Fallacies in evaluating decentralized systems

Andreas Haeberlen[†‡] Alan Mislove[†‡] Ansley Post[†‡] Peter Druschel[‡]

Rice University[†] Max Planck Institute for Software Systems[‡]

{ahae,amislove,abpost,druschel}@mpi-sws.mpg.de

ABSTRACT

Research on decentralized systems such as peer-to-peer overlays and ad hoc networks has been hampered by the fact that few systems of this type are in production use, and the space of possible applications is still poorly understood. As a consequence, new ideas have mostly been evaluated using common synthetic workloads, traces from a few existing systems, testbeds like PlanetLab, and simulators like ns-2. Some of these methods have, in fact, become the "gold standard" for evaluating new systems, and are often a prerequisite for getting papers accepted at top conferences in the field.

In this paper, we examine the current practice of evaluating decentralized systems under these specific sets of conditions and point out pitfalls associated with this practice. In particular, we argue that (*i*) despite authors' best intentions, results from such evaluations often end up being inappropriately generalized; (*ii*) there is an incentive not to deviate from the accepted standard of evaluation, even if that is technically appropriate; (*iii*) research may gravitate towards systems that are feasible and perform well when evaluated in the accepted environments; and, (*iv*) in the worst-case, research may become ossified as a result. We close with a call to action for the community to develop tools, data, and best practices that allow systems to be evaluated across a space of workloads and environments.

1. INTRODUCTION

As in other engineering disciplines, the relevance of experimental computer systems research is dependent on the choice of appropriate experimental conditions and environments. Ideally, a new system or method should be evaluated in the precise environment in which it is intended to be used. However, doing this is often impractical for several possible reasons: (*i*) Performing an experiment in a live system may be unsafe; (*ii*) replicating the live system in its entirety may be impractical for reasons of cost, time, intellectual property, or privacy issues; (*iii*) a comprehensive evaluation may be impractical due to the number of possible configurations of the live system; or, (*iv*) the live system may not yet exist and its precise characteristics may be unknown.

The design of decentralized systems, including ad hoc networks and peer-to-peer overlays, is a prominent example of work that attempts to develop an infrastructure for a space of applications that is still poorly understood and includes new and yet-to-be-invented instances. The current practice is to evaluate such systems using synthetic workloads, simulation environments such as `ns-2`, traces from existing systems like Gnutella, and testbeds like PlanetLab. While this type of evaluation is of considerable value, it tests the system only at a few specific points in the space of possible workloads and conditions that a deployed system might experience.

This research was supported in part by the Max Planck Society, and by the National Science Foundation under Cooperative Agreement No. ANI-0225660, `http://project-iris.net/`.

Consider just a few of the factors that influence the performance of a distributed system: network topology; link bandwidth, loss rate, and delay; node performance, reliability, uptime, and mobility; storage capacity, performance and reliability; and user workload. If we view each of these factors as a dimension in a space of environments, then a single experiment evaluates a system at one point in this space. For instance, the combination of a workload trace and the PlanetLab testbed fixes one point; the combination of the `ns-2` simulator with its models for radio propagation, mobility, and traffic pattern fixes another point.

Today, it is common practice in the systems research community to evaluate a new system at just a few well-established points in this space, defined by a selection of widely used traces, testbeds, simulation and emulation environments. The use of these artifacts has considerable appeal: they are readily available, their frequent use lends them credibility in the community, and they allow comparisons with other published results. Moreover, traces and testbeds reflect, after all, "real" systems.

We explicitly acknowledge that these methods of evaluation have been, and continue to be, of tremendous value to systems research and we do not suggest to discard them. Rather, our goal is to raise awareness of the limitations of such point evaluations, and to encourage the community to raise the standards of evaluation for decentralized systems. Specifically, we are concerned about the following pitfalls:

Generalization Despite authors' best intentions, the community may be tempted to extrapolate from the results of point evaluations to the entire space of workloads and environments. Such generalization can be misleading and may even close off interesting avenues of research.

Gravitation The convenience and the rewards for using established evaluation points may bias research towards systems that perform well at those points. As a result, other parts of the space may be neglected, even though they could lead to the discovery of useful and practical systems.

De facto standardization There is an incentive not to deviate from accepted points of evaluation due to the perceived credibility of established methods, and due to the difficulty of obtaining new, credible testbeds and trace data. As a result, systems may be evaluated at points that are not commensurate with their intended use.

Ossification In extreme cases, a research area may become ossified because the focus on specific standardized points of evaluation hinders new discoveries.

Unknown robustness properties The focus on individual evaluation points may not expose a system's robustness to changes in its environment. Therefore, it can be difficult to predict how a system behaves in an environment different from the one in which it was evaluated.

These dangers are not unique to research on decentralized systems. They can affect all experimental systems research, and have been pointed out elsewhere [8, 20]. However, we think that the situation is particularly precarious in decentralized systems, due to the vast number of factors affecting a system's performance, the large design space, and the comparatively small number of deployed systems, workload traces, simulation, emulation and testbed environments.

The rest of this paper is structured as follows. We elaborate on the dangers of the current practice and cite examples in the following five sections. Section 7 then recommends a path towards improving the current standard of evaluation in experimental systems research. Section 8 concludes.

2. GENERALIZATION

Generalization is an important tool of any experimental science. In essence, it allows the researcher to use the outcome of one experiment to predict the outcome of other, similar experiments. By carefully choosing an experiment that represents the characteristics of an entire class of configurations, the researcher can make claims without having to repeat the experiment for all possible configurations, which is often difficult or even impossible.

2.1 The problem

Unfortunately, there is an inherent danger to overstretch a generalization, i.e. to apply results to configurations that are only vaguely similar to the one tested experimentally, or even to apply them to configurations whose precise characteristics are not yet known. As a result, the potential of the proposed solution is either overestimated or underestimated. While both types of errors are problematic, the latter is slightly more dangerous because optimistic claims are usually discovered and corrected in later experiments, whereas pessimistic claims can lead others to abandon the approach altogether, and may thus remain undetected for a long time.

The generalization problem is particularly dangerous in distributed systems research, for two reasons: First, large-scale distributed systems are expensive and tedious to build, so it is far more attractive to reuse and generalize existing results than to build an entirely new system. Second, as in any emerging area, the space of potential applications is still poorly understood, so it is very difficult to judge whether or not a particular experiment is representative of an entire class of applications.

2.2 Example: File sharing traces

One instance of this problem is the inordinate popularity of file sharing traces for evaluating peer-to-peer systems. File sharing was the first widely used P2P application; in the late 1990s and early 2000s, millions of users were sharing music and videos on systems such as Napster and Gnutella. Consequently, the first measurement studies of P2P systems [3, 10, 24] used data from these applications. These studies provided much useful data, such as uplink bandwidth, session time, availability of member nodes, and popularity of objects. However, this data has since been used in a large number of other publications on a variety of different systems, and it is not clear that this extent of generalization was warranted.

The reason is that these traces specifically describe file sharing systems; other decentralized systems may have very different characteristics. For example, users typically ran the file sharing software on their home PCs. The traces therefore contain relatively few office workstations, which are generally less resource constrained and more available than home PCs – a natural selection bias. Also, many users were selfish in the sense that they only downloaded content but offered nothing for others (according to [24], the fraction of these nodes was 25% in Gnutella). These users saw no reason to remain in the system once their searches and downloads were complete. As a consequence, traces of these systems show an extreme level of churn, which by far exceeds the levels seen in traces from other environments, such as [7]. It is reasonable to expect far lower levels of churn in, say, a P2P telephony system such as Skype [25], where users want to be reachable and therefore have an incentive to stay online.

2.3 Problematic consequences

The file sharing traces have been used in many influential publications, e.g. [3, 5, 22], which has put them into the limelight of the community's attention. The problem is that the traces may have biased the general perception of decentralized systems. An expert on decentralized systems knows that file sharing systems represent just one point in a large space of applications and environments; however, somebody not as familiar with the area might come to the conclusion that decentralized systems *always* have high churn and low availability, which is not true. The fact that one particular type of environment appears so often in published work makes it easy for the reader to generalize where it is not appropriate.

For example, the Bamboo paper [22] lists typical churn rates in Gnutella, Napster, Kazaa and Overnet, and then argues that DHTs should deliver good performance under churn rates at least as high. This is problematic because there is a price to pay for the ability to handle high amounts of churn. For example, the paper proposes proactive recovery to avoid positive feedback cycles under high churn; however, reactive recovery converges more quickly and is therefore desirable for some applications. Somebody not intimately familiar with the topic might follow this recommendation even if their target environment had very low churn, and thus lose some potential performance.

In their study of Overnet, Bhagwan et al. [3] noted that Overnet had a significant node turnover, and that node availability decreased over longer periods of time. They found that, under these conditions, a system such as OceanStore would require frequent and periodic file refreshes to maintain high file availability. Again, an expert would immediately relate this to the particular characteristics of the Overnet environment, while a casual reader, who is not familiar with other traces such as [7], might take this to be a typical feature of decentralized systems.

Blake and Rodrigues [5] have shown that node availability and membership times limit the amount of data that can be maintained by a p2p storage system. However, their analysis is based (i) on a nine-day trace of 33,000 Gnutella nodes, which has extremely low node availability (only 5,000 nodes were usually available), and (ii) on the assumption that nodes do not re-join the system with their state intact. The paper states that under these conditions, a similar level of service could be provided by a few dedicated, well-provisioned hosts; in its conclusion, it raises many questions about the DHT research trajectories at the time. It is true that it is extremely challenging to implement a p2p storage system in an environment with a Gnutella-style node population. However, somebody not as familiar with the area might come to the conclusion that p2p storage systems are generally infeasible, and might question that line of research.

3. GRAVITATION

It is natural for a mature research area to eventually focus on a particular space of problems or potential applications. This is desirable because it leads to a deeper understanding of that area and creates solid foundations for future research. It is also efficient, because resources are not wasted on other areas that have turned out to be not as promising.

3.1 The problem

However, research on decentralized systems is not yet ready to focus on a particular area. There are two reasons for this: First, the set of 'real' traces and testbeds is still small, so a large part of the space of possible environments remains unexplored. Second, because the area is still young, it is not even clear how large the space of environments is, or which are the interesting areas. There is a danger of missing an opportunity simply because nobody is aware of its existence.

Unfortunately, there is a tendency for research to 'focus on itself'. We call this phenomenon *gravitation*. Successful research projects tend to uncover new, related research questions; also, they create an attractive force towards other projects, until many of them end up in the same area. Finally, their combined gravitation may attract a lot of funding, which could cause other parts of the design space to be abandoned entirely.

The availability of testbeds and traces also creates a considerable amount of gravitation. New testbeds are built for the emerging area of interest, and fresh data is collected only in that area. On the other hand, systems that would not perform well on the existing testbeds, or with the existing traces, are proposed less frequently.

3.2 Example: Unix

In his famous 2000 polemic [20], Rob Pike has called systems software research 'insular, ossified, and irrelevant', among other things because it was mostly focused on Unix for a long time. He argued that PhDs at the time were being exposed only to Unix, whereas twenty years before, they would have encountered a wide variety of operating systems, all with good and bad points of their own. As a consequence, he claimed that nobody even considered anything other than Unix any more, which was why most new operating systems tended to re-implement Unix in one way or another.

3.3 Could this happen to decentralized systems?

There is some evidence that decentralized systems research may face a similar problem in the near future. The PlanetLab testbed is already generating much new research; for example, the CoDNS [18] cooperative DNS lookup system was motivated in part by failures observed while running CoDeeN on the PlanetLab testbed. Similarly, Bamboo [22] was motivated in part by the file sharing traces. Thus, research may be biased towards systems suitable for environments like Gnutella and PlanetLab.

So what are we missing? It is easy to find examples of systems that have been influenced by previous work; however, it is inherently hard to say what *would* be built if the bias did not exist. One can only speculate what systems could be invented if there was a testbed containing many cell towers and mobile phones in active use, or one with a thousand freely programmable routers in a high-speed network. It is evident, though, that whenever a new testbed (such as the MIT Roofnet [1]) is created, a series of fascinating new ideas is published, which often become the foundation of an entirely new line of research.

4. DE FACTO STANDARDIZATION

As a community matures, a set of common evaluation methods usually emerges, e.g. in the form of traces, simulators or models that are considered to provide a sufficient evaluation environment. This has several benefits: Results in different publications are more easily comparable, there is consensus in the community over what is considered an acceptable evaluation, and the burden of proof on researchers is lessened, since the methodology in question has already been validated by others and can be considered established. Thus, a certain amount of standardization is beneficial.

4.1 The problem

Unfortunately, a standard can reach a point where it hampers research instead of benefitting it. As a first step, the standard methodology becomes very popular, such that more and more published work uses it, which in turn increases its popularity. Thus, the standard can acquire a 'critical mass' of recognition that allows it, in a second step, to dominate an entire research area. Reviewers now tend to expect the standard in conference and journal submissions, and it becomes difficult, although not yet impossible, to justify the use of alternate standards. At this point, researchers face the choice of whether to fight an uphill battle in order to push the standard of their choice (which is hard), or simply to use the established standard (which is easy). Eventually, the standard can reach the final stage and become a 'gold standard', i.e. the only accepted methodology in an entire field.

If the gold standard were perfect, its existence would not be a problem. However, real methodologies are seldom perfect; they almost always use particular abstractions, reflect certain biases, or make simplifying assumptions. If these assumptions do not hold for a project, it cannot use the gold standard, which dramatically diminishes its chances of getting published. Moreover, if the gold standard has imperfections or flaws, there is a danger that research may be led astray.

So far, a gold standard does not seem to have emerged in distributed systems. However, the PlanetLab testbed is close to acquiring a 'critical mass' of recognition and may develop into a gold standard in the future. In order to demonstrate the dangers of this possibility, we describe an example from a related area of research.

4.2 Example: The `ns-2` simulator

One example of a gold standard is the Network Simulator (`ns-2`) [17], which is used to evaluate a range of network protocols, e.g. from TCP to mobile ad-hoc networking protocols. Without a doubt, `ns-2` has been a big help for this community; however, its model of wireless networks is a considerable abstraction, and, as has been pointed out elsewhere [1,4,6,16,28], some of its properties are very different from those of a real wireless network.

Two aspects of `ns-2`'s model in particular have evoked criticism: Its mobility model, and its signal propagation model. The default (and only) mobility model that comes with the standard distribution of `ns-2` is the random waypoint model [14], which has several problems [6, 16, 28]. First of all, it essentially models random mobility which is very different from the movement patterns of pedestrians or cars for which many ad-hoc routing protocols are intended. The random waypoint model show no locality among nodes, as movement is random; however, actual movement patterns have been shown to exhibit significant locality [12]. Second, as destinations are picked randomly, nodes tend to frequently move through the middle of the simulation area, which results in a non-uniform density. Finally, as nodes randomly pick the speed at which they move towards their destination, the system is biased towards lower speeds, and thus the average node speed decreases over time. Thus, protocols simulated in `ns-2` with the random waypoint model tend to perform better under longer simulations, as the degree of mobility decreases monotonically over time. This can lead to unfounded confidence in evaluated protocols, which may perform well in the simulator but poorly in the real world [1,4,28].

The signal propagation model in `ns-2` is also problematic because the probability of successful delivery is based almost exclusively on the distance between the source and destination; however, studies have shown that distance is not a particularly good predictor [1], as short links are sometimes just as likely to lose packets as long ones. Additionally, the loss rates on particular links range from constant rate to extremely bursty, where the loss rate varies from time to time [1]. Since `ns-2` only models loss rate based on signal strength, the evaluation of protocols in this environment does not include these effects. Thus, protocols simulated in `ns-2` do not need to take this effect into account, and consequently can perform poorly in the real world. In fact, the authors of the study that pointed out many of these effects [1] found it necessary to create a new routing and MAC protocol [4].

4.3 Could this be happening to PlanetLab?

At the time of this writing, PlanetLab [21] is the most widely used tool for evaluating new distributed systems. It has grown substantially over the years, and currently consists of 629 nodes in 297 sites, which are distributed over five continents. Without a doubt, PlanetLab has been extremely useful to the community, allowing experiments of a type and scale that would have been infeasible without it. The ever-increasing popularity of the testbed speaks for itself.

However, it would be dangerous to allow PlanetLab to become the gold standard of the distributed systems community. The reason is that the characteristics of PlanetLab are not representative of the general Internet – something that the PlanetLab designers themselves acknowledge [19]. For example, PlanetLab is centrally administered and monitored for problems, so failures are rare, but can be massively correlated when they do occur [15]. The member nodes are highly available and rarely leave the system, which results in very low churn. They are also typically well-provisioned and well-connected; for example; the available bandwidth is capped at 10 Mbps per slice, and most nodes are connected via the global research and education network (GREN) [2]. Finally, most users of PlanetLab are cooperative, which is why no freeloading or security attacks have been reported so far.

These examples show why PlanetLab, while being an invaluable resource, may not be an appropriate testbed for *all* kinds of distributed systems, and why it may be counterproductive to expect a PlanetLab evaluation as a requirement for any strong publication in the field.

5. OSSIFICATION

We have argued earlier that there is considerable benefit in agreeing on a standard evaluation method (as long as that method does not become a gold standard and dominates all others). However, once such a standard has been established, there is a disincentive to change it, since change reduces its credibility and the comparability of results.

5.1 The problem

While updating the standard too frequently diminishes its value, updating the standard not frequently *enough* is dangerous. The standard must be kept up to date with recent research findings, and it must be evolved continually to keep in sync with the latest developments in technology. If this is not done, the standard may become *ossified* and cease to reflect the state of the art. Thus, it may actually guide research in the wrong direction because it creates an incentive to improve existing systems towards a goal that is no longer relevant.

The decentralized systems community has not yet accepted a common standard, so there is no danger yet of such a standard becoming ossified. However, it may be helpful to point out experiences from other fields as a cautionary tale.

5.2 Example: Andrew and SpecCPU2000

An interesting example of an ossified standard is the Andrew Benchmark used to evaluate file system performance. Andrew was designed to stress a file system by first creating a directory hierarchy, then copying files to that hierarchy, examining them, and finally compiling them [13]. However, it did this using a fixed-size data set, which was not updated with time, so the entire data set eventually fit into the buffer cache of most systems. As a result, most requests could be satisfied from the cache, so the benchmark was no longer limited by I/O operations. Instead, Andrew became limited by the compile phase, which is CPU bound. As Tang and Seltzer [8] have pointed out, it is unclear what Andrew measures today.

A similar effect can be seen in CPU benchmarks, most of which attempt to model the types of programs users will execute. The most commonly used benchmark is the SpecCPU2000 benchmark suite [26], containing a mix of programs that has not been updated in six years. In these intervening years, the tasks that users value highly have changed significantly: the growth of digital music and video, as well as the recent voice-over-IP (VoIP) trend, have changed the requirements for a "fast" CPU. A benchmark that is not updated to reflect what it is trying to measure is another example of an ossifying standard.

5.3 How is this relevant for distributed systems?

Once the community accepts a standard evaluation technique, it must be ensured that the technique is 'kept alive' and regularly updated to reflect recent developments. For example, the community should strive to obtain traces of newly deployed peer-to-peer systems which can be used in parallel with the existing traces from Gnutella and Napster. In particular, additional traces, e.g. [11], are slowly becoming available; however, they still do not have the same visibility as the original file sharing traces.

6. UNKNOWN ROBUSTNESS PROPERTIES

If, for the reasons discussed earlier, an evaluation can only be based on a small number of experiments, it is important to establish that the evaluation is *robust*, i.e., that small changes in the operating environment do not cause significant changes in the system's behavior. Without this property, it is not safe to generalize results to a large range of environments.

6.1 The problem

Unfortunately, robustness is still difficult to establish for decentralized systems. There are two main reasons for this: First, large parts of the design space are still unexplored, so it is difficult to predict the likely effect of small changes without actually implementing them. Second, and more importantly, there is no infrastructure yet with which to perform a sensitivity analysis.

In the absence of such an infrastructure, the only way to examine a system's sensitivity to changes in its environment is to perform a small series of point evaluations; for example, a system might be evaluated with trace-driven simulations as well as in the PlanetLab testbed. While good performance at two different points in the problem space is clearly a good sign, it does not strictly say much about the system's behavior at intermediate points, or even at points fairly close to the ones evaluated. Thus, there sometimes are surprises when a system is deployed in a new environment.

6.2 Example: Routing consistency

One instance of this problem affected the key-based routing substrates that have been built in recent years [23, 27, 29]. These implement a consistent mapping from a key space to a dynamic set of nodes with assigned identifiers, and they provide a primitive that delivers a message and a key to the live node whose identifier is numerically closest to that key.

While these substrates showed excellent performance in simulation under a wide range of parameter settings, they suffered badly when they were deployed on wide-area networks such as the Internet. In particular, the non-transitive connectivity of these networks caused problems [9]. Since the protocols implicitly assumed that any two nodes with global IP addresses would be able to route packets to each other, they were not robust when even just a few pairs of nodes were unable to communicate. Messages routed to the same key were suddenly delivered to different nodes, or not at all [15]. While the key-based routing protocols had been shown in simulation to be fault-tolerant and self-stabilizing, the introduction of a new type of fault broke an implicit assumption and thus led to faults that could persist indefinitely or even grow worse over time.

7. A CALL TO ACTION

Developing guidelines for experimental evaluation that avoid all of the pitfalls we outlined in this paper is difficult and perhaps impossible. However, we think there is room for improvement and it is our hope to focus the community's attention on how to realize improved best practices.

A key problem is the difficulty of systematically evaluating a prototype system against the full set of relevant, available simulation and emulation environments, testbeds, traces, workload models and benchmarks. One way to address this problem could be the development of (i) a standard interface between prototype systems and experimental platforms; (ii) a set of simulation environments, network emulators and testbeds that support the interface; and, (iii) a library of workload models, trace data and benchmarks that can be plugged into the simulators and emulators.

The standard interface would enable the evaluation of a prototype using multiple evaluation environments, from simulator to testbed, with little effort. It would encourage the development of new simulation and emulation environments to advance the state-of-the-art in experimental evaluation and avoid de facto standardization. Finally, the library could be continually augmented by the community with the latest available trace data, models and benchmarks, thus avoiding gravitation and ossification.

Of course, such an environment is not a silver bullet. To prevent the pitfalls lamented in this paper, the community will also have to do its part in using the environment's capabilities in evaluating new systems, will have to contribute to the development of new environments, and to keeping the library up-to-date.

8. CONCLUSION

This paper points out potential problems with current evaluation standards for decentralized systems: because much attention is focused on a small set of traces and testbeds, a large part of the space of environments and workloads remains unexplored, and many interesting systems may never be considered because they cannot be made to work in these environments. This problem is compounded by the fact that the space of possible applications is still poorly understood, so it is not clear at all what classes of applications are potentially being ignored. We call the community to action, in an effort to develop better evaluation environments, standards and best practices. We think that an environment that makes it easier to evaluate prototype systems using a variety of simulation and emulation environments, and against the latest available set of workload traces and models, could go a long way towards avoiding the pitfalls associated with the current practice.

REFERENCES

[1] Daniel Aguayo, John Bicket, Sanjit Biswas, Glenn Judd, and Robert Morris. Link-level measurements from an 802.11b mesh network. In *Proceedings of the 2004 Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'04)*, Portland, OR, August 2004.

[2] Suman Banerjee, Timothy G. Griffin, and Marcelo Pias. The Interdomain Connectivity of PlanetLab Nodes. In *Proceedings of the 2004 Passive and Active Measurement Workshop (PAM'04)*, Antibes Juan-les-Pins, France, April 2004.

[3] Ranjita Bhagwan, Stefan Savage, and Geoffrey M. Voelker. Understanding availability. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems*

(IPTPS'04), Berkeley, CA, February 2003.

[4] Sanjit Biswas and Robert Morris. Opportunistic routing in multi-hop wireless networks. In *Proceedings of the 2005 Annual Conference of the ACM Special Interest Group on Data Communication (SIGCOMM'05)*, Philadelphia, PA, August 2005.

[5] Charles Blake and Rodrigo Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. In *Proceedings of the 9th Workshop on Hot Topics in Operating Systems (HotOS'03*, pages 1–6, Lihue, HI, May 2003.

[6] Douglas M. Blough, Giovanni Resta, and Paolo Santi. A statistical analysis of the long-run node spatial distribution in mobile ad hoc networks. In *Proceedings of the 5th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM'02)*, Atlanta, GA, 2002.

[7] William J. Bolosky, John R. Douceur, David Ely, and Marvin Theimer. Feasibility of a serverless distributed file system deployed on an existing set of desktop PCs. In *Proceedings of the 2000 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'00)*, Santa Clara, California, United States, 2000.

[8] Yasuhiro Endo, James Gwertzman, Margo Seltzer, Christopher Small, Keith A. Smith, and Diane Tang. VINO: The 1994 fall harvest. Technical Report TR-34-94, Harvard Computer Center for Research in Computing Technology, 1994.

[9] Michael J. Freedman, Karthik Lakshminarayanan, Sean Rhea, and Ion Stoica. Non-transitive connectivity and DHTs. In *Proceedings of the 2nd Workshop on Real, Large, Distributed Systems (WORLDS'05)*, Dec 2005.

[10] Krishna Gummadi, Richard Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, NY, October 2003.

[11] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of BitTorrent-like systems. In *Proceedings of the 2005 Internet Measurement Conference (IMC'05)*, Oct 2005.

[12] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. In *Proceedings of the 10th Annual Conference on Mobile Computing and Networking (MobiCom'04)*, Sep 2004.

[13] John H. Howard, Michael L. Kazar, Sherri G. Menees, David A. Nichols, M. Satyanarayanan, Robert N. Sidebotham, and Michael J. West. Scale and performance in a distributed file system. *ACM Transactions on Computer Systems*, 6(1):51–81, Feb 1988.

[14] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[15] Alan Mislove, Ansley Post, Andreas Haeberlen, and Peter Druschel. Experiences in building and operating a reliable peer-to-peer application. In *Proceedings of the 1st Conference of the European Professional Society for Systems (EuroSys'06)*, April 2006.

[16] William Navidi and Tracy Camp. Stationary distributions for random waypoint models. *IEEE Transactions on Mobile Computing*, 3(1), January 2004.

[17] ns-2 Network Simulator. `http://www.isi.edu/nsnam/ns/`.

[18] KyoungSoo Park, Vivek S. Pai, Larry Peterson, and Zhe Wang. CoDNS: Improving DNS performance and reliability via cooperative lookups. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, San Francisco, CA, Dec 2004.

[19] Larry Peterson, Vivek Pai, Neil Spring, and Andy Bavier. Using PlanetLab for network research: Myths, realities, and best practices. Technical Report PDN–05–028, PlanetLab Consortium, Jun 2005.

[20] Rob Pike. Systems software research is irrelevant. Talk at the Columbia CS Colloquium, Feb 2000.

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

[22] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling churn in a DHT. In *Proceedings of the 2004 USENIX Annual Technical Conference (USENIX'04)*, June 2004.

[23] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of the 2nd International Middleware Conference (Middleware'01)*, pages 329–350, Heidelberg, Germany, Nov 2001.

[24] Stefan Saroiu, Krishna Gummadi, and Stephen Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking 2002 (MMCN'02)*, San Jose, California, January 2002.

[25] Skype. `http://www.skype.com/products/`.

[26] SpecCPU2000 benchmark suite. `http://www.spec.org/cpu2000/`.

[27] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *IEEE Transactions on Networking*, 11, February 2003.

[28] Jungkeun Yoon, Mingyan Liu, and Brian Noble. Random waypoint considered harmful. In *Proceedings of IEEE Infocom 2003 (InfoCom'03)*, 2003.

[29] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Sean C. Rhea, Anthony D. Joseph, and John D. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.