An Architecture for Large-Scale Internet Measurement

Vern Paxson
Network Research Group
Lawrence Berkeley National Laboratory*
Berkeley, CA 94720
vern@ee.lbl.gov

Jamshid Mahdavi, Andrew Adams and Matt Mathis
Pittsburgh Supercomputing Center
Carnegie Mellon University
Pittsburgh, PA 15213
mahdavi@psc.edu, akadams@psc.edu, mathis@psc.edu

Abstract

Historically, the Internet has been woefully under-measured and under-instrumented. The problem is only getting worse with the network's ever-increasing size. We discuss the goals and requirements for building a "measurement infrastructure" for the Internet, in which a collection of measurement "platforms" cooperatively measure the properties of Internet paths and clouds by exchanging test traffic among themselves. The key emphasis of the architecture, which forms the underpinnings of the National Internet Measurement Infrastructure (NIMI) project, is on tackling problems related to *scale*. Consequently, the architecture emphasizes decentralized control of measurements; strong authentication and security; mechanisms for both maintaining tight administrative control over who can perform what measurements using which platforms, and delegation of some forms of measurement as a site's measurement policy permits; and simple configuration and maintenance of platforms.

1 Introduction

Historically, the Internet has been woefully under-measured and under-instrumented. The problem is only getting worse with the network's ever-increasing size. Several research efforts are tackling this problem by working on developing Internet "measurement infrastructures" [LJJMH+97, Al97, HGDL97]. Such an infrastructure consists of a collection of measurement "platforms," which are dedicated workstations sited at different locations around the network. The platforms cooperate with one another to measure the properties of Internet paths and clouds by exchanging test traffic among themselves.

There are several general uses for a measurement infrastructure: to diagnose performance problems in the interior of the network by probing the network's characteristics from different vantage points; to measure the properties of a wide range of network paths to facilitate research into how the network behaves and evolves [Pa97a, Pa97b]; and to assess the performance delivered by different Internet service providers (ISPs), by measuring what happens to traffic injected into one end of an ISP's network for delivery across the ISP's "cloud" to a remote point on the other end of the network. By systematizing the assessment of ISP performance, this last function holds considerable promise for spurring ISPs to optimize their networks.

We are currently developing the "National Internet Measurement Infrastructure" (NIMI). NIMI differs from the other infrastructure projects in that the emphasis with NIMI is on building a generalized architecture for scalable measurement infrastructure, rather than performing a specific set of measurements for specific analysis goals.[1] By "scalable" we mean that our hope is to devise an architecture that might one day see ubiquitous deployment throughout the Internet. However, even building a smaller-scale measurement infrastructure is difficult (as we discuss in § 2).

We note that the "end-to-end" nature of Internet performance measures (such as quantifying packet loss rates along the path from a data sender to its receiver) often entails analyzing the behavior of a chain of distinct ISPs, no single one of which can by itself fully monitor the traffic. Consequently, elements of the infrastructure must be able to cooperate with other, administratively diverse elements.

*This paper is to appear in *IEEE Communications*, 1998. This work has been supported through funding by National Science Foundation Grant No. NCR-9711091. V. Paxson's efforts were also supported by the Director, Office of Energy Research, Office of Computational and Technology Research, Mathematical, Information, and Computational Sciences Division of the United States Department of Energy under Contract No. DE-AC03-76SF00098.

[1] Indeed, if NIMI is fully successful, then it would be natural to build the other measurement infrastructures on top of it.

A natural question when considering deploying a large-scale infrastructure is then: who will own, control and maintain the platforms? One possible answer is that the infrastructure will belong to a single organization. However, we argue that the heterogeneity of the Internet makes this approach unlikely to work in practice, and, instead, if we hope to achieve widespread deployment, then the network operators themselves must decide to deploy measurement platforms. We see several important incentives for them to do so. First, the infrastructure will aid in detecting and debugging performance problems within their network and with their peers, enabling them to better operate their network. Second, we hope that eventually "service-level agreements" between the network operators and their customers will include provisions for on-going measurement so that the customers can verify that they really do obtain the service stated in their contracts. Third, while an operator might want to limit the access provided by their platforms for measuring their *own* network, we imagine that in some cases operators will be happy to make their platforms available for measuring the performance of their *competitors*. This last fits with a major goal of NIMI, namely to facilitate *public* access to Internet measurements. This does not mean *unlimited* access, which is politically infeasible, but instead *sufficient* access, i.e., access sufficient for debugging problems, studying overall Internet behavior, and, in some circumstances, assessing the service delivered by different ISPs.

The possibility discussed above of an operator perhaps making their platforms publicly available for limited types of measurement is an example of the notion of "measurement policy." We use this term to mean defining the specifics of "who can perform what measurements using which platforms." With NIMI we do *not* attempt to define any particular measurement policies. Doing so, we believe, is hopeless, if our goal is widespread deployment in the Internet. Instead, we emphasize the development of general *mechanisms* for defining and enforcing a wide range of policies.

In the remainder of this paper we discuss the NIMI architecture in high-level terms: the requirements for a large-scale measurement infrastructure, and how different NIMI mechanisms combine to address these. We spend considerable time developing the requirements, as we believe these are crucial to understanding and shaping the architecture.

We confine our discussion to the architecture of the NIMI platforms themselves. We do not discuss here the specifics of how measurement is coordinated between multiple platforms, nor how analysis is conducted, except to note that we view the platforms as "measurement engines" whose proper job is to accurately conduct different types of measurements, and nothing beyond that.

We first discuss our experiences with a measurement infrastructure prototype (§ 2), as these experiences greatly aided us in formulating the set of goals and requirements we believe a large-scale infrastructure must meet (§ 3). In § 4 we give an overview of the NIMI architecture and discuss how it attempts to address these goals and requirements. In § 5 we briefly sketch the status of the NIMI implementation (discussed in greater detail in [AMMP98]), which is preliminary, and in § 6 offer thoughts on a number of challenging areas for future work.

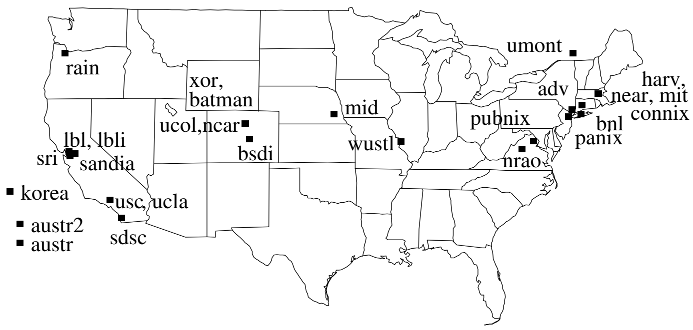

Figure 1: North American and Asian NPD sites

2 Experiences with a prototype

In this section we discuss experiences gained by creating a prototype of a measurement infrastructure, because these experiences highlight several problems that in turn shaped our development of the requirements and goals for building measurement infrastructure (discussed in § 3).

The original interest of one of us (Paxson) in the problem of developing a measurement infrastructure arose from the observation made by Danzig and colleagues that traffic mixes (i.e., which protocols comprise what fraction of the traffic) vary greatly from site to site [DJCME92]; and from the finding in our follow-on work that, in addition to mix, even the basic characteristics of TCP connections such as the distribution of their sizes or durations vary so much from site to site that we cannot soundly use the term "typical" to try to characterize Internet traffic [Pa94]. There is simply too much variation. Instead, we must attempt to soundly characterize the full range of behavior we might encounter at different points in the Internet.

Clearly we must measure Internet traffic properties from a large number of sites in order to capture the full range of behavior. With this in mind, when we embarked on a large-scale study of end-to-end Internet routing and packet dynamics, we developed a specialized measurement program, the Network Probe Daemon (NPD), which then—with the much-appreciated help of numerous volunteers—was deployed at more than 30 sites around the Internet, as shown in Figure 1. Each NPD functioned as a "measurement server" that would accept requests to either measure the route between the NPD host and a remote host using the `traceroute` utility [Ja89], or to source or sink a TCP bulk transfer and record the packets using `tcpdump` [JLM89].

A key property of the NPD measurement infrastructure is that it exhibits N^2 scaling: if the infrastructure consists of N sites, then we can measure $O(N^2)$ Internet paths between

Figure 2: N^2 mesh scaling: North American links measured by the NPD study

the sites. This scaling property means that a fairly modest (in terms of N) infrastructure can potentially observe a wide range of Internet behavior. Figure 2 shows the different links measured in the study. The N^2 scaling effect is readily apparent—a few dozen sites allowed us to study hundreds of paths through the network.

The results of the routing and packet dynamics studies are discussed in [Pa97a] and [Pa97b]. Here, our interest lies in the lessons learned from building and operating this prototype infrastructure.

First, we found that running on-going "mesh" measurements between the NPD sites took an inordinate amount of effort simply to fix problems that inevitably cropped up. While the likelihood of a single site exhibiting a problem was low, once the number of sites becomes at all appreciably large (say, more than a dozen), the probability approaches 1 of at least one of them suffering from some sort of error (system crashed and failed to restart; daemon configuration accidentally overwritten; updates required to NPD or measurement software; packet filter misconfigured; disk space exhausted; clock running berserk; firewall changed to not allow the measurement traffic; account privileges revoked; daemon unable to obtain lock on log file). Since our goal was to sustain full mesh measurements, we needed to attend to these problems, a cumbersome process involving the exchange of a great deal of email, since we did not have direct administrative access to most of the NPD hosts.

These experiences suggest several steps in order to minimize the maintenance burden of an infrastructure: *(i)* keep the elements of the system homogeneous to the extent possible, as a number of the problems were caused by differences among the varying operating systems hosting the NPD software; *(ii)* build into each measurement point as much self-diagnosis as possible; *(iii)* facilitate remote access for administration such as automating software upgrades; *(iv)* design measurement procedures and analysis to accommodate inevitable outages.

A second lesson concerned how to coordinate measurement. In the NPD studies, a single workstation W in our office was used for all measurement scheduling: it ran through a (randomized) list of which measurements should be conducted at which times, and whenever the time arrived for a new measurement from NPD host A to host B, W would contact A and B to initiate the measurement. Clearly, such centralized coordination will fail when the infrastructure becomes too large.

However, it also suffers from a second serious shortcoming. If our goal is to conduct *unbiased* measurement of the characteristics of various paths, then a centralized approach for scheduling measurement fails: whenever W cannot connect to *either* of A or B, no measurement will be performed. It is quite possible that the problem preventing W from reaching A or B coincides with a problem along the path from A to B, i.e., a problem that we would have measured and included in our analysis, if only W could have contacted both A and B to arrange for the measurement. Thus, the centralized design introduces a form of *bias*: our measurements will be artificially skewed towards underreporting serious connectivity problems, because we will sometimes lose the opportunity to measure that the problem is occurring.

This problem of bias is actually less rooted in the use of a centralized scheduling agent, and more in the notion of scheduling *immediate* measurement. If W could instead instruct A and B to conduct a measurement at some point fairly far off in the *future*, then if W failed in its initial attempt to schedule the measurement, it could continue to try to do so. If it ever finally managed to contact both A and B, it would then still schedule the measurement for the original desired time, preserving the sampling properties designed into the original measurement schedule. With this approach, we introduce bias due to failure to schedule measurement only in the case of major, long-lived outages.

The upshot of this observation is to highlight the importance of incorporating the notion of a measurement schedule not only for use by the coordinating agent (W in our example), but also in the measurement platforms themselves (A and B), and that as soon as the agent decides on a future measurement, it should begin attempting to schedule it with the platforms.

Similarly, if we only support measurement-on-demand, then when we want to diagnose network problems, we will sometimes lose the ability to perform the measurements necessary for doing so. If, however, we schedule routine diagnostic measurements in advance, then we will have measurements of problems available for retrospective analysis.

A final lesson concerned what we thought at the time was a prudent mechanism to include in NPD, namely a limited lifetime for each measurement of 10 minutes, after which the NPD would automatically terminate the measurement under the presumption that something had gone wrong and action was required lest the NPD wedge. The problem with such a lifetime is that it again biases the measurement results by eliminating any measurements that happen to require more time than the lifetime—often losing opportunities to measure times of particularly poor network performance.

The measurement lifetime was motivated by the fact that the NPD was designed to only perform one measurement at a time. Consequently, if an NPD ever wedged, the NPD site would be lost to future measurement opportunities until it could be manually unwedged. A better way to deal with this problem is to eliminate the basic limitation: the measurement platforms should be able to conduct concurrent measurements. This functionality, however, also requires that the platforms can diagnose when two concurrent measurements will conflict (for example, in their access to common resources), and thus we must include resource locks associated with particular types of measurement.

Two other noteworthy features of the NPD prototype were its use of fairly strong authentication of measurement requests, based on shared private keys and a challenge/response protocol; and its built-in "self-test," which would attempt to diagnose configuration problems. In retrospect, the authentication mechanism was only a slight burden (it required the development of a simple debugging mode for the coordination agent, one that first carried out the necessary authentication steps); and the self-test proved very valuable, but could have been made more extensive (for example, it didn't check disk free space levels).

3 Design goals and constraints

In light of our experiences, we formulated a number of design goals and constraints that we believe a measurement infrastructure must meet:

1. **Work in an administratively diverse environment.** Part of what makes large-scale Internet measurement difficult is the presence of many different administrative domains, each of which will have their own measurement policies and needs. When devising an architecture for a measurement infrastructure, it is vital to keep this basic requirement in mind, as it shapes many of the subsequent design decisions.

2. **Work in the commercial Internet.** If the goal is to support truly large-scale Internet measurement, then we cannot sidestep the reality that much of the Internet is run by commercial entities that have significant business considerations that must be recognized.

3. **Support a wide range of measurements.** There are many different interesting measurements one might want to perform to assess the performance of an Internet path: one-way and round-trip loss and delay, available bandwidth, and routing stability, to name a few. Surely the future will find us thinking about others. For a measurement system to serve as an *infrastructure* rather than focussing on specific assessments of network performance, it must easily accommodate a number of different types of measurement. This requirement leads to the important notion of a "measurement module"—that is, carefully designing the interface between the generalized measurement "engine" and the various specific measurement functions it can be used to invoke.

4. **Conduct active measurements rather than passive.** An "active" measurement is one in which part of the measurement process is to generate new traffic and inject it into the network, while a "passive" measurement is one in which existing network traffic is recorded and analyzed. There are a number of tradeoffs between using active or passive techniques [Pa96]. However, given the constraint of working in the commercial Internet, we must focus on active measurement rather than passive, as the use of passive measurement raises thorny privacy and security problems.

5. **Scale to 1000's of measurement platforms.** The Internet currently consists of an estimated 30,000,000 hosts and continues to grow at a furious pace [Lo98]. An Internet measurement infrastructure project runs some risk of becoming a "success disaster," in which a system that works fine for a limited scope takes off and rapidly disseminates throughout the much larger general network. Consequently, it greatly behooves us to attempt to address problems of very large scale (such as minimizing measurement and control traffic) even when designing a fairly small-scale prototype.

6. **Give platform owners full administrative and policy control over their platforms** . . . An immediate consequence of accommodating large-scale administrative diversity is to recognize that the infrastructure will consist of many different parties who have many different interests and levels of trust between them. The infrastructure will never attain widespread deployment, and thus major utility, unless the different parties are assured that they retain full control over whatever elements of the infrastructure that they provide. This requirement means that the owner of a measurement platform must be able to both understand how the platform is being used, and completely control its use.

7. . . . **but make it easy for platform owners to delegate control** . . . While the previous requirement means that platform owners must have full control over their platforms, we do not want to *require* them to routinely exercise that control. If we do, then orchestrating measurements across multiple administrative domains will rapidly bog down in the difficulties of obtaining the necessary permissions. Thus, we have a tension between assuring platform owners that they have full control, and yet wanting them to usually avoid having to exercise it. We address this tension by emphasizing the need for a *delegation* mechanism, by which a platform owner can decide to trust a third party for a *specific subset* of the platform's operation.

   Such delegation decisions can be quite "coarse-grained" (but see below), and thus potentially require coordination only on lengthy time scales. For example, a third party interested in conducting routing stability studies across a mesh of measurement sites might need to only once contact each of the sites for their administrative approval. Once obtained, the measurements can proceed without further involvement by the different site administrators. However, a site administrator must be able at any time to revoke such delegations; otherwise, they lose ultimate control over their measurement platforms, conflicting with the preceding requirement.

8. . . . **and provide fine-grained control when needed.** We need to recognize that, due to considerations such as business concerns, some platform owners will have quite particular requirements for specific measurements. For example, they may want to ensure that a given platform is only used as a measurement point for conducting "cloud" measurements of their competitors, and not used to measure their own network infrastructure; or, that such internal measurements are conditionally enabled, but that their results are guaranteed to remain confined to internal access only. If we fail to provide control mechanisms sufficiently fine-grained to express these sorts of requirements, then the platform owners will simply fall back on more coarse-grained control, such as "no public access."

9. **Build in solid security and authentication from the beginning.** Since one of our hopes for the measurement infrastructure is that the platforms within it will allow some forms of public access, we must take particular care to ensure that this access cannot be abused to subvert the platform to compromise its measurement policy, allow unintended access to confidential measurement results, become a stepping stone for attempts to subvert other hosts, or become a traffic source for denial-of-service attacks. In addition, we have already identified a major requirement being that the owner of a platform has full control over who uses the platform to do what, which requires solid authentication mechanisms in order to determine exactly what rights to grant a given requester. Experience has shown time and time again that system design integrity suffers when security mechanisms are added late in the design process. Consequently, we must recognize that we instead need to include such mechanisms from the beginning of our design; in particular, we must avoid the temptation to use simplistic, non-scalable security and authentication mechanisms when developing our first small-scale prototype, even though it can get away with quite limited mechanisms.

10. **Require minimal administration, maximal self-configuration.** From our earlier experiences (§ 2), we found that manually administering even a modest number of measurement platforms rapidly becomes a serious headache. Given our requirement of scaling to potentially thousands of platforms, we clearly must include in our architecture mechanisms for reducing administration to as great a degree as possible. One powerful technique for doing so is to emphasize "self-configuration," in which platforms automatically discover their correct modes of operation (such as their measurement policies) without requiring any but the most high-level human intervention to do so.

    An additional level of self-configuration would be to automate the process of determining *what* to measure *when*, and by *whom*. We view this form of configuration as very important for very large-scale measurement, but also a very challenging research problem.

4 The NIMI architecture

We now turn to how the NIMI architecture attempts to address the requirements and constraints discussed above, incorporating the experience gained from the NPD prototype.

4.1 Measurement requests

As with NPD, the basic operation of a NIMI platform is to accept requests for measurement. Unlike with NPD, however, requests fundamentally include a notion of *when* the platform should execute the request, to address the issues raised in § 2. Furthermore, requests can also include a notion of *on-going* measurement, that is, measurements that should be conducted either at a series of future times, or at a specified rate, which can be either periodic or (preferred, per the discussion in [PAMM98]) Poisson.

4.2 Credential-based authentication

A central element of the NIMI architecture is the role played by cryptographically-strong *credentials*. A credential consists of an identifier that names the credential, a public key, and a private key, the latter two comprising a public-key cryptography key pair.

NIMI requests contain within them the name of the credential held by the requester. The requests must be cryptographically signed using the private key. A NIMI platform receiving a request looks up the public key it holds for the given credential identifier and uses it to verify the signature. If the signature check fails, then the request is discarded without further processing. If the check succeeds, then the request is presumed to indeed have come from a requester who holds the corresponding credential.

4.3 Policy based on ACLs and credentials

Another key element of the NIMI architecture is the use of *access control lists* (ACLs) to represent a NIMI platform's measurement and control policies. In its simplest form, an

ACL is represented as a table. The rows of the table correspond to credentials, and the columns to possible actions. Actions may be particular measurement requests, manipulation of measurement schedules, retrieval or deletion of measurement results, uploading of new software versions, or the ability to manipulate a subset of the ACL table (see below).

Each entry in the ACL table designates the policy for whether credential C has the right to invoke action A. The entry $\langle C, A\rangle$ is either "true" if the action is always permitted, "false" if it is never permitted, or the name of a decision program (or script) to run. The decision program is given C, A, and the full request, and returns an exit status of either "true" or "false."

Thus, a NIMI platform's entire measurement and control policy is contained within its ACL table. Deleting a row from the table completely eliminates any access granted to a given credential. Deleting a column completely eliminates the ability of *anyone* to execute a given action.

It is important to note the NIMI authentication is *entirely* based on possession of the name and private key of a credential. *Anyone* possessing these elements of a credential has the same access to a NIMI platform as anyone else possessing the credential. Thus, the trust boundaries in the NIMI architecture lie exactly at who holds and shares which credentials.

Whenever we need a new type or level of access to a NIMI platform we control, we can generate a new credential name and key pair, and disseminate the private key to only those whom we wish to have the corresponding access. We must trust them to not further disseminate the private key; or, more realistically, we must decide in advance the degree to which we trust them to both use the access and disseminate the credential appropriately. If we do not trust them very much, then it behooves us to grant them only very limited access to our platforms. Using ACLs to express a platform's policy makes doing so straightforward.

Alternatively, we can allow our platform to participate in a measurement study administered by someone else, by agreeing to install the public key they give us for a credential corresponding to the measurement study.

Potentially, one could add an action to the table that allows credential holders to upload code to the platform, either for new types of measurement, or even new types of measurement administration, along with another action allowing access to such dynamically loaded measurements. Doing so obviously entails major security concerns. We plan to investigate the degree to which such access can be made secure and controlled in a fine-grain manner.

In summary, the ACL table itself is a powerful *mechanism* that can be used to express a wide range of measurement and control *policies*.

4.4 Ensuring security and privacy

The use of public key cryptography for NIMI credentials also makes it easy to ensure that NIMI requests are secure from interceptions that might either constitute "man in the middle" attacks or violate the privacy of NIMI measurement requests and results. (This last can be quite important, since measurement results can be sensitive if they reflect poorly on a particular network operator.) A NIMI request to a given platform is encrypted using the platform's public key. When the platform replies to a request, it encrypts its response using the public key associated with the credential in the request.

4.5 Delegating trust

In § 3 we discussed the tension between giving full, potentially fine-grained policy control over how a platform is used to the platform's owner, versus wanting to encourage platform owners to delegate trust when they deem it appropriate to do so. The NIMI access model provides a mechanism to make delegation simple and easy to understand in terms of its consequences.

A subset (particular rows and columns) of the ACL table can be designated a *subtable*. A given set of credentials can be confined to only potentially having access to the actions (columns) of a given subtable. (Here we say "potentially" because the credential's specific access is still checked against the $\langle C, A\rangle$ entry in the subtable.) One particular action to which a credential can be granted access is the ability to *modify a subtable*. Thus, a particular credential can be entrusted to manage a given subtable, including adding new rows to it (but not new columns, as these correspond to new actions), while ensuring that this privilege cannot be abused to modify the larger, encompassing ACL tables. Consequently, a platform owner can safely delegate trust and control for a well-defined subset of a platform's functions, and can revoke this authority at any time by simply deleting the subtable.

Subtables can have embedded within them additional subtables. Thus, NIMI supports hierarchical delegation of trust, which we anticipate will prove useful for managing large measurement experiments that might otherwise bog down in the sheer complexity of managing their access policies.

4.6 Autoconfiguration

As noted earlier, it is vital that the architecture support a form of autoconfiguration, because the burden of manually configuring a large number of measurement platforms quickly becomes overwhelming.

The basic principle underlying NIMI autoconfiguration is that each platform can at any time be told to conduct a "cold start" (if, of course, the request contains an appropriate credential). At the end of the cold start, the platform is fully configured with its current measurement policy.

When executing a cold start, the platform has exactly four pieces of configuration information available to it: a credential unique to the platform; the public key of the credential for its "master" configuration server; URLs for locating one or more instances of its master; and an ACL table with a single row, corresponding to the master credential, and a single column, corresponding to the action of being allowed to modify the entire ACL table.

The platform then contacts an instance of its master and requests that the master configure it. As with all requests, this one is signed using the private key of the platform's credential, and encrypted using the public key of the master's credential.

The configuration master in turn issues requests to the platform to update its ACL table with rows and columns that reflect the access policy for that particular platform. (This may be the same policy that is used for a number of other platforms, or a policy tailored to the specific platform; from the platform's perspective, the distinction is irrelevant. In particular, platforms do not "know" about other platforms.)

In addition, the master might instruct the platform to *redirect* part of its configuration to a different set of URLs. From each of these additional configuration-points, the platform again requests configuration, using the exact same procedure as with the configuration master. However, the degree to which these additional configuration-points can further configure the platform depends entirely on the access that the configuration master set up for them as additional rows and columns in the platform's ACL table. Thus, the master can potentially *delegate* part of the task of configuring the platform to other parties; the master fully controls the extent to which these parties can further configure the platform, by the ACL entries the master creates on their behalf. Again, the architecture here emphasizes full control over a platform by its owner (since its owner will set up the platform's cold-start configuration information, including the credential and URLs of its master and any redirects); but with it remaining easy for the owner to delegate as much of the work as they see fit, without compromising the overall integrity of the platform.

Thus, for a site to participate in someone else's measurement study potentially only involves defining a subtable limiting the possible actions the study can invoke, adding a corresponding configuration redirect (with an associated credential), and reconfiguring the site's platforms. The decision process and configuration changes need only be made once, if the site is happy to delegate further responsibility for the study.

Finally, the configuration-points to which a master redirects a platform can in turn further redirect the platform to still other configuration points. This hierarchical structure again facilitates scaling of the configuration process to quite complicated, large-scale configuration needs.

5 Implementation status

In this section we briefly sketch the implementation status of NIMI, which is preliminary. See [AMMP98] for a more detailed discussion.

We have currently deployed six NIMI platforms at five sites, four in the United States and one in Europe. We anticipate increasing the level of deployment by an order of magnitude over the coming months, as we solicit volunteers to run the first stable release of the NIMI software. The implementation consists of two parts: first, a standard configuration, consisting of Pentium hardware running the FreeBSD or NetBSD operating systems; second, a software distribution that implements the measurement scheduler, which accepts NIMI requests, authenticates them, and schedules them for later execution, and the current suite of measurement modules: `traceroute` [Ja89], `TReno` [MM96], and "poip." This latter is a generalized "ping" utility that implements versions of standard IPPM metrics [PAMM98] for host connectivity and packet loss and delay.

Currently, the software implements ACL-based authentication; scheduling measurements for future execution; encryption for confidentiality; and autoconfiguration. It does not yet support delegating trust via ACL subtables; manipulation of measurement requests once they've been scheduled (for example, suspending or deleting them before they are executed); auto-maintenance in terms of uploading new measurement tools; resource locks and arbitration for concurrent measurement; nor manipulation of measurement results (managing how they are disseminated and removed from the platforms). All of these are scheduled for near-term implementation.

6 Future work

We finish by briefly sketching some areas for longer-term future work.

One NIMI-associated project just beginning is the generation of multicast traffic from a NIMI platform that is transmitted to a large number of NIMI receivers. The goal is to analyze the patterns of received packets in order to pinpoint performance problems along the shared portion of the multicast tree between the single sender and the multiple receivers.

We envision possibly implementing a completely separate use of multicast in NIMI, namely as a means to efficiently coordinate large numbers of NIMI platforms and to disseminate their results. This use of multicast is problematic, though, because it requires *reliable* multicast transport, still an area of active research.

In addition to addressing the mechanics of result dissemination, a larger, more difficult, yet critical problem is understanding how to effectively disseminate results so that we can construct large-scale views of the network's performance without finding the sheer volume of the results overwhelming.

Another area for future research concerns the "mapping problem." Once the infrastructure reaches a sufficiently large size, we may need NIMI platforms to automatically discover the overall NIMI measurement topology, and, in particular, the relationship between each platform and its neighbors.

Since NIMI platforms are fully equipped to measure network paths, they are in some ways uniquely suited for self-organization, since the first step of self-organizing is to discover exactly where one is with respect to one's peers. On the other hand, the process of adding NIMI platforms to the infrastructure may occur at a sufficiently low rate that maintaining centralized databases of probe locations suffices.

Related to discovering the measurement topology is the problem of *aggregating* a set of measurements made for a single region or for neighboring regions into a higher-level statement concerning the region(s). For example, we would like the set of NIMI platforms located in New York City to *(i)* discover each other's presence, *(ii)* realize that they are all close to one another, and *(iii)* collaborate to form high-level statements about network conditions within, to, and from New York City, when possible.

Another general, hard problem concerns how to take a set of individual measurement results and from them diagnose performance problems internal to the network. One aid for doing so is, again, the availability of a map of the measurement topology; and, again, this is something that the NIMI platforms themselves are well positioned to produce.

A quite different problem concerns ways to ensure that we can *trust* measurements made by a particular platform as indeed being accurate, and not subject to tinkering by an unethical provider seeking to improve their apparent service, or unjustly denigrate that of their competitors.

Lastly, we offer a perhaps hopelessly ambitious vision of what a ubiquitous measurement infrastructure might deliver to an end-user: when they encounter a problem, they simply click on a graphical interface button requesting a diagnosis. This request in turn generates queries to a distributed database filled with recent measurement history, in an attempt to diagnose the problem, and, critically, to avoid “measurement implosion” when many users encounter the same performance problem and might all attempt to launch measurements in order to diagnose it.

If the database contents prove insufficient to diagnose the problem, then back (to a subset of the end-users querying about the problem) comes instead a suggestion as to what additional measurements the user's host might contribute to the database. If the user okays these additional tests, their host performs the measurements, and, consequently, not only is their own problem diagnosed, but now the distributed database can answer others' queries, both about the specific problem, and about other performance problems upon which the new measurements shed light.

7 Acknowledgements

The authors are grateful to David Martin and Les Cottrell for providing NIMI probes at different sites, and to Craig Leres for assistance in configuring some of the NIMI probes. Valuable input for the NIMI project has come from many sources, including David Martin, Les Cottrell, Guy Almes, Van Jacobson, and Craig Labovitz. We would also like to thank Sally Floyd for her helpful comments on this paper.

References

[AMMP98] A. Adams, J. Mahdavi, M. Mathis, and V. Paxson, “Creating a Scalable Architecture for Internet Measurement,” *Proc. INET '98*, Geneva, July 1998.

[Al97] G. Almes, “IP Performance Metrics: Metrics, Tools, and Infrastructure,” *http://io.advanced.org/surveyor/*, 1997.

[DJCME92] P. Danzig, S. Jamin, R. Cáceres, D. Mitzel, and D. Estrin, “An Empirical Workload Model for Driving Wide-area TCP/IP Network Simulations,” *Internetworking: Research and Experience*, 3(1), pp. 1-26, 1992.

[HGDL97] C. Huitema, M. Garrett, J. DesMarais, and W. Leland, ”Project Felix: Independent Monitoring for Network Survivability,” *ftp://ftp.bellcore.com/pub/mwg/felix/index.html*, 1997.

[Ja89] V. Jacobson, `traceroute`, *ftp://ftp.ee.lbl.gov/traceroute.tar.Z*, 1989.

[JLM89] V. Jacobson, C. Leres, and S. McCanne, `tcpdump`, *ftp://ftp.ee.lbl.gov/tcpdump.tar.Z*, 1989.

[LJJMH+97] C. Labovitz, F. Jahanian, S. Johnson, R. Malan, S. Harris, J. Wan, M. Agrawal, A. Klink, and N. Scala, “The Internet Performance and Analysis Project,” *http://www.merit.edu/ipma/docs/team.html*, 1997.

[Lo98] M. Lottor, *http://www.nw.com/zone/WWW/top.html*; February 1998.

[MM96] M. Mathis and J. Mahdavi, “Diagnosing Internet Congestion with a Transport Layer Performance Tool,” *Proc. INET '96*, Montreal, June 1996.

[Pa94] V. Paxson, “Empirically-Derived Analytic Models of Wide-Area TCP Connections,” *IEEE/ACM Transactions on Networking*, 2(4), pp. 316-336, Aug. 1994.

[Pa96] V. Paxson, “Towards a Framework for Defining Internet Performance Metrics,” *Proc. INET '96*, Montreal, 1996.

[Pa97a] V. Paxson, “End-to-End Routing Behavior in the Internet,” *IEEE/ACM Transactions on Networking*, 5(5), pp. 601-615, Oct. 1997.

[Pa97b] V. Paxson, "End-to-End Internet Packet Dynamics," *Proc. SIGCOMM '97*, Cannes, France, Sep. 1997.

[PAMM98] V. Paxson, G. Almes, J. Mahdavi and M. Mathis, "Framework for IP Performance Metrics," RFC 2330, Internet Society, May 1998.