in the specific context of the f_{min} objective. However, these techniques can be applied to other network-wide objectives as well.

A. Intelligent Provisioning

The theoretical bound in Section IV assumes that each router is given $\gamma\times$ more resources. However, it is expensive to add $\gamma\times$ more SRAM to *all* routers. Instead, we selectively augment a few routers and still get the same performance. The insight here is that it suffices to upgrade a small number of heavily loaded routers.

We consider the following provisioning problem. The operator gives a memory budget $Budget$ to be distributed across routers, determined by the ISP's monetary constraints and router SRAM cost. Each router R_j has a lower bound LB_j for the default memory configuration and an upper bound UB_j for the maximum feasible amount [30]. Our goal is to decide the allocation of resources to routers (the L_js) that will boost the f_{min} objective.

However, it is difficult to model the coverage C_i that the greedy algorithm gives for each OD-pair under a given set of constraints. Thus, we make a simplifying assumption that the hash ranges across the different SamplingSpecs on a given path are non-overlapping. That is, if u_k denotes the size of the hash range assigned for a_k, we express C_i as the sum of the u_ks (Eq 9). Under this assumption, the provisioning problem can be expressed as the linear program shown below. While this is less desirable than modeling the C_is exactly, this is a reasonable approach to generate general provisioning guidelines. As we will see in Section VI-D, this heuristic works well in practice.

Maximize $\mathbf{min}_i\ C_i$, subject to

$$\forall j, \quad \sum_{k:a_k\in R_j.\mathsf{specs}} u_k \times t_k \leq L_j \tag{6}$$

$$\sum_j L_j \leq Budget \tag{7}$$

$$\forall j, \quad LB_j \leq L_j \leq UB_j \tag{8}$$

$$\forall i, \quad C_i = \sum_{k:a_k\in P_i} u_k \tag{9}$$

$$\forall k, u_k \geq 0; \forall i, \quad C_i \leq 1 \tag{10}$$

Given the optimal memory allocations after solving the LP, we run the greedy algorithm in Figure 4 with $\gamma = 1$ to ensure that we operate within the resource constraints.

B. Partial OD-pair identification

Next, we consider a scenario in which the network operator can upgrade some border routers. This can be achieved using a software update to the router or by adding a middlebox that processes each packet, modifies the header, and forwards it to the router. These upgraded nodes have the ability to determine and add OD-pair identifiers to packet headers. We assume that all routers run both cSamp and cSamp-T sampling algorithms. That is, a router logs a flow if its hash falls in a hash-range corresponding *either* to the OD-pair or the SamplingSpec.

Let $\mathcal{P}_e$ be the set of *enabled* OD-pairs whose packets carry OD-pair identifiers and let $\mathcal{P}$ be the set of all OD-pairs. As in Figure 4, we find the maximum minimum fractional coverage using binary search over the parameter τ. The key difference in the new algorithm is that each iteration of the binary search has two steps. In the first step, we solve a cSamp-style linear program over the enabled OD-pairs. In the second step, we define the capped functions $\hat{C}_i(\tau) = \min_i(C_i, \tau)$ for the non-enabled OD-pairs and use the greedy algorithm to maximize $\hat{F} = \sum_i \hat{C}_i$.

$$\text{Minimize } \sum_j L_j, \text{ subject to}$$

$$\forall j, \quad \sum_{i \in \mathcal{P}_e : R_j \in P_i} (d_{ij} \times T_i) \leq L_j \tag{11}$$

$$\forall i \in \mathcal{P}_e, \quad C_i = \sum_{j : R_j \in P_i} d_{ij} \tag{12}$$

$$\forall i \in \mathcal{P}_e, \quad \theta \leq C_i \leq 1 \tag{13}$$

$$\forall i \in \mathcal{P}_e, \ \forall j, \quad d_{ij} \geq 0 \tag{14}$$

In each iteration, for the current value $\tau_{current}$, the first step solves the LP shown above. The input to the LP is the set of enabled OD-pairs $\mathcal{P}_e$ and the target coverage $\theta = \tau_{current}$. The LP minimizes the total resources used across the routers while ensuring that each $OD_i \in \mathcal{P}_e$ has $C_i \geq \theta = \tau_{current}$. Solving the LP returns the resources allotted to each router or an infeasible status if there is no feasible solution.

If the LP is infeasible, we proceed to the next iteration of the binary search. If the LP is feasible, we obtain the new resource constraints per router by subtracting the resources used in the LP stage from the original resource limits. Next, we run the greedy algorithm with these reduced resources and the modified objective $\hat{F}$ specified over the non-enabled OD-pairs. By construction, the maximum value of $\hat{F}$ is $(M - |\mathcal{P}_e|) \times \tau_{current}$ where M is the number of OD-pairs and $|\mathcal{P}_e|$ is the number of enabled OD-pairs. $\hat{F}$ reaches this value if and only if each non-enabled OD-pair in $\mathcal{P} \setminus \mathcal{P}_e$ has fractional coverage $\geq \tau_{current}$. If the greedy algorithm achieves this value, then $\tau_{current}$ is feasible and we try a higher value in the next iteration; else we try a lower value in the next iteration.

Topology (AS#)	PoPs	OD-pairs	Flows $\times 10^6$	Packets $\times 10^6$
NTT (2914)	70	4900	51	204
Level3 (3356)	63	3969	46	196
Sprint (1239)	52	2704	37	148
Telstra (1221)	44	1936	32	128
Tiscali (3257)	41	1681	32	218
GÉANT	22	484	16	64
Internet2	11	121	8	32

TABLE III
PARAMETERS FOR THE EXPERIMENTS

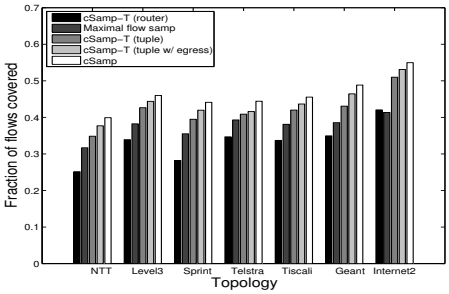


Fig. 5. Total flow coverage

VI. EVALUATION

Setup: We use PoP-level topologies of educational backbones and tier-1 ISPs [31] to evaluate the performance of cSamp-T with each PoP as a node in the network (Table III). We use shortest-path routes for each OD-pair and model the traffic matrix using a gravity model based on city populations [19]. We assume that each node can log $L = 400,000$ flow records.[7] For cSamp-T, we discretize the hash-range with $\delta = 0.02$.

A. Coverage and Overlap

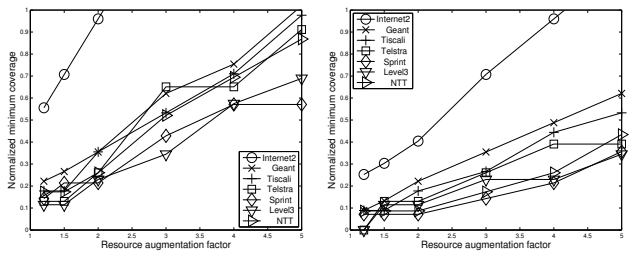
Total flow coverage: We consider three granularities of SamplingSpecs: router, router 3-tuple, and router 3-tuple augmented with egress information. The first two SamplingSpecs can always be inferred from local information but there may be some ambiguity in resolving the egress (Table I). We use the tuple+egress as a hypothetical solution to emulate the effect of MPLS-based forwarding. We also compare these to cSamp and maximal (uncoordinated) flow sampling.[8]

Figure 5 shows that using 3-tuple SamplingSpecs provides significant improvement (25-30%) over the router-only case. cSamp-T (3-tuple+egress) is closer to cSamp, but the gap between the 3-tuple and egress-added cases is small. We also verified that the performance of the greedy algorithm is close to the theoretical upper bound for cSamp-T. (Not shown; please see the extended report [8] for additional results).

The theoretical guarantee for total flow coverage depends on running two instances of the greedy algorithm: with and without the cost-benefit flag. We found that both configurations have similar performance and that the instance with the cost-benefit flag $cbflag = \textbf{false}$ is slightly better.

[7] Assuming 12 bytes per flow record [7], this requires $400,000 \times 12 = 4.8$ MB of SRAM *per PoP*, which is within the 8 MB technology limit *per linecard* suggested by Varghese [30].

[8] In maximal flow sampling, each router's flow sampling rate is $\min(1, \frac{l}{t})$, where l is the number of flows it can log and t is the number of flows it observes.



(a) Tuple (b) Router

Fig. 6. Normalized min. fractional coverage with resource augmentation

Minimum fractional coverage: Section IV showed that it is infeasible to maximize f_{min} without resource augmentation. Thus, we evaluate the performance as a function of the resource augmentation factor γ, where each router's SRAM is $\gamma \times 400,000$. We normalize the minimum fractional coverage by the optimal value achieved by cSamp at $\gamma = 1$. That is, if cSamp-T has value 0.2 at $\gamma = 3$ and cSamp has value 0.4 at $\gamma = 1$, the y-value corresponding to $\gamma = 3$ is $\frac{0.2}{0.4} = 0.5$.

Figure 6 shows the result for the router and tuple granularities. The tuple+egress was almost identical to the tuple case; we do not show this for brevity. With $\gamma \geq 4$, cSamp-T achieves $\geq 50\%$ of cSamp for all topologies. We see that the difference between the router and tuple formulations is more pronounced in the minimum fractional coverage result. With router-level SamplingSpecs, even at $\gamma = 5$, four out of the seven topologies only reach 40% of cSamp's performance. For the same $\gamma = 5$, using 3-tuple SamplingSpecs, five out seven topologies achieve $\geq 90\%$ of cSamp's performance. Also, the γ at which cSamp-T has good performance is much better than the theoretical bound in Section IV. Section VI-D shows that targeted provisioning reduces this even further.

We see that 3-tuple SamplingSpecs perform much better than router SamplingSpecs, and are very close to the tuple+egress case. Thus, we focus on 3-tuples for the rest of the evaluation.

Duplicated flow reports: A secondary objective in cSamp is to avoid duplicate flow reports to reduce the overhead in processing duplicated measurements. Maximal uncoordinated sampling can have $\geq 30\%$ duplicate reports (expressed as a fraction of the number of unique flows logged). Compared to the uncoordinated case, cSamp-T with 3-tuples has $3\times$ fewer duplicated flow reports (not shown). Relative to cSamp which has no duplicate reports, this is not ideal. However, this is unavoidable since cSamp-T operates at a much coarser granularity.

B. Algorithm Running Time

In order to be responsive to traffic dynamics, we want the time to compute sampling manifests to be within a few minutes. (Configurations are typically recomputed across epochs spanning several minutes.) Table IV shows the time to greedy solution on a 4-CPU (Intel Xeon 3.20GHz) machine. Lazy evaluation provides more than an order of magnitude reduction compared to the naive algorithm. The reduction is more significant for the minimum fractional coverage since it involves multiple calls to the greedy subroutine. With this reduction, cSamp-T scales to larger PoP-Level topologies.

Topology	Total coverage (sec)		Min. Fractional (sec)	
	Naive	Lazy	Naive	Lazy
NTT	207.12	4.15	39632	154.1
Level3	205.36	3.30	48269	84.3
Sprint	75.30	2.21	14211	71.6
Telstra	50.53	1.65	6997	45.0
Tiscali	35.18	1.16	8518	33.7
GÉANT	3.06	0.28	542	7.6
Internet2	0.22	0.05	38.4	1.9

TABLE IV
TIME TO COMPUTE SAMPLING MANIFESTS: VANILLA GREEDY VS. LAZY EVALUATION (WITH A SINGLE THREAD) FOR POP-LEVEL TOPOLOGIES

Topology	# Routers	Total Cov. (sec)	Min. Frac. (sec)
NTT	350	345.9	994.7
Level3	315	224.1	540.2
Sprint	260	174.0	554.6
Telstra	220	180.7	267.6
Tiscali	205	77.0	327.4

TABLE V
COMPUTE TIMES FOR ROUTER-LEVEL TOPOLOGIES (4 THREADS)

Next, we evaluate how the algorithms scale to very large router-level topologies. We generate router-level topologies by treating each PoP as a "core" router and adding 4 edge routers to each such core router. As described earlier, we use two extra optimizations: parallel execution and tighter upper bounds for the binary search. Table V show that even for very large topologies, the compute times are within reasonable bounds. This can be further reduced by increasing the degree of parallelization.

C. Size of sampling manifests

Compared to cSamp, cSamp-T increases the size of the sampling manifests because the hash-ranges assigned for each SamplingSpec need not be contiguous. We use a simple compression procedure to merge hash ranges after the greedy algorithm. This looks for maximally contiguous hash ranges in the original sampling manifest and merges them into a single hash range. Table VI shows that this compression procedure reduces the manifests roughly $10\times$. Also, the total bandwidth overhead after compression is only 25KB in the worst case.

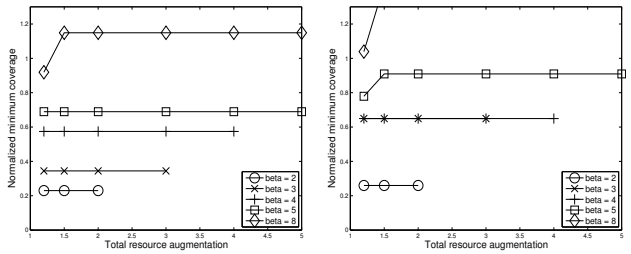
D. Intelligent Provisioning

As a specific scenario, we set $LB_j = L = 400,000$ for all j in the formulation from Section V-A. We specify the total SRAM budget as $Budget = \gamma \times N \times L$, where N is the number of PoPs, and the technology limit as $\beta \times L$. We vary the parameters γ and β. Figure 7 shows the minimum fractional coverage normalized w.r.t cSamp for two topologies, Level3 and Telstra. We choose these because the greedy algorithm performs poorly compared to cSamp in Figure 6. An interesting result is that the curve levels off as a function of γ; i.e., increasing the total budget does not add much benefit. However, increasing the upper bound β provides significant improvement. In fact, even with a moderate total increase $\gamma = 1.2$, we see that the performance is within 80% of cSamp.

Since β is more crucial than γ, for the remaining topologies we fix $\gamma = 1.5$ and analyze the normalized minimum fractional

Topology	Total (KB)		Max. per PoP (KB)	
	Naive	Merged	Naive	Merged
NTT	178.5	16.3	5.6	1.0
Level3	341.9	25.2	34.1	3.3
Sprint	140.9	13.0	10.3	0.6
Telstra	112.3	7.2	3.3	0.5
Tiscali	110.9	12.6	9.8	0.6
GÉANT	45.5	6.5	5.6	0.6
Internet2	14.5	5.0	4.5	0.7

TABLE VI
SIZE OF THE SAMPLING MANIFESTS IN CSAMP-T IN KILOBYTES OF TEXT CONFIGURATION FILES



(a) Level3 (b) Telstra

Fig. 7. Understanding the impact of total resource augmentation (γ) and technology upper bound (β) in the resource allocation formulation.

coverage as a function of β in Figure 8. With $\beta = 5$, all topologies achieve $\geq 60\%$ of cSamp's performance. Contrasting this with Figure 6, the main difference is that we do not require all PoPs to be augmented with $5\times$ more resources – the total resource budget is $\leq 1.5\times$.

E. Partial OD-pair identification

We try three strategies for selecting the enabled OD-pairs $\mathcal{P}_e$ by upgrading the top-k PoPs that (a) observe the maximum amount of traffic, (b) lie on most number of routing paths, or (c) originate the most traffic. Here, upgrading implies that we enable OD-pair identifiers on all OD-pairs having one of these top-k PoPs as origins. For each k, we run the two-step procedure from Section V-B for all values in $1, \ldots, k$ and pick the configuration with the highest f_{min}.

Figure 9 shows the normalized minimum fractional coverage for the Level3 and Telstra topologies as a function of k (number of top-k PoPs). First, we observe that upgrading just a few PoPs ($< 6\%$) significantly improves the performance. Second, enabling identifiers on nodes that observe the most traffic performs much better than the other two strategies.

F. Hybrid Coverage Objective

cSamp maximizes a hybrid objective: maximizing the total flow coverage subject to getting the highest minimum fractional coverage per OD-pair. In cSamp-T, we considered these two objectives separately. A natural question is if we can also maximize this hybrid objective. It is easy to extend the algorithm in Figure 4 to do this. We run the greedy algorithm to optimize the capped minimum fractional objective ($\hat{F}$) and then switch the objective function to optimize the total coverage if $\tau_{current}$ is feasible.

To evaluate this hybrid approach, we consider the configuration obtained after targeted provisioning with $\gamma = 1.5$ and

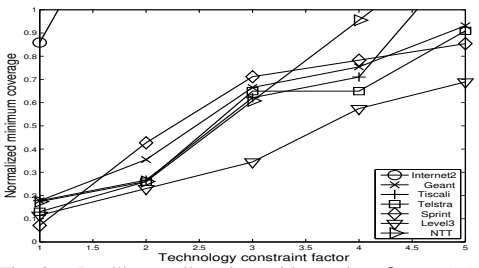
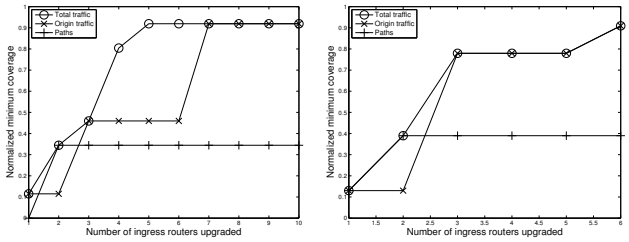


Fig. 8. Intelligent allocation with varying β at γ=1.5



(a) Level3 (b) Telstra

Fig. 9. Min. fractional coverage with partial OD-pair identification. This can be viewed as incremental deployment of cSamp via cSamp-T.

$\beta = 5$. Table VII compares the total coverage obtained with three strategies: maximizing the minimum fractional coverage, maximizing the total flow coverage, and the above two-step procedure. Maximizing the fractional coverage alone does not work well. This is because the greedy algorithm terminates when it achieves the target fractional coverage, even if it can increase the total coverage. Also, the total coverage obtained by the hybrid approach is very close to the greedy algorithm for maximizing the total coverage alone. While it is hard to provide theoretical guarantees in this case, Table VII shows that the two-step optimization works well in practice.

VII. DISCUSSION

More fine-grained local information: We can bring the performance of cSamp-T even closer to cSamp by providing more hints to routers. One possible approach is to distribute IP-prefix to ingress-egress maps [32], to enable more fine-grained sampling decisions.

Sensitivity of router upgrades: The formulations in Section V for router upgrades, as presented, assume static routing and traffic configurations. Evaluating the sensitivity of the upgrades and designing upgrade strategies robust to dynamics are topics of future work. One strategy is to leverage the fact that real-world routing and traffic matrices have some dominant patterns that are largely invariant to dynamics. Thus, we can use these invariants as inputs to the formulations.

VIII. OTHER RELATED WORK

The closest related work is cSamp [7], which we discussed in Section II. Here, we discuss other related work.

Sampling: Most related work focuses on the single-router case to work around limitations of packet sampling. This includes work on adaptive sampling [33], inverting sampled measurements [6, 14], and data streaming algorithms (e.g., [15, 21]). cSamp and cSamp-T depart from these approaches by taking a network-wide coordinated approach for flow monitoring.

Topology	Greedy-Minfrac		Greedy-Total
	NoHybrid	Hybrid	
NTT	0.13	0.58	0.58
Level3	0.10	0.60	0.60
Sprint	0.22	0.61	0.64
Telstra	0.13	0.59	0.62
Tiscali	0.23	0.60	0.63
GÉANT	0.35	0.63	0.68
Internet2	0.60	0.71	0.78

TABLE VII
COMPARING THE HYBRID MAXIMIZATION TO THE GREEDY ALGORITHM FOR MAXIMIZING THE TOTAL FLOW COVERAGE

Greedy algorithms for monitor placement: Prior work has applied greedy algorithms for monitor placement to cover all routing paths using as few monitors as possible [16, 17]. The authors show that this is NP-hard and propose greedy algorithms. These formulations can be extended to incorporate packet sampling [17, 18]. However, these do not satisfy flow coverage objectives, and by relying on packet sampling, they can result in a large amount of redundant flow measurements. cSamp-T provides more fine-grained flow coverage objectives and reduces duplicated flow reports.

Sensor network monitoring: There has been recent work applying the theory of maximizing submodular functions in sensor networks [34, 35]. The problem of placing sensors robust to adversarial objectives [11] is conceptually similar to maximizing the minimum fractional coverage.

IX. CONCLUSIONS

cSamp is a recent proposal to meet the demand for fine-grained flow monitoring capabilities in ISPs. However, ISPs cannot realize the benefits of cSamp in practice because of its reliance on OD-pair identifiers. In its current form, this would require changes to packet headers, impose additional overhead at ingress routers, and may require ISPs to overhaul their routing infrastructures.

We presented cSamp-T, a framework that provides benefits comparable to cSamp, in which the sampling decisions at routers are based only on local information, and do not rely on global OD-pair identifiers. However, obtaining exact solutions to maximize the total flow coverage (f_{tot}) and minimum fractional coverage (f_{min}) in this framework is NP-hard. We achieve near-optimal performance for f_{tot} by leveraging its submodularity. For f_{min}, getting good performance without resource augmentation is provably hard. However, targeted provisioning achieves near-ideal performance with low overhead. Alternatively, upgrading a small number of border routers to provide OD-pair information also yields good performance. cSamp-T thus makes the benefits of a coordinated network-wide monitoring solution like cSamp more immediately available to ISPs. It also provides an incremental deployment path for ISPs to transition to cSamp.

ACKNOWLEDGMENTS

We thank Daniel Golovin and Matthew Streeter for suggesting the NP-hardness proof, and Aman Shaikh for discussions about MPLS and IP routing. This work was supported in part by NSF awards CNS-0756998 and ANI-0331653.

REFERENCES

[1] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing Network-Wide Traffic Anomalies," in *ACM SIGCOMM*, 2004.
[2] M. P. Collins and M. K. Reiter, "Finding Peer-to-Peer File-sharing using Coarse Network Behaviors," in *ESORICS*, 2006.
[3] K. Xu, Z.-L. Zhang, and S. Bhattacharya, "Profiling Internet Backbone Traffic: Behavior Models and Applications," in *ACM SIGCOMM*, 2005.
[4] A. Ramachandran, S. Seetharaman, and N. Feamster, "Fast Monitoring of Traffic Subpopulations," in *IMC*, 2008.
[5] Y. Xie, V. Sekar, D. A. Maltz, M. K. Reiter, and H. Zhang, "Worm Origin Identification Using Random Moonwalks," in *IEEE Symposium on Security and Privacy*, 2005.
[6] N. Duffield, C. Lund, and M. Thorup, "Charging from sampled network usage," in *IMW*, 2001.
[7] V. Sekar, M. K. Reiter, W. Willinger, H. Zhang, R. Kompella, and D. G. Andersen, "cSamp: A System for Network-Wide Flow Monitoring," in *NSDI*, 2008.
[8] V. Sekar, A. Gupta, M. K. Reiter, and H. Zhang, "Coordinated Sampling sans Origin-Destination Identifiers: Algorithms, Analysis, and Evaluation," Technical Report, CMU-CS-09-104, CMU, 2009.
[9] G. Nemhauser, L. Wosley, and M. Fisher, "An analysis of the approximations for maximizing submodular set functions," *Mathematical Programming*, vol. 14, pp. 265–294, 1978.
[10] M. Minoux, "Accelerated Greedy Algorithms for Maximizing Submodular Set Functions," in *8th IFIP Conference, Springer-Verlag*, 1977.
[11] A. Krause, B. McMahan, C. Guestrin, and A. Gupta, "Selecting Observations Against Adversarial Objectives," in *NIPS*, 2007.
[12] A. Anand, V. Sekar, and A. Akella, "SmartRE: An Architecture for Coordinated Network-Wide Redundancy Elimination," in *SIGCOMM*, 2009.
[13] Y. Zhang, S. Singh, S. Sen, N. Duffield, and C. Lund, "Online Detection of Hierarchical Heavy-hitters," in *IMC*, 2004.
[14] N. Hohn and D. Veitch, "Inverting Sampled Traffic," in *IMC*, 2003.
[15] A. Kumar, M. Sung, J. Xu, and J. Wang, "Data Streaming Algorithms for Efficient and Accurate Estimation of Flow Distribution," in *ACM SIGMETRICS*, 2004.
[16] C. Chadet, E. Fleury, I. Lassous, H. Rivano, and M.-E. Voge, "Optimal Positioning of Active and Passive Monitoring Devices," in *CoNeXT*, 2005.
[17] K. Suh, Y. Guo, J. Kurose, and D. Towsley, "Locating Network Monitors: Complexity, heuristics and coverage," in *IEEE INFOCOM*, 2005.
[18] G. R. Cantieni et al.,, "Reformulating the Monitor Placement problem: Optimal Network-Wide Sampling," in *CoNeXT*, 2006.
[19] M. R. Sharma and J. W. Byers, "Scalable Coordination Techniques for Distributed Network Monitoring," in *PAM*, 2005.
[20] N. Duffield and M. Grossglauser, "Trajectory Sampling for Direct Traffic Observation," in *ACM SIGCOMM*, 2001.
[21] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting," in *ACM SIGCOMM*, 2002.
[22] H. Ballani and P. Francis, "CONMan: A Step Towards Network Manageability," in *ACM SIGCOMM*, 2007.
[23] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Meyers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A Clean Slate 4D Approach to Network Control and Management," *ACM SIGCOMM CCR*, vol. 35, no. 5, Oct. 2005.
[24] A. Feldmann, A. G. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience," in *ACM SIGCOMM*, 2000.
[25] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast Accurate Computation of Large-scale IP Traffic Matrices from Link Loads," in *ACM SIGMETRICS*, 2003.
[26] M. Ramakrishna, E. Fu, and E. Bahcekapili, "Efficient Hardware Hashing Functions for High Performance Computers," *IEEE Transactions on Computers*, vol. 46, no. 12, pp. 1378–1381, 1997.
[27] R. Kompella and C. Estan, "The Power of Slicing in Internet Flow Measurement," in *IMC*, 2005.
[28] Personal Communication with Aman Shaikh, AT&T Research.
[29] L. A. Wolsey, "Maximising real-valued submodular functions: Primal and dual heuristics for location problems," *Mathematics of Operations Research*, vol. 7, no. 3, pp. 410–425, 1982.
[30] G. Varghese, *Network Algorithmics*. Morgan Kaufman, 2005.
[31] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP Topologies with Rocketfuel," in *ACM SIGCOMM*, 2002.
[32] K. Argyraki, P. Maniatis, O. Irzak, S. Ashish, and S. Shenker, "Loss and Delay Accountability for the Internet," in *ICNP*, 2007.
[33] C. Estan, K. Keys, D. Moore, and G. Varghese, "Building a Better NetFlow," in *ACM SIGCOMM*, 2004.
[34] C. Guestrin, A. Krause, and A. Singh, "Near-optimal Sensor Placements in Gaussian Processes," in *ICML*, 2005.
[35] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, "Near-optimal Sensor Placements: Maximizing Information while Minimizing Communication Cost," in *IPSN*, 2006.