Places to stay on the move

Software architectures for mobile user interfaces

extended abstract

Alan Dix
aQtive limited and Staffordshire University

Devina Ramduny
Staffordshire University

Tom Rodden, Nigel Davies.
Lancaster University

http://www.hiraeth.com/alan/topics/mobile/

Abstract

Architectural design has an important effect on usability, most notably on temporal properties. This paper investigates software architecture options for mobile user-interfaces, in particular those for collaborative systems. One of the new features of mobile systems as compared with fixed networks is the connection point to the physical network, the point of presence (PoP), which forms an additional location for code and data. This allows architectures that bring computation closer to the users hence reducing feedback and feedthrough delays. A consequence of using PoPs is that code and data have to be mobile within the network leading to potential security problems.

Keywords: mobile computing, collaborative work, CSCW, software architecture, client-server

Introduction

At first sight it seems that software architectures are about the internals of system design and not a necessary concern for the user interface. However, the merging of computing and communication systems and the maturing of distributed computing techniques has started to altered the way in which we view our interaction with computer systems. With the growing prominence of an almost universally accessible information infrastructure we are increasingly seeing our interaction focus on the communication infrastructure rather than the devices that access it.

The rapid development of the World Wide Web represents the most dramatic example of this shift in how we view our interaction with computer systems. The popular acceptance of massively interconnected computer systems has in the nature of computers has seen computer and communication systems being seamlessly interwoven within the everyday life of the general public. Interaction is now routinely with the web (rather than the machine used to access it) and our everyday experience of this interaction makes inherent assumptions about the architecture of the infrastructure.

As the opportunity to access the underlying infrastructure increases with the development of mobile devices the link to the underlying architecture is likely to become even more significant and the need to consider the architecture of the infrastructure underpinning user interaction will grow.

This paper considers the issues of software architectures and their implications for interaction with a particular emphasis on the implications for mobile interfaces. Our particular concern is the dynamic nature of the infrastructure required to support mobile and context sensitive applications and the impact on the relationship between interaction and architecture.

Software Architectures in HCI

Concerns about the architecture of interactive systems are not new to HCI. Experience has shown that the internal structure of a system has a dramatic effect on its external behaviour [Gram 1996]. Because of this user interface architecture has been a concern for many years including the development of Seeheim, MVC, PAC and Arch/Slinky models as well as the whole stream of UIMS systems [Pfaff, 1985; Lewis, 1995; Coutaz, 1987; UIMS, 1992].

In collaborative systems research the interest in software architectures has continued with CSCW architectures and toolkits including Rendezvous, MEAD, Suite and Groupkit [Hill, 1994; Bentley, 1994; Dewan, 1990; Greenberg 1994]. These collaborative systems almost always imply some form of networked solution, and increasingly single user systems also involve access to central information giving rise to a whole industry in client-sever and n-tier applications.

The emergence of distributed architectures to support interaction across geographically disparate communities of users has seen a series of debates about the nature of these architectures and their impact on interaction. This debate has tended to centre on the propagation of the effects of the actions of users to others involved in the activity being supported. The core issue in this discussion has been the tension between the responsive nature of replicated architectures that allow feedback to be provided locally and the need for some centralised component to make users aware of the action of others.

The majority of these arrangement tend to assume 'control' over the entire system, with bespoke software running at the users' own workstations and at various central servers. The implicit assumption in many of these systems is that the machines are connected to a single local area network. However, two developments have challenged this assumption and hence the whole basis for network-based user interfaces.

The first is the World Wide Web which in addition to highlighting the importance of the infrastructure has suggested alternative architectural arrangements for applications. A Web 'application' may include code running at a web-server (via CGI scripts or other server side technology), web pages displayed on browsers of many different kinds, and applets or similar downloaded code [Dix 1998]. In previous work we and others have investigated the ways in which CSCW architectures can be married to the web infrastructure [Bentley, 1997; Clarke, 1999; Palfreyman, 1996; Ramduny, 1997].

The second development is the massive growth in mobile communications and mobile computing. Although the end points here may be well understood (although in the case of small mobile devices difficult to design for), the network itself is far less controlled than even the Internet, with limited bandwidth, temporary disconnection, and an ever changing network topology. The design of appropriate user interfaces for this environment is becoming an increasingly important topic [Borovoy, 1998; Davies, 1994; Dix, 1995; Johnson, 1997, 1998; Joseph, 1995; Long, 1996; Want, 1995].

The need to consider the dynamic nature of this infrastructure places new demands on the software architecture and the overall role of the architecture. Essentially, software architecture is about 'what goes where'. In stationary networks, the 'where's tend to be fairly obvious and are normally characterised as either clients or servers. Even this can lead to a rich set of architecture alternatives. In mobile systems the changing network topology suggests a much richer set of alternative possibilities.

Considering these new and emerging arrangements provides the focus for this paper. We wish to consider what possible arrangements exist for mobile and dynamic infrastructures and the implications for future networks. To achieve this we will start by revisiting the topology of interactive single-user and collaborative systems on fixed networks. We will then go on to see how this changes when we consider mobility.

Architectures for static networks

In the traditional single-user arrangement for networked interaction, a single user client usually interacts with a single server (Figure 1). The user interface sits at the client-end while the data is on the server. The choice between opting for a thin or thick client affects the performance of the application and depends on many factors, not least the volume and rate of change of central information – which is itself a form of weak collaboration.

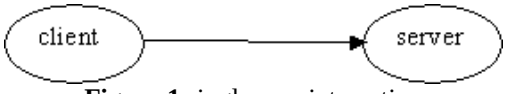

Figure 1 single user interaction

In multi-user collaborative systems the arrangement becomes more complex as we may have one of more clients handling the demands of the community of users. These clients may in turn interact with one of more servers via some inter-process communication mechanism. For the sake of simplicity we will limit interaction to take place through a single server as shown in Figure 2.

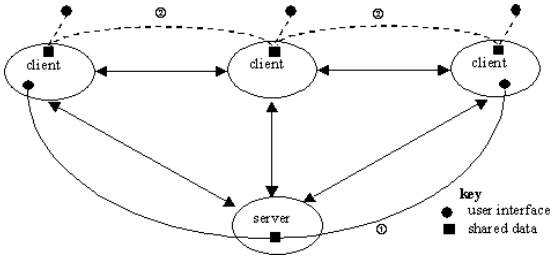

Figure 2 collaborative networked system

The two main architectures that emerge from the above are ① a centralised architecture and ② a peer-peer architecture.

In a centralised architecture, each user's client manages the screen layout and accepts inputs. The server holds the shared data and receives all users input events. Output events are broadcast by routing them through local client programs to all the users. As all the data is held centrally, access management, concurrency control and data consistency is simplified. However, a server failure due to a network breakdown or to a delayed feedback especially in a distributed setting, can give rise to a deadlock state. Client-server architectures has been adopted in conferencing systems, such as

MMConf [Crowley., 1990] and shared window systems like shared X [Gust, 1988]. Centralised architectures can easily support WYSIWIS or presentation level sharing as the server broadcast the output to all the clients. View level sharing can also be supported as demonstrated by the Rendezvous system [Hill, 1994].

In a replicated or peer-peer architecture, a separate copy (or replica) of the application runs on each workstation, which executes the application code and send output to the local user. To ensure synchronisation among the different replicas, input from each workstation is sent to each replica. The copies then communicate with each other to maintain data and interface consistency. Each replica handles its own screen management and user's feedback locally and must also update the screen in response any change in application data from other replicas. The replicated approach offers the advantages of a centralised architecture with the added benefits of performance as the output of a workstation is produced by a local workstation. Because the clients can be managed locally, alternate views are supported and it is relatively easy to provide end-user interface tailoring (Bentley et al). However, the major difficulties with replicated architectures lie with synchronising and maintaining data consistency. For example, if a user deletes a selected object in a WYSIWIS group drawing program while another user is changing the selection to a different object, inconsistent interfaces can result due to events arriving in a different order at each workstation.

WWW-based collaboration

The web has had a significant impact in increasing the prominence of the underlying infrastructure but many ways the impact on the architecture has been less dramatic. However, applet security mechanisms make true peer-peer architectures difficult except via some form of 'post office' server as found in various chat programs [Welie, 1996; Yahoo!] or a client-end plug-in [ICQ].

The use of applets on the web also opens up the possibility that code may be executing on a client, but have its permanent home on a server. The interactions between applet-based code mobility and the movement of data via caching has been used to classify the different modes of web-based architecture [Ramduny, 1997].

The movement of code and functionality inherent within applets also starts to alter our consideration of architectures. If architectures are about deciding where things are what happens when the supporting mechanisms allow them to easily move. The need for the infrastructure to exhibit dynamic becomes even more acute when we start to consider the support of devices that are also mobile.

Mobile Architectures

On the web applet code may be mobile itself, but usually runs on static computers. With truly mobile computing the devices themselves move. In a paper at the previous workshop on 'Human Computer Interaction with Mobile Devices', we investigated various kinds of 'mobility' [Rodden, 1998], but for the purposes of this paper we focus on physical mobility of devices. However, we will find that like the web, this tends to also require computational mobility.

Computational mobility means that computation may start at one network site, but then move and continue to execute at another network site. Mobility may involve:

- *code mobility*: very useful as demonstrated by the increasing use of Java applets.
- *control mobility*: moving a thread of control from one network point to another which then returns to its originating point, for example Remote Procedure Call (RPC)
- *data mobility*: data is exchanged over the network in the form of parameters
- *link mobility*: endpoint of one network connection is sent to another network connection to allow the receiving party to connect to it.

Computation is not limited to code. Rather computation is the combination of the code and the context of its execution. When code is moved from one network point to another, the current state of the execution is lost and the connections that the computation had at its original site no longer exist. The code can only execute at the remote end if state and connectivity is re-established at the receiving site. Therefore control must move through some form of dynamic binding and data must also migrate in order to preserve the state of the computation. As network links form part of the state information they must move as well. The location of computation is crucial in determining the effectiveness of mobile applications. Computational location influences application behaviour and resource usage.

Points of Presence

To understand the need to consider location and mobility within the architecture let us introduce the notion of a *Point of Presence* (PoP). Again, let's begin with the case of a single-user application over a mobile network. In the static case this involved simply the user's client machine and the central server. In the mobile case, in addition to the client and server there is some form of *Point of Presence* (PoP) where the client machine has its first connection to the physical network (Fig. 3).

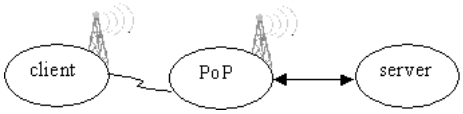

Figure 3 Points of Presence

In the case of cellular mobile-phone-based connectivity this PoP would be the local cell's base station accessed via radio, in the case of a small hand-held PDA this may be a desktop computer accessed by an infra-red port or fixed cradle.

This effectively forms a third place where computation and data may reside. For mobile devices a PoP has both better network connectivity and potentially greater computational power than a hand-held or body-adorning device, but it is also close to the user and will be able to engage in a faster pace of interaction than a server-based interaction. It is therefore a natural place for part of a user-interface.

The role of a PoP is to make the interactions of the user present on the network and to mediate in appropriately presenting the effects of interaction from the network to the user. This is somewhat like what is being seen with proxy-based web services for hand-held computers [Fox 1996]. When used in this classic client server arrangement it is in fact yet another server rather than the actual PoP, but is one that is 'mobility aware' as compared with the web server itself.

Just as the single server in client–server applications may in fact be several servers for different databases or in the case of the web several web-servers, we will use the term PoP to include not just literally the first point of contact, but also 'close' points. The defining feature is that these are locations determined by the mobility and location of the device rather than the intrinsic location of shared data.

Collaborative Points of Presence

In considering Points of Presence let us turn now to collaborative systems. Again we may have many users' client computers (hand-held devices, wearables etc.) and one (or more) central servers. However, for each client we now have a PoP and these clients become present in the network through these Points of Presence. Some clients may be close enough to have a shared PoP others may have completely different PoPs. Also very close clients may even be able to communicate directly rather than via the fixed network, for example, Palm Pilots that can communicate via iRDA.

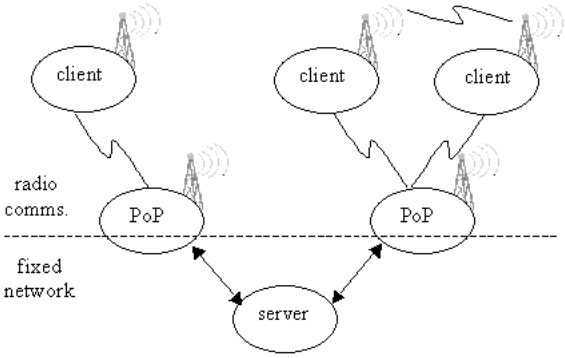

Figure 4 Mobile collaborative network

In this arrangement we have three possible places where data and programs may sit: client, server and PoP; and also a variety of communication paths. In the following section we will investigate six possibilities in terms of three different cases.

Arranging Points of Presence

To make the three different cases we wish to consider clearer we will draw upon a running example. Imagine two people, Alison and Brian, using some form of digital paper. The target application is a shared drawing tool – as Alison draws on her pad with a stylus, marks appear on her own digital paper and also (with some feedthrough delay) on Brian's pad. The question then is what forms of software architecture may best support the interaction needed for our shared digital paper.

Case 1 – Static network PoPs

The first pair of architectures are those when we effectively ignore the mobile nature of the network and treat it exactly like the static network.

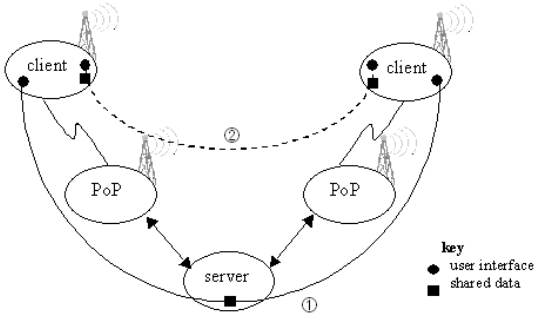

Figure 5: The static network arrangement

For fast and reliable networks, such as the roving radio-based ethernets for use within offices this arrangement makes considerable sense. Two sensible application configurations are possible under this arrangement.

① *server-based centralised architecture*

Shared data is held at the server end.

② *peer-peer architecture*

Communication is from the client down to the network and back to the other client, just as when two users communicating with each other via a mobile phone.

The PoP has no computational role in either case, being merely a router or post office passing on communication. This is effectively the same as ① and ② in figure 2 except that the network has both mobile and fixed links. The issues are the same as for fixed networks, except the delays may be longer. In case ① Alsion may experience some delay in getting feedback for her actions (disconcerting on digital paper) in case ② both Alison and Brian's digital pads contain all the shared data and have to communicate to maintain a consistent replicated state.

Case 2 – Power in the PoP

Now consider a similar case, but where the client computation is spread between it and the PoP. This may be useful if the drawing pads supporting the digital paper have only a small amount of on-board memory or limited computational power.

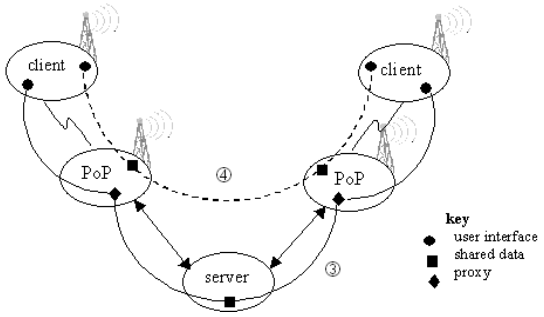

Figure 6: The use of computational PoPs

In figure 6, we see versions of ① and ② where the PoP has a greater role. Two potential configurations emerge.

③ *centralised data, PoP as intermediary*

The data sits at a central server, but the PoP takes an active role as a proxy client/server. The PoP runs part of the application and also communicates with the client for I/O.

④ *decentralised replicated solution*

Replicated data sits at the PoP (and possibly part of the application) – the PoP acts as a virtual server. Although the data resides at the PoP, it is likely to reside their only temporarily, so long as there are local users. (The exception being ubiquitous data such as telephone directories.)

One example of ③ would be if the digital pads were configured as X servers (this in fact is the arrangement used in the early versions of the PARC ubicom environments), the shared drawing program could then be comprised of X clients at each PoP which each access a shared server. As in case ①, Alison will experience feedback delays, unless there is additional caching.

Although we described ④ as a virtual server, it will have a far greater pace of feedback than a centralised solution – Alison's stylus strokes only have to register on the replica at the local PoP before being echoed back to her pad.

Case 3 –Moving PoPs together

Finally, there are two options that are only possible when devices are physically close. For example, imagine that Alison and Brian have brought their digital pads with them to a meeting. As they talk they start to sketch on their pads. If any of the previous architectures are used the feedthrough of Alison's actions on Brian's pad will experience full network delays, little different than as if they were hundreds of miles apart. However, being so close they might reasonably expect virtually instantaneous response.

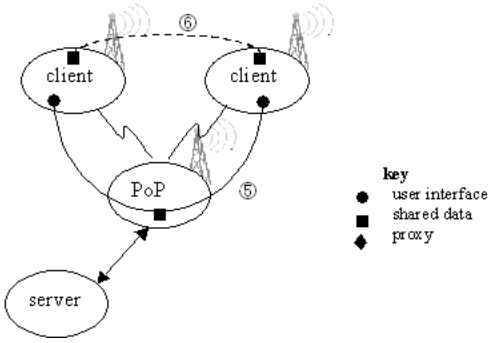

Figure 7: colocated PoPs

Two arrangements exist that might enable the level of response anticipate by Alison and Brian.

⑤ *distributed centralised solution*

The server gives some of the functionality to PoP to allow the clients to communicate with the PoP – the PoP therefore assumes the role of server

⑥ *peer-peer communication*

Direct communication using local communication (e.g. infrared). The shared data is at the client end.

⑤ and ⑥ are similar to ① and ② in figure 2 but the communication takes place entirely in a mobile environment, thus ensuring faster feedback and feedthrough.

Of the two ⑥ is very simple in that it is really only a local network version of ②, the only difference being a greater pace of feedthrough for Brian and the problems of hand over etc. within the network.

Although in some ways ⑤ is the same as ① it has the crucial difference that the machine to act as the server for the shared application, the PoP, is dynamically configured rather than at a fixed

location. For example, if Alison and Brian are in a reactive meeting room, the PoP may be in a device embedded within the wall that senses the presence of the two pads and downloads relevant software from a remote server. Thereafter Alison will experience almost instant feedback (just a roundtrip to the wall and back) and Brian equally fast feedthrough. In addition, being a centralised application is has all the advantages of software simplicity that go with server-based solutions.

Issues

As we expected, the mobility of physical devices means a constantly shifting topology within the infrastructure and so, as with web-based solutions, issues about data and code mobility resurfaces in mobile settings.

In cases ④ and ⑤, some of the functionality of the server has migrated to the PoP. In case ⑥, it is some of the client functionality that is in the PoP. So how does it get there? Unless the application is ubiquitous it will not be sensible to have copies of the code sitting at every PoP, so it must either download from the client or a server. Also in cases ④ and ⑤ shared data must migrate to the PoPs.

The resulting security and management implications are somewhat daunting: will public carriers allow foreign code to run *inside* their network, will users trust devices embedded in the environment to run parts of their (perhaps private) applications.

However, the usability (and indeed infrastructure efficiency) gains of moving computation close to users is obvious. The demands to move a reconfigure netorks in tis way suggest that much of the on-going work in 'active networks' within the communications community may hold some promise.

References

1. Bentley, R., Rodden, T., Sawyer, P. and Sommerville, I. (1994) Architectural support for cooperative multi-user interfaces. In *IEEE COMPUTER special issue on CSCW*, **27**(5), pp 37-46.
2. Bentley, R., U. Busbach, D. Kerr and K. Sikkel (Eds.), (1997). *Groupware and the World Wide Web*, Dordrecht, Kluwer
3. Borovoy, R., Martin, F., Vemuri, S., Resnick, M., Silverman, B. and C Hancock (1998) Meme tags and community mirrors: moving from conferences to collaboration, *Proceedings of the ACM 1998 conference on Computer supported cooperative work*, pages 159-168
4. Clarke, D. and A. Dix (1999). Proceedings of The Workshop on the Active Web. 20th January 1999.
5. Coutaz, J. (1987). PAC, an object oriented model for dialogue design. *Human–Computer Interaction – INTERACT'87*, Eds. H.-J. Bullinger and B. Shackel. Elsevier (North-Holland). pp. 431-436.
6. Crowley (1990) … MMConf …
7. Davies, N., G. Blair, K. Cheverst, and A. Friday. "Supporting Adaptive Services in a Heterogeneous Mobile Environment." *Proc. Workshop on Mobile Computing Systems and Applications (MCSA)*, Santa Cruz, CA, U.S., Editor: Luis-Felipe Cabrera and Mahadev Satyanarayanan, IEEE Computer Society Press, Pages 153-157. December 1994.
8. Dewan, P, A tour of the Suite user interface software, *in Proceedings of UIST'90* (1990), ACM Press, 57-65.
9. Dix, A. J. (1995). Cooperation without (reliable) Communication: Interfaces for Mobile Applications. *Distributed Systems Engineering*, **2**(3): 171–181.
10. Dix, A. (1998). The Active Web - Parts 1&2. *Interfaces*, 38:18-21, 39:22-25.
11. Glasgow University (1998). *Workshop on Human Computer Interaction with Mobile Devices*, Glasgow, 21st & 22nd May 1998
12. Gram, C. and G. Cockton, Eds. (1996). *Design Principles for Interactive Software*. UK, Chapman and Hall.
13. Greenberg S.,Marwood D., 'Real Time Groupware as a Distributed System; Concurreny Control and its effect on the Interface' Proceedings of CSCW'94, North Carolina, Oct 22-26, 1994, ACM Press.
14. Gust (1988) … shared X …
15. Hill, R.D., Brinck, T., Rohall, S.L., Patterson, J.F. and Wilner, W. (1994). The Rendezvous architecture and language for constructing multi-user applications. *ACM Transactions on Computer-Human Interaction*, **1**(2), 81–125.
16. ICQ ("I Seek You"). http://www.icq.com/
17. Johnson, C. W. (1997). The impact of time and place on the operation of mobile computing devices. *Proceedings of HCI'97: People and Computers XII*, Bristol, UK, pp. 175–190.
18. Johnson, C. (ed.), (1998). *Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices*. University of Glasgow, 21-23rd May 1998., GIST Technical Report G98-1.
19. Joseph, A., A. deLespinasse, J. Tauber, D. Gifford, and M.F. Kaashoek. "Rover: A Toolkit for Mobile Information Access." *Proc. 15th ACM Symposium on Operating System Principles (SOSP)*, Copper Mountain Resort, Colorado, U.S., ACM Press, Vol. 29, Pages 156-171. 3-6 December 1995.
20. Lewis (1995). *The Art and Science of Smalltalk*. Prentice Hall.
21. Long, S., R. Kooper, G.D. Abowd, and C.G. Atkeson (1996). "Rapid Prototyping of Mobile Context-Aware Applications: The Cyberguide Case Study." *Proc. 2nd ACM International Conference on Mobile Computing (MOBICOM'96)*, Rye, New York, U.S., ACM Press,
22. Palfreyman, K. and Rodden, T. A Protocol for User Awareness on the World Wide Web, in *Proceedings of CSCW'96*, (Boston, Massachusetts, Nov. 1996), ACM Press, 130–139.
23. Patterson, J.F, Day, M. and Kucan, J. Notification Servers for Synchronous Groupware, in

Proceedings of CSCW'96 (Cambridge Massachusetts, 1996), ACM Press, 122–129.

24. Pfaff, G. and Hagen P.J.W., editors (1985) Seeheim Workshop on User Interface Management Systems, *Springer-Verlag, Berlin*.
25. Ramduny, D. and A. Dix (1997). Why, What, Where, When: Architectures for Co-operative work on the WWW. *Proceedings of HCI'97*, Bristol, UK, Springer. pp. 283–301.
26. Rodden, T. , K. Cheverst, N. Davies and A. Dix (1998). Exploiting context in HCI design for Mobile Systems. Workshop on Human Computer Interaction with Mobile Devices, Glasgow, 21st & 22nd May 1998.
27. UIMS (1992) The UIMS tool developers workshop: A metamodel for the runtime architecture of an interactive system. In *SIGCHI Bulletin*, **24**(1), pp 32-37.
28. Want R., Schilit, B. N., Adams, N. I., Gold, R., Petersen, K., Goldberg, D., Ellis, J. R. and M. Weiser (1995) An Overview of the ParcTab Ubiquitous Computing Experiment. IEEE Personal Communications, December 1995, Pages 28-43.
29. Welie, V.M. and Eliëns, A. (1996) Chatting on the Web. In ERCIM workshop on CSCW and the Web (Sankt Augustin, Germany), GMD/FIT.
30. Yahoo! Chat. http://chat.yahoo.com/
31. Fox, A., S.D. Gribble, E.A. Brewer, and E. Amir. 1996 "Adapting to Network and Client Variation via On-Demand, Dynamic Distillation." Proc. ASPLOS-VII, Boston, Masachusetts, U.S.,