Seamless Mobility with Personal Servers

Markus Bylund[1] (bylund@sics.se) and Zary Segall[2] (zary@zary.com)

[1]Swedish Institute of Computer Science, Box 1263, SE-164 29 Kista, Sweden
[2]Royal Institute of Technology, Forum 100, SE-164 40 Kista, Sweden

Abstract. We describe the concept and the taxonomy of personal servers, and their implications in seamless mobility. Personal servers could offer electronic services independently of network availability or quality, provide a greater flexibility in the choice of user access device, and support the key concept of continuous user experience. We describe the organization of mobile and remote personal servers, define three relevant communication modes, and discuss means for users to exploit seamless services on the personal server.

Introduction

The term Ubiquitous Computing [1, 2] refers to a vision of invisible computers being embedded in our environment and participating in our lives. The envisioned usage however, is far from the way that we use computers today. Instead of having multi-purpose computers such as desktop or laptop PCs, computers will blend into our environment and turn into invisible and special purpose devices that will help us to accomplish tasks everywhere, not only when sitting by our desk. Using computers for various purposes, according to the vision, will be as much of an unconscious activity as using the nowadays-ancient technology of writing for long-term storage of information.

It is anticipated that next-generation personal computers will support a variety of more interactive and proactive modes of computation than we can see today. As embedded systems and networks become mobile, personal mobile computers will act as both agents and intermediaries between their users and the embedded system infrastructure. Miniature, portable server systems will allow their users to seamlessly access networks, retrieve data from embedded sensors, control embedded actuators, perform functions proactively on the user's behalf, and interact with other people's personal systems as people come into proximity. For such systems, the locus of control moves through the environment, interacting proactively and often autonomously in response to real-time, environmental inputs. For the purposes of this paper, we will call such a personal control system a *personal server*.

Personal servers provide a variety of new, dynamic modes of operation and interaction depending on the environment and the user's needs – in other words, it is potentially highly suitable for realizing seamless mobility. In the simplest case, we can visualize the use of a personal server in terms of the user equipped with such device, walking up to a monitor, keyboard and other peripherals with wireless interfaces and beginning to work. His or her personal server connects to the surrounding peripherals, devices, and networks, allowing access to local data or control systems. Since the locus of control moves with the user, it arrives fully customized and able to personalize the user's interaction with environmental systems (e.g., to the user's job, capabilities, goals, age, etc.) and employ the user's own set of application software.

The remainder of the text is outlined as follows. In the next section, we expand the term seamless mobility to include aside from network capabilities and coverage, also UI and device flexibility as well as continuity of user experience. This expansion is later exemplified with the use of a sample service – the aWare Messenger. The concept of personal servers is then treated in some detail, including aspects such as service access and software support. Finally, a discussion about how personal servers support seamless mobility concludes the paper.

Expanding Seamless Mobility

Seamless mobility is often referred to as the marriage between 3G (or 2.5G) and 4G[1] technologies. With the wide coverage of 3G technologies, combined with the high but local performance of 4G, it is argued that electronic services can be used ubiquitously. However, we believe that there is more to it than performance and coverage of network connectivity. In this section we highlight three factors that we view as particularly important in order to achieve seamless mobility: the first being network capabilities and coverage, the second user interface (UI) and device capabilities, and the third the importance of true user experience continuity.

Network Independence

Many, if not most, discussions about mobility have so far been about network issues, ranging from how to provide support for mobility in network protocols (for example Mobile IP [3]) to discussions about different solutions for wireless networking. The main reason for this is that different technical constraints influence what can be done in mobile settings. Bandwidth for example, can easily vary with a factor of 1000 depending on network connection (from GPRS with less than 100 kb/s to wired networks with 100 Mb/s or more). Variations in latency[2] are almost as dramatic – a factor of 100 can easily be found (from several seconds with GPRS down to milliseconds with wired networks).

However, the performance of network connection in terms of bandwidth and latency is not the only factor that should be considered when evaluating the impact on seamless mobility. The possibility to roam between different network operators will in practice also influence the mobility. The pricing of different types of network connections is yet another factor.

All these factors make for a quite uncertain situation resulting in the fact that an adequate network connection everywhere cannot be taken for granted. Considering this, we argue that to achieve seamless mobility we must not only consider how to provide for mobile network connectivity, but also how to enable users to operate services independently of (Internet) network connectivity. This would unburden users from issues such as low bandwidth, high latency, roaming issues, and high traffic costs. We recognize that some services require a real-time connection (services for inter-person communication or data intense services for example), but most services include at least parts that could work without network connection.

Below, we further elaborate on some of these issues.

Testing the Performance of Mobile Network Connections

For quite a few classes of applications, the performance of GPRS is too low – for example, highly interactive applications such as games, where the latency is the most limiting factor. This is however also the case for as simple applications as web browsers. A web page is typically made out of several dozens of data entities which all need to be fetched with HTTP requests. For example, it typically requires several minutes to load the first page of the Swedish newspaper Dagens Nyheter (a total of about 300 kB and several dozens of entities to download) to a Sony Ericsson P800 smart phone over a GPRS connection. Loading the same page on a desktop computer with a high speed wired connection requires less than a second. By introducing a proxy close to the wired-wireless border, the performance of GPRS can improved [4], but the gap to 4G technologies and wired networks is still huge.

As part of an evaluation of the Sony Ericsson P800, we made a few tests of the combined impact of these differences (a complete description of these tests can be found in a separate technical report [5]). A simple Java application running on the phone made a series of HTTP

---

[1] In this context, it would be equally true to replace 2.5G with General Packet Radio Services (GPRS), 3G with Universal Mobile Telecommunication Service (UMTS), sometimes referred to as 2.75G, and 4G, sometimes referred to as "beyond 3G", with integrated network technologies including 802.11 and other standards for wireless communication.

[2] Throughout this discussion, we define latency as the roundtrip time of a small (<512 bytes) message.

			HEAD		GET		
			Latency	Std. Dev.	Latency	Std. Dev.	Bandwidth
Sony Ericsson P800	**USB**		220 ms	7 ms	2,200 ms	290 ms	200 kb/s
	BT		290 ms	14 ms	2,400 ms	710 ms	180 kb/s
	GPRS		2,600 ms	250 ms	15,000 ms	3,700 ms	29 kb/s
iPaq 3870	**WLAN**		140 ms	4 ms	260 ms	4 ms	2 Mb/s
Laptop (PIII, 600MHz, Win 2k)	**LAN**		<10 ms	5 ms	10 ms	5 ms	32 Mb/s
	BT/GPRS		1,600 ms	150 ms	14,000 ms	2,100 ms	32 kb/s

Table 1. A comparison between different means for accessing Internet based services from different mobile devices. The HEAD request type gives a practical estimate of the minimal latency of the connection while the GET request type also takes the factor of bandwidth into account. Each number is an average of 25 repetitions of each request. The bandwidth is estimated based on the GET response time.

requests with the phone connected to the Internet via GPRS as well as Bluetooth and USB cable via a laptop computer (which was connected to a high speed wired LAN). The response time between the network connections turned out to be equal for USB and Bluetooth but many times as high for GPRS (see Table 1). A single HTTP HEAD request, which included less than 1 kB of content, required several seconds to complete when operating via GPRS. When the request was changed to HTTP GET, which in addition downloaded a 55 kB large entity, response times were raised to 15 s on average.

For comparison, a Compaq iPaq 3870 PDA with a WLAN card was also tested. The PDA displayed a latency that was only 60% of the phone with its fastest connection (USB) and a ten times as high bandwidth. The PDA differ from the phone in a number of ways (processor, OS, and Java VM), but the differences are not big enough to explain the huge difference in network performance that the test reveals. It is therefore likely that the phone would display similar performance (as the iPaq) if it could be equipped with a WLAN network connection.

The application was also tested on a laptop with two different network connections in order to find out the maximum impact of other factors. The laptop test with the LAN connection concluded that the impact of delays caused by the Web server that was targeted throughout all tests was less than 10 ms, which makes this delay negligible. Tests on the laptop with the BT/GPRS (Bluetooth connection to P800 phone, GPRS connection to the Internet) connection revealed that the overhead due to (unknown) issues with the P800 phone was at most 1 s per request (independently of request type). The latter test turned out to be irrelevant since the P800 phone was provably capable of executing a request in less than 300 ms, which is far less than the 1 s difference between phone and laptop.

The small difference between the USB and Bluetooth connections came as a surprise. With a theoretical maximum bandwidth of 2 Mb/s for the USB connection, and 723 kb/s for Bluetooth, we expected the USB connection to be two to three times as fast as the Bluetooth connection, at least in the case of the GET request. One possible reason for this not being the case is that the Sony Ericsson Phone Connection PPC Suite, which provides network pass trough from laptop to phone, is implemented in a way that limits the performance of the USB connection.

It was also interesting to notice that the variation in response time for the slow GPRS connections was exceptionally high. These variations could be explained by variations in GPRS channel availability, but also TCP retransmissions due to the high latency of the connection [4].

Other Factors but Performance

The performance of network connections however, is only one of several factors that influence mobility. The possibility to roam between different types of networks and networks of different service providers with a continuous connection is another factor. On the one hand, this is also a technical matter with several existing solutions [3, 6]. On the other hand, it is a matter of more practical nature. In order for users to get seamless access to network connectivity from every possible place, all service providers need to agree on automatic roaming or users need to subscribe to all service providers. Given the diversity and large number of service providers, neither factor is likely to work well enough in order to take mobile network access for granted.

Pricing of network connectivity is yet another factor that influence how users actually end up making use of connections in mobile settings. In the case of GPRS for example, the cost of service is usually a combination of a flat and traffic based rate, while the cost of most wired connections (for example cable TV and ADSL) is usually only based on a flat rate. This means that users that make use of for example both GPRS and ADSL connections have reasons to plan their usage of bandwidth intense services to situations when ADSL is available. It is even possible that some users reject GPRS only because of the variable cost associated with its usage.

UI and Device Flexibility

It is not productive to talk about seamless mobility if one does not consider that while being mobile, the user context is bound to vary over time. However, different contexts and situations require different types of devices and user interfaces. This has been acknowledged by the manufacturers of access devices. There exist today a wide range of devices for mobile computing – ranging from smart phones with calendar, contact list, and all sorts of messaging applications, PDAs of different kinds, to tablet PCs and laptops. These devices differ in a great number of ways, but most of the time, these differences are well motivated. The screen of a smart phone, and its adapted UI, is much smaller than that of the desktop computer simply because it would be burdening to carry a large display while walking around. Nevertheless, while sitting in an office most users prefer a device with a large display. These differences can be seen on all levels of design of the artifacts that we use to access electronic services. Some artifacts have hard buttons that immediately trigger specific applications, while others have full size keyboards. Some devices have scroll wheels for navigation while others have mice, and so on.

Judging by the diversity of mobile devices available today, it would seem as if the range of available artifacts provides good support for mobility as it is. This is not the case however, because the flexibility is not there yet. If one is to take full advantage of the differences between various artifacts, one must also be able to switch seamlessly between them. This is seldom possible with the infrastructure for electronic services and range of artifacts available today. Usually it is not even possible to use the same service and data on different artifacts at all, let alone in a seamless manner. In some cases it is possible to synchronize data between similar, but *different*, services on different devices. This is only true for the data however; the state of the service is seldom, if ever, included in the process.

Therefore, we argue that in order for seamless mobility to become a reality, we must find the means to support flexibility in the choice of device and UI by allowing access to the same services and data from many different types of devices.

User Experience Continuity

As stated above, seamless mobility requires some kind of network independence and flexibility in choice of user access device. These two factors make it possible to introduce *continuous user experience* for electronic services as we change network connectivity and switch devices as the situation requires. This would allow users to start working with a task on one device with a particular network connection, to continue the work on another device

completely without network connection (on an airplane or at a hospital for example), to finally finish the task on a third device and network connection.

From the users' point of view, this kind of continuity can take on two different shapes: continuity as in the *remote control* case, and continuity with *adapted user interfaces*. In the first case, the different access devices can be seen as remote controls to the electronic services of the user. The UI is identical in all cases, but the hardware running the UI changes. Applications such as the Virtual Network Computing (VNC) client [7] and Microsoft's Remote Desktop Connection[3] can provide exactly this. Thin clients that are small enough to execute on PDAs and smart phones (in addition to desktop/laptop computers), mirror the screen, keyboard, and mouse of another computer. The electronic services of the user executes on one and the same computer (usually a desktop computer or a server) and as the user switches access devices, only the stream of screen, keyboard, and mouse events need to change origin and destination. The execution state and data of the service is kept the same over time since that is never moved between the devices.

The second alternative, to adapt the UI to different user access devices, is more complex. In this case the UI of the electronic service is adapted to the capabilities and constraints of the device currently in use. A laptop may provide a wide overview of all functionality of the service including full access to input features while a much smaller smart phone might require a simplified UI. Output may be filtered (certain kinds of media may for example not be possible to show) and the means for input may be reduced because of the limited keyboard. On some devices, the adaptation may go so far as to abandon the prevailing desktop and windows metaphor in favor of something more suitable. This could for example be the case for a pure voice-based interaction device. However, this alternative for providing continuity is expensive and complicated to realize, mainly because it requires implementation and maintenance efforts that grow rapidly with the number of devices that should be supported. Another problem is that when a new device is introduced, a new adaptation (or version of the service) needs to be implemented. In Nylander et al. [8], we describe a methodology, including a fully functioning system, which reduces these difficulties when adapting UIs to different types of devices.

From the user perspective, these two alternatives have both advantages and disadvantages. The remote control alternative is preferable because the UI always stays the same – users immediately recognize the service when using it on a new device and there is no learning time that needs to be accounted for. However, since no adaptation of the UI is done the usability may suffer a lot on some devices. An example of the latter is the use of a complex application such as for example Microsoft Word on a smart phone. The small screen provides an exceptionally poor overview of all actions and capabilities of the rich UI, and input is extremely cumbersome with only scribbling or a virtual keyboard available. The adapted UI alternative is powerful since it allows services to take full advantage of the unique features of each device, therefore building on the knowledge and design expertise that was put into the development of each device. This means that special features such as hard buttons and scroll wheels can be assigned functionality that ties closely to the intentions of the designer of the device. However, since the one and the same service will appear differently on different devices, some learning time will be required for each new device being used.

The aWare Messenger – Seamless Mobility Exemplified

The aWare Messenger is a sample service that is designed for a mobile setting, and in this text, it serves to illustrate the different factors of seamless mobility stated above.

The aWare Messenger aims at reducing the barriers that physical distances generate between co-workers and peers. The system achieves this by limiting the drawbacks with not being physically close by creating a sense of presence, adding cues to reminders much in the same way as seeing a peer in a corridor or coffee room would, and revealing the degree by which a

---

[3] The Remote Desktop Connection is a standard feature of Microsoft's Windows XP.

Figure 1. aWare Messenger ultra-lite device worn as a wrist band.

peer is susceptible to a chat or other activity. In this way, the system will reduce the likeliness of misunderstandings, increase the inspiration that peers get from each other, reduce the need to "catch up", and create a greater sense of group belonging.

The aWare Messenger realizes this in two ways. First, by supporting simple asynchronous communication between remote peers, much in the same way as an instant messenger such as ICQ, the service allows for quick and simple questions or comments to be made. In contrast to traditional instant messengers though, the aWare Messenger allows communication over a wide range of UIs and modalities – be it a desktop computer with a GUI or a cellular phone through which messages can be dictated vocally and read out load. Alternatively, the ultra-lite aWare Messenger device (see Figure 1), which is worn on the wrist or attached to the sleeve, can be used. In situations where peers are available via different modalities, the service automatically makes the necessary translations.

Second, the system generates a sense of presence by mediating to all users which other (relevant) users are present in the system. In addition, by making use of sensors the system is able to mediate which mood a peer is in (in real life mediated by face expression or posture) as well as how susceptible each peer is to an activity. These inferences made by the system are of course possible to set manually or turn off.

A first prototype of the aWare Messenger has been implemented. This version bases its adaptation of susceptibility on the position of each user. Users are able to communicate with text messages asynchronously, and they can manually set their mood for other users to see. The service is accessible from desktop/laptop computers, Compaq iPaq PDAs and Sony Ericsson P800 smart phones. For the purpose of the discussion below however, we disregard the real-life limitations of the prototype and consider the full-featured vision described above instead.

From a network independence perspective, the aWare Messenger is capable of a number of adaptations in order to make the user experience as seamless as possible. With a high quality network connection, the service provides voice-based asynchronous communication in order to give users the possibility of operating the service hands-free. As the quality of the network connection drops, voice communication is no longer available and text-based messaging is offered instead. Finally, being completely without a network connection, no messaging can

occur and no information about peers can be received. However, the last known state of all peers is kept and users can still browse history logs of messaging and adjust preferences and settings.

The aWare Messenger also addresses UI and device flexibility. Mobility is clearly an important design criterion for a service of this kind. This makes it important to provide solutions that work with small devices that are designed for mobile use. However, for some users an equally important use situation will be in the office next to a desktop computer. In such a case it is not necessary to trouble the user with tiny displays and minimalist input peripherals. The service is therefore accessible from desktop/laptop computers as well.

In order to face the issue of user experience continuity, the state of the service is kept the same independently of access device. This means that the messaging history as well as preferences and settings (except for settings that only apply to a particular device) remain the same across all access devices. If the user makes changes on one device, for example enters a new nickname, the changes will be automatically mirrored to the user's other devices. It is even so that if the user is entering a message on one device, and switches halfway through the task, the user can continue with the editing on the new device exactly where it was left.

Personal Server

Recognizing the relevance of the three factors of seamless mobility outlined above, we can now introduce the concept of a *personal server*. A personal server is a host for the electronic services of a single user. The type of hardware is not important on this level; it could be a desktop computer just as well as a virtual server on a multi-user machine or a PDA. The important aspects of the concept are instead that (1) it is uniquely tied to a single user, and (2) that the user only needs to view the collection of functionality and data that the server constitutes as one single entity. The second aspect should be put in contrast to something that is (possibly poorly and incompletely) duplicated on several different devices, such as the calendar on the desktop computer, the one in the PDA, and the one on the screen fridge at home.

To use the services of the personal server, *user access devices* of different kinds are needed. Note that the user access device can be, but are not necessarily, the same device as the personal server is running on. In contrast to the uniqueness of the personal server (from the user perspective), user access devices are many, one for each context or situation.

Other definitions of the personal server concept exist, such as the one of Want et al. [9, 10] for example. Our concept is somewhat more comprehensive than that of Want et al., although theirs is included in the Personal Server and identified as a wearable server (see below).

Personal Server Characteristics

The personal server concept encompasses a large number of solutions for electronic service mediation. In order to evaluate the concept from a seamless mobility perspective, we need to explore some of its characteristics somewhat deeper (see Figure 2). The most obvious characteristic is whether the personal server is *mobile* (i.e. something that the user can bring along and access locally) or *remote* (i.e. something that is placed on a fixed location and accessed remotely).

Remote Server

The advantage of having a remote server is that services get access to a wired, and possibly high quality, network connection at all times. Services can execute continuously assisting the user even when the user is not in contact with the remote server (e.g. a broker service buying and selling stocks on behalf of its user). However, being remote also implies the need of a network connection between server and user access device in order for the user to interact with the services. The latter violates the desire to gain independence of network connection as described above, but depending on user needs this may be balanced by the possibility to have personal services executing continuously regardless of the user's whereabouts. The remote
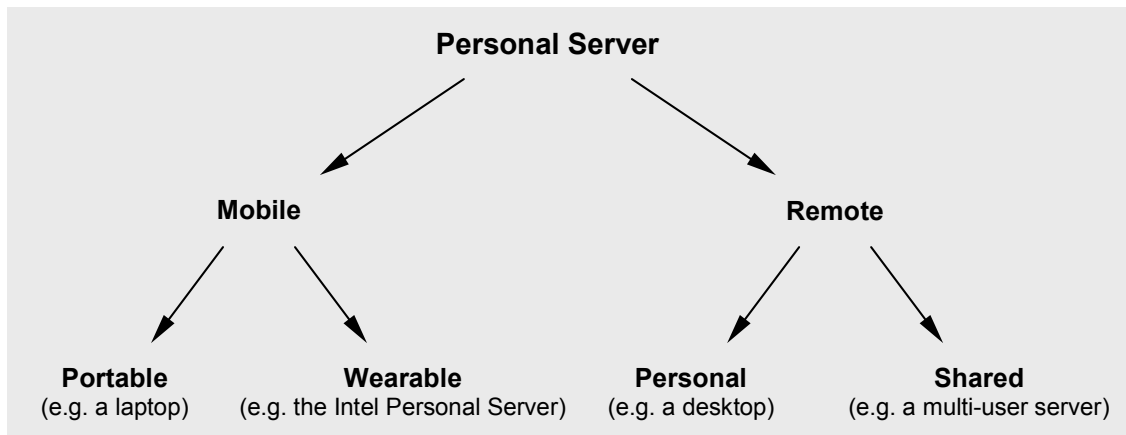

Figure 2. A taxonomy of different types of personal servers.

server can be further divided into two groups depending on whether it is *shared* or *personal*. A shared server would typically be owned and maintained by a service provider or a corporation that would host services for individual users, while the personal server could be an individual user's desktop computer at home or at work. This distinction however, is not significant when evaluating the concept from a seamless mobility perspective.

Mobile Server

The advantage of the mobile server is that it can be brought along by its user. This allows the user to access services even when there is no network connection available at all. Of course, parts of the functionality of some services require a network connection in order to work, but most services can at least offer some functionality in a completely disconnected mode. A chat service for example, inherently requires a network connection to other chat peers in order to function. But the service could at least offer history browsing and the editing of preferences when no network connection is available. As discussed above, the user might even prefer a local off-line mode even if a network connection is available, this in order to minimize the cost of network traffic. In such a case, services can be programmed to minimize network traffic by only sending and receiving necessary information. An e-mail client for example, could be set to only download the headers of new e-mails and the e-mail bodies only upon explicit user request (just as most e-mail clients on PDAs and smart phones already work). Another network related advantage of the mobile server is reduced latency. Highly interactive services such as games benefit from local execution; especially if the only available network connection is a GPRS connection with latency in the range of seconds.

The mobile server can be further divided into *portable* or *wearable* depending on how mobile they are. A portable server is typically a laptop computer that can be brought along by the user, but it is large enough to include full size user I/O peripherals such as screen and keyboard. They are however difficult to use when on the move or when e.g. driving a car, but if they are close to the user (e.g. in a back-pack or in the trunk of a car), more suitable user access devices can be used. In this case, all that is needed is a short-range wireless network connection such as Bluetooth or peer-to-peer WLAN. In contrast, the wearable server [11] is small enough to actually be worn by the user at all times. This is likely to imply that there is no room for I/O peripherals, in which case all user interaction need to be maintained via user access devices (just as in the case with the portable server on the move). In some cases, such as in the case of the aWare Messenger, special purpose devices can provide dedicated user interfaces to certain services running on the wearable server (see Figure 3).

Figure 3. A possible design for the wearable server to be used with the aWare Messenger ultra-lite device.

So far, there are few examples of hardware that can realize wearable servers of this kind, at least if high requirements on performance are placed. One exception is the Intel Personal Server [9, 10], which is a computer of about the size of a deck of cards[4]. It comes completely without user I/O peripherals, but includes a Bluetooth module that makes it possible to connect to user access devices.

Communication Modes

As the discussion above clearly shows, use of electronic services is largely a matter of communication; this is particularly true in a mobile setting. The nature of this communication differs with its purpose and the situation of the user. In the context of personal servers, three distinct modes of communication clearly stand out: user access device to personal server, personal server to another personal server, and personal server to application server.

User Access Device to Personal Server

In most cases, users of personal servers will not interact with their services on their personal server directly, but rather via user access devices. Ideally, the network connection for this kind of communication would provide high bandwidth and low latency, but this cannot always be assumed. The network connections differ substantially between the case of the mobile server and the remote. With a mobile server, it is assumed that the user access devices always are in the immediate physical vicinity of the server. This makes for a situation with small variations in connectivity characteristics. Medium high bandwidth and low latency can be assumed since the wireless networking standards available today already support this. The bandwidth and latency of Bluetooth for example, is good enough for most applications except for some games with very high requirements on media presentations. This stability makes it straightforward to implement services since no adaptations of the communication between user access device and personal server needs to be performed. It is much more difficult to predict the connectivity characteristics in the case of a remote server though. From the very

---

[4] At the time of writing (April 2003), the Intel Personal Server is only a prototype and it is not available to the open market.

high bandwidth and low latency of a high speed LAN, to the low bandwidth and high latency of e.g. GPRS, to complete disconnectedness: all possibilities are equally probable. With these changes in connectivity characteristics, there is a need to adapt the communication between user access device and personal server since rich interactivity will not work over poor connections, and since users with high quality connections will probably not be satisfied with the limited interactivity that the poor quality connections can offer.

Personal Server to Application Server

In theory, services executing on the personal server are self-contained. In most cases however, this is not possible to realize in full. This is either because performance or storage limitations of the personal server require services to make use of networked resources, or because the service itself inherently requires communication. An example of the former is a Yellow Pages phone book service, in which case it is not practical to store all phone book entries locally on the personal server. The database of phone numbers therefore needs to reside on a networked application server, while a history of previously looked up phone numbers could be stored locally. An example of the latter is an e-mail service which messaging functionality is completely dependent on a network connection, regardless of performance and storage capability of the personal server.

In both of the above examples, services on the personal server need to communicate with some kind of networked functionality. Depending on the type of service and the type of personal server, different requirements are placed on this communication channel. Requirements on latency and bandwidth are quite the opposite of the requirements placed on network connectivity in the user access device to personal server communication mode described above. Since the characteristics of the network connection between a mobile server and an application server are difficult to predict, services that require fast and reliable network connections could suffer in such a context. The connection between a remote server and an application server is likely to be stable and fast, making it more suitable for services with such requirements. For the same reason, a remote server is likely to be a better host for services that act autonomously on its user's behalf. This because when the user is not available (e.g. while sleeping or traveling), chances are the mobile server is unavailable as well. The remote server however, is less dependent on the actual doings of its user, making it suitable to serve the user 24 hours a day.

Personal Server to Another Personal Server

If the two previous communication modes resemble different versions of traditional client-server computing, the third is more in the line of peer-to-peer computing [12]. In this case, a service on a personal server communicates with a service (usually of the same kind) running on another personal server. Sometimes this is a direct connection between the two peers, but it could also be that the communication is routed with the help from other peers – creating a chain of connections leading from source to destination. Services that implement this communication mode are for example games in which peers play against each, and services for exchanging for example business cards, notes, and calendar entries.

In general, this mode of communication practice under the same conditions as personal server to application server communication (uncertain and varying conditions for mobile servers and reliable conditions for remote servers). However, when the two personal servers are close enough for direct communication via for example Bluetooth or peer-to-peer WLAN, high bandwidth and low latency between the peers can be assumed. Another advantage with this setup is that this type of network connection is usually for free since it is a direct connection between two personal servers – no network service provider is involved.

User Access Devices

In some cases, as for example the portable or remote server, the same hardware that executes the services also provides user I/O capabilities such as screen, keyboard, and mouse. When speaking of personal servers in general though, this is not the case. Instead, users need to rely

on user access devices in order to interact with their services. As suggested above, these should be chosen based on the context and situation that the user is in at the time of usage. When sitting in an office at work, a desktop computer with a large screen, a full keyboard, and a mouse is probably one of the better alternatives. However, while on the move a smart phone or perhaps a headset and a head-up display is more suitable.

Examples of currently available user access devices are 2.5G smart phones, PDAs, Tablet PCs, and desktop computers in the form of Web kiosks. The strengths of the smart phones are their small form factor that makes them highly mobile, and the nearly constant network access over GPRS. These two features are also the weaknesses of the device – being so small also make them unsuitable for many tasks (for example word processing, reading text, and viewing images), and the network connection is unreliable, slow, and with high latency. Many PDAs are slightly bigger than smart phones. This makes them somewhat less mobile but at the same time more useful for certain tasks. PDAs are usually better equipped in terms of network connectivity – PDAs with both WLAN and Bluetooth, and in addition GPRS as an optional feature, has been available for some time. This further makes them more capable in a mobile setting since more network solutions, over a wide price-performance range, are available. Tablet PCs resemble PDAs in how they are used, but they are much larger, usually with full-size displays. They are also more powerful and thus often capable of hosting services locally (i.e. acting as a portable server) in addition to acting as access devices. Finally, Web kiosks can be found in many public places ranging from cafés to airports and public libraries. For users without access devices of their own these can provide access to remote servers that host services with HTML user interfaces. They can also be used in cases were other access devices are too limited in terms of I/O capabilities.

Software Support for Personal Servers

There have been few attempts to provide serious software support for personal servers as described herein. Alan Dearle describes *ubiquitous environments* [13], which could work as a software platform for personal servers, and some technical problems (from a software point of view) that need to be solved in order to realize them. Dearle also lists a number of platforms that address at least parts of what is needed in order to implement them (for example Grasshopper [14], Telescript/Odyssey[5], and Aglets[6]).

For the Intel Personal Server [9, 10], Want et al. have chosen to rely mostly on already existing support such as Web servers and file sharing mechanisms as a means for connecting personal server with user access devices. This makes the software platform for the personal server trivial – an ordinary OS (in their case Linux), a Web server, and services with HTML user interfaces, is enough. This choice also makes the personal server quite limited. Available access devices, for example, are limited in this case since they must have a Web browser. This also limits how well the UI can be adapted to the access device at hand, and hard buttons and other special I/O peripherals cannot be assigned special functionalities.

In the sView project however, we have developed a system [15, 16] that supports all aspects of personal servers as described above. The system builds on the notion of personal service environments [17] that store electronic services and data of individual users. When residing on a computer, the services can be accessed locally via the I/O peripherals of the computer, or remotely via network connected access devices (such as PDAs, cell phones, and Web kiosks). This allows the system to implement both mobile and remote servers. The service environments are mobile, which makes it possible to move all services to a different computer if needed. During the migration, the execution state of each service is saved in order to achieve continuous user-service interaction. This makes it possible to combine the qualities of mobile and remote servers by running the service environment on a server (acting as a remote server) when needed, and moving it to a mobile device (acting as a mobile server) when that is more suitable.

---

[5] General Magic Inc.
[6] IBM Tokyo Research Laboratory

	Mobile Server	**Remote Server**
Service access	Great opportunities since services can execute locally.	Limited opportunities since a network connection is required in order to access the services. Off-line execution is not possible.
Autonomous execution	Good opportunities, but services requiring a continuous network connection may suffer.	Great opportunities since services can rely on a continuous high-speed network connection.
Remote Control	Great opportunities since remote control UI typically require low latency and at least medium high bandwidth.	Good opportunities, but with connections between server and user access device with high latency and/or low bandwidth, the performance may suffer.
Adaptive UI	Equal opportunities. However, services with none or modest need of networked data that output large amounts of data to the user (e.g. single-user games) benefit from mobile server execution. On the other hand, services that require large amounts of networked data but only output a limited amount of data to the user (e.g. a personal search engine) benefit from remote server execution.	

Table 2. A comparison between the two major categories of personal servers with regard to some seamless mobility challenges.

As part of our work with sView we have also developed solutions to support device independent software components [8] and peer-to-peer communication between personal service environments [18]. The former is a key component when realizing continuity as in the case of adapted user interfaces (described above), while the latter is vital when implementing services that make use of the personal server to another personal server communication mode.

Discussion

All in all, we argue that the concept of personal servers in general, and wearable servers in particular, is highly attractive as a general solution for providing seamless mobility. First, the concept allows services to be (partially) independent of network connection and quality. Second, the concept is virtually built on the notion of UI and device flexibility, as a means of treating the variation in user needs a mobile setting offers. And third, the concept allows different solutions for providing continuity, both network and UI/device oriented, to coexist.

However, being such a wide concept, not all variants of personal servers are equally well suited to handle all challenges of providing seamless mobility. In Table 2 we list a few such challenges in order to compare how well the two main categories of personal servers can handle them. The first two challenges (service access and autonomous execution) are mostly related to network independence. They address how dependent the solution is of network connectivity in order for the user to access services and for the services to operate autonomously. The last two categories (remote control and adaptive UI) address how well the solutions support continuity as in the remote control case or adaptive UI respectively (se above).

Acknowledgements

References

[1] G. Kortuem and Z. Segall, "Wearable Communities: Augmenting Social Networks with Wearable Computers," IEEE Pervasive Computing, vol. 2, pp. 71-78, 2003.

[2] M. Weiser, "The Computer for the 21st Century," Scientific American, pp. 94-104, 1991.

[3] C. Perkins, "Mobile IP," in IEEE Comm., vol. 35, 1997, pp. 84-99.

[4] R. Chakravorty and I. Pratt, "Performance Issues with General Packet Radio Service," Journal of Communications and Networks, vol. 4, pp. 266-281, 2002.

[5] M. Bylund, "An Empirical Evaluation of the Performance of Mobile Network Connections," Swedish Institute of Computer Science, Kista, Sweden SICS Technical Report T2003:06, May 2003.

[6] J. S. Hansen, T. Reich, B. Andersen, and E. Jul, "Dynamic Adaptation of Network Connections in Mobile Environments," in IEEE Internet Computing, vol. 2, 1998, pp. 39-48.

[7] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper, "Virtual Network Computing," IEEE Internet Computing, vol. 2, pp. 33-38, 1998.

[8] S. Nylander and M. Bylund, "Providing device independence to mobile services," in Universal Access: Theoretical Perspectives, Practice, and Experience, Lecture Notes in Computer Science, N. Carbonelle and C. Stephanidis, Eds., LNCS 2615 ed: Springer-Verlag, 2003, pp. 465-473.

[9] R. Want, G. Borriello, T. Pering, and K. I. Farkas, "Disappearing Hardware," IEEE Pervasive Computing, vol. 1, pp. 36-47, 2002.

[10] R. Want, T. Pering, G. Danneels, M. Kumar, M. Sundar, and J. Light, "The Personal Server: Changing the Way We Think About Ubiquitous Computing," presented at 4th International Conference on Ubiquitous Computing (UbiComp'2002), Göteborg, Sweden, 2002.

[11] Z. Segall, "Wearable Server," Department of Computer Science, University of Oregon, Eugene OR 97403, Proposal to Intel Research Council March 2002.

[12] A. Oram, Peer-To-Peer: Harnessing the Power of Disruptive Technologies, 1st ed: O'Reilly, 2001.

[13] A. Dearle, "Toward Ubiquitous Environments for Mobile Users," IEEE Internet Computing, vol. 2, pp. 22-32, 1998.

[14] A. Dearle, R. di Bona, J. Farrow, F. Henskens, A. Lindström, J. Rosenberg, and F. Vaughan, "Grasshopper: An orthogonally persistent operating system," Computing Systems, pp. 289-312, 1994.

[15] M. Bylund and F. Espinoza, "sView - Personal Service Interaction," presented at 5th International Conference on The Practical Applications of Intelligent Agents and Multi-Agent Technology (PAAM'2000), Manchester, UK, 2000.

[16] F. Espinoza, "Individual Service Provisioning," in Dept. of Computer and Systems Sciences. Kista, Sweden: Stockholm University and Royal Institute of Technology, 2003.

[17] M. Bylund, "Personal Service Environments – Openness and User Control in User-Service Interaction," in Computing Science Department, Information Technology. Uppsala, Sweden: Uppsala University, 2001.

[18] F. Espinoza and L. Hinz, "Generic Peer-to-Peer Support for a Personal Service Platform," presented at Saint'2003, Orlando, Florida, 2003.