On Community-Oriented Internet Measurement

Mark Allman[†], Lann Martin[‡], Michael Rabinovich[‡], and Kenneth Atchinson[¶]

[†]*International Computer Science Institute,* [‡]*Case Western Reserve University,*
[¶]*Baldwin-Wallace College*

Abstract. In this paper we describe a new measurement framework that researchers can use to abstract away some of the mundane logistic details that tend to dog every measurement project. The measurement community has outlined the need for better ways to gather assessments from a multitude of vantage points and our system is designed to be an *open community-oriented* response to this desire. While many previous efforts have approached this problem with heavy-weight systems that ultimately fizzle due to logistical issues (*e.g.,* hosts breaking and no money to replace them) we take the opposite approach and attempt to use the lightest possible weight framework that allows researchers to get their work done. In particular, we take the approach of designing a system without any sort of central "core" component and therefore the system has no single point of failure. In addition, our proposed system is *community-oriented* in that there is no central control and we build just enough mechanism for the community to get their work done and police the infrastructure. In addition, our proposed system works in an open fashion such that results from the community's infrastructure are immediately provided to the community through publicly available "live feeds".

1 Introduction

The Internet has become a vastly complex and heterogeneous system that defies simple characterization or measurement. Researchers gain fundamental understanding from detailed measurements spanning a wide variety of vantage points around the network. A thriving sub-community of networking researchers has emerged that focuses on Internet measurement and analysis. Arguably, this sub-community has greatly enhanced global understanding of a wide variety of aspects of how networks work "in the wild" (*e.g.,* operations of the routing system, better understanding of peer-to-peer transfers, how various attacks operate, etc.). With this understanding come new and better techniques for designing and deploying Internet technologies. Our goal is to both enhance this sub-community's ability to provide further understanding, as well as enhance the entire community's ability to assess the efficacy of new ideas through live measurements.

The research community has clearly stated its need for more and better measurement data. An NSF-sponsored workshop on "Community-Oriented Network Measurement Infrastructure" brought together a set of measurement experts who noted a variety of community needs [5]. Among the needs articulated were both the need to more easily run large-scale Internet measurements and the need for datasets from a broad range of networks. In this paper we provide an initial sketch of a system that addresses both of these desires.

Internet measurement studies often fall into one of two camps: (i) those that require researchers to expend large amounts of time on a formidable amount of mundane logistical details in order to run their measurement tools and collect data from a variety of locations and (ii) small-scale studies that only consider small pockets of the network and therefore may not be indicative of large-scale behavior. Our goal is to provide a *community-based* measurement framework to address some of the problems associated with large-scale measurement. We intend to form a *lightweight* measurement platform that maintains *no* dedicated infrastructure. Instead, it relies on using a distributed hash table (DHT) (*e.g.,* OpenDHT [9], an overlay substrate used by a variety of other applications) to handle all communication needs for a mesh of measurement hosts. We will provide tools and libraries to aid in the communication tasks specifically required to undertake Internet measurements (*e.g.,* find measurement points, form measurement requests and collect results). Our over-arching goals are to ease the pain involved with conducting large-scale measurement studies such that researchers can both (a) spend more time focused on gaining insight about the network and how their new technologies work and less time on logistics and (b) have better access to large-scale infrastructure and data such that researcher will be incentivized to move away from small-scale studies. The Internet has benefited from community effort for a number of innovations. Community members proved willing to contribute resources to projects of individual research groups, such as seti@home, traceroute@home and DIMES. We expect that they would be even more willing to take part in an effort that benefits the entire community.

Abstracting the mundane details of measurement away from researchers should not be taken as a small contribution. In fact, our experience is that much of the effort associated with large measurement studies is spent getting the mundane logistics right. While we do not provide a framework to rid researchers of all the logistical headaches of network measurement we provide a framework that takes care of a number of the issues. With this in mind we sketch several aspects of our framework:

– The infrastructure can support a fluid set of measurement points that are provided and administered by the community. Unlike other efforts there is no central management required, making this system a truly community-oriented effort that is *of* and *for* the network research community.
– Our envisioned system uses a general-purpose DHT for the “glue” that (loosely) connects the system components. The lack of a central core is a feature in that no central maintenance is required and no single point of failure exists.
– Researchers will be freed from many, but not all, of the logistic details of recruiting measurement points to focus their attention on the important details of the measurements themselves (techniques, data analysis, etc.).
– The system requires no centralized maintenance beyond keeping the DHT running. We envision that the research community will keep a DHT such as OpenDHT running for a variety of purposes anyway and so using the DHT to coordinate measurements is not an extra burden. If this is not the case and yet the community still desired such a system for measurement purposes a DHT can be readily built from existing and available DHT software.
– Long-running measurements that benefit the entire community can be run with community-wide resources. For instance, a setup similar to CAIDA’s *skitter* sys-

tem [4] could be built and be supported by a distributed set of organizations—none of which control, or can hinder (*e.g.*, due to funding or manpower issues), the overall data collection operation.

- Small and focused sets of measurements can be *easily* taken between a consenting group of measurement points. That is, the group of measurement points used in a particular experiment may be organized specifically for that experiment and not assembled from generic measurement hosts donated to the community. This can help with measurements that are too unknown or specialized to run on shared measurement points (*e.g.*, due to security concerns or because an experiment requires a specialized kernel).
- Since the measurement results are reported through the DHT, anyone can pull down the results of measurements as they are completed. This allows the entire community to benefit from measurements involving the community's shared infrastructure immediately, rather than waiting for the raw data to be posted to some measurement repository and indexed in systems like DatCat [3] or PREDICT [1]. (Note, as discussed in § 3 our system provides immediate but *short-term* storage of results. Therefore, archiving and indexing measurement results in long-term repositories is orthogonal to our framework.)
- The "barrier to entry" for doing large-scale Internet measurement studies is quite high due to the need for a distributed set of measurement points and the time required to coordinate measurements and observations. This shuts many researchers with small-to-modest resources out of the entire area of research (or, relegates them to conducting limited studies, as noted above). Our proposed system will open the field of sound, large-scale Internet measurement to a much broader community of researchers than are currently engaged in this field.
- In addition, having a lightweight measurement infrastructure that can be easily used can encourage researchers who are not engaged in "Internet measurement" per se to both (i) test their ideas out on the real network and (ii) take broad measurements to solidly ground their work in the actual operation of the network.
- While we are proposing an "open" infrastructure the security implications of nefarious use of the platform must be taken into account. We discuss mitigating such concerns in § 3.2.

2 Related Work

Our work is related to two classes of previous efforts: (i) measurement taking infrastructures and (ii) data dissemination systems.

A number of measurement taking infrastructures have been developed, each with their own wrinkle (*e.g.*, NIMI [7], Surveyor [6]). Generally these systems have been more heavyweight than the system we propose. These systems have features that we do not include in our design, such as allowing for the updating of tools, coordinating measurements, stronger and more fine-grained notions of access control, etc. Our system is in some sense on the opposite end of the spectrum—making up for a lack of features by making the *key* tasks as easy as possible. In addition, we note that in many cases these heavyweight all-encompassing measurement infrastructures have ultimately required more upkeep than their designers and operators could handle (*e.g.,* due to the cost of replacing worn out portions of the infrastructure) and so have withered.[1] Again, we take the opposite approach and focus on designing a framework that can be used without any sort of central authority and without relying on any particular organization other than the community at-large to maintain infrastructure. Similarly, the current DipZoom project [8] uses a peer-to-peer approach, but aims to leverage Internet users at large as measurement providers and uses a central core that must be maintained. This approach distributes the cost and effort to maintain the entire system.

Taking measurements is only one part of our system. Since we are using an open DHT for all communication, the results of the measurements can be retrieved directly from the DHT by the community at-large.[2] These "live feeds" of data then benefit the entire community. A number of efforts provide access to archived measurement data (*e.g.,* as indexed in CAIDA's Data Catalog [3]). While our system provides direct access to data without such a catalog, the systems are actually orthogonal. We do not envision keeping measurement results in the DHT indefinitely. Rather, we envision that the data will age out on the order of days after it was produced. Therefore, while the community can latch on to live feeds, longer term archival and indexing systems will still be required.

3 System Architecture

As outlined above, the proposed measurement system is centered around an open distributed hash table such as OpenDHT [9]. Our only requirement for the DHT is that it support a *get()/put()* interface. That is, *put (k,value,t)* places *value* into the DHT under hash key k with a time-to-live of t. Note that multiple values can be placed in the DHT for a given key. The DHT is queried using *get (k)* to retrieve all the values stored under the hash key k.

Fundamentally, there are three types of actors and three operations for a measurement system. The actors consist of (i) measurement requesters who desire some assessment of the network, (ii) measurement points (MPs) that provide certain types of measurements upon request and (iii) so-called "watchers" that do not request measurements, but do track the "live feeds" by retrieving measurement results from measurements scheduled by others. The operations that must be supported are: (a) identifying a remote measurement point suitable to provide the desired measurement, (b) requesting a measurement be conducted by a remote measurement point and (c) retrieving measurement results when available. In the following subsections we discuss in detail how the system works and several additional considerations.

[1] Note that not all infrastructures have met this fate. For instance, the *skitter* infrastructure [4] (and its descendant *archipelago* [2]) has been kept running for close to 10 years (through much hard work).

[2] Clearly, a researcher could encrypt measurement results before placing them in the DHT to prevent community access, but this runs counter to the spirit of the system.

3.1 Tables

Since we employ a DHT to loosely couple all the entities in our system, all communications happen through entries in various tables held in the DHT. The various actors in the system are responsible for inserting new table entries, maintaining existing entries and polling the DHT tables periodically to find new entries. Our system does not call for the explicit removal of items from the DHT, but rather assumes they will be aged out (based on the time-to-live described above.[3] We now discuss the three basic operations provided by the platform.

Identifying Measurement Points. The first key task for researchers wishing to make use of our system is finding the names of the tables to deposit measurement requests into and finding the names of the tables that can be monitored for results. Methods that would accomplish this goal depend on the usage scenarios of our proposed framework. If the experimenter is simply using the framework to interact with a set of well-known nodes that have been constructed for a particular study then the task of finding MPs is unnecessary. However, if a researcher wishes to make use of a set of community-provided MPs (*e.g.,* hosts setup to run *wget* on request) then some discovery process needs to be put in place. Our system contains a master table *AllMPs* that includes information about each measurement provider. At a minimum this master list will indicate the measurement type including version (*traceroute*, *wget*, etc.), acceptable arguments, the name of the DHT key monitored for measurement requests and the name of the DHT key under which results will be deposited. In addition, ancillary information may also be given (*e.g.,* tool version, operating system and version, location of the measurement host, etc.). The entries in the master table are populated and maintained by the MPs as they come online. If an MP becomes inactive, its entry in the master table will age out, so MP failures may only cause some number of measurements requests to go unfulfilled—already possibility due to the best-effort nature of our system. Also note if some host provides multiple measurement types (*e.g., ping* and *pathload*) then it will have multiple entries in the master list. That is, each entry in the list is scoped to one measurement type.

Requesting Measurements. Requesting a measurement involves simply inserting an appropriate record into a table that a given measurement point regularly consults—as determined, for instance, by consulting the master list of measurement providers discussed above. The time-to-live of measurement requests should be fairly short (minutes) since MPs are assumed to be polling for new requests regularly. Each request will give the time the measurement point should run the measurement[4], the arguments to run the tool with and the name of a DHT table to place the results into (in addition to the table where all a given MP's results are deposited).

Reporting Measurement Results. Similar to issuing a measurement request, reporting measurement results involves putting the data into the DHT and then placing at least one pointer to that result into appropriate tables. First, each measurement result is put

[3] Note that OpenDHT has a built-in TTL limit to deal with overly long TTL requests. A TTL limited coupled with a web-of-trust (see § 3.2), mitigates the potential clogging problems that would result from overly long TTL values.

[4] MPs will be expected to be roughly time synchronized (*e.g.,* to within seconds, not minutes). This could be tracked with a heartbeat measurement built on top of the generic platform.

into the DHT under a unique hash key, U. For instance, a Universal Unique Identifier (UUID) could be used for U (as returned by *uuidgen* or similar). Using a unique identifier for each measurement both avoids name clashes between MPs and allows the measurement results to reside in the DHT only once, but be indexed in a variety of ways. The key U will be placed into both the results table given in the measurement request and the results table advertised by the measurement point as the depository for all its results.[5] Additional pointers could be placed in other tables as the measurement point deems appropriate (*e.g.,* a table for all *ping* measurements taken in Europe).

We note that DHTs often have a limitation on the size of each entry. For instance, OpenDHT has a 1024 byte limit on the size of the records that can be placed into the system. Obviously, this may be inadequate for many measurement results and therefore the MPs will have to split the results across a number of entries with the consumers of those results being required to reassemble the pieces. As discussed in § 3.3 we intend to make this process seamless for MPs and measurement consumers by providing fragmentation and reassembly primitives. Therefore, instead of using the DHT's standard *put()* and *get()* functions, alternate forms will be available that abstract away any required fragmentation and reassembly.

3.1.1 Example

We now step through an example usage of our framework. This example is meant to be illustrative and help the reader gain intuition in the system, rather than exhaustively showing all possible behavior and capabilities.

MP Registration. When a measurement point for a particular measurement comes on line it registers four pieces of information in the "AllMPs" master table: (i) the type of measurement and version being provided (*e.g., ping-0.45b*), (ii) the name of the request queue the measurement point services (*e.g.,* "reqQ"), (iii) the name of the list that the measurement point adds results to upon completion (*e.g.,* "respQ") and (iv) other ancillary data that may aid researchers (*e.g.,* location of measurement point, operating system version, etc.).[6] Example:

put ("AllMPs","ping-0.45b reqQ respQ extra_info")

Finding MPs. When a researcher wants to run a particular kind of measurement they can access the "AllMPs" table to obtain a list of the MPs, their capabilities and the tables they use. Example:

get ("AllMPs")

$\Rightarrow$

ping-0.45b reqQ respQ extra

Measurement Request. After a researcher has determined MPs that meet their needs they request a particular MP perform a measurement by adding an entry to the MP's request queue that gives (i) the time the measurement should be undertaken (*e.g.,* 184866301), (ii) the name of a result queue the researcher will monitor for results

[5] Note that there will inevitably be additional details included with the results, such as a checksum of the results, time the measurement was taken, etc. These details are omitted here, where we focus on the high-level design.

[6] Note: We have not yet added structure to this information, but such structure (or partial structure) would likely be needed to make this field useful for automated processing.

(*e.g.,* “MyResults”) and (iii) arguments to the particular measurement tool (*e.g.,* “-n www.icir.org”). Examples:

put (“reqQ”,”184866301 MyResults -n www.icir.org”)
put (“reqQ”,”184866601 MyResults -n www.icir.org”)

Measurement Point Polling. Periodically, the measurement point polls the DHT to retrieve its request queue. If the MP sketched above polled it would find the two measurements inserted into the queue in the last step. Example:

get (“reqQ”)
$\Rightarrow$
1184866301 MyResults -n www.icir.org
1184866601 MyResults -n www.icir.org

Running Measurements. Upon receiving requests the MP schedules and executes the requested measurements. Upon completion assume the results will be held in some local variable **R**. The MP will generate a universally unique identifier as the key under which to place the measurement result (*e.g.,* **U**). After having placed the results in the DHT the MP then places pointers to the results in its own result queue and the queue requested in the researcher’s measurement request. Example:

put (U,R)
put (“respQ”,U)
put (“MyResults”,U)

Researcher Retrieving Results. The researcher who requests some measurements simply polls on the result queue provided in their request to retrieve pointers to measurement results. Following these pointers will then yield the results. Example:

get (“MyResults”)
$\Rightarrow$
U
get (U)
$\Rightarrow$
measurement results (**R**, in this case)

Watcher Retrieving Results. An uninvolved researcher can simply watch results roll into the DHT based on other’s requests. In order to do this the watcher will first have to identify MPs conducting desirable measurements (as shown above in the “Finding MPs” step. From this information the passive observer can then poll on the MP’s response queue for pointers to measurement results as shown in the previous example above (but starting with retrieving the “respQ” table instead of “MyResults”).

3.2 Security

As sketched above, the system has a number of security vulnerabilities. First, a measurement requester can attempt to increase the load on a measurement point simply by requesting large quantities of measurements. Even more problematic is the distributed nature of the system which could allow a requester to coax many MPs to simultaneously send (potentially large volumes of) traffic towards a particular victim. Finally, an attacker could launder requests through the measurement infrastructure in an attempt to gain a layer of anonymity. We offer several approaches to mitigate such problems.

First, we note that measurement requests are just that: *requests*. We make no assumptions that the MPs *must* satisfy all (or any) requests. The requests will receive "best effort"-like service. That is, a measurement point should do its best to conduct the requested measurement at the requested time, but does not make any guarantees. Given this notion, every measurement point can implement local policy related to its willingness to conduct measurements to mitigate some of the security concerns. For instance, a measurement point can both limit the rate of requests that will be serviced (in the aggregate and from a given requester) and can monitor and limit the host and network resources a particular measurement consumes—terminating the measurement if certain thresholds are eclipsed (a la ScriptRoute [10] and DipZoom [8]).

Protecting against nefarious use of a given measurement point is difficult within our framework because we do not have a central authority through which requests can be vetted. For instance, an attacker could coax a large number of measurement points to engage in a DDoS of some service. Or, an attacker could launder their web connections through such a service to add a layer of anonymity. The MPs themselves each only understand a small part of an attack which could look nothing at all like an attack from their viewpoint. Rather than trying to somehow vet all requests that are inserted into the system in a centralized fashion, we again take a community-based approach to the problem and offer two mitigations.

– MPs could inform each other about the measurements they are conducting. For instance, a measurement point executing a measurement towards some target host H could insert that fact into a table in the DHT. Before MPs run measurements they consult this target-based table to assess the load already being placed on the target before deciding whether to execute the given measurement.
– A second mechanism is that we impose the requirement that all entries placed in the DHT be cryptographically signed. We then construct a table in the DHT whereby researchers can recognize each other's cryptographic keys as legitimate (i.e., to build a web-of-trust). MPs can then only act on requests from known well-intentioned researchers. There is a one-time cost in getting on such a list, but the cost is small (getting a small number of colleagues to vouch for you in the system). This web-of-trust provides a reasonable sense of a requester's intention before running a measurement and accountability afterwards.

While the general problem of trusting requesters in an open measurement system is difficult, our intention is to leverage the fact that when scoped to a system by and for researchers the problem becomes tractable to suitably mitigate with simple techniques.

3.3 Primitives

Our system does not attempt to provide a stock measurement system that will satisfy all researchers and all tasks. Rather, researchers will have to integrate new tools and new data collection techniques into the system as they are needed. The following primitives are designed to aid researchers in this integration task by providing high-level abstractions to the low-level details required to interact with the DHT. In addition, we will provide tools for common tasks that use these primitives.

Registration. As noted above, MPs will register and maintain their presence and information about the measurement tools they provide. While the specific contents of the registration will be tool specific, the process will be common across tools.

Removing Duplicates. Since we rely on polling and on entries in the DHT to simply time out there must be a way for actors to discover entries that have already been processed when retrieving a table. For instance, a measurement point would only want to schedule one measurement no matter how many times a given request is retrieved. Also, retrieving a measurement result once is sufficient and just because a result pointer is observed multiple times does not mean the result needs to be fetched multiple times. Our design calls for an abstraction that only exposes previously unseen items.

Assessing Trust. As sketched above, one common task across measurement types will be interacting with a web-of-trust to assess a requester's legitimacy. Primitives to aid in this process will be key to making such a trust model work.

Fragmentation and Reassembly. As discussed above, measurement results may need to be fragmented across a number of DHT entries due to limitations on the size of a single entry (*e.g.,* 1024 bytes in OpenDHT). Therefore, primitives for fragmentation and reassembly will be required such that researchers and developers can be provided with an abstraction that works on entire measurements and are not bothered by the details of how they are placed into the DHT.

Miscellaneous Tasks. There are important common measurement-oriented primitives that will aid researchers in setting up their measurements. For instance, a common task is to derive a measurement schedule, and primitives for this task that will allow for direct use with the overall measurement framework will be crucial. Another example is for a measurement point to implement an event loop that polls the DHT for needed information and executes measurements at the appointed times. Having a primitive for easily constructing such an event loop will inevitably aid those integrating new measurement tools into the system.

The above primitives are designed to work across a variety of measurement applications. We note, however, that the above list is likely incomplete. As we progress beyond our proof-of-concept implementation (see § 4) we will likely find additional primitives that are broadly useful and we will include these in the released toolkit.

3.4 Passive Measurements

We note that our discussion above is in terms of active measurements. However, passive monitors can also be used in our system. We envision these manifesting themselves in two forms: by-request monitoring and continuous monitoring. In the first category a researcher can place a request into an appropriate DHT table to have a measurement point monitor some facet of the network for some prescribed amount of time. For example, an MP can register as being capable of monitoring a local Web site, and a request could be to watch a web log for the next 10 minutes. The latter category allows for passive monitors to simply run continuously and dump their results into the DHT for public consumption via live feeds (*e.g.,* a distributed dark address space monitor that provides a wide view of malicious activity). As with sharing of any passive measurements the provider may wish to apply anonymization and sanitization policies to the data before release. For instance, a passive monitor might provide the length (in bytes or seconds) of each TCP connection without providing the IP addresses (even in anonymized form).

4 Summary

Because of the established difficulties in maintaining coherent measurement infrastructures, we propose to build a measurement platform with *no* dedicated infrastructure at all. Instead, we utilize an *existing* overlay substrate *already maintained* for a variety of other purposes, and we concentrate all functionality that is specific to our platform in the end hosts. We further make end hosts totally autonomous and loosely connected to the platform: they can join and depart at will without any reconfiguration in the rest of the platform. This allows the platform to grow and shrink naturally with the needs of the community and be resilient to failures (either technical or logistical). It is important to note that we do not tackle all the hard problems associated with measurement, but rather provide a reasonable platform as a basis.

We have built a small prototype of our system that includes a generic client and an MP that provides *traceroute* measurements on request. While modest, this small prototype is aiding us as we flesh out the details of the system as we work towards providing a toolkit for the broader community.

Acknowledgments

We thank Ethan Blanton, Josh Blanton and Yaohan Chen for discussions of the system described in this paper. Vern Paxson and the anonymous reviewers provided valuable suggestions on a draft of this paper. This work was sponsored by NSF grants ITR/ANI-0205519, NSF-0722035 and NSF/CNS-0721890 for which we are grateful.

References

1. PREDICT: Protected Repository for the Defense of Infrastructure Against Cyber Threats. http://www.predict.org.
2. CAIDA. Archipelago measurement infrastructure. http://www.caida.org/projects/ark/.
3. CAIDA. Internet Measurement Data Catalog. http://www.datcat.org.
4. CAIDA. Skitter. http://www.caida.org/tools/measurments/skitter/.
5. k. claffy, M. Crovella, T. Friedman, C. Shannon, and N. Spring. Community-Oriented Network Measurement Infrastructure (CONMI) Workshop Report. *ACM Computer Communication Review*, 36(2):41–48, Apr. 2006.
6. S. Kalidindi and M. J. Zekauskas. Surveyor: An infrastructure for internet performance measurements. In *INET'99*, 1999.
7. V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An architecture for large-scale internet measurements. *IEEE Communications*, 36(8):48–54, 1998.
8. M. Rabinovich, S. Triukose, Z. Wen, and L. Wang. Dipzoom: the internet measurements marketplace. In *9th IEEE Global Internet Symp.*, 2006.
9. S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *SIGCOMM*, 2005.
10. N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A public internet measurement facility. In *Usenix Symp. on Internet Technologies and Systems*, 2003.