Towards a High Quality Path-oriented Network Measurement and Storage System

David Johnson Daniel Gebhardt Jay Lepreau

University of Utah, School of Computing

Abstract. Researchers need current and historical measurements of Internet paths. We built and deployed a complete system designed to fill these needs: a safe, shareable, multi-user active network measurement system probes network paths and reliably records measurements in a storage facility with multiple levels of caching, providing users with fast, flexible querying. Our system, deployed on PlanetLab for over 20 months, has accumulated 940 million measurements and made them publicly available in a separate, federated data repository. Our experience shows that building and running such a valuable research tool poses significant engineering and practical challenges.

1 Introduction

For a multitude of reasons, researchers need current and historical measurements of Internet paths. These reasons include creating or validating network models, using those models to perform experiments under Internet conditions in network testbeds, studying trends and stationarity in network conditions, and selecting Internet paths that tend to exhibit certain properties. We explored one design point on the spectrum of path-oriented network measurement and storage systems, motivated by 1) the needs of a network emulation environment and 2) the need to provide permanent, public repositories of historical measurement data. The result is Flexmon: a shareable, multi-user active measurement system that collects pairwise path data between sites in a network at tunable frequencies, yet protects the network from excess traffic. Its architecture allows multiple clients to schedule their own probes on subsets of nodes in the network, while sharing probe results among clients, amortizing the costs. Furthermore, Flexmon provides a reliable storage path for probe data and stores them permanently in a separate federated data repository.

Flexmon's design provides a measurement infrastructure that is *shared, reliable, safe, adaptive, controllable,* and *accommodates high performance data retrieval.* Each feature is not novel, but the design, the lessons learned, and its eventual more mature implementation should provide an important community resource—and indeed, our accumulated measurements are already publicly available. Flexmon has some features in common with other measurement systems such as S^3 [16] and Scriptroute [13], but is designed to support different goals including shared control over measurements and easy public data availability.

Flexmon measurement is controllable and dynamic in that an authenticated user may adjust the measurement frequency of any particular path, while Flexmon itself caps and adjusts the rates based on overall network resources consumed. These features allows higher frequency measurement than what would be possible globally, and thus a more accurate path emulation, while still preserving its safety to large networks like PlanetLab.

Our experience shows that building and running such a system poses major engineering and practical challenges, some of which Flexmon does not yet meet. We have been running Flexmon on the PlanetLab network testbed for over twenty months, accumulating 940 million publicly available measurements between PlanetLab sites. In this paper, we describe the Flexmon architecture and implementation, discuss its reliability, outline our dataset, and draw lessons for developing high quality network measurement and storage systems.

2 Design

In this section, we present Flexmon's architecture and discuss our design choices. We originally built Flexmon to serve as the measurement infrastructure for Flexlab [10]. Flexlab allows researchers to experiment in the controllable, repeatable environment provided by the Emulab [15] testbed, but using link characteristics that are dynamically updated using traffic models derived from a real network, specifically the PlanetLab [8] testbed embedded in the Internet. Consequently, Flexmon inherits several design goals from Flexlab.

2.1 Design Choices

Flexmon's design was driven by four primary requirements: the need to obtain and provide "raw measurements" to researchers, the need to operate within unreliable networks, the desire to obtain data that are useful for many researchers, and the need to protect the network from excess traffic.

First, Flexmon is designed to measure end-to-end network path properties, using (typically standard) external programs wrapped by a script that canonicalizes the parameters. Our initial experiments with packet-pair and packet-train tools, including pathload [3] and pathchirp [9], produced poor results on PlanetLab due to its extremely overloaded hosts. Consequently, we currently use two simple, controllable tools: a modified version of `iperf` measures bandwidth, and `fping` measures connectivity and latency. However, this experience reveals the importance of allowing multiple measurement tools. Flexmon's design supports the addition of new measurement tools, as they prove desirable in the future.

The design also supports two methods for running these tools, either *one-shot* or *continuous*. In one-shot mode, the tool is spawned at regular intervals to produce a single measurement that is captured by the system. This mode makes it easy to integrate existing probe tools such as ping and iperf that can produce a "summary" result before exiting. However, this mode can cause high overhead when the probe frequency is high. In continuous mode, the tool is spawned once, for a longer duration, and its periodic results are collected by the path prober. Many probes can benefit from collecting state over an extended period, while still reporting periodic results.

Our system must function over unreliable networks. PlanetLab is a heavily utilized, multi-user overlay testbed with hundreds of nodes distributed around the world. Thus, like the Internet it is a part of, PlanetLab nodes are often unresponsive. To be useful such an environment, Flexmon must be reliable in a number of key ways. When nodes return from an unresponsive state, they must quickly rejoin the measurement network and continue probing. During a network outage, Flexmon nodes must continue probing and reliably store results to persistent storage for delivery when connectivity returns.

Potential users of Flexmon will likely often require the same types of measurements. However, frequent or simultaneous probing with tools such as `iperf` can be both costly and cause self-interference, affecting accuracy. Thus, it is important for Flexmon to allow users to share probe results. In Flexmon, probing costs can be amortized across several users, which can alleviate the larger bandwidth costs incurred by `iperf` and similar network measurement tools.

Finally, both the underlying network and the measurement system itself must be protected from excess measurement traffic. Given our trust model, such traffic is typically caused accidentally, but our experience shows that it is a real problem. Our design can limit global and per-user bandwidth usage, providing both defense in depth and reflecting the reality of traffic limiting on PlanetLab.

2.2 Software Architecture

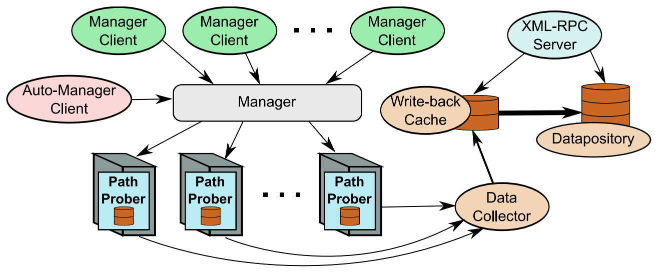


Fig. 1. Flexmon architecture.

Flexmon consists of six types of components: *path probers*, a *manager*, *manager clients*, the *auto-manager client*, a *data collection* subsystem, and a federated, permanent *data repository*. Two additional components from the base Emulab system are essential: a reliable but lightweight control system, and a database that includes static and dynamic state of the measurement nodes. Figure 1 shows an overview of the communication between these components.

Flexmon users request that measurement probes be performed among sets of nodes using manager clients. They indicate the nodes to be probed, the type, frequency, and duration of measurements, and any other parameters required by the tool. A special type of manager client, the auto-manager client, attempts to maintain a fully connected measurement set for the entire network. The centralized manager receives client probe requests, performs safety checks on them, and forwards the requests to path probers.

Path probers run on each network node and receive control commands from the manager. Commands mirror the client requests, allowing changing of the measurement destination nodes, the type of measurement (e.g., latency), and the measurement frequency and duration.

Path probers send results to a data collection service that caches and batches before sending them on to the permanent federated repository, the Datapository [1]. To speed up both queries and updates, the data collector places the results in a smaller database, which functions as a write-back cache of the measurements taken in the last 24 hours. New data in the cache is flushed hourly to the `Datapository` database. Finally, an XML-RPC server provides publicly available query functionality to users, allowing queries to its pairwise internal cache, the 24 hour cache, and the `Datapository` database.

We anticipate that researchers will typically use manager clients to schedule probes at higher frequencies between nodes of specific interest, while the auto-manager client maintains low-frequency, low-cost, network-wide measurements. As an example, the Flexlab [10] testbed uses a manager client to run a high-fidelity path measurement between PlanetLab nodes to determine the initial conditions for link shaping within an emulated network.

Multiple manager clients may send requests specifying different measurement frequencies for the same path. The path prober maintains a queue of probe requests ordered by frequency, and serially executes the highest frequency probe request per type (i.e., available bandwidth estimation, latency, connectivity). Once the duration of that request expires, it is removed from the queue, and the prober executes the next probe. Since the prober runs only the highest frequency probe request per type, all users of the system will see at least the rate of measurements that they requested.

Flexmon can reject probe requests based on resource constraints in the network. For instance, network administrators for probed nodes may request that Flexmon limit its bandwidth consumption to a specific rate. Flexmon maintains a per-path available bandwidth average based on prior available bandwidth probes, and performs admission control on probe requests by computing the expected bandwidth utilization of the probe. If the expectation is that the request will exceed a global limit, or a per-user limit, the probe request would be rejected.

3 Implementation

Flexmon currently serves as the measurement infrastructure for Flexlab, a service running on Emulab. Many of its central components run on Emulab servers, and it measures a subset of the PlanetLab network. In this section, we provide background on how Flexmon is deployed across Emulab and PlanetLab, and discuss implementation details for key system components.

Deployment. Emulab provides a portal [14] to PlanetLab, through which PlanetLab resources can be used in much the same way as other Emulab resources. To deploy Flexmon's path probers on PlanetLab nodes, we create an Emulab experiment containing all PlanetLab nodes considered "live" by Emulab. When this experiment swaps in, Emulab automatically deploys the measurement scripts and tools and starts the node's path prober daemon. Although the number varies over time, Flexmon typically monitors 275–350 PlanetLab nodes .[1]

Background Measurements. From the set of PlanetLab nodes in the Flexmon experiment, the auto-manager client chooses a subset so that each node represents a unique PlanetLab site. The auto-manager client requests all site-pairs latency, connectivity, and bandwidth probes to run with infinite duration between all nodes in this set. The auto-manager client chooses a single node per site because all nodes at one site should exhibit similar path characteristics to nodes at another site. The auto-manager client prioritizes site nodes based on least CPU utilization to minimize the effect of observed latencies in process scheduling on PlanetLab nodes [10, 12], and updates priorities as loads change.

The auto-manager client parameterizes its latency probing by period (each path is measured after an interval), and bandwidth probing by duty cycle (the fraction of time a path prober is measuring bandwidth to any destination). All-sites latency probing is inexpensive when compared to all-sites bandwidth probing, even in a large-scale network such as PlanetLab. However, all-sites bandwidth probing can become extremely costly and could cause Flexmon to run afoul of PlanetLab bandwidth caps. Although Flexmon allows its administrator to set global and per-user caps , it remains important for background probing to leave space under the cap for high frequency, per-user manager client probing. We have found that by setting the latency probing period to 600 seconds, and the bandwidth probing duty cycle to 10%, auto-manager client probes do not exceed PlanetLab caps, and leaves sufficient resources for manager clients.

Probing. A path prober runs on each node in the measurement network. Each path prober receives commands from the manager that control probe execution. Commands result from probe requests to the manager, and set the destination nodes, the probe types to be executed (e.g., latency or bandwidth) between the source and each destination node, the mode in which the probe should run (one-shot or continuous), the period between individual tests, the total duration of probing, and any additional arguments. The path prober spawns wrapper scripts that translate the canonicalized parameters to tool-specific arguments, runs the tool, and converts its results and error conditions into a generic form. For one-shot probes, a result message is sent to the data collector once

[1] Regular PlanetLab users will note that this number is lower than the number of nodes typically reported as live by CoMon [6]. Emulab's portal to PlanetLab creates a richer environment on nodes than the default PlanetLab sliver creation method, and thus requires a higher level of node health.

the probe has finished execution. For continuous probes, the path prober monitors the output of the probe and sends periodic messages to the data collector. Messages contains the node pair, probe type, the probe result (including error information), the time at the start of probe execution, and a magic protocol ID.

When deciding how to estimate available bandwidth, we first experimented with several packet-pair and packet-train tools, including `pathload` [3] and `pathchirp` [9]. Others report those two programs to work acceptably well on PlanetLab [5], but in our experience they often returned extremely unreliable results, or none. Therefore, we estimate available bandwidth with `iperf`, a so-called Bulk Transfer Capacity method. `iperf` consumes much bandwidth during tests, but it has the advantage that, by using a real TCP flow, it obtains a highly accurate measurement of the bandwidth seen by a TCP flow, including taking into account the reactivity of other flows. We extended `iperf` with our own iperf daemon, since iperf produced memory leaks during long runtimes. Each path prober runs an iperf daemon to handle incoming probes from other nodes.

Flexmon uses the `fping` utility to measure latency and detect path outages. When a loss occurs, a state machine drives a frequency-adaptive probing process to distinguish packet loss from true connectivity failure (in four seconds), and to subsequently detect connectivity restoration (within ten seconds).

Probing tools may experience errors due to underlying network behavior. Path probers capture certain errors and report them as anomalous measurements since errors provide users with useful information about the state of the network. We currently capture timeout, unresolvable hostname, and host unreachable errors. However, our system can flexibly record arbitrary error conditions.

Flexmon is designed to run safely on unreliable networks. Since PlanetLab is a large, heavily-utilized network, it is inevitable that nodes will periodically reboot or become unresponsive for extended periods of time. If a PlanetLab node is rebooted, or the sliver is reset, the path prober restarts when the sliver does. However, we chose not to have the path prober checkpoint the current running state and resume from it during failure recovery. Each path prober has no knowledge of the overall system goals, and the manager or auto-manager client may have already adapted to deal with the loss of particular nodes.

Measurement Transfer and Storage. Flexmon's reliability and availability are greatly improved by strategic buffering, reliable probe result transfer, and caching. Each path prober sends probe result to the data collector over UDP. Before a prober sends a result, it first inserts it into a Berkeley database, which acts as a stable storage buffer. The prober then sends the message to the data collector and retransmits after five seconds if the result message is not acknowledged by the data collector. The data collector does not acknowledge the result message until it has been successfully inserted into a SQL database; therefore, it must be aware of duplicate measurements, and will drop them when they are detected. Through this mechanism, Flexmon largely ensures application reliability; however, measurements may be lost if a path prober node's disk fails while the data collector is not running.

The data collector maintains the caching database containing measurements taken within the last 24 hours, making result polling and queries on recent data much faster than if all results were inserted directly into the `Datapository`. A script reliably "flushes" measurements in this cache back to the `Datapository` each hour, ensuring that measurements older than 24 hours are not aged out of the cache until they are successfully entered into the `Datapository`.

Query Interfaces. To facilitate efficient and easy data access, while minimizing security concerns and increasing functionality, Flexmon provides several interfaces to query measurement result data. First, as mentioned in the previous section, Flexmon maintains the results from the previous 24 hours in a database that acts as a write-back cache. This database, into which the data collector inserts new measurements, provides a reasonably efficient query mechanism for services that require access only to recent data. At this time, the `users` database is accessible by any researcher with an Emulab account. However, due to security and performance risks, we only provide raw SQL access to the `Datapository` database to those "power users" who must be able to compose their own queries.

Instead of providing raw SQL access to the Flexmon databases to all potential users, we provide a safer, controlled means of accessing measurement data. Flexmon runs a simple XML-RPC server that periodically polls the `users` database and keeps an in-memory cache of the single most recent latency and bandwidth measurements for each known path. The XML-RPC server ages measurements out of its internal cache once they reach an age of 24 hours. It also provides a simple API that can be used to query either the `users` database or the `Datapository` database, depending on the specified time interval . The `getMeasurements` method requires an interval to restrict query responses to a manageable size, and can filter results based on source and destination node and site, as well as on basic measurement value constraints. Another method, `getFullyConnectedSet`, finds a max-clique from the data in the in-memory cache, and can restrict its search to a specific set of nodes.

4 System Status

Flexmon has been monitoring sets of PlanetLab nodes since February 2006, and has placed approximately 940 million measurements of pairwise latency and available bandwidth in the `Datapository`, accounting for approximately 89% and 11% of total measurements respectively. Until December 2006, Flexmon ran in "beta" mode, and underwent several architectural changes. After this time, although several bugs were fixed, the system remained largely unchanged, aside from occasional changes in the set of PlanetLab nodes monitored.

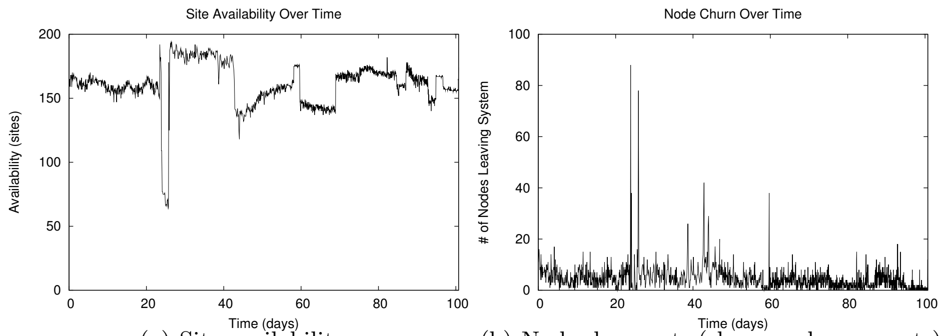
The number of nodes in the experiment can change over time. When we change the experiment configuration, we normally restart key Flexmon daemons, such as the auto-manager client and the manager. Depending on when restarts occur, there may be slight anomalies for a brief window of time in the measurement archive. Prior to fixing a bug, many of our PlanetLab nodes were given bandwidth caps by PlanetLab for a short time due to detected overuse.

Since Flexmon path probers may not always be able to conduct a pairwise site measurement successfully (i.e., due to unexpected node unavailability, path outages, packet filtering at PlanetLab sites), measurement results stored in the `Datapository` also include pairwise errors. For instance, over the measurement history, approximately 17% of latency measurements and 11% of bandwidth measurements represent errors. Of known latency errors, approximately 74% are timeouts; 18% are DNS lookup failures; and 6% are ICMP unreachable errors.

5 Metrics

We analyzed a snapshot of logfiles produced by Flexmon during a 100-day period to evaluate key properties of our system. There are times when the number of available sites is much lower due to the inherent unreliability of distributed systems and PlanetLab. The number of nodes in the experiment changes over time. During the period analyzed, we monitored a median of 151 sites.

We first evaluate the number of available sites in our system during the period. We extract the number of available sites from periodic liveness checks performed by the auto-manager client. Figure 2(a) shows the available *sites* over the period. Near Day 23, there is a sharp drop in the available sites, caused by an experiment restart. Overall, this graph demonstrates a relatively stable number of available sites.



(a) Site availability. (b) Node churn rate (shown as leave events).

Fig. 2. Measurement node availability and churn rate.

We also analyzed the churn rate of nodes in our system. We compute the churn rate as the number of nodes that "left" (were newly unresponsive) in a single period of liveness checks. Figure 2(b) shows the churn rate over the period. First, we see large spikes near Day 23 that are caused by the experiment restart, where a large number of nodes suddenly left the system. These correlate well with the decreased site availability shown in Figure 2(a), as do other spikes in this graph, which may represent transient failures.

6 Related Work

Researchers have built a number of network measurement infrastructures, each with its unique spin on the measurement task.

Scriptroute [13] provides a safe, publicly-available network probe execution environment. Users submit probe scripts written in an interpreted language to Scriptroute servers, which execute the scripts while ensuring that scripts do not exceed resource limits and do not send malformed packets. Flexmon also constrains probes according to resource limits. However, since Flexmon prevents users from running arbitrary tools or algorithms, and only allows probes between nodes within the monitored network, a malformed packet filtering mechanism such as that provided by Scriptroute is unnecessary. Due to its more permissive trust model, Scriptroute does not allow node-local data storage, which compromises reliability. It is also not linked to a public data repository.

`ANEMOS` [2] is an extensible measurement system which stores results in a central database. However, task scheduling is done on the centralized *Coordinator* rather than in a distributed fashion, and results are not buffered at the *Workers*. These two traits limit its usefulness on an unreliable network. While Flexmon demonstrates scalability at least to hundreds of nodes, `ANEMOS` has only been tested to tens. The work in `ATMEN` [4] describes an open framework for providing network measurement services and querying the results. Its distributed architecture provides scalability for measurement, but does not provide functionality for a general user or application to access the entire history of all collected data.

`NIMI` [7] is a well-known measurement framework for measuring Internet traffic, with a trust model that is more restricted than Flexmon's. It is secure, scalable, and extensible, but lacks a central result repository.

TCP Sidecar [11] is the foundation of sideping, a tool designed to passively estimate RTTs. Sidecar snoops on sent TCP packets and retransmits them, with subtle changes. When Sidecar receives a duplicate ACK from the remote host, sideping can estimate the RTT. Flexmon differs from sideping since it is designed to service probe requests from its users with the given frequency and duration arguments, between specific nodes, so it cannot passively wait for a TCP connection to occur between the target nodes.

7 Conclusion

We have described Flexmon, a shareable, path-oriented, active network measurement system with reliable measurement storage. Our experience shows that building such a system poses significant engineering and practical challenges. It should ensure reliable measurement storage, avoid overloading monitored nodes and networks, and function in unreliable networks with heavily loaded nodes.

It will require a major effort to provide such a system that is truly reliable, safe, and efficient, with high availability and high performance. The level of effort is probably similar to the effort we ourselves expended in building the initial version of the Emulab network emulation testbed. However, Flexmon is a real, working system that has collected 940 million measurements, it has been successfully used by the Flexlab network testbed, and we plan to evolve it to a permanent and production-quality measurement and storage system.

Acknowledgments

We thank many of our colleagues in the Flux Research Group for their significant contributions. Robert Ricci, Mike Hibler, Leigh Stoller, and Sachin Goyal added functionality to Flexmon. Robert, Sachin, and Kirk Webb also helped operate the deployed system. Pramod Sanaga and Kevin Atkinson assisted in gathering performance numbers. Eric Eide provided feedback and editing support. We thank David Andersen for his work on the Datapository.

References

[1] D. G. Andersen and N. Feamster. Challenges and Opportunities in Internet Data Mining. Technical Report CMU–PDL–06–102, CMU Parallel Data Laboratory, Jan. 2006. `http://www.datapository.net/`.

[2] A. Danalis and C. Dovrolis. ANEMOS: An Autonomous Network Monitoring System. In *Proc. PAM*, San Diego, CA, Apr. 2003.

[3] M. Jain and C. Dovrolis. End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput. *IEEE/ACM Trans. Networking*, 11(4):537–549, Aug. 2003.

[4] B. Krishnamurthy, H. V. Madhyastha, and O. Spatscheck. ATMEN: A Triggered Network Measurement Infrastructure. In *Proc. WWW*, May 2005.

[5] S.-J. Lee et al. Measuring Bandwidth Between PlanetLab Nodes. In *Proc. PAM*, Mar.–Apr. 2005.

[6] K. Park and V. Pai. CoMon: A Mostly-Scalable Monitoring System for PlanetLab. *OSR*, 40(1):65–74, Jan. 2006.

[7] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis. An Architecture for Large-Scale Internet Measurement. *IEEE Comm.*, 36(8):48–54, Aug. 1998.

[8] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. HotNets-I*, Oct. 2002.

[9] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell. pathChirp: Efficient Available Bandwidth Estimation for Network Paths. In *Proc. PAM*, Apr. 2003.

[10] R. Ricci et al. The Flexlab Approach to Realistic Evaluation of Networked Systems. In *Proc. NSDI*, Apr. 2007.

[11] R. Sherwood and N. Spring. A Platform for Unobtrusive Measurements on PlanetLab. In *Proc. of WORLDS'06*, Seattle, WA, Nov. 2006.

[12] J. Sommers and P. Barford. An Active Measurement System for Shared Environments. In *Proc. IMC*, Oct. 2007.

[13] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A Public Internet Measurement Facility. In *Proc. USITS*, Mar. 2003.

[14] K. Webb et al. Implementing the Emulab-PlanetLab Portal: Experience and Lessons Learned. In *Proc. WORLDS*, Dec. 2004.

[15] B. White et al. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. OSDI*, Dec. 2002.

[16] P. Yalagandula et al. S3: A Scalable Sensing Service for Monitoring Large Networked Systems. In *Proc. Workshop on Internet Network Mgmt.*, Sept. 2006.