Scalable Coordination Techniques for Distributed Network Monitoring

Manish R. Sharma and John W. Byers

Dept. of Computer Science, Boston University, Boston MA 02215, USA

Abstract. Emerging network monitoring infrastructures capture packet-level traces or keep per-flow statistics at a set of distributed vantage points. Today, distributed monitors in such an infrastructure do not coordinate monitoring effort, which both can lead to duplication of effort and can complicate subsequent data analysis. We argue that nodes in such a monitoring infrastructure, whether across the wide-area Internet, or across a sensor network, should coordinate effort to minimize resource consumption. We propose space-efficient data structures for use in gossip-based protocols to approximately summarize sets of monitored flows. With some fine-tuning of our methods, we can ensure that all flows observed by at least one monitor are monitored, and only a tiny fraction are monitored redundantly. Our preliminary results over a realistic ISP topology demonstrate the effectiveness of our techniques on monitoring tens of thousands of point-of-presence (PoP) level network flows. Our methods are competitive with optimal off-line coordination, but require significantly less space and network overhead than naive approaches.

1 Introduction

In monitoring applications ranging from wide-area traffic monitoring to event detection in sensor networks to surveillance by a set of pan/tilt/zoom cameras located at distributed vantage points, a growing challenge involves appropriate coordination of the activities of the individual monitors. In the applications above, monitors are *resource-constrained*, and thus it is essential to minimize the effort monitors expend on monitoring tasks. For example, when any of a set of monitors may perform a monitoring task equivalently well, a cost-saving strategy is to elect a single leader to perform the job. Of course, such a leader election process must be efficient, must be robust to losses and failures, and must err on the side of conservatism to ensure that all observable activities are monitored by at least one monitor. Avoiding duplication of effort has a secondary advantage for applications in which observed data is subsequently aggregated and processed, since complications associated with the presence of duplicate observations are avoided. In this work-in-progress paper, we consider the problem of minimizing duplication of effort in distributed event monitoring in which monitors are connected by a high-speed network. While we believe that both the statement of our problem and our methods are much more broadly applicable, we focus here exclusively on wide-area network traffic monitoring.

Wide-area Network Monitoring: Current technology to monitor network traffic by passively collecting flows or samples or logging packet headers either compromises router performance or incurs high costs due to costly measurement infrastructure. We argue that a brute force, non-adaptive approach to monitoring network traffic misses an opportunity to better manage resource consumption. Instead, we advocate a lower-cost alternative, i.e. developing scalable techniques to coordinate and distribute the monitoring effort. For example, if the monitoring effort can be distributed in such a way that each monitor monitors only a small subset of network flows, substantial savings can be achieved in terms of storage and processing overhead. Our work attempts to achieve the above goal without introducing too much control traffic overhead.

We assume a passive network monitoring infrastructure comprised of multiple monitoring systems, that coordinate to monitor network traffic traversing through them. Such systems are not expected to be ubiquitous or directly integrated into routers but are specially equipped with traffic capture and storage capabilities. We assume that: all monitors can communicate with all other monitors periodically, the monitors have sufficient memory to perform the monitoring functionality, and the monitors can compute the set of monitors on the route of a flow. We model the incoming traffic at monitors as a datastream consisting of items in the form of key-value pairs. Here, the key is taken to be a network flow at the Point of Presence (PoP) level, i.e. an ingress/egress pair, and the value is the size of each packet in bytes.

Problem Statement and Contribution: Let S denote a set of events, let $M = \{m_1, m_2, \ldots m_K\}$ be a set of monitors, and let $V_i \subseteq S$ be the set of events (flows) observable by m_i. Now let L_i denote the set of events monitored by m_i. Our objective is: Minimize $\sum_i |L_i|$ subject to $\bigcup_i L_i = S$ and $\forall i, L_i \subseteq V_i$.

In other words, monitors must monitor only flows they can observe, all observable flows must be monitored at least once, and the goal is to minimize duplication of effort. In the next sections, we describe our approach to this problem using Bloom filter-based summarization techniques to coordinate between a set of network monitors, quantify the cost, and present simulation results.

2 Coordination Algorithms

We now provide a brief overview of our data structures, algorithms and key results; full details and the analysis are in [5]. Each monitor locally maintains two data structures. The first is a lookup table of active flows marked either as actively monitored or monitored by someone else. The second is a counting Bloom filter, that approximately represents the set of active flows at a network monitor. Our approach starts by having each monitor represent the set of flows it is currently monitoring with a counting Bloom filter [1], a compact randomized data structure that supports lookup operations on keys. With a Bloom filter, lookups for inserted keys are always correct, but lookups for keys not present in the filter may yield a false positive, with a tunable false positive probability p. Full details are in [2]. Monitors use a simple gossiping protocol to periodically disseminate their Bloom filters to all other monitors in the system. Use of Bloom

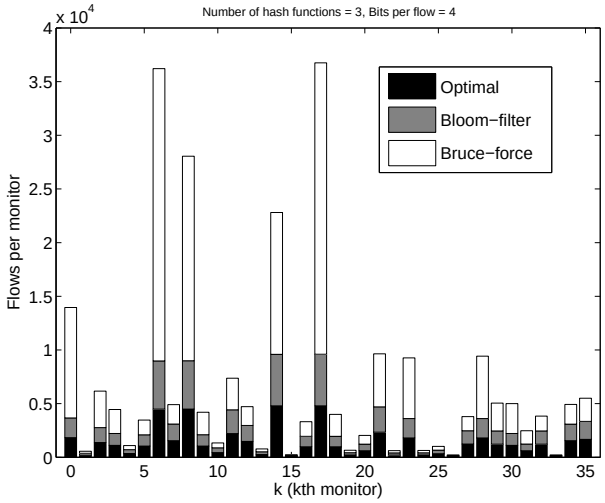

Fig. 1. Performance analysis of monitoring approaches

filters not only reduces the size of the summaries by orders of magnitude as compared to a full-fidelity representation but also keeps the network overhead of gossiping the summaries to other monitors manageable.

On the arrival of a new flow (whether new to the system or due to a route change), a monitor checks if it is the first monitor on the flow's route. If so, it inspects the Bloom filters of other monitors along the route. If the flow appears to be monitored elsewhere, it leaves the flow for the appropriate monitor. Otherwise, the monitor creates new state for the flow and maintains the state from that time onward. If the monitor is not first on the route, then it is not initially responsible for monitoring that flow. However, the approximate nature of the summaries makes them vulnerable to errors in the form of false positives, i.e., a flow is not actually monitored by a monitor but its summary reflects that it is.

The simple, but elegant solution to this potential problem is a central contribution of our work: we eliminate false positives by applying an idea of *self-inspection*: if a given monitor finds that an observable flow produces a false positive in its own Bloom filter, then it immediately starts to monitor that flow. The cost of this method is a small amount of redundant monitoring: in the event that two or more Bloom filters have a match on a given flow, they must all monitor that flow (redundantly). Our analysis in [5] shows that for a flow traversing j monitors using Bloom filters with false positive probability p, the expected number of monitors that will monitor the flow using our method is $(1-p)^j + pj$.

3 Experimental Results

We simulated deployment of our monitoring infrastructure over one of the PoP level topologies generated by Rocketfuel [4]. The topology consisted of 36 PoPs, producing 1296 origin-destination (OD) pairs. Next, we made use of inferred backbone link weights [3] to run Dijkstra's single-source shortest path algorithm

at all PoPs to determine the route from one PoP to any other PoP. To create a plausible distribution of network flows between PoPs in our topology, for all PoP pairs, we compute a value l_{ij} that is the fraction of flows which originate at PoP i and terminate at PoP j. Using a gravity model, we take $l_{ij} \propto P_i \times P_j$, where P_i is the population of node i, and normalize $l_{ij} = \frac{P_i P_j}{\sum P_i P_j}$ to ensure $\sum_{i,j} l_{ij} = 1$. We simulated our proposed network monitoring technique with 50,000 network flows distributed amongst different PoP pairs using the gravity model, and compared the following three different monitoring approaches.

- **Brute force:** Each monitor monitors every flow that is visible to it.
- **Optimal:** Each monitor is given full information about the workload, and the flow is assigned to the least loaded monitor on its route.
- **Bloom filter:** Monitors have no prior information, flows arrive one by one, and our proposed methods do the online assignment of flows to monitors.

Figure 1 plots the number of flows monitored for each approach and at each of the 36 monitors. Our online Bloom filter approach is nearly as good as the offline optimal in terms of overall load reduction and load balance, and significantly improves worst-case load over the brute force approach. Using a simple back-of-the-envelope calculation (omitted for lack of space), we estimate that an unoptimized version of our approach affords more than a factor of two memory savings on average, and more than a factor of five at the worst case monitor in this scenario. Unlike brute force, our methods have an extra cost associated with data exchange to ensure continuous monitoring of all visible flows under route changes and to maintain load balance. In a naive all-pairs exchange of Bloom filters, the total aggregate traffic load is 3.24 MB (200 KB per pair) in our simulation setup.

Future work: Our ongoing work involves experimental evaluation, validation and refinement of our methods over large, realistic datasets. Along with additional evaluation, key considerations that we intend to further investigate in the full version of the paper are: further reducing network overhead when periodically exchanging summaries, refining load balancing mechanisms to improve their robustness, and specifying how data structure parameters can be set automatically.

References

1. Bloom, B. H.: Space/time trade-offs in hash coding with allowable errors. In *Comm. of the ACM*, volume 13(7), pages 422–426, 1970.
2. Broder, A. Z., and Mitzenmacher, M.: Network Applications of Bloom Filters: A Survey, In *Proc. of the 40th Annual Allerton Conference*, 2002.
3. Mahajan, R., Spring, N., Wetherall, D., and Anderson, T.: Inferring Link Weights using End-to-End Measurements. In *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002.
4. Spring, N., Mahajan, R. and Wetherall, D.: Measuring ISP Topologies with Rocketfuel. In *Proceedings of ACM SIGCOMM*, August 2002.
5. Sharma M. R. and Byers J. W.: Scalable Coordination Techniques for Distributed Network Monitoring. *Tech. Report BUCS-TR-2005-001, Boston U.*, January 2005.