Experiment coordination for large-scale measurement platforms

Mario A. Sanchez[†*] Fabian E. Bustamante[‡] Balachander Krishnamurthy[◦] Walter Willinger[⋄]
[†]HP Labs [‡]Northwestern University [◦]AT&T Labs Research [⋄]Niksun, Inc.

ABSTRACT

The risk of placing an undesired load on networks and networked services through probes originating from measurement platforms has always been present. While several scheduling schemes have been proposed to avoid undue loads or DDoS-like effects from uncontrolled experiments, the motivation scenarios for such schemes have generally been considered "sufficiently unlikely" and safely ignored by most existing measurement platforms. We argue that the growth of large, crowdsourced measurement systems means we cannot ignore this risk any longer.

In this paper we expand on our original lease-based coordination scheme designed for measurement platforms that embrace crowdsourcing as their method-of-choice. We compare it with two alternative strategies currently implemented by some of the existing crowdsourced measurement platforms: centralized rate-limiting and individual rate limiting. Our preliminary results show that our solution outperforms these two naive strategies for coordination according to at least two different intuitive metrics: resource utilization and bound compliance. We find that our scheme efficiently allows the scalable and effective coordination of measurements among potentially thousands of hosts while providing individual clients with enough flexibility to act on their own.

1. INTRODUCTION

The risk of placing an undesired load on the network through probes originating from measurement platforms has always been present. Several scheduling schemes have been previously proposed to avoid undue loads or DDoS-like effects from uncontrolled measurement campaigns. So far, the motivation scenarios for such schemes have generally been considered "sufficiently unlikely".

In practice, most traditional measurement platforms have opted for indirectly managing this risk by limiting the number of probes each individual measurement node can perform (e.g., with local rate limits) or by including an *Acceptable User Policy* appealing to good network etiquette to minimize complaints from network administrators [8].

We argue that the growth of new, crowdsourced, large-scale platforms [4, 11–13] make this approach no longer feasible. The scale and inherent volatility of these platforms mean that these simple approaches will, with time, translate into either valuable resources or overloaded networks. A simple scheme that generates a centralized optimal probing schedule for nodes to follow is not feasible, given the lack of control over clients' availability and will naturally result in an overly conservative use of the platform. Limiting the number of probes and clients assigned to a specific destination is almost guaranteed to yield sub-optimal results, as there's no assurance of when clients will launch the assigned probes (due to limited resources), or even how many of the assigned probes will actually be completed (a client might simply disappear in the middle of an experiment).

We have been exploring an alternative lease-based approach for measurement coordination in the context of our work on Dasu [11]. With our approach, clients are periodically assigned "measurement budgets" through the use of "experiment leases". The budget specifies the maximum number of probes that a client is allowed to launch (on a per-destination basis) before the lease expires. These budgets are dynamically computed for individual clients based on the aggregate behavior of the system.

In this paper, we expand on our original description and evaluate our proposed solution. We compare our approach with two alternative strategies currently implemented by some of the existing crowdsourced measurement platforms: centralized rate-limiting and individual rate limiting. Our analysis illustrate the value of our flexible approach in terms of both resource utilization and bound compliance. We find that our scheme efficiently allows the scalable and effective coordination of measurements among potentially thousands of hosts while providing individual clients with enough flexibility to act on their own.

[*]Written while at Northwestern University

2. PROPOSED SOLUTION

In the following paragraphs, we describe the two constructs employed in our solution —*Experiment Leases* and *Elastic Budgets*— and provide a high-level explanation of our coordination approach. We conduct our analysis in the context of Dasu [11], a software-based measurement platform hosted by voluntary nodes located at the edge of the network and supports both controlled network experimentation and broadband characterization.[1]

Experiment Leases. To support the necessary fine-grained control of resource usage, we use the concept of experiment leases. In general, a *lease* is a contract that gives its holder specified rights over a set of resources for a limited period of time [5]. An *experiment lease* grants to its holder the right to launch a number of measurement probes, using the common infrastructure, from or toward a particular network location. Origin and/or targets for the probes can be specified as IP-prefixes, domain names or website urls; other forms, such as geographic location, could be easily incorporated.

To coordinate the use of resources by measurement clients taking part in an experiment, we rely on a distributed coordination service [6]. This coordination service runs on well-provisioned servers (PlanetLab nodes) using replication for availability and performance. Clients receive the list of coordination servers as part of the experiment description.

Before beginning an experiment, clients contact a coordination server to announce they are joining the experiment and obtain an associated lease. As probes are launched, the clients submit periodic updates to the coordination servers about the destinations being probed. This information is used to compute estimated aggregate load per destination and to update the associated entries in the experiment lease. Before running a measurement, each client checks whether it violates the constraint on the number of probes allowed for the associated destination, and if so, delays it. After a lease expires, the host must request a new lease or extend the previous one before issuing a new measurement. The choice of the lease length presents a trade-off between minimizing overhead on the coordination service versus minimizing client overhead and maximizing the use of clients' resources.

Elastic Budget. An experiment lease grants to its holder the right to launch a number of measurement probes (i.e., *a budget*) from or toward a particular network location. Due to churn and user-generated actions, the number of measurement probes a client can launch before lease expiration (i.e., the fraction of the allocated budget actually used) can vary widely. To account for this, we use the idea of *elastic budgets* that expand and contract based on system dynamics.

Elastic budgets are computed by the coordination service and used to update bounds on experiment leases distributed to clients. The service calculates the elastic budget periodically, based on the current number of clients participating in the experiment and the number of measurement probes allowed, assigned, and completed by each client. The coordination service uses this elastic budget to compute measurement probe budgets for the next lease period for each participating client.

The budget is computed in the following way:

Let,

d,	*destination*
M,	*aggregate max # probes per unit time to dest d*
m,	*max # of probes per unit time a client will launch*
n,	*# of clients in the experiment*
a_i,	*# of probes to dest d assigned to client i*
c_i,	*# of probes to dest d completed by client i*
p_i,	*completion rate of allowed probes in recent past*

Then,

$$Budget = \begin{cases} M/n & \text{if } M/n < ppm \\ ppm & \text{if } M/n > ppm \end{cases}$$

where,

$$ppm = \sum_{i=1}^{n} p_i * f(i)$$

$$f(i) = \begin{cases} a_i - c_i & \text{if } (a_i - c_i) < m \\ m & \text{if } (a_i - c_i) > m \end{cases}$$

This approach is well suited for experiments where the server knows a priori what destinations each client should probe. In the case of experiments where the destinations to be probed are not assigned by the server, but obtained by the clients themselves (through a DNS resolution, for example), the same approach can be used if we conservatively assume that a client will launch the maximum number of probes per unit of time whenever it is online.

3. APPROACH

To evaluate the efficacy of our approach, we run extensive tests using simulated Dasu clients. The experiment consists of a number of Dasu clients probing a common /24 prefix, with an imposed maximum load in the number of aggregate probes per minute allowed towards the destination. The experiments highlight how our coordination scheme behaves in terms of resource utilization, scalability, and bound compliance.

[1]For a more detailed explanation of Dasu's architecture and our coordination approach please refer to [10].

We define two key metrics to measure the efficiency of our proposed solution, Bound compliance and Resource utilization. *Bound compliance* refers to both how quickly the algorithm reacts to violations and what penalty is paid when a violation was reached, i.e., how many excess probes were launched towards the destination during that time period. *Resource utilization* refers to how effectively the solution utilizes existing system resources. Given that Dasu clients are available for limited amounts of time, we want to maximize utilization of their resources. As such we look to see if clients could have launched a larger number of probes while remaining under the aggregate limits imposed by the system.

3.1 Experimental setup

When a number of probes is assigned to a client, it is necessary to consider how long it will take for it to probe those destinations or whether it will probe them at all. The completion time of an experiment can vary sharply between Dasu clients, mainly because the number of probes that a client sends (from those scheduled) depends both on the client's uptime (will the client complete the entire experiment?) and the client's load.

For our experiments, we provision a set of coordination servers that clients periodically contact as experiments progress. To achieve greater flexibility on the parameters we evaluate, we chose to conduct our experiments using simulated Dasu clients. While the communication library used by clients to contact the coordination servers is unmodified, the simulated clients do not actually launch the requested probes. Instead, clients follow a fixed-interval periodic probing schedule that simulates probes and attempts to launch the maximum number of probes possible per unit of time, as permitted by the client's local rate limits. Probe delays are introduced into this periodic probing mechanism by simulated client CPU and bandwidth load.

This setup allows us to independently manipulate the different parameters that impact experiment completion times, as well as isolate their impact on the different components of our coordination solution. The four variable parameters we evaluate are: (a) clients' resource utilization (both in terms of CPU and bandwidth), (b) inter-arrival times of clients joining the experiment, (c) client-server communication delays, as well as (d) clients uptime. We model these parameters based on the population of real Dasu clients. This ensures that the mix of clients in the experiment reflects the correct distribution with respect to real Dasu-client population, and that clients' behavior is based on real Dasu traces.[2]

[2]For a detailed discussion of the distribution of the Dasu-client population please refer to [10]

3.2 Experiment

The server assigns each of the 40 Dasu clients 100 probes to be launched towards a common /24 prefix as soon as possible. The target maximum aggregate probe rate for the destination prefix is set to 100 probes per minute. Depending on the number of clients actively participating in the experiment, the number of probes per client is dynamically adjusted as the experiment progresses, as negotiated through the Experiment Lease and assigned Elastic Budget. Clients probe the destination /24 prefix according to their own local rate-limits and CPU/bandwidth load. To reduce the duration of the experiment simulation, we scale it down by reducing the basic unit time from 1 minute to 10 seconds: the local rate-limiting at the simulated clients will send up to five probes within the 10-second window, instead of the usual 60.

Parameter	**Value**
Lease duration	60 seconds
Budget interval	10 seconds
Report interval	10 seconds
Elasticity	0-0.5-1
Number of clients	40
Number of probes per client	300
Client Inter-arrival Times	0-10 seconds
Communication delay	0-1000 ms
Local client rate limit	5 probes
Bandwidth profile	60% download
CPU profile	10% utilization

As clients join the experiment, the coordination service dynamically adjusts the maximum number of probes the clients are allowed to launch (their individual elastic budgets) based on the current number of clients participating in the experiment, the number of measurement probes allowed, and the number of probes assigned and completed by each client.

4. EVALUATION

In this section we evaluate the impact of individual components on both resource utilization and bound compliance.

4.1 Impact of Elastic Budget limits

We start by looking at how different elastic budget thresholds affect the overall performance of our solution, both in terms of bound compliance and experiment completion time. Recall that elastic budgets expand and contract based on system dynamics and are used to update bounds on experiment leases distributed to Dasu clients by the coordination service. As such, the elasticity of the budget assigned to a client depends in part on its completion rate of allowed probes in the

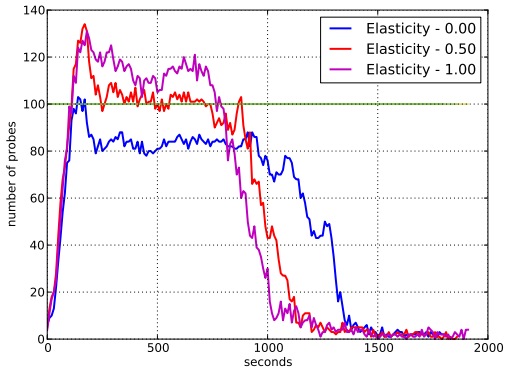


Figure 1: Impact of Elastic Budget limits on bound compliance. Higher elasticity values increase the probability of going over the specified bound.

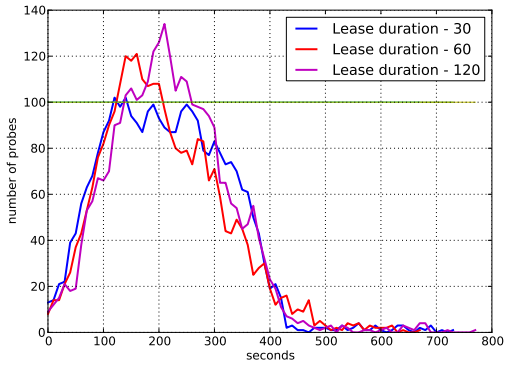


Figure 2: Impact of lease duration on bound compliance. Shorter lease durations translate into tighter control on the client's aggregate behavior and into tighter budget limit compliance.

recent past; i.e. clients that were allowed to send x number of probes in the previous lease period but only sent y probes (where $y < x$) will be allowed some *elasticity* to exceed their allocated limit in the next lease period. This limit is expressed as the fraction of probes over their allocated budget that clients are allowed to exceed.

We evaluate the impact of different elasticity settings by launching the same experiment consecutive times, while allowing the coordination service to assign different elasticity thresholds to the clients. In the tighter scenario –an elasticity of 0.00– clients are never allowed to exceed their allocated budget, even when the number of probes sent in the past was smaller than allowed. Similarly, an elasticity of 1.00 allows the coordination service to assign clients a flexibility of up to 1.00, i.e. twice the amount of probes specified in their budget for the next lease period, depending on their past performance. Note that an elasticity of 1.00 does not mean clients are allowed to exceed the limit all the time. Instead the elasticity is computed based on their performance and can reach up to 1.00 if they sent no probes in the past lease period. For example, a client that was allowed 20 probes in the previous lease period but only managed to launch 15 probes in the allotted time will be allowed an elasticity of 1-15/20 = 0.25, for the next lease period.

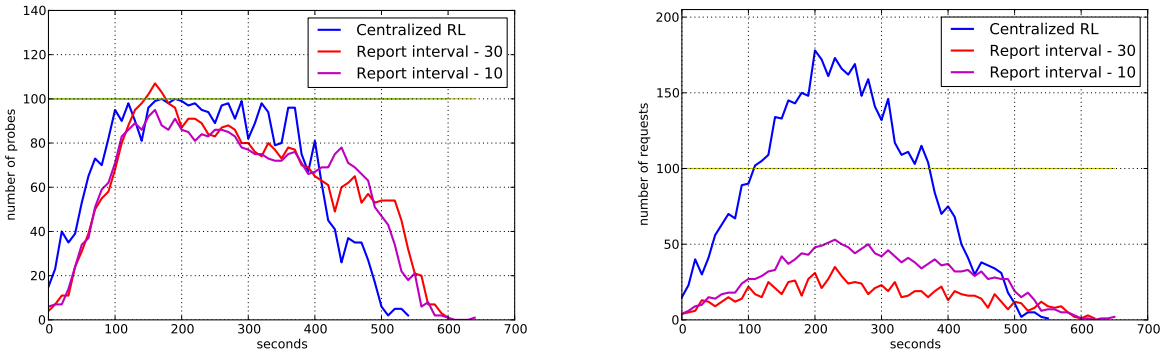
Figure 1 shows the impact of Elastic Budget limits on bound compliance for three different elasticity settings. The figure plots the number of aggregated probes towards the destination (y-axis) against time (x-axis). The figure shows that an elasticity setting of 0.00 allows for tighter control of the aggregate number of probes sent around the specified bound (set to 100 probes). As expected, higher elasticity values increase the probability of going over the specified bound, but only briefly, as the subsequent lease updates drives the probing rate to expected values below the bound. It can be seen that an elasticity value of 1.00 allows for a temporary spike of up to 20% over the specified limit, for a long period of time; the much lower elasticity of 0.5 allows a smaller penalty in terms of number of probes over the limit and the amount of time required to reduce the aggregate probing rate.

4.2 Impact of lease duration

The duration of the Experiment Lease directly impacts how quickly clients are made aware of updated limits from the coordination service based on the aggregate system behavior. Now we turn to the impact of lease duration on bound compliance for three different lease durations and two different elasticity settings. Figure 2 shows the aggregate number of probes from participating clients towards the destination (y-axis), versus the experiment duration time, plotted on the x-axis. For a given elasticity setting, the figures show how shorter lease durations translate into tighter control on the client's aggregate behavior and hence into tighter budget limit compliance. For instance, for an elasticity of 1.00, Fig. 2 shows how a lease duration of 30 seconds ensures the aggregate number of probes from clients never violates the predetermined limit of 100 probes, whereas a longer lease duration of 120 seconds causes the limit to be surpassed by almost 20 percent at its peak. This is one of the expected trade-offs that must be carefully managed when designing different experiments.

5. COMPARING ALTERNATIVE SCHEMES

We now compare the efficiency of our solution to two other alternative coordination schemes: *centralized rate-limiting* and *individual rate limiting*. To compare the three, we look at three different critical metrics: (a) *overhead*: how much control traffic is shared between clients and coordination servers, which effectively impacts how scalable the solution is, (b) *timeliness*: the



(a) Bound Compliance (b) Communication Overhead

Figure 3: Performance comparison between Elastic Budget and Centralized Rate Limiting (CRL) approaches on bound compliance (a) and communication overhead (b).

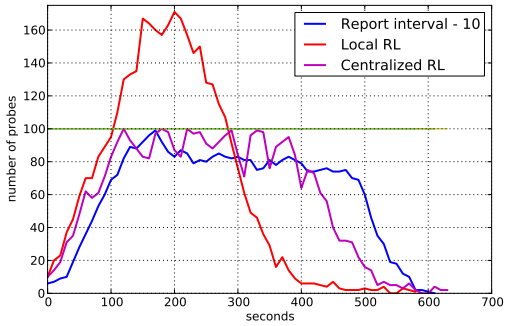


Figure 4: Bound compliance performance comparison between three different approaches: Elastic Budget (blue), Centralized Rate Limiting (purple), and Local Rate Limiting (red).

elapsed time between when the constraint was violated and when it was detected, and (c) *penalty*: how many excess probes were sent to the target when the aggregate constraint was violated.

Figure 3 compares the performance of Centralized Rate Limiting (CRL) against that of the Elastic Budget with zero Elasticity approach, both in terms of bound compliance and communication overhead. For this comparison we set up an experiment involving 50 different Dasu clients with an assigned load of 100 probes each towards a common /24 prefix, and an aggregate maximum limit of 100 probes. In the case of the CRL approach, clients contact a central server before launching any measurement that has been allowed by their local rate limits. Figure 3a shows that, with this approach, the pre-specified limit of 100 probes is never exceeded, providing a more accurate bound compliance than either of the other two alternatives: Elastic Budget with report intervals of 10 seconds (EB10) and 30 seconds (EB30). Given that the centralized server is contacted before clients launch any measurement, the server contains perfect knowledge of the load on the destination; hence the strict bound enforcement is expected.

Figure 3b, on the other hand, compares all three approaches in terms of the aggregate number of communication interactions between coordination servers and the clients. It can be seen that while CRL provides no *penalty* and perfect *timeliness* when it comes to bound compliance (Fig. 3a), it suffers the highest *overhead* of all three options, over 3x higher than the second closest performing approach. Although EB10 takes an extra 30 seconds to complete, it provides similar performance to CRL in terms of *penalty* and *timeliness*, while providing a much reduced communication *overhead*. Finally EB30 provides the smallest communication *overhead* but this at the expense of higher *penalty* and reduced *timeliness*.

Finally, Figure 4 compares these two approaches against simple local rate limiting approach (LRL). Given LRL does not communicate with any coordination servers, we compare the three only in terms of bound compliance. The figure shows that LRL exceeds the predefined limit the moment the combined probing rate of the VPs joining the experiment exceeds this threshold. While this can be minimized by conservatively adding a number of VPs that ensure the threshold can never be exceeded, this will most likely yield unpredictable results given the volatility of the clients.

This comparison shows that our Elastic Budget approach can be a scalable solution to the problem of client coordination for next generation large-scale measurement platforms.

6. RELATED WORK

There exists a rich body of literature on the subject of scheduling active network monitoring activities. Several of these efforts have concentrated on preventing simultaneous scheduling of activities that would interfere with one another leading to inaccurate measurements.

For instance, a scheduling algorithm based on EDF (Earliest Deadline First) was proposed in [3] that provides an offline measurement schedule given a task that potentially involves multiple measurement nodes running numerous concurrent measurement tasks.

More closely related are studies on the development of scheduling algorithms to orchestrate network-wide active measurements [2, 9]. However, most of these presuppose that the destinations to be probed and the availability of the measurement nodes is stable, making them more relevant to campaigns that perform Internet-wide periodic monitoring tasks like those of CAIDA's Ark monitors [1] used to map the Internet topology[3].

More relevant to our work, [7] proposes a scheduling algorithm for probing measurement targets that respects a predefined maximum probing rate. This work focuses on completing probing experiments as quickly as possible while imposing a limit on the probe rate introduced on the network. However, the implementation of such an algorithm in our context would require a centralized entity to assign probes individually to each measurement node one at a time. Aside from the obvious scalability constraints of such an approach, measurement nodes are left with little independence and end up becoming simple measurement extensions of the centralized server.

7. CONCLUSION

In this paper we evaluated a lease-based scheme introduced in previous work [10,11] to control the impact that measurement experiments collectively have on the underlying network and system resources designed for large-scale crowdsource measurement platforms. We compared our solution with two alternative strategies currently implemented by some of the existing crowdsourced measurement platforms: centralized rate-limiting and individual rate limiting. Our preliminary results showed that our solution outperforms these two naive strategies for coordination according to at least two different intuitive metrics. We further found that our scheme efficiently allows the scalable and effective coordination of measurements among potentially thousands of hosts while providing individual clients with enough flexibility to act on their own.

8. REFERENCES

[1] Caida. Ark. `http://www.caida.org/projects/ark/`.

[2] Calyam, P., Kumarasamy, L., and zgner, F. Semantic scheduling of active measurements for meeting network monitoring objectives. In *Proc. of CNSM* (2010), IEEE.

[3] Calyam, P., Lee, C.-G., Arava, P. K., and Krymskiy, D. Enhanced EDF Scheduling Algorithms for Orchestrating Network-Wide Active Measurements. In *Proc. of RTSS* (2005), IEEE Computer Society.

[4] Cappos, J., Beschastnikh, I., Krishnamurthy, A., and Anderson, T. Seattle: a platform for educational cloud computing. In *Proc. of the 40th ACM technical symposium on Computer science education* (2009), SIGCSE '09.

[5] Gray, C. G., and Cheriton, D. R. Leases: An Efficient Fault-Tolerant Mechanism for Distributed File Cache Consistency. In *Proc. ACM SOSP* (1989).

[6] Hunt, P., Konar, M., Junqueira, F. P., and Reed, B. ZooKeeper: wait-free coordination for Internet-scale systems. In *Proc. USENIX ATC* (2010).

[7] Kumar, N. D., Monrose, F., and Reiter, M. K. Towards Optimized Probe Scheduling for Active Measurement Studies. In *Proc. of ICIMP* (2011).

[8] PlanetLab. PlanetLab. `http://www.planet-lab.org/`.

[9] Qin, Z., Rojas-Cessa, R., and Ansari, N. Task-execution scheduling schemes for network measurement and monitoring. *Computer Communications 33* (2010).

[10] Sánchez, M. A., Otto, J. S., Bischof, Z. S., Choffnes, D. R., Bustamante, F. E., Krishnamurthy, B., and Willinger, W.

[11] Sánchez, M. A., Otto, J. S., Bischof, Z. S., Choffnes, D. R., Bustamante, F. E., Krishnamurthy, B., and Willinger, W. Dasu: Pushing experiments to the Internet's edge. In *Proc. of USENIX NSDI* (2013).

[12] Shavitt, Y., and Shir, E. DIMES: Let the Internet measure itself. *SIGCOMM Comput. Commun. Rev. 35*, 5 (October 2005).

[13] Wen, Z., Triukose, S., and Rabinovich, M. Facilitating Focused Internet Measurements. In *Proc. ACM SIGMETRICS* (2007), ACM.

[3] Ark monitors probe IP addresses from every routable IPv4 /24 prefix in cycles of approximately 48 hours.