Pervasive Services Infrastrucutre

Dejan S. Milojicic
Mobile Systems and Services Laboratory
HP Laboratories Palo Alto
HPL-2002-79
April 2nd , 2002*

E-mail: dejan@hpl.hp.com

distributed systems, pervasive systems, peer-to-peer systems, mobile systems, research and development

Technology has been advancing in the past century at an incredible rate. All predictions about the limits have been consistently broken for processor speed, network bandwidth, and memory size. Nevertheless, the way we use computers has not kept pace with the technology itself. Computers are still being used in a similar way as a couple of decades ago.

We need to change the research from technology-centric to human-centric. Invisible computing, deployed for common users, and in new markets, can dramatically change the technology landscape. It can further influence the way we perceive and use computers and make a disruptive push of technology development. Pervasive and peer-to-peer computing are two efforts in this direction. We compare four different types of computing (distributed, mobile, pervasive, and peer-to-peer) with respect to disruptiveness, benefits, and challenges. Then we analyze research, technology, and management. Finally, we discuss four case study projects as examples of four types of computing and ways of how to conduct research.

* Internal Accession Date Only Approved for External Publication
To be published in the Proceedings of the Workshop on Trends in Information Technologies in the Beginning of the 21st Century

Pervasive Services Infrastructure

A Case Study in Research and Development

Dejan S. Milojicic

HP Labs[1]

Abstract. Technology has been advancing in the past century at an incredible rate. All predictions about the limits have been consistently broken for processor speed, network bandwidth, and memory size. Nevertheless, the way we use computers has not kept pace with the technology itself. Computers are still being used in a similar way as a couple of decades ago.

We need to change the research from technology-centric to human-centric. Invisible computing, deployed for common users, and in new markets, can dramatically change the technology landscape. It can further influence the way we perceive and use computers and make a disruptive push of technology development. Pervasive and peer-to-peer computing are two efforts in this direction. We compare four different types of computing (distributed, mobile, pervasive, and peer-to-peer) with respect to disruptiveness, benefits, and challenges. Then we analyze research, technology, and management. Finally, we discuss four case study projects as examples of four types of computing and ways of how to conduct research.

1 Introduction

Have we learned how to recognize successful technologies from failures? Hamming advised us how to select important research [11]; Christiansen taught us how to identify disruptive technologies [6]; Lampson introduced principles of system design [14]; and visionaries predicted the future of computing [7, 20, 30]. What assumptions can we make about research and technology today?

First, it is services that matter to users, not technology itself; next, it is convenience to users, not power of the systems being used; furthermore it is the time and place where users need service. The computers today represent a source of friction to users – it is hard to keep pace with all versions of software, hardware becomes obsolete soon after being purchased, and managing it is a nightmare. Wireless networks are still not ubiquitous. Services are being offered by service providers that can be down, or the network can be down, preventing service delivery to users.

Technology is moving towards mobile, pervasive, and peer-to-peer solutions. Laptops have already replaced desktops, and it is a matter of time when handhelds will take

1. *Address:* HP Labs 1501 Page Mill Road, MS 1183, Palo Alto, CA 94304, USA. *Tel.:* +1 (650) 236-2906. *Fax:* +1 (650) 857-7029. *Email:* dejan@hpl.hp.com. *WWW:* http://www.hpl.hp.com/personal/dejan_milojicic/

Table 1: Comparison of Disruptive Computing

Computing Trend	Disruption *(detaching from)*	Benefits		Challenges	
distributed	decentralization of computing and data *(a single computer)*	remote access replication for high availability redundancy for failover	Increased deployment, accessibility and usability	remote nodes dependency distributed state end-to-end argument	heterogeneity, interoperability, security, trust, transparency, scalability, etc.
mobile	attaching data and computing to user *(fixed computer/network)*	mobile information access mobile computing access ad-hoc networking		disconnection battery lifetime context awareness	
pervasive	user- v. technology-centric focus *(technology)*	broad availability computing invisibility context-adaptivity		personalization scalability in locale user interfaces	
peer-to-peer	peer-provided service/ computing/data *(fixed service providers)*	shared cost-of-ownership ad-hoc component availability self-organization		reputation release of control fragmentation of peer base	

their place. Limited user interfaces, battery lifetime, and lack of ubiquitous connectivity are key disabling factors. On these devices, users want to have rich content delivered and access services as on traditional computers. There is imminent convergence of PDAs and mobile phones, enabling service access from (almost) anywhere.

To better understand the directions technology is evolving, we selected four different types of computing (distributed, mobile, pervasive, and peer-to-peer) that we compare in the light of the disruptions they have brought to users, their benefits, and their challenges. Next we discuss how to conduct and manage the research, and how to distinguish it from development. Finally, we present a few projects in the Pervasive Services Infrastructure (PSI) program at HP Labs and position them in the light of the four technologies and the ways of conducting research.

2 Retrospective of Technology Trends

The trend in future computing can be simply stated as more and more distributed, mobile, pervasive, and peer-to-peer. One simplified comparison of the four types of computing is based on the disruptions relative to traditional centralized computing, as well as the benefits and challenges each type introduces. The comparison is motivated by Satyaranayan's analysis of pervasive computing [27], and it is summarized in Table 1. In the rest of this section we discuss some of the disruptions, benefits and challenges of each type of the computing.

Distributed computing detaches computing from a single location by distributing data and computation [5]. The benefits and challenges of distributed computing are taken to the extreme with contemporary software and hardware. The benefits include remote access to data, replication, and redundancy. The challenges are even greater because of the increased distribution and dependency on larger number of components. Furthermore, the state is also more distributed introducing additional penalties for synchronization, and making it harder to ensure end-to-end guarantees [25].

Mobile computing offers disruption by separating computing from any fixed location and by enabling a user to take computing and data with her [16]. The benefits include mobile information and computing access, eliminating the need for wires (resulting in more cost-effective deployment and usability), and support for ad-hoc networking. The challenges encompass the support for disconnection due to the lack of global connectivity, limited battery lifetime, and the need for context awareness.

Pervasive computing is a historically more recent type of computing [20, 27]. The premise of pervasive (or ubiquitous) computing is that it should become invisible, threaded into the lives of users. Pervasive computing detaches focus from technology-centric and makes it user-centric. The benefits of pervasive computing include broad availability of computers, their invisibility, and adaptation to context. At the same time, the introduced challenges include the need for personalization, scalability in localized environments (so called downward scalability), and the need for improved user interfaces (voice, gesture, etc.)

Finally, peer-to-peer computing is a relatively new trend, even though the paradigm has existed almost as long as distributed computing [1, 16]. Peer-to-peer computing detaches service from fixed service providers and enables any peer in the system to offer the service. The primary benefits of peer-to-peer computing are in the shared cost of ownership (peers contribute a large part of the infrastructure), ad-hoc peer availability (peers are expected to join in the system on an ad-hoc basis), and self-organization (the system adjusts or self-organizes based on the load, availability, and peer needs). At the same time, the challenges are posed by: reputation of the peers serving the needs and those who do not contribute to the system (so called free-riders); the release of control (not only resources, but also control is decentralized in a peer-to-peer system, which is a cause of concern for IT), and the fragmentation of the user base (the success of a peer-to-peer system is ultimately tied to the existing user base).

Across all four types of computing, common benefits have improved over time, resulting in an increased deployment, accessibility, and usability of computing. However, these benefits have also increased some of the common challenges, such as scalability, heterogeneity, transparency, security, and trust. For example, *heterogeneity* and *interoperability* were first emphasized in distributed computing, but these requirements changed with the introduction of mobile, pervasive, and peer-to-peer computing. Issues involve how to support all kinds of devices with different capabilities, hardware, software, and networks. Similar arguments apply to *scalability and security* because suddenly any number of other computers and/or users is able to connect to a service provider. Users can connect from any location, scaling from the whole world, down to numerous computers in a single room, and possibly without centralized points of control and trust. Some of the requirements had to be relaxed in order to make some headway, e.g. security in wireless networks. If security were a major concern in wireless technology, the area would probably not have advanced as it has today.

3 Research, Development, and Management

This section addresses some of the trends in conducting research, relationship between research and development, and managing research.

3.1 Conducting Research

In this subsection, we analyze some of the common pitfalls in doing research and compare it with alternatives.

Fashion v. Substance. Over time, the author of this paper has encountered and personally experienced a number of fashionable activities. Artificial intelligence and expert systems were very fashionable in the eighties, Massively Parallel Processors (MPP) in the nineties, then pervasive computing, and most recently peer-to-peer computing. Over the time, there were a lot of efforts to rewrite all software in an object-oriented way, microkernels were dominating the "future of the operating systems," and there were many distributed system environments offered. It was really hard to distinguish between the real contribution (substance) and jumping on the bandwagon. The impact often lies in the eye of beholder. For example, there was a huge investment in the Mach microkernel, including OSF/1, but it has not resulted in a widely used operating system (earlier versions propagated into Apple's operating system, and OSF/1 was used as Digital UNIX). On the other hand, some of its ideas and research motivated other work, including features of the NT operating system.

There are also certain benefits in joining the bandwagon, such as a bigger pool of researchers having the same or similar mindshare, sharing results, tools, common events, etc. Nevertheless, at the end of any project, the important question to answer is really what contributions have been made and not how fashionable the results are. The ideal approach is to work in a contemporary area, yet attempt to give a unique and substantial contribution in the field.

Solution in search of the problem. One of the most common failures of researchers is being attached to a certain area, having some ideas about a potential solution and then searching for a problem to apply the solution to. For example, the author of this text tried to apply the process migration solution to the field of mobile agents. The best way to avoid this pitfall is to ask oneself: what is the problem I am trying to solve, rather than how/where can I apply this (great) idea.

Hard problems v. making impact. Another common mistake of researchers (myself included) is a tendency towards doing something intrinsically hard, rather than primarily useful to users. Examples include hard real-time, distributed shared memory, process migration, and many other problems in almost any research area. A good way to overcome this problem is to evaluate the impact of the work, rather than how hard the problems are. Still, there will continue to exist research in critical, but non-cutting edge areas, such as fault tolerance and compilers.

Newest gadgets. Another common pitfall is always to use the newest possible technology. While it is important to be contemporary, many of the latest technologies may not have a lasting impact, examples include MPP and Myrinet. The early adopters affect, where the cost for the newest devices significantly dominates the prices in stable markets, also applies to the risk of using the newest technologies for research.

Putting it all together: the "aha" effect. There are rare occasions when it is possible immediately to recognize successful research. That is one that inspires us to say "aha" immediately as we hear about it. Typically, it is research that address some substantial problem in a contemporary setting, and the impact is obvious. However, in

many cases, impact comes after hard work and many years of persistent contributions, rather than a brilliant idea. Examples of big impact in HPL are inkjet printers and VLIW processors. Each effort took almost 10 years to make an impact on HP.

3.2 Research v. Development

This subsection addresses differences between research and development.

Caricature v. complete solution. The goal of research is to create only enough solution to recognize its potential. It is frequently compared to a caricature: enough of the picture to demonstrate the theme, in comparison to a complete picture that requires significantly more effort [28]. Once a caricature with a useful theme is created, it can be turned over to development to productize a complete picture. This is a win-win situation. Researchers are best at drawing many caricatures and developers are best at picking the most promising ones and turning them into very good products.

Success v. failure of research/development. Frequently, it is believed that research serves only to find promising solutions for development. There is a less known benefit of research in identifying inappropriate solutions and thereby saving the effort of development. Because development requires significantly more effort, identifying wrong solutions in research saves the significantly greater investment that could have been spent in development.

Making big impact. From research, industry expects results that can potentially have big impact and disrupt the market, rather than incremental improvements. For example, research is expected to come up with solutions that can result in multiple performance improvements, dramatically better scalability, and self-organization. Another rule of the thumb is that typically 90% of the solution can be achieved by 10% of the effort, whereas the last 10% of the solution may require 90% of efforts.

When and what to transfer? It is a hard decision when to transfer and when to engage developers with the results of the research. If it is too early, the benefits may not yet be obvious enough; if it is too late, the benefits might be too late to leverage. Sometimes, only idea is transferred, early in its inception. Other times, it can be a complete prototype. Yet other times, it is a whole solution, together with the people who developed it.

What to leverage? In doing research, it is required to leverage work of others as much as possible. Making additional assumptions about the future is one way of doing it. Extending the caricature analogy further, research is like a scaffolding that can hold the whole architecture, but only crucial pieces need to be developed and evaluated.

Putting it all together. Research and development need to work in harmony, each is best in its own domain: researchers in predicting the field and developers in making it happen. Researchers have better sense for vision and developers have it for business.

3.3 Managing Research

This section summarizes experience the author had while managing research projects in HP Labs, in the period of 1998-2002.

Surveying the field. Before starting to work in a new area, it is a requirement to understand the prior art. This can range from studying recent and classic literature, and

writing overview papers, up to writing thorough surveys. The former approach serves to educate the author about related work, to make sure that the intended work hasn't already been done, and to learn about alternative and potential solutions to leverage. The latter end of spectrum can be research in its own right, by identifying the high level view of an area and making new observations.

Creating the vision/mission/purpose/values statements. Setting vision (where to), mission (what and by when), purpose (why and for whom), and values (guiding principles) is a traditional approach to running projects. This is equally applicable in conducting research projects. It helps to clarify the goals and gather the teams behind the same viewpoint. It also helps in communicating the research to others.

Identifying the key research questions. The next important step is to identify the key research questions. This helps to focus the research into specific areas. If there are no key research questions, there is no opportunity for research. However, identifying them is research in its own right, and typically the hardest part of starting a new effort.

Working with others and globalization. Both research and development are less and less localized efforts and more and more globalized. Throughout the career of the author of this paper, he frequently worked on projects with participants from more than one continent, sometimes making it even very hard to organize meetings at meaningful times of the day for everyone. Cultural differences are yet another problem.

Getting Feedback from competition. One of the best analysis of your results is typically obtained from competition. While it is not common approach in industry, the research community is very generous in providing honest feedback.

Knowing when to stop a research. It is hard to start new projects, but it is an equally hard decision to stop an effort that does not make any substantial contributions any more. One of the best indications of making impact is having others to use your results. This is useful in many ways, including to verify the results.

Putting it all together: impact. The final step in managing research is to evaluate your effort. When are you successful, and what can you learn about it to be more successful next time? One measure is the number and quality of publications. This is especially valued in academia, where "publish or perish" is a driving motivation. Another measure is by the number and quality of patents. Fundamental patents in specific areas are especially valued. The last one and probably the most important one is if you have impacted any business. This is particularly valued in industrial labs, although academia also adopts a similar approach lately. Professors are becoming valued for starting a successful business during sabbaticals. Inktomi, Akamai, and VMware are a few examples, but there are many more. Governments also support this model, by consistently supporting those academic efforts that will result in transfer to industry.

4 A Case Study: Pervasive Services Infrastructure

The Pervasive Services Infrastructure (PSI) program started in 2001 [17]. The PSI vision is "any service to any device". This vision remained the same across the life of the program even with various projects starting and stopping and going through transformations over time. It also contributed to position PSI with respect to other programs.

For example, the broader HPL program has a vision to deliver “right service at the right time”. To achieve this bigger vision, the PSI vision “any service to any device” is still the right one at the systems level. PSI consists of a few projects briefly described in the rest of this section. A more detailed description is beyond the scope of this paper.

4.1 Adaptive Offloaded Services (AOS)

The AOS project enables transparent offloading of applications from handheld devices to surrogate servers [18]. AOS overcomes handheld devices diversity due to their age, configuration, and different producers. It enables offloading of memory and processing for improved performance and increased battery lifetime. AOS achieves this by transparently monitoring applications. The changes have been made to the underlying JVM (Chai) to enable the splitting of applications, offloading the execution of part of the applications, and forwarding requests between the distributed parts.

AOS increases the number of services that can execute on wireless devices and increases their performance. The key research questions include: how to effectively split and offload Java programs; monitoring distributed service execution; selecting the right granularity of service components for offloading; and exploring different criteria for offloading, such as memory, execution, or power. Related work to AOS includes Coign [10], yBase [15], and JavaParty [21].

AOS selected a hard problem to address and has quickly demonstrated its ability to transparently offload memory. Currently, other angles of offloading (processing and energy saving) are investigated. AOS confirmed that hard problems are difficult to bound (see hard problems in Section 3.1 and completeness of solutions in Section 3.2).

4.2 Services on Demand (SoD)

SoD represents an infrastructure architectured around service brokers and client run times [2]. The brokers index services and match the resource capability of devices. The client run time interacts with service brokers, which deploy services and transparently store data on storage servers. SoD goal is to enable users to start services on zero-installed devices and to match services to user preferences and device resources. As a part of installation, the service components are transparently bytecode-edited to support remote storage and disconnection.

The objective is to increase service pervasiveness across a range of personal mobile devices, enabling easy retrieval, composition, and deployment of services. It also makes personal data available across all of a person’s devices and helps to cope with intermittent connectivity. SoD facilitates the use of personal devices, it increases the service base, and it eliminates the complexity of installing, administering, backups, sync-ing, etc.

The key research questions that SoD addresses are: service publishing and discovery; how to exchange services between the devices; service trust delegation, collaborative services framework; and private storage sharing model. Work related to SoD includes Oxygen [7], Ninja [9], and Coda [26].

The main challenges in SoD are related to the wealth of development in Web services domain, such as OSGI or UDDI. Compared to other PSI projects, SoD has the most

complete implementation becoming a good candidate for transfer (see research v. development in Section 3.2). SoD was primarily conducted in HP Labs in Grenoble, France, with bi-weekly meetings with and quarterly visits to Palo Alto, making it a truly global project (see globalization in Section 3.3).

4.3 Peer-to-Peer Ad Hoc Communities (PAC)

The PAC project supports communities of peers that could be users who communicate and collaborate [19] or aggregated user appliances, such as a mobile phone, laptop, or personal digital assistant [23]. The objectives are an infrastructure for ad-hoc group activities and a platform for aggregation of the resources of user appliances. The benefits include removing a fixed infrastructure as a requirement, and a (nearly) single-system image of appliances. PAC demonstrated the handoff between infrastructure mode and pure ad-hoc mode, where in the former advantage can be taken of the infrastructure servers for performance, battery lifetime, and availability. A different set of membership and multicast protocols are used in ad-hoc mode. PAC also demonstrated aggregation of the appliance file systems, bandwidth aggregation, and saving battery lifetime.

The key PAC research questions include user-friendly aggregation of appliances, single-system view, improving communication latency and battery life, and ad-hoc/infrastructure transitions. The work related to PAC includes: Magi [3], .NET [29], and MOPED [12].

There is also a significant industrial presence in this field (Groove, Magi, JXTA), and initial concerns were how to distinguish contributions from others (see Section 3.1 on conducting research). Our approach was to select specific enough contributions (ad-hoc, aggregation) and to be user- vs. technology-centric. PAC eliminates user friction, such as infrastructure awareness or having to deal with multiple personal appliances.

4.4 Planetary-Scale Peer-to-Peer (P^3)

The objectives of the P^3 project are to provide infrastructure support for Internet-based services by harnessing commodity components (on the Internet), to achieve unlimited scalability/availability with zero-administration cost, and to provide appropriate level of safety, security and privacy. The key components of P^3 include application-level (overlay) networks; core services such as scalable naming, storage/computing services, monitoring and scheduling; resource virtualization; and service advertisement, discovery, and composition.

The P^3 key research questions include: how the application-level overlay can take advantage of underlying network conditions and applications' needs; how to provide storage and computation services at the Internet scale; how to scale monitoring/scheduling; how to achieve self-configuration, -tuning and -healing; and how to provide safety/QoS guarantees.The work related to P^3 includes: P2P data placement systems, such as PAST [24], CAN [22], and Oceanstore [13].

One of the biggest challenges of P^3 is in differentiating from other numerous academic efforts in the P2P space (see Section 3.1). The approach is to try to offer improved algorithms over the competitors in the field.

4.5 Summary

All PSI projects revolve around the general trends of distributed, mobile, pervasive, and peer-to-peer computing. One of the primary criteria for starting projects was to make an impact, while still addressing hard problems. In all of the efforts, we primarily addressed the users of the technology and not technology itself, for example adapting the services to user devices for more convenient use (AOS), delivering services to users on-demand (SoD), hiding the infrastructure from users and aggregating her appliances (PAC), and offering planetary scale aggregation of components (PAC). In each case, we tried to make computing invisible to users.

5 Summary

In this paper we described trends in computing, by comparing the disruptive nature, benefits, and challenges of distributed, mobile, pervasive, and peer-to-peer computing. We also presented some insights in research, development, and management, by discussing how to conduct research, by comparing research versus development, and by evaluating management of research. Finally, we presented four case study projects of the Pervasive Services Infrastructure (PSI) program.

Conducting research is a challenging, but also a very fulfilling effort. It is challenging in that one always has to be on the edge, competing with the top minds around the world of academic and industrial competitors. It is fulfilling in that one can see the impact of his ideas. Part of it is art and intuition, but there are also some common practices, such as how to select good topics, when to start prototyping, when to transfer results, how to engage production units and how to manage research in general.

Acknowledgements

Alan Messer and Ira Greenberg are principal contributors to AOS; Philippe Bernadat conceived SoD; Kiran Nagaraja, Sami Rollins, Dan Muntz, and Vanish Talwar are contributors to PAC; Zhichen Xu is leading P^3. The author is indebted to Ira Greenberg, Alan Messer, Kiran Nagaraja, Sami Rollins, Greg Snider, and Zhichen Xu for reviewing the paper. Their comments significantly improved the content and presentation.

6 References

[1] Oram, A., “Peer-To-Peer, Harnessing the Power of Disruptive Technology,” O’Reilly. 2001.

[2] Bernadat, P., Greenberg, I., Messer, A., Milojicic, D., “Tailoring Java for a Pervasive Service Infrastructure,” HPL Technical Report, HPL-2002-24, 2002.

[3] Bolcer, G. et al., “Peer-to-Peer Architecture and the Magi Open Source Infrastructure,” Endeavours Technologies White Paper, 2000.

[4] Brooks, F., “Mythical Man Month, Essays on Software Engineering,” Addison-Wesley Publishing, 1975.

[5] Couloris, G., Dollimore, J., and Kindberg, T., “Distributed Systems. Concepts and Design,” Addison Wesley, 2001.

[6] Christiansen, C., “The Innovator’s Dilemma,” Harper Business, 1997, revised 2000.

[7] Dertouzos, M.L., “The future of computing,” Scientific American, July 1999.

[8] Gong, L., “JXTA: A Network Programming Environment,” IEEE Internet Computing, (5)3:88–95, May/June, 2001.

[9] Gribble, S., et al., “The Ninja Architecture for Robust Internet-Scale Systems and Services,” Computer Networks, vol. 35, no. 4, pp 473-497, 2001.

[10] Hunt, G.C. and Scott, M.L., “The Coign Automatic Distributed Partitioning System,” Proceedings of the 3rd USENIX OSDI, pp. 187-200. New Orleans, LA, Feb. 1999.

[11] Kaiser, J.F., “Richard Hamming: You and Your Research,” Transcription of Bell Communications Research Colloq. March 1986.

[12] Kravets, R., et al., “A Cooperative Approach to User Mobility,” ACM CCR, vol. 31, 2001.

[13] Kubiatowicz, J., et al., “OceanStore: An Architecture for Global-Scale Persistent Storage,” Proceedings of the 9th ASPLOS 2000, November 2000.

[14] Lampson, B., “Hints for Computer System Design,” Proc. SOSP 1983, pp 33-48.

[15] Li, Z., Wang, C., Xu, R., “Computation Offloading to Save Energy on Handheld Devices: A Partition Scheme,” ACM CASES’01, Atlanta, November 16-17, 2001.

[16] Milojicic, D., et al., “Peer-to-Peer Computing,” HPL technical report, HPL-2002-57, 2002.

[17] Milojicic, et al. Pervasive Services Infrastructure, Proceedings of TES’2001, pp 187-200.

[18] Messer, A., et al., “Towards a Distributed Platform for Resource Constrained Devices,” to appear in ICDCS, Vienna, Austria, 2002.

[19] Nagaraja, K., et al., Adaptive Infrastructure for Mobile Ad-hoc Communities,” submitted for publication, 2002.

[20] Norman, D. A., “The Invisible Computer,” Cambridge, MA, MIT Press, 1998.

[21] Philippsen, M., and Zenger, M., “JavaParty - Transparent Remote Objects in Java,” Concurrency: Practice and Experience, 9(11):1125–1242, November 1997.

[22] Ratnasamy, S., et al., “Scalable Content-Addressable Network,” Proceedings of the SIGCOMM, pp 161-172, 2001.

[23] Rollins, S., et al., “Pervasive P2P: Aggregation of Resource Devices a Peer-to-Peer Environment,” submitted for publication.

[24] Rowstron, A. and Druschel, P. 2001. Storage Management and Caching in PAST, a Large-Scale, Persistent, Peer-to-Peer Storage Utility. *Proceedings of SOSP*, pp 188-201.

[25] Saltzer, J.H., Reed, D.P., Clark, D.D., “End-To-End Arguments in System Design,” ACM TOCS, vol. 2, no. 4, November 1984, pp 277-288.

[26] Satyanarayanan, M., “Fundamental Challenges in Mobile Computing,” Proc. of 15 ACM Symp. PODC, May 1996, pp 61.

[27] Satyanarayanan, M. 2001., “Pervasive Computing: Vision and Challenges,” IEEE Personal Communications, August 2001.

[28] Scaglia, P., personal communication.

[29] Thai, T., Lam, H., “.NET Framework Essentials,” O’Reilly, 2001.

[30] Weiser, M., “Some Computer Science Problems in Ubiquitous Computing,” CACM, July 1993, 75-84.