Network Architecture Testbeds as Platforms for Ubiquitous Computing

By Timothy Roscoe

EH Zürich, Switzerland

Distributed Systems research, and in particular Ubiquitous Computing, has traditionally assumed the Internet as a basic underlying communications substrate. Recently, however, the Networking research community have come to question the fundamental design or "architecture" of the Internet. This has been led by two observations: firstly, that the Internet as it stands is now almost impossible to evolve to support new functionality, and secondly, that modern applications of all kinds now use the Internet rather differently, and frequently implement their own "overlay" networks above it to work around its perceived deficiencies. In this paper I discuss recent academic projects to allow disruptive change to the Internet architecture, and also outline a radically different view of networking for Ubiquitous Computing that such proposals might facilitate.

Keywords: Ubiquitous Computing, Distributed Systems, Internet, Network Architecture

1. Introduction

In Mark Weiser's compelling original vision of Ubiquitous Computing (1991), the computers disappear. Instead, ordinary people interact with computing infrastructure in a seamless, almost unconscious way. For example, in research like Xerox PARC and Olivetti Research in the early 1990s, one could walk up to large displays like the Xerox LiveBoard (Elrod *et al.* 1992) and immediately use them as a personal tool.

Despite much progress, with a few exceptions this vision has not been realized outside small-scale deployments among small user groups or in research labs. A primary factor in this is that there are deep systems problems in Ubiquitous Computing which are not obvious from the descriptions of user experiences familiar from popular articles. The computers disappear, but where do they go? Where does the code that creates a user experience run, and how does it know enough about the user who has just appeared in a particular context? How can such computing resources facilitate and mediate communication between themselves and between human users? For Weiser, a researcher in operating systems, the vision of ubiquitous computing posed fundamental challenges in resource management, security, networking, and distributed systems which have had an impact on most fields in computer science.

In this paper, I will discuss one of the many current systems challenges which stand in the way of ubiquitous computing becoming an everyday reality: the difficulty of scaling laboratory-scale deployments of ubiquitous computing infrastructure to the global scale, where devices, communication mechanisms, and applications can never be fully standardized, are dynamic, evolve relatively independently, and can not be foreseen in advance.

By way of foundation, I will talk about how the communication needs of new (not necessarily ubiquitous) applications have put pressure on the architecture of the Internet, the response of the systems research community, and the directions this response might lead in for future distributed systems, and ubiquitous computing in particular. If ubiquitous computing is to become truly ubiquitous, we will have to abandon the controlled and relatively uniform world of computer science laboratories and small-scale field trials, find a way to embrace the complexity and diversity of the real world at scale, and tackle how to manage this changing heterogeneity.

2. Internet ossification and Peer-to-peer systems

In some ways, the Internet itself has become a pervasive and almost "invisible" technology in recent years. However, reconsideration of the Internet's basic architecture, and of the kinds of networking infrastructure that can best support widely distributed applications, has been a constant undercurrent in the systems and networking community for many years. This feeling became crystallized around 2001 and 2002, for several reasons.

First of these was a sense of frustration among some members of the networking research community that the Internet and its protocols had become "ossified": so much infrastructure, commerce, and communication by now depended on the Internet that experimenting with radically new approaches at scale was impossible. At the same time, routing protocols like BGP had become so embedded in the network infrastructure that changing them even in an incremental manner became a delicate issue which most Internet Service Providers shied away from. This was at a time when deficiencies in the Internet's functionality were becoming painfully clear in the form of spam, denial-of-service attacks, outages caused by routing anomalies, and the difficulties of transmitting media like voice conversations with quality guarantees.

The corresponding effect on networking research in general was a narrowing of scope, with a reluctance to explore radically different approaches on the one hand, and a reluctance to build real networks and systems on the other. Researchers relied instead predominantly on analytical results, simulations from programs like `ns2`†, or network emulation using platforms like EmuLab (White *et al.* 2002). This was particularly ironic at a time when widespread of deployment of data networks very different from the Internet (in particular, packet data over the GSM phone network and wireless sensor networks) was clearly imminent.

This issue was captured succinctly in a report commissioned by the U.S. National Academy of Sciences entitled "Looking Over the Fence at Networks" (2001). The report also suggested a potential way out of the impasse: networking research could use the concept of an *overlay* to deploy, and attract users to, a new network architecture without needing to explicitly change the underlying Internet. This naturally led to discussion of what new applications might attract users to a

† See `http://www.isi.edu/nsnam/ns/`.

new network, and how researchers could deploy overlay networks at sufficient scale to gain both experience and real users.

The second trend was rather more optimistic: the emergence of Peer-to-Peer systems both as a major research agenda, and at the same time the source of a significant proportion of traffic on the Internet. Early 2002 saw the end of the "Internet boom", and there was a feeling that client-server applications over the wide-area was now a mature and well-understood area, and P2P systems provided a new direction in research.

However, this sense of excitement with a rich new research agenda encountered similar problems to the classical networking community: how to try out and validate ideas with such systems at scale. As with networking, ideas such as structured overlay networks were investigated mostly by simulation; in the rare cases that a real deployment took place, it involved a very small number of distributed hosts (for example, in the original Chord paper (2001) Stoica *et al.* reported experience deploying the system on only 17 sites).

Thirdly, a variety of related technologies had matured at approximately the same time. These ranged from techniques for managing systems composed of computers and networking elements, in the fields of cluster-based computing and data center provisioning, to operating system virtualization techniques and resource control mechanisms. It was felt that these mechanisms could form the basis of an approach to moving research agendas in networking and distributed systems out of the current impasse.

3. PlanetLab

In this context, Larry Peterson (of Princeton University) and David Culler (of the University of California at Berkeley, and then director of the Intel Research Lab in Berkeley) met to discuss (based on an earlier proposal by Tom Anderson of the University of Washington) the idea of building a distributed computing platform to allow the development of large-scale network overlays and distributed applications.

An informal workshop convened at Intel Berkeley in March 2002 was well-attended at short notice by most of the major US Universities in the field. Consensus was reached will within the day on the basic details of the scheme, which came to be known as PlanetLab, and the reasoning was published later that year (Peterson *et al.* 2002). The key ideas were:

- PlanetLab is a shared, community-built platform. Each research institution that wishes join PlanetLab agrees to purchase and host at least two (many institutions provide more) PC-architecture server machines outside their sites firewall. These machines run the PlanetLab software, a modified Linux kernel and associated support and management software.

- In return for hosting a small number of machines, researchers at an institution can acquire *slices*: collections of virtual machines at other institutions and sites around the world. In effect, a modest outlay in local computational resources can be used to gain access to a small share of a very wide range of resources planet-wide.

- Slices are the basic container for services. Each virtual machine in a slice appears to be complete Linux operating system, allowing almost any software written for Linux to be executed. Since the virtual machines make comprise a slice are distributed over the Internet, a slice provides a vehicle for worldwide deployment of research software in a way that was previously impossible for most research institutions.

- PlanetLab has two usage models. Many users of PlanetLab deploy the software for a short period of time, run experiments, gather measurement data, and stop. However, from the beginning PlanetLab was designed to also support long-running services with real users.

This latter usage mode had a deliberate agenda: to shift the direction of networked systems research away from simulations and toward the practical validation of theoretical ideas through building and operating real systems.

It is interesting to contrast PlanetLab's aims with those of the Grid (Foster & Kesselman 2004). PlanetLab's goal is to support research into the design and implementation of widely-distributed network services, not e-Science. The Grid caters to scientists who have very large computation batch jobs (often with very large associated data sets) to run in reasonable elapsed time, but have little interest *per se* in where the job executes. PlanetLab, on the other hand, is used by computer scientists who wish to run their software for very long period simultaneously in a large number of *specific* geographical locations, but who do not need large computational resources in any one particular place.

Typical PlanetLab applications spread their computation geographically for some subset of the following reasons:

- **Removing latency.** To serve a large, dispersed user population and still provide fast end-to-end response time, computation must be moved towards users to reduce the round-trip time for messages between users and the service. Examples are commercial content distribution networks (CDNs) like Akamai and the web crawlers used by search engines like Google to populate their indexes.

- **Spanning domain boundaries:** the service executes in many geographical locations so as to have a presence in many physical areas, legal jurisdictions, financial domains, etc. Examples of this kind of requirement include the censorship-resistant publishing systems used by human-rights organizations, and archival storage systems designed to survive the physical destruction or financial dissolution of any participating provider.

- **Multiple vantage points:** the application needs to process and correlate data in real time from many physical locations. Network mapping and measurement applications were among the first services deployed on PlanetLab and continue to be major users of the platform.

With a small amount of seed funding from Intel Research, the first PlanetLab nodes came online in June 2002, just three months after the initial meeting† PlanetLab's update was enthusiastic, and it succeeded rapidly in changing the publishing

† A brief history of PlanetLab can be found at `http://www.planet-lab.org/history`.

climate for top-tier networking systems conferences, raising the bar dramatically for evaluations.

Perhaps the most important contribution has been as a source of unexpected research challenges—very few distributed applications work as expected when deployed on PlanetLab, and finding out why has focussed researchers on systems problems of great importance to industry. To take one example, a current topic of research among several groups is replacing, or augmenting, the current Internet Domain Name System (DNS) with functionality that is more flexible, scalable, and secure.

Some applications on PlanetLab have also been successful in attracting (non-research) users in large numbers. For example, two experimental content-distribution networks (Coral, from New York University, and CoDeeN, from Princeton) regularly serve more than 100,000 individual users a day. As well as the valuable operational experience this gives researchers, this traffic has also been used to conduct measurement-based research into the dynamic behavior of the Internet itself, as in Zhang *et al.* (2004).

PlanetLab has been legitimately criticized as unrepresentative of the Internet as a whole (most of the its nodes are on well-connected academic networks or in commercial colocation centers rather than on broadband connections) and (ironically) for skewing distributed systems research towards its own peculiar scenario. Nevertheless, its impact has been undeniable.

4. GENI

Despite PlanetLab's impact and success (it is now a standard part of a systems researcher's toolbox, and at time of writing consists of 840 nodes spread over 416 sites), it also failed in one of its explicit goals: to facilitate innovation in the architecture of the Internet itself (Anderson & Roscoe 2006). In retrospect, PlanetLab's applicability was limited in this regard, resulting more in a focus on Internet-based distributed systems rather than fundamental network architecture research:

- PlanetLab uses the Internet as its interconnect. Overlays above the Internet are therefore limited in the facilities they provide: in particular, they can offer no more guarantees for Quality of Server (bandwidth, end-to-end delay, loss rate, delay jitter, etc.) than the Internet itself. Since the Internet eschews such guarantees as a design principle (Clark 1988), this imposes a real limit as to how different a new network architecture deployed above PlanetLab can be.

- PlanetLab's community model means a relative paucity of resources, and PlanetLab machines are regularly oversubscribed (particularly those in interesting locations). A routing infrastructure which aims to provide reliability, or performance guarantees, will have a hard time running over PlanetLab.

- While PlanetLab nodes are widely dispersed globally, and their Internet connectivity varies, they are still relatively homogeneous: they are all PC-based servers, offering a Linux execution environment, and their interface for communication is still limited to the Internet Protocol.

Anderson *et al.* made a case (2005) for addressing this issue by extending the PlanetLab model to include not just relatively homogeneous computers spread throughout the network, but also routers, switches, radios, links, other kinds of computers, etc. In short, a research slice would include parts of all the physical plant needed to construct a complete wide-area network architecture, which spanned many kinds of networking technologies (including wired and wireless networks) and many kinds of computing devices (servers, desktops, phones, wireless sensor nodes, etc.).

Such a platform requires a much greater capital outlay than PlanetLab for hardware and software development, dark fiber across continents, radio spectrum permits, etc. To this end, the National Science Foundation in the US is currently backing a large-scale program called the Global Environment for Network Innovation or GENI (BBN 2008) to build such a platform.

The goal of the GENI program is to enable fundamental research into the architecture of the Internet, but allowing research groups to implement new network architectures and deploy them along side each other at scale to evaluate their usefulness, potentially with real users. Several possible successful outcomes have been suggested.

In one, ideas generated and validated through the research are retrofitted to the Internet, which thereby evolves into a better network. In another, a better network architecture is eventually arrived at by consensus. This new network runs alongside the Internet for an extended period of time, but eventually replaces it. A further possible outcome is that several architectures emerge in different domains, and coexist.

Funding for the GENI program in the US is undecided at time of writing. However, regardless of the decision, it is representative of an emerging way of thinking about networking infrastructure and associated research. Similar programs are planned in Europe, Korea, Brazil, and elsewhere.

5. Ubiquitous computing and network architecture

Programs like GENI aim to create a better network environment (more secure, more reliable, more amenable to real-time traffic) for the development of ubiquitous computing applications. But the GENI project itself is focused on the development of new network architectures, and at first sight might seem to have little to do with more user-centric ubiquitous computing as it is commonly portrayed.

However, I want to argue that the basic requirements of the GENI platform open up wide-ranging interesting possibilities for the global deployment of true ubiquitous computing applications.

The most fundamental challenge facing the designers of GENI is one of heterogeneity: we are trying to build a facility to allow networking researchers to exploit a wide range of networking and computing hardware, including unforeseen equipment in the future. This means that GENI will have to present slices of its resources (routers, computers, links, radios, etc.) in as detailed a way as possible to “users” (networking researchers deploying a new architecture), and that those users will have to build the tools to synthesize a useful network out of these diverse resources.

This is very close to the problem facing implementers of the ubiquitous computing vision: how to knit a dynamically changing collection of networked devices (or shares of them) into a coherent user experience.

In turn, this raises a fourth success scenario for GENI, in addition to the three listed above: GENI's model of slice acquisition and virtualized, but individual resources, itself becomes the next 'network' architecture, and services run directly over this substrate. Ubiquitous Computing applications directly acquire and release slices of hardware resources as needed.

In the Systems Group at ETH Zürich, we view this as a highly productive context in which to pursue broad research into supporting ubiquitous computing. The wide-area setting forces us to engage with many systems-level challenges (like those thrown up by experiences with PlanetLab) which do not manifest in a lab or small-scale deployment, and the heterogeneous nature of the underlying platform forces us to engage with a complex set of evolving technologies with general techniques rather than point solutions.

Our research goal is to determine the appropriate runtime and support environment for modern ubiquitous computing applications over a heterogeneous infrastructure. GENI represents an extreme point in this space, but even today's Internet, the mobile phone network, PlanetLab, and increasingly prevalent utility computing services like Amazon's Elastic Compute Cloud (EC2) present a deployment environment where dynamicity and heterogeneity are key challenges.

A promising avenue for addressing the latter at present is the use of subsets of first-order logic coupled with constraint programs for describing resource *requirements* by applications, resource *advertisements* by resource providers, and *contracts* between applications and resource providers as to commitment and use of resources like virtual machines for computing, network links and radios for communication, storage services, forwarding engines, sensors, displays, etc.

One concrete instantiation of this, which we are pursuing in collaboration with colleagues at Microsoft Research in Cambridge, might equally be termed the "personal overlay network", or the "disaggregated personal computer". Instead of one's data, applications, and communications being tied to a physical PC on a desk, or a large provider of high-level applications like Google or Yahoo!, we are exploring a model where a user's data and applications are hosted by a small overlay network consisting of physical devices (such as a phone or home server) and rented virtual machines (such as those provided by PlanetLab or Amazon EC2).

The loss of any one of these devices should not jeopardize the user's data or the functionality of their computing infrastructure, but equally this overlay should be able to adapt, for example to:

- move heavyweight computation in the form of virtual machines closer to the user when he or she travels, so as to minimize interaction round-trip time,
- route information efficiently between personal overlays for inter-personal communication and data exchange
- acquire large amounts of resources for short periods of time, for instance for image recognition or synthesizing 3D views from photographs,
- take advantage of devices near the user, such as large displays, microphones, etc.

While at first sight this may seem somewhat distant from the large-scale testbed efforts like GENI, the principal challenges are remarkably similar to those inherent in building new network architectures above such a shared, virtualized infrastructure. In this way, we hope to provide part of the software substrate needed to move the ubiquitous computing vision out of the laboratory and more into our daily lives.

References

Anderson, T., Peterson, L. L., Shenker, S. & Turner, J. 2005 Overcoming the Internet Impasse through Virtualization. *IEEE Computer* **38.4** 34–41

Anderon, T. & Roscoe, T. 2006 Learning from PlanetLab. *Proc. 3rd Workshop on Real, Large Distributed Systems (WORLDS)* November 2006

Bolt, Beranek & Newman 2008 GENI: Global Environment for Network Innovation. `http://www.geni.net/`. Accessed March 2008

Clark, D. 1988 The design philosophy of the DARPA internet protocols. *SIGCOMM Comput. Commun. Rev.* **18.4** 106–114

Computer Science and Telecommunications Board, National Academy of Sciences 2001 *Looking Over the Fence at Networks: A Neighbor's View of Networking Research.* Washington, D.C.: National Academies Press

Elrod, S. *et al.* 1992 Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. *Proc. SIGCHI Conf. on Human factors in computing systems, 1992,* pp. 599–607

Foster, I. & Kesselman, C. 2004 *Grid 2: Blueprint for a New Computing Infrastructure.* 2nd edn. Morgan Kaufmann

Peterson, L. L., Culler, D., Anderson, T. & Roscoe, T. 2002 A Blueprint for Introducing Disruptive Technology into the Internet. *Proc. 1st Workshop on Hot Topics in Networking (HotNets-I), Princeton, NJ, USA, October 2002.*

Stoica, I., Morris, R., Karger, D., Kaashoek, F. & Balakrishnan, H. 2001 Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *Proc of the 2001 ACM SIGCOMM Conf., 2001* pp. 149–160

Weiser, M. 1991 The Computer for the Twenty-First Century. *Scientific American* **265**(3) 94-104

White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C. & Joglekar, A. 2002 An Integrated Experimental Environment for Distributed Systems and Networks. *Proc. 5th Intl. Symp. on Operating Systems Design and Implementation, Boston, MA, USA, December, 2002,* pp. 255-270

Zhang, M., Zhang, C., Pai, V., Peterson, L. & Wang, R. 2004 PlanetSeer: Internet Path Failure Monitoring and Characterization in Wide-Area Services. *Proc. 6th Intl. Symp. on Operating Systems Design and Implementation, San Francisco, CA, USA, December 2004*