Towards Optimized Probe Scheduling for Active Measurement Studies

N. Daniel Kumar, Fabian Monrose, and Michael K. Reiter
Department of Computer Science
University of North Carolina
Chapel Hill, NC, USA
{ndkumar, fabian, reiter}@cs.unc.edu

***Abstract*—Internet measurement studies often require prolonged probing of remote targets to collect information, yet almost all such studies of which we are aware were undertaken without considering the polluting effects of their unconstrained probing behavior. To help researchers conduct experiments in a more responsible fashion, we present a framework and technique that enables efficient execution of large-scale periodic probing without exceeding pre-set limits on probing rates. Our technique employs a novel scheduling algorithm and leverages knowledge of diurnal traffic patterns to make data collection more efficient (e.g., by probing servers only during periods in which probe results will be most useful). We evaluate our technique in the context of a real-world study and show that it substantially outperforms naïve probing strategies for accomplishing the same goal, sending more probes during useful periods, fewer probes overall, and probing at more precise intervals as required by our measurement applications.**

***Keywords*-probe scheduling; responsible network experiments.**

I. Introduction

Over the past several decades, researchers and practitioners have proposed countless techniques for understanding various characteristics about the Internet at large. For the most part, these pursuits have been grounded in empirical measurements that are either passive or active in nature. As their names imply, passive measurements typically involve observations taken from some form of capture device, while active measurements require the injection of specially crafted packets (so-called *probes*) into the network to infer characteristics of the phenomenon under scrutiny.

Recently, there has been a marked increase in the use of active measurements for understanding Internet topology and mapping (e.g., geo-location), path-variance and other end-to-end performance dynamics (e.g., bandwidth estimation), the spread of security-related events (e.g., worm outbreaks), and client and server demographics (e.g., website popularity), etc. To better enable such studies and improve data collection efforts, a number of scalable measurement infrastructures (e.g., CAIDA's Archipelago Measurement [10], Google's M-lab open platforms, and the Flexible Lightweight Active Measurement Environment [25]) have been made available to the research community for controlled experiments. Archipelago, for example, provides a centrally managed framework that supports a distributed shared-memory architecture that can be used to coordinate network measurements across the globe.

A common requirement in many of the Internet-wide studies being conducted today is the need to repeatedly probe some set of targets over time. Collectively, we rely on each experimenter's own restraint to limit the volume of traffic they inject into the network and/or target at a given resource. While some experimenters exercise such restraint to limit collateral damage, in the absence of accepted guidelines for such measurements and tools to enable experiments that respect them, the injection of billions of probes into the network during a short period of time is not uncommon (e.g., [3]); indeed, the literature is rife with examples of arguably egregious practices.

Such disregard for the collateral damage caused by network measurement experiments has raised enough concern that it has led to a call urging the community to move towards a set of best practices for active measurements. As Papadopoulos and Heidemann note [17], as practitioners it should be *our* job to design experiments carefully, in a manner that significantly lowers load without sacrificing measurement fidelity. In this paper, we present a technique for attempting to do just that. Specifically, we propose an efficient scheduling algorithm for probing measurement targets, which is efficient in that it can be tuned towards expeditious completion of the experiment while also observing some predefined maximum probing rate.

Of specific interest to us are studies of Internet demographics (e.g., inferring website popularity rankings [19], [24], exploring the prevalence of malicious domain name system (DNS) servers [6], and client-density estimation [7], [20]) where the fidelity of the experiment can increase if the targets are probed at ideal times. We aim to maximize the utility of each probe that we send, in contexts where the utility of a probe can be correlated with time-of-day, e.g., when it is expedient to probe targets during their peak (or off-peak) traffic periods. Our approach may also be extended for applications requiring that certain subsets of targets be probed contemporaneously, such as RadarGun [2] for IP alias resolution. We show that for experiments taking several days, and given limited probing resources, our scheduling algorithm outperforms several naïve strategies for probe scheduling according to intuitive metrics provided in §IV.

The remainder of this paper is organized as follows. Related work is presented in §II. In §III we outline our scheduling framework and algorithm, which we evaluate in §IV. We conclude in §V.

II. Related Work

As mentioned earlier, some experimenters have exercised restraint to limit collateral damage from the probes they inject. Bender *et al.* [2], for example, attempt to constrain their probing rates to avoid triggering rate limiters. Others [21], [11] have tried to take advantage of opportunistic measurement techniques in order to unobtrusively make network measurements without triggering alarms from intrusion detection systems. Unfortunately, these instances of restraint seem rare, and the academic literature contains numerous examples of experimenters probing at high rates (e.g., 260 packets per second [22]) or injecting high volumes of probes (e.g., 220 million [6] and 27 billion probes [3]),[1] which are arguably indistinguishable from attacks.

Previous work on scheduling active network monitoring activities has focused on preventing the simultaneous scheduling of activities that would interfere with each other's results and lead to inaccurate measurement reports [4], [8], [18]. Here we are not concerned with probes interfering with one another, but we focus rather on respecting a limit on the probe rate we induce on the network, while completing the probing experiment as quickly as possible. That is, because our targets are all distinct, by imposing limits on our probing rate we are able to mitigate (to some extent) packet loss due to network congestion. To complete the experiment quickly, we leverage knowledge of the *useful interval* in which each type of probe should occur, which is characteristic of the type of probing activities considered here but that is not utilized in these prior works. Moreover, we do not assume that information about the underlying network topology is readily available.

Additionally, prior work in this area provides no guidance on how to prioritize probes for scheduling in the absence of network topology, and so is not helpful in our setting. The work of Calyam *et al.* [4] applies the basic *earliest-deadline-first* (EDF) heuristic [15] that we use, but their scheduling is done in an offline setting; they do not utilize efficient data structures to leverage the scalability of the simple online heuristic. Additionally, Calyam *et al.* do not allow for the possibility of dropped probes, and so their algorithm simply fails if all of their probes cannot be issued on time.

Also related to our work (i.e., arranging as many targeted probe sequences as possible to fit within some maximum rate limit) is the literature on perfectly periodic or cyclic scheduling on parallel processors. The fundamental scheduling problems are still NP-hard [16], but efficient approximations [1] could be used for applications not requiring precisely periodic probes.

III. Approach and Assumptions

Our work is motivated by the need to find more efficient ways to schedule network probes. Specifically, we consider the common case wherein the prober needs to issue a series of probes to a given target, j, at some periodic interval, π_j. In some cases, each target might have its own probing period (e.g., the time-to-live- (TTL-) based intervals assumed in various types of DNS cache snooping investigations [19], [24]), or the period might be uniform across all measurement points (as is the case with several ID-based alias resolution studies for creating router-level topology graphs [2], [13], [22]).

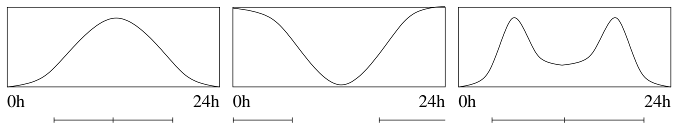


Fig. 1: Hypothetical curves illustrating the volume of DNS requests (for different websites) observed by a DNS resolver. In the cache inspection example, the goal is to probe the resolver about a given website w, but only during peak periods of activity for w.

We also assume that for some of the experiments, there is a notion of an idealized measurement window (i.e., a contiguous window of time) that we call a target's "useful interval", denoted $[s_j, e_j)$. The intuition here is that the fidelity of the experiment may be improved by probing targets during some region of time when we can expect the most utility out of our probes (see Figure 1). In the case of DNS cache inspection, for example, it makes little sense to probe a caching resolver [23] to infer density-based estimates of client populations using the server when most of its clients may be asleep or idle [7]. (We do not adapt these windows dynamically, although doing so would be useful for applications such as bandwidth estimation.) Lastly, we assume that the experiment dictates that a certain number of probes be sent to each target.

For the remainder of the paper, we assume that network traffic follows diurnal patterns—i.e., it is largely similar from day to day, but not from hour to hour, and it exhibits no major differences from week to week. Accordingly, we schedule probes to our targets in real time, using these diurnal traffic patterns. Our solution makes use of a simple and efficient priority queue, described in §III-C.

A. Constraints

Our scheduling algorithm must meet the following constraints:

1) periodic probes must be evenly spaced at intervals of $(\pi_j + \delta)$, for some relatively small δ compared to the periodicity π_j;
2) a predetermined number of probes N_j must be issued to each target by the end of the probing experiment; and
3) no more than L probes per second may be sent.

In what follows, we meet the first two constraints heuristically, but observe the probing rate limit constraint strictly.

B. Goals

Ideally, we would like to probe each unfinished target (a target j to which we have sent fewer than N_j probes) every day at times $\{s_j, s_j{+}\pi_j, s_j{+}2\pi_j, \ldots, \sim e_j\}$ within the target's useful interval $[s_j, e_j)$ until we have sent N_j probes to each target or the time for the experiment is exhausted. This would allow us to achieve perfect periodicity of π_j between our

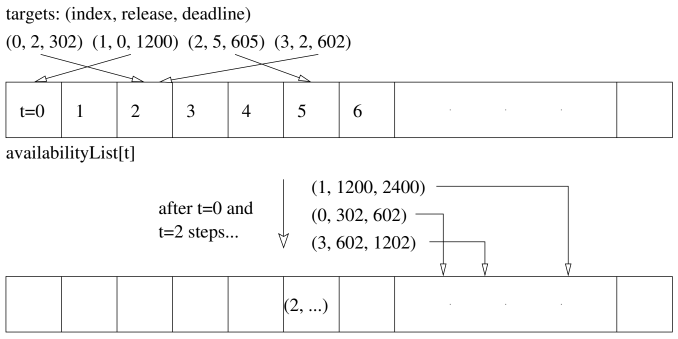


Fig. 2: An individual probe to a target j is initially placed into `availabilityList`[s_j] according to the beginning of its useful interval $[s_j, e_j)$. If the probe is issued, the succeeding probe to target j is placed into `availabilityList`[$s_j + TTL_j$].

probes, and we would conserve probes by only issuing them within $[s_j, e_j)$. Unfortunately, with a large number of targets, there is no guarantee that we can do this without exceeding our rate limit of L probes/sec at some point in time. Since we must observe our probing rate limit (3) strictly, the other two constraints are met only heuristically.

Not every unfinished target will be probed on every day, but if a target j is probed on a given day, we require that it be hit with a sequence of (at most $\lceil \frac{e_j - s_j}{\pi_j} \rceil$) probes within the time interval $[s_j, e_j)$, such that in every fixed (non-sliding) window of duration π_j beginning at time s_j, there will be exactly one probe to the target j, until e_j is reached or a probe is dropped (described below). By the end of the experiment, up to N_j probes will have been issued to each target j, but no more.

C. Earliest-Deadline-First (EDF) Scheduling

We characterize a single probe by the tuple (j, r_j, d_j), where j is the target to be probed, and $[r_j, d_j)$ is the fixed window of duration π_j within which the probe should be issued. Thus, the first probe in a sequence is $(j, s_j, s_j + \pi_j)$, the second $(j, s_j + \pi_j, s_j + 2\pi_j)$, and so on. Recall that we do not enforce the periodicity π_j strictly; in principle, two probes may be as close together as 1 timestep or as far apart as $2\pi_j - 1$ timesteps.

We initialize our algorithm by populating an array called `availabilityList` of length $T = 86400$ (seconds in a day) with the first probes for each target, assigning a probe $(j, r_j = s_j, d_j = s_j + \pi_j)$ to position `availabilityList`[s_j], the time at which the probe will become available for issuing. Starting at time $t = 0$, we push the probes from `availabilityList`[$t \bmod T$] onto a priority queue `Q` ordered by increasing d_j—“earliest deadline first”. In our notation, the time t is definite and monotonically increasing; the interval edges $\{s_j, e_j, r_j, d_j\}$ are all within $[0, T)$ and represent times between 00:00:00 and 23:59:59.

At each timestep t, we pop and send as many probes as possible from `Q` without exceeding our rate limit L. For each probe (j, r_j, d_j) that we send, we place its successor probe $(j, d_j, \max(d_j + \pi_j, e_j))$ into `availabilityList`[d_j] if the target j remains to be probed. We thus aim to issue the successor probe in the π_j-length window immediately following $[r_j, d_j)$. If the first probe (j, r_j, d_j) is issued near the end of the window $[r_j, d_j)$, this means a lot of probing resources are being consumed around time t, and the succeeding probe is likely to be issued near the end of the window $[d_j, d_j + \pi_j)$, so that the probing periodicity π_j is attained heuristically.

The probes left in `Q` are carried over onto timestep $(t + 1)$, and then combined with the probes pushed onto `Q` from `availabilityList`[$t + 1$]. If a probe to j is dropped, i.e., it is popped off of `Q` at a time $t \geq d_j$, this means that the rate limit has been reached for the last π_j timesteps, and remaining probes to j are postponed until t re-enters the useful interval $[s_j, e_j)$ the following time, in order to ease the workload and reduce probe delay. More precisely, if a probe (j, r_j, d_j) is dropped, the probe $(j, s_j, s_j + \pi_j)$ is placed into `availabilityList`[s_j] and reconsidered at time $t \equiv s_j \pmod T$.

Thus, probes may be dropped from the ends of sequences, but never from the middle; we succeed in guaranteeing that there will be exactly one probe sent to j in every fixed window of length π_j from s_j until the end of the probing sequence. We also strictly observe our probing rate limit L (constraint (3) in §III-C). Our heuristic success in evenly spacing our probes at $(\pi_j + \delta)$ intervals (constraint (1)) and sending N_j probes to each target (constraint (2)) is illustrated in §IV.

D. Complexity

The use of a priority queue with amortized constant-time pushes and log-time pops (in the length of the queue) makes our approach scalable: for an experiment of duration D days, our worst-case time complexity is O($J \cdot N \cdot D \cdot \log J$), where J is the number of targets to probe and $N \doteq \max_j N_j$ is the maximum number of probes to send to a target. When scheduling online at each timestep, the time complexity is O($J \log J$). The space requirement of our algorithm is O(J).

IV. Evaluation

We evaluate our approach via a simulation of a week-long measurement experiment that aptly demonstrates a type of Internet-wide study which uses DNS cache inspection [9] to infer demographic information. For example, both Wills *et al.* [24] and Rajab *et al.* [19] used cache inspection techniques to infer the relative popularity of a set W of websites of interest. The basic idea is that the caches of a number of DNS resolvers across the globe are probed at regular intervals, and the responses are used to compute the request rate for each website $w \in W$. Intuitively, DNS entries of websites with higher hit rates will be refreshed more quickly than those with lower hit rates, and so the targets in W can be ranked accordingly. The probe periodicity is determined to match the authoritative TTL of the targeted website w. It is also assumed that the ideal probing interval (the “useful interval”) for a pair (v, w) is the period when the frequency of requests for the website w from clients using resolver v is highest.

In the analysis that follows, we simulate probing 100,000 targets, each with their own useful interval and probing periodicity. Our probing periodicities, then, are target-specific as opposed to timezone-specific ($\pi_{(v,w)} = TTL_w$), as are our useful intervals ($[s_{(v,w)}, e_{(v,w)})$). We set the number of probes required per target to be $N_j = 50$ (for all targets j). The probing limit is set conservatively at $L = 100$ probes/second.

Lastly, client DNS traffic patterns were modeled [12] using a 21-day trace of outgoing `http` requests from a university campus network [5]. For each target, we calculated the mean μ and standard deviation σ of the target's DNS traffic pattern (Figure 1). We evaluated useful interval settings of $[\mu-\sigma, \mu+\sigma)$ (the mean window length being 9h30m, containing 70% of traffic) and $[\mu-2\sigma, \mu+2\sigma)$ (mean windows of 19h, 95% of traffic).[2]

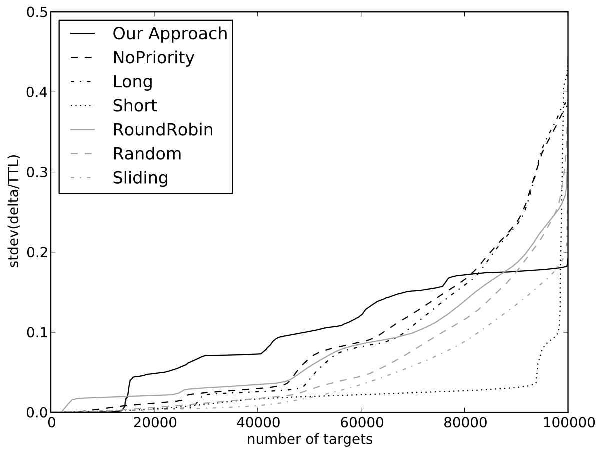
A. Comparative Schemes

To evaluate the effectiveness of our approach, we compare it with six alternative strategies for accomplishing the same goal of issuing 50 probes to each target during the target's useful interval. The strategies follow the same restrictions as we do (i.e., maximum probing rate, number of targets, periodicity, etc.), but are naïve in that they do not take useful intervals or other diurnal traffic patterns into account when scheduling their probes.

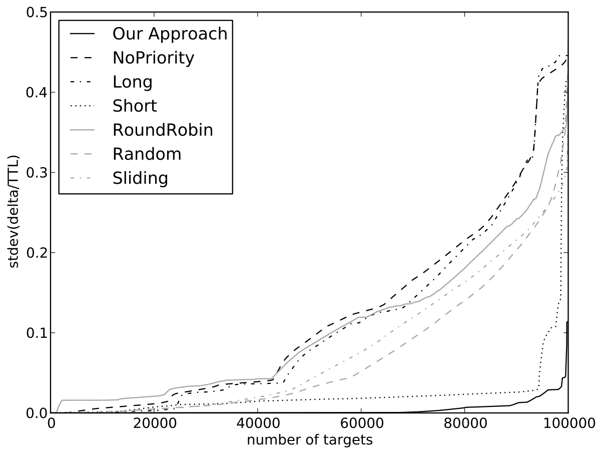
- SHORTESTPERIODSFIRST: prioritize targets with the shortest periodicities. This approach might be useful in the contexts of several measurement studies in the literature [6], [22], to collect as much measurement data as possible early in the experiment.
- LONGESTPERIODSFIRST: prioritize targets with the longest periodicities in order to complete the experiment in as few days as possible. Note that with variable probing periodicities, the overall length required to complete the experiment will be determined by the "long tail" caused by late probing of a few targets with long periodicities. Just like the above approach, shortening this long tail would also be desirable in many studies [6], [22].
- SLIDINGWINDOW: begin probing a random subset of targets on day 1, adding another subset on day 2, and so on until all targets have been added. This strategy is inspired by the approach of Keys *et al.* [13] to allow certain subsets of targets to overlap in probing.
- ROUNDROBIN: prioritize targets that have been probed least, so that each day the minimum fraction of probes sent across all targets is as high as possible.
- RANDOM: prioritize targets according to a predetermined random order.
- NOPRIORITY: use a simple first-in-first-out (FIFO) queue instead of a priority queue. Newly available probes and probes following those issued are placed at the end of the queue, without prioritization.

B. Analysis

Our results suggest that, if a probing rate limit must be observed, our approach will yield more accurate and efficient data collection than naïve probing techniques.



(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m

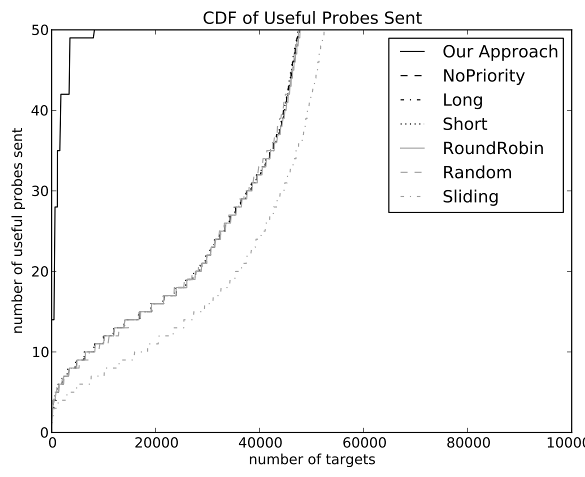


(b) $\mu \pm 2\sigma$, mean useful interval is 19h

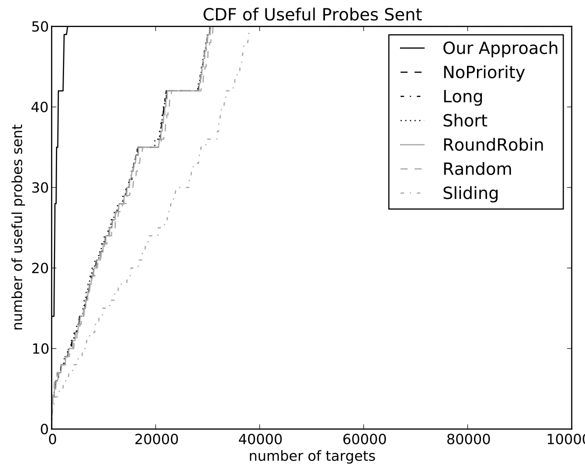
Fig. 3: CDF of the standard deviation of inter-probe delay. Many of the naïve strategies probe their targets at intervals that deviate significantly in length, which would lead to poor measurement results for several applications.

In this simulated probing experiment, probes are issued at intervals of $TTL_j + \delta$, for some small δ. Empirically, the mean of the δ values is always near 0, but we present in Figure 3 a CDF of the deviation in the δ values for each of the 100,000 targets. Figure 3 thus illustrates that our approach issues probes at more regular intervals than the naïve strategies, which will lead to more accurate data collection.

Our strategy also allows us to probe more targets—and issue more probes to those targets we do hit—than the naïve strategies. Figure 4 is a CDF of the total number of probes issued during useful intervals ("useful probes") for each of the targets, as of the end of the experiment. It shows that we are able to completely finish probing 92–97% (all but 8,228–3,070) of our targets by the end of the week, compared to about 50–70% of targets completed by the naïve strategies. Moreover, we manage to send at least 14 of 50 useful probes to all targets, while the naïve strategies send only 3–5 useful

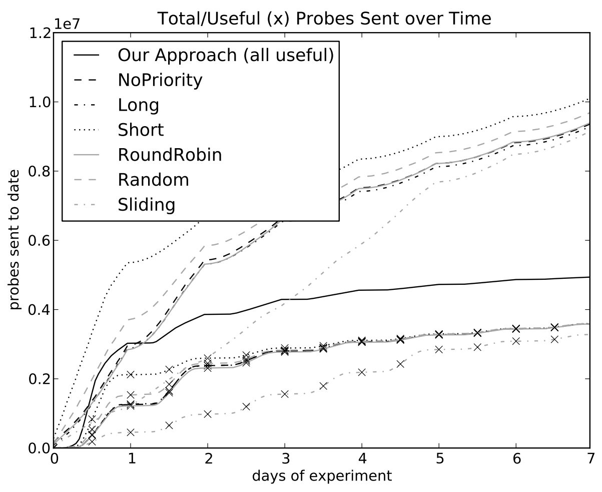


(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m

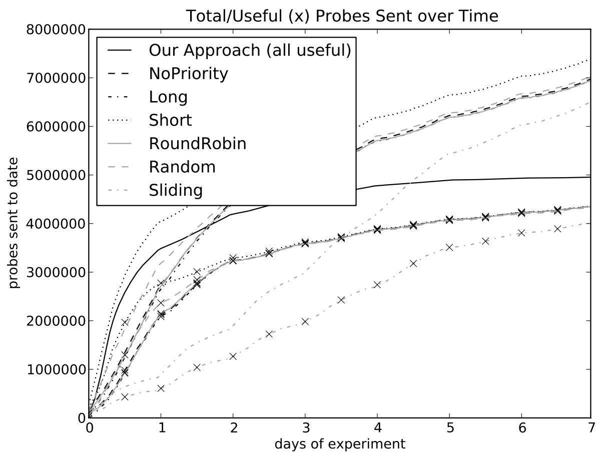


(b) $\mu \pm 2\sigma$, mean useful interval is 19h

Fig. 4: CDF of the number of useful probes sent per strategy. Depending on the useful interval length ((a) vs. (b)), we finish probing 92–97% of our targets, compared to 50–70% of targets finished by the other strategies.



(a) $\mu \pm 1\sigma$, mean useful interval is 9h30m



(b) $\mu \pm 2\sigma$, mean useful interval is 19h

Fig. 5: Total vs useful number of probes sent over time. Total probes are indicated by unmarked lines; useful probes by criss-crossed lines. Notice that we send 4.9m probes, compared to the 7–10m probes issued by other strategies.

probes to all of the targets by the end of the week. As Figure 4b shows, lengthening the useful intervals does not significantly affect these results.

Notice as well that we are able to send more probes during useful intervals while sending fewer probes overall. Figure 5a illustrates our conservative probing behavior. Over the course of the week, we send a total of 4.9m probes, all during useful intervals—compared to the over 9.6–10m probes sent by naïve strategies, only 40% of which can be considered useful.

Two further observations may be made from Figure 3: first, with respect to the periodicity standard deviation being measured, our approach improves when the useful intervals are lengthened, while the naïve strategies perform worse. Second, the naïve strategies of SHORTESTPERIODSFIRST and SLIDINGWINDOW perform relatively well based on their inter-probe delay. For SHORTESTPERIODSFIRST, this performance is likely due to the ease with which the short periodicity jobs interleave with one another, as well as how easily they can be deferred if the rate limit is reached. However, notice that in Figure 5, the SHORTESTPERIODSFIRST strategy also issues the most probes overall. The SLIDINGWINDOW strategy limits inter-probe delay by spreading the workload evenly over the course of the experiment. That said, by naïvely staggering the targets, SLIDINGWINDOW fails to send as many useful probes to each target as the other strategies do (Figure 4).

Our advantage over the other strategies stems largely from the fact that we probe only during the useful intervals and that the intervals themselves are often not closely aligned with one another. For example, targets in various timezones will likely have useful intervals staggered throughout the day, providing a small subset of targets to probe during each hour of the day. Additionally, while we do not assume that probes in some part of a useful interval might be more useful than in other parts of the same interval (i.e., similar to the notion of "weighted tardiness scheduling" [14]), it should be clear that our approach could be augmented to work more effectively in such contexts since, by design, we have the flexibility to constrain the useful intervals as much as is necessary, and we can assign different scheduling priorities to probes depending upon where they fall within the intervals. Finally, our application of the EDF scheduling heuristic improves the precision of our periodic probing more than naïve prioritizing or a FIFO queue could do.

We believe the results show that our scheduling approach can be used to conduct active measurements on the Internet in a more responsible fashion. In particular, we argue that it offers a good solution for experimenters interested in distributing their workloads over time and across limited resources while keeping probing restrictions in mind — yet still maintaining their over-arching goal of finishing their experiments expeditiously.

V. Conclusion and Future Work

In this work we have introduced a technique for scheduling periodic network probes for Internet measurement, towards providing a means for researchers to collect measurement data while probing responsibly. By leveraging diurnal traffic data for our targets, we demonstrated that our technique could be used to collect more useful data (while sending fewer probes overall) than naïve strategies in the context of a simulated 7-day large-scale measurement experiment that observed a probing rate limit at all times.

As part of future work, we are exploring specific enhancements to our framework to accommodate subsets of targets whose probe sequences need to overlap. In doing so, we would extend our approach to be of significant practical benefit to some IP alias resolution applications [2] whose efficient linear probing complexity depends on probe sequences to targets that overlap in time. We are also interested in performing a large-scale empirical evaluation of the real-world benefits of using our scheduling strategy to coordinate probes from different measurement platforms (e.g., Archipelago [10] and FLAME [25]).

VI. Acknowledgements

We thank the anonymous reviewers for their comments and suggestions for improving the paper. This work was supported in part by the National Science Foundation under award number 0831245.

Notes

[1]Our intention is not to call attention to these particular studies, as they are by no means exceptions to the norm; they do, however, illustrate the sheer volume of probes sent in many active Internet measurements today.

[2]Note that due to limited data we estimated one useful interval for each website w, not one interval for each DNS server-website pair (v, w).

References

[1] Amotz Bar-Noy, Vladimir Dreizin, and Boaz Patt-Shamir. Efficient algorithms for periodic scheduling. *Computer Networks*, 45(2):155–173, 2004.

[2] Adam Bender, Rob Sherwood, and Neil Spring. Fixing Ally's growing pains with velocity modeling. In *Proceedings of the* 8th *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 337–342, October 2008.

[3] John Bethencourt, Jason Franklin, and Mary Vernon. Mapping Internet sensors with probe response attacks. In *Proceedings of the* 14th *USENIX Security Symposium*, pages 13–29, 2005.

[4] Prasad Calyam, Chang-Gun Lee, Phani Kumar Arava, and Dima Krymskiy. Enhanced EDF scheduling algorithms for orchestrating network-wide active measurements. In *Proceedings of the* 26th *IEEE International Real-Time Systems Symposium*, pages 123–132, 2005.

[5] Crawdad: A Community Resource for Archiving Wireless Data at Dartmouth. http://crawdad.cs.dartmouth.edu.

[6] David Dagon, Niels Provos, Christopher Lee, and Wenke Lee. Corrupted DNS resolution paths: The rise of a malicious resolution authority. In *Proceedings of the* 15th *Network and Distributed Systems Security Symposium*, 2008.

[7] David Dagon, Cliff Zou, and Wenke Lee. Modeling botnet propagation using time zones. In *Proceedings of the* 13th *Network and Distributed Systems Security Symposium*, February 2006.

[8] M. Fraiwan and G. Manimaran. Scheduling algorithms for conducting conflict-free measurements in overlay networks. *Computer Networks*, 52(15):2819–2830, 2008.

[9] Luis Grangeia. DNS Cache Snooping, or, Snooping the Cache for Fun and Profit, February 2004.

[10] Young Hyun. The Archipelago Measurement Infrastructure. In 7th *CAIDA-WIDE Workshop*, 2006.

[11] Tomas Isdal, Micheal Piatek, Arvind Krishnamurthy, and Thomas Anderson. Leveraging BitTorrent for end host measurements. In *Proceedings of the* 8th *Passive and Active Measurement Conference*, pages 32–41, 2007.

[12] Jaeyeon Jung, Arthur W. Berger, and Hari Balakrishnan. Modeling TTL-based Internet caches. In *Proceedings of IEEE INFOCOM*, pages 417–426, April 2003.

[13] Ken Keys, Young Hyun, and Mathew Luckie. Internet-scale alias resolution with MIDAR. www.caida.org/tools/measurement/midar/, 2010.

[14] Joseph Y-T. Leung. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman Hall/CRC, 2004.

[15] C. L. Liu and James W. Layland. Scheduling algorithms for multi-programming in a hard real-time environment. *Journal of the ACM*, 20:46–61, 1973.

[16] A. Munier. The complexity of a cyclic scheduling problem with identical machines and precedence constraints. *European Journal of Operational Research*, 91(3):471–480, 1996.

[17] Christos Papadopoulos and John Heidemann. Toward best practices for active network measurement. In *Workshop on Active Internet Measurements*, 2009.

[18] Zhen Qin, Roberto Rojas-Cessa, and Nirwan Ansari. Task-execution scheduling schemes for network measurement and monitoring. *Computer Communication*, 33(2):124–135, 2010.

[19] Moheeb Abu Rajab, Fabian Monrose, Andreas Terzis, and Niels Provos. Peeking through the cloud: DNS-based estimation and its applications. In *Proceedings of the* 6th *Conference on Applied Cryptography and Network Security*, pages 21–38, 2008.

[20] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the* 6th *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 41–52, October 2006.

[21] Rob Sherwood and Neil Spring. Touring the Internet in a TCP sidecar. In *Proceedings of the* 6th *ACM SIGCOMM Conference on Internet Measurement*, pages 339–344, 2006.

[22] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[23] Paul Vixie. DNS complexity. *ACM Queue*, 5(3):24–29, April 2007.

[24] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. Inferring relative popularity of Internet applications by actively querying DNS caches. In *Proceedings of the* 3rd *ACM SIGCOMM/USENIX Internet Measurement Conference*, pages 78–90, November 2003.

[25] A. Ziviani, A. T. A. Gomes, M. L. Kirszenblatt, and T. B. Cardozo. FLAME: Flexible lightweight active measurement environment. In *Proceedings of the* 6th *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities*, 2010.