A Lightweight UI Software Infrastructure for Wrist-based Displays: If your microwave oven could talk to your watch, what would it say?

Peter Hutterer†, Mark T. Smith‡, John Ankcorn‡, Wayne Piekarski†, and Bruce H. Thomas†

†Wearable Computer Lab
School of Computer and Information Science
University of South Australia
Mawson Lakes, Adelaide, SA, 5095, Australia
office@who-t.net
{bruce.thomas, wayne.piekarski}@unisa.edu.au

‡Hewlett Packard Labs
Hewlett Packard Corporation
1501 Page Mill Rd
Palo Alto, Ca 94304, United States of America
msmith@hpl.hp.com
jca@hp.com

Abstract

Supporting a rich array of information sources is a key element to making highly mobile computing devices usable by the wider community. It is our belief that there will not be one specific killer application for this form of computing device, but an array of applications that the user can easily access. These applications will be context sensitive and associated with a range of activities. We have developed a custom watch platform that acts as a display for presenting this type of information to the user. In order to make the watch as small and low powered as possible, we have offloaded the processing onto an external mobile device we term a personal server, which is also carried by the user. We present our lightweight software infrastructure supporting a wrist-based display communicating with a portable personal server.

1 Introduction

We are investigating application domains for wrist-based displays coupled with computation systems. Ubiquitous mobile computing devices offer the opportunity to provide easy access to a rich set of information sources. Placing the display on the user's wrist allows for quick, easy, and pervasive access to this information. The aims of our user interfaces are to support lightweight interactions and the construction of interfaces that are as simple to use as a common wristwatch today. We have developed a prototype watch that contains only a display with very limited process capabilities, see Figure 1. Rather than perform processing on the watch, the processing is offloaded to a small external server that then sends display information via Bluetooth. The advantage is that the watch display device is only required to support very simple processing, allowing the use of very low powered microprocessors. Any input devices that are attached to the watch communicate only with the external server, and any changes are then reflected on the display.

The scenario we used for inspiration in our research is someone watching television who decides to cook microwave popcorn in the kitchen. During a commercial break, the user leaves the room to place the popcorn in the kitchen microwave oven. The user sets the timer on the microwave oven and presses the start button. When oven starts to cook the popcorn, the oven communicates with the user's watch to start a countdown timer. The user may return to viewing the television, and their watch will then inform them when the popcorn is ready.

The paper starts with a background to a number of data watch technologies. The watch display hardware we have developed for this research is then described. Our personal server software (PerServ) is then described, and an overview of a set of applications we have developed for the watch with the PerServ is put forward. We finish the paper with some concluding remarks on our experience with this architecture.

Figure 1. The prototype watch display

2 Background

Several commercial projects have been targeted at replacing the watch with a multi-function device. Some research work has been presented on the social weight of using a watch device [1], and Narayanaswami et al [2] ask the question - What are the applications an all-purpose watch could be used for? This section presents an overview of a selection of

the current watches that are commercially available. However, this is only a small selection of watches since detailing every watch available is beyond the scope of this paper.

2.1 Commercial Watches

We investigated six different current watch devices - five commercially available watches and the IBM Linux Wristwatch research project. All six have slight differences in functionality; the Matsucom OnHand PC [3] and the IBM Linux Wristwatch [4-6] try to create a fully functional miniature computer on the wrist. Fossil's Wrist PDA [7] has the approach of creating a miniature PDA device, whereas the Wrist Net uses a very selective channel-based data reception mechanism. The Timex USB Datalink [8] tries to fill the gap between sports watches and PDAs, but with the main focus still being on the watch functions. The Field Technology CxMP Smart Watch is the only one of the commercial watches with a colour display and supports image viewing and sound output. Although five of these watches have been released commercially, each of them contains a number of limitations that should be avoided in the development of future watches.

2.2 Processor

The processing unit used in a watch effects both the size of the watch and the battery life. While low powered CPUs such as StrongARM chips are available for small devices, their power consumption is still excessive for highly miniaturised platforms such as watches. StrongARM chips, as in the Linux Wristwatch, only operate for a few hours on a single watch battery before requiring a recharge. To provide improved battery life, the processor must be severely reduced in complexity, but with associated losses in functionality and programming ease. When designing a watch device, trade-offs must be made between functionality, performance, and power consumption, and it is not possible to have the design optimised for all three criteria.

2.3 Connectivity

Of the watches described here, connectivity to an external device is implemented in a number of different ways. USB, RS-232 and infrared have the limitation of requiring either a cable or line of sight between the watch and external devices. A Bluetooth wireless personal area network (PAN) connection does not require direct line of sight to the external device and may work up to approximately ten metres [9]. The important restriction of wireless PAN connections is that none of them are able to support connections to the Internet at the distances supported by a wireless LAN (WLAN). While WLAN technology may be used for wireless Internet with a laptop or PDA, the power required is too great for a small watch platform.

2.4 Interaction

Interaction with commercial watches has been performed in various ways. The OnHand PC and WristNet watches provide buttons around the edge of the watch, while the IBM watch supports a touch screen surface that is divided into quarters so the user can select options on the screen itself. Other interactions are supported using three way rocker switches in the Fossil Wrist PDA and a small joystick employed on the Matsucom watch. The joystick and rocker button inputs allow the quick traversal of menus. While limited speech recognition has been implemented in devices such as mobile phones, watch platforms are even further limited in processing capability and speech recognition is still not currently feasible [6].

3 Wrist Device

Our watch device prototype, as shown in Figure 1, was designed and manufactured by Hewlett Packard Labs. The watch described in this section consists of the following three elements: the microcontroller, the LCD display, and the Bluetooth chipset. The embedded software on the watch is then explained.

3.1 The Hardware

The layout of the watch hardware is the LCD display on top, and the microcontroller and Bluetooth chip mounted underneath the display. The microcontroller used on the watch prototype is a Texas Instruments MSP430. The two main reasons for this choice of microcontroller are its low power requirements and availability. We employ the F1491 model [10] because it contains various features that we deem suitable for this project: 64 Kb ROM, 2 Kb RAM, and two serial UARTs for connecting to the Bluetooth chip and LCD controller. The 16 bit MSP430 processor operates at 4 MHz and only needs 280 μA in active mode, which is excellent for battery efficiency. The MSP430 also supports development using the C programming language, which helps to simplify the implementation of complex functionality such as TCP/IP stacks.

The display in the prototype is an Epson L2F50176T00. The resolution of this display is 120x160 pixels at a screen size of 2.0 x 2.6 cm, which is approximately 150 DPI. The LCD supports RGB colour with 16-bits per pixel. The microcontroller communicates with the LCD screen via the UART interface, which is slower in performance than if the display was updated via dedicated memory attached to the microcontroller.

For communication between the watch and the PerServ, we employ the Mitsumi WML-C09NBR Bluetooth chipset, without using an external antenna. Like the LCD, the Bluetooth chip is connected to the microcontroller via a UART interface, and supports a maximum data rate of 721 kbps [11], although this is limited by the speed that the microcontroller is able to service the UART. The WML-C09NBR is a Bluetooth class 2 chip with a theoretical range of ten metres, but during testing with a standard USB Bluetooth adapter and a laptop we found that the maximum distance was closer to three to four metres.

3.2 Watch Software

The microcontroller operates a simple software application to control the following two functions: 1) initializing the microcontroller, the LCD, and the Bluetooth network stacks; and 2) the display of data received from the Bluetooth network onto the watch's LCD. The Bluetooth connection provides the ability to transport packets, and we implement UDP and IP layers on top of this. The UDP payload contains data in our custom Image Transfer Protocol (ITP) packet format. The ITP protocol supports the redrawing of specific sub-regions of the LCD. Screen updates can therefore be reduced to only those regions of the screen which have changed,. Compared to transmitting the whole screen, this brings a significant reduction of network transfer time and increases the overall update rate of the watch when used with most of our applications. When testing the ITP, we discovered that the transmission times were quite large even though the screen was mostly blank. Run Length Encoding (RLE) was then implemented to compress the images in a way which would be easy to implement in the watch CPU, while at the same time giving reasonable transmission time for images with large areas of flat colours.

4 Personal Server Model

Future architectures like the watch represent a large advance in the way consumer IT devices can be used to manage networked personal resources. In this context, the definition of a personal resource is a remotely accessed entity that enhances or extends those personal attributes that are physically co-located with the user and represented by the watch. For example, other reasons for choosing and wearing a wristwatch can include such personal attributes as a statement of fashion, expressing a sense or value of self, and other forms of symbolic communication. IT devices worn on the body represent a significant advance to managing personal resources because they combine the functionality of a networked service with the personal attributes of the device, resulting in the networked service becoming a personal resource. If the watch itself is an expression of self, then that expression can now be extended beyond the wrist into the networked space.

The functionality of personal resources may be as varied as the users, and so a practical way to support personalized functionality is necessary. Watches, smart-phones, and other relatively small IT devices have by necessity limited computing, communication, power, GUI, and storage capability associated with them compared to a larger platform such as a personal computer. These limitations are generally imposed by physical size constraints, but still they limit what the device can do and what applications it can support. In the architecture presented here, this problem is addressed though the deployment of a personal server. A personal server is a physically small, mobile, remotely managed device that can provide functionality to a personal device as if it were built in. The device obtains this functionality through a network connection, typically wireless, to the personal server. In this way, the personal server and the device form an aggregate that to an application looks like a single inseparable device. This is one way that personal servers are unique when compared to a general client-server computing model. By performing these aggregation services, personal servers allow applications to be written for a single virtual device and do not need to explicitly deal with the connectivity between the aggregated components.

4.1 Services

To allow personal resources to exist, a personal server needs to provide other services in addition to aggregation and basic networking. The prototype personal server in this architecture provides the following services: 1) application services, 2) storage services, media caching, 3) transcoding & rendering, 4) network diversity, 5) wireless and wired connectivity, 6) security of data, VPN, authentication to services, 7) personalization support and internet content adaptation framework, 8) user device power management support, 9) multiple device aggregation, and 10) session management. Application services allow application code to be run either on the watch, the personal server, or split between both. This is especially useful when the watch’s processor does not have a Memory Management Unit, or other means of supporting a virtual memory model, making the loading and running of multiple code modules difficult. Running an application entirely on the personal server with the watch as the UI is a useful model for applications requiring data processing or storage requirements beyond the capabilities of the watch itself. To support this, data storage, transcoding,

and rendering services are provided. Transcoding and rendering are used in video and imaging applications where image frame resolutions, frame rates, data encoding, and related parameters of source data need to be altered to accommodate the watch's GUI hardware.

Network diversity is accomplished by the use of multiple radios in the personal server. Using multiple radios of different types accommodates connectivity requirements across a wide range of possible options, for example bridging between the watch's Bluetooth PAN radio and an 802.11 WLAN. By using multiple radios of the same type, the personal server can accommodate emerging WLAN mobility models such as Inter-Access Point Protocol [12].

The personal server provides security services on behalf of the watch, accommodating the needs of the application, internet service provider, and user. The server can act as a personal proxy or firewall performing VPN or other data security services. It can also authenticate the user to networked resources on behalf of the watch, becoming a trusted user information data repository for all the user's small personal devices. Personalization support is an important aspect of managing personal resources, including context aware services such as location or other sensor sourced data, and on-line data personalization which is done through a framework for Internet Content Adaptation based on the ICAP 1.0 protocol [13]. This framework provides a proxy-based solution for online adaptation of content obtained from the Internet.

The personal server allows power management in the watch to be managed at a system level. This can be much more effective than the watch managing its power alone. In addition to offloading processing and higher power communication from the watch, the server can provide hints as to the type and complexity of data the watch may be required to process, allowing it to lower its clock rates or supply voltages for relatively non-demanding tasks [14].

4.2 Aggregation Model

The aggregation model that results in the personal server and the watch becoming a single virtual device can be used with multiple devices. One example could be the aggregation of the watch and server with a wireless audio earpiece and a free-space locator or pointer that would then all appear as a new single wearable virtual device. The aggregated devices would also include the ones the user desires to command or control, which in this study are home appliances and entertainment sources. One method to support this is to use session management such as Session Initiation Protocol (SIP) [15] to allow multiple devices to join or leave an application as required. SIP is also widely used as a session management protocol for instant messaging and other communication applications such as voice over IP.

The basic functionality of a personal server may be modelled using off the shelf computing platforms such as PDAs or PCs. In this study, initial development of the personal server was performed on a Hewlett Packard iPAQ PDA. Studies based on this have led to new designs of practical and cost effective personal servers that focus on desired application and communication functionality and not on traditional PIM applications. Two types of prototype personal servers have been constructed and are being used in studies at Hewlett Packard Laboratories. **Error! Reference source not found.** depicts the general block diagram followed by both architectures. The first architecture uses a Pentium class processor and rotating magnetic storage, while the second uses a much lower powered design based on the Intel StrongARM processor and semiconductor storage [16]. Both designs use multiple modular radios for connection diversity, and include radios for personal area (Bluetooth), local area (802.11b), and wide area (CDMA) wireless services. By using modular radios, new radio technology can be deployed almost instantly. Both platforms run embedded Linux OS as the operating system.

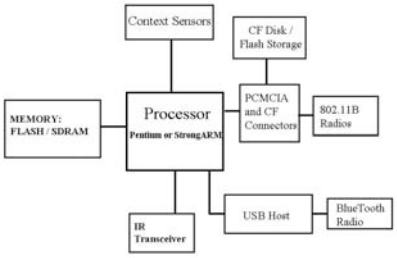

Figure 2. Personal server block diagram

Physically, the designs are intended to be as small and lightweight as possible. It is envisioned that personal servers may be deployed in a number of ways, including carried with the user or opportunistically co-located with the user, such as built into a car or an office cubicle. As the user is not expected to directly interact with the server, it can in effect be hidden. A single user may deploy multiple personal servers. The server itself has no external UI, and appears as a small box. It is remotely managed via any of its communication interfaces and this provides a potential for value added services. The design based on the StrongARM processor is a board measuring 6.4 x

13.7 cm and consumes under 0.6 Watts of power at 200 MHz, not including the radios. Future designs will reduce the size and power consumption much further, add extra media computing resources, and add 802.15.4 (ZigBee) wireless personal area connectivity.

5 Controlling The Personal Servers

Our personal server architecture supports multiple personal server devices, but the watch communicates to only one personal server at a time. This requires the ability to switch personal servers while an application is in operation and to transfer the application's data and context to maintain continuous operation between the servers. We envision some personal servers being fixed to a location and broadcasting on a series of public wireless hotspots; some examples of these locations are the home, company building, or a regular café. In situations where there are no available fixed personal servers, the watch employs the user's IPAQ as a mobile personal server device. Fixed personal server infrastructure allows the watch access to greater processing, data, and networking capability. The mobile personal server provides greater portability and flexibility. It is necessary to implement both a method to recognize other servers and to transfer data between servers. This section provides an overview of the design issues in the swapping of personal servers.

5.1 Synchronizing Servers

To be able to exchange application data and maintain continuous operation of an application across multiple personal servers, the alternative servers must be known to the user's personal server, i.e. by maintaining a list of allowed IP addresses. In the ideal case, the tool for shifting servers is an application using the plug-in framework instead of being built into the personal server. The application then listens on a specific port and maintains connections to remote servers while simultaneously synchronizing the data. If an existing connection between two PerServs is lost, as can happen when the user with the watch leaves the range of one PerServ, all servers try to connect to the watch. The server which still has a connection takes over the display of the watch. In the case of multiple servers being able to communicate with the watch, a user-defined hierarchy designates the active server.

5.2 Synchronizing Data

In the context of synchronizing we define two types of applications; those that are transferable and those that are non-transferable. The decision of which applications are transferable depends on both the current system configuration and on the user's context. The synchronization application has to provide a list of the transferable programs on both the sender and the receiver side. By comparing this list, the decision is made which applications have to be started on each server to keep the data synchronized. The applications are then pushed by the synchronization application to provide exportable data. Consequently, the synchronization application sends the data to the second personal server. On the other side this data is received and then presented to the matching application.

6 User Interface and Examples

This section describes some applications that we have developed and their user interfaces to the watch. A more complete description of these applications may be found in "Lightweight User Interfaces for Watch Based Displays" [17]. Each of these applications is an example for a technique to process certain forms of input data. The Timer application is described which connects to household devices. The MP3 application shows how to connect the PerServ with an external application. The RSS Feeds explains how applications can fetch live data from the Internet.

The PerServ should be able to communicate directly with other hardware. The Timer application is designed to work with common household devices - we decided to emulate these devices with standalone computers to simplify our initial trial. This is an example of how future household devices only require slight modifications to be able to make use of the PerServ. The use of a countdown timer is ambiguous as for example a coffee machine or a microwave oven could use it to show the remaining time of food to be finished. To accomplish this task, the household device needs to send a message to the Timer application indicating the remaining time. The users can meanwhile focus their attention on something else, and the countdown is displayed on their watches. The Timer application is built to show more countdowns at the same time, sorted by finishing time.

One example to show outside applications communicating with PerServ applications is the MP3 Title Display that consists of a PerServ application and a plug-in for XMMS[1], a popular MP3 Player for Linux. The plug-in retrieves information about the current song and passes it on the PerServ application for display on the watch. A similar system may be used to connect a mail client with an application on the PerServ.

Live data streams are vitally important to many businesses today, such as the latest stock quotes or headlines of industry news. A widely used method is RSS [18] that is a subset of XML and is supported by a

[1] http://www.xmms.org/

large number of news sites. Even stock prices, which often use proprietary protocols, are available via RSS. A considerable advantage in the RSS XML structure is the possibility of using pack-and-go XML parsers. This also eases the use of multiple streams at the same time. The output of an RSS news feed can be either graphical or textual, depending on the user's context and the information itself. If it is necessary to keep an eye on stock prices over a longer time interval, a graphical interface such as line diagrams are a good solution. However, if there are too many lines on the diagram it becomes fuzzy and difficult to read, and in this case it is better to display the information in text format.

7 Conclusion

In a time where the ubiquitous access to information becomes vitally important, the concept of personal servers offers a high-quality solution. A notebook computer, currently the most widespread personal computation device, is a production-based device. Its main task is to offer a portable, but rather immobile computing environment to let the user work in a location independent manner. The information-based personal server approach does not focus on the user's work but on providing personalized information, independent of the location but always available. Its infrastructure, based on simple and swappable assets, is a great advantage compared to the fixed configuration of a notebook. The different wireless connection protocols the personal server offers allow it to blur the borders between a personal area network, a local area network, and the global Internet. From the perspective of the watch, the source of the information is irrelevant.

This paper has described a number of contributions we have made in the domain of ubiquitous mobile computing devices. We present our new personal server architecture, which is extendable to a number of application domains. Because the protocol is open and simple, the servers, watches, and applications can be modified, changed, or exchanged easily in the future. The application framework is well designed in its ability to support a wide range of different applications. We feel the ability to accommodate new applications and infrastructure is key to making the architecture useful in the wider-world information village.

8 Acknowledgements

We would like to thank Hewlett Packard Labs for their support on this project. This project was made possible by the continuing support of the School of Computer and Information Science at the University of South Australia. Aaron Toney also provided a great sounding board and a wealth of information about programming with low powered devices.

9 References

1. Toney, A., et al., Social Weight: Designing to minimise the social consequences arising from technology use by the mobile professional. Personal and Ubiquitous Computing, 2003. 7(5): p. 309-320.
2. Narayanaswami, C., M.T. Raghunath, and N. Kamijoh, What Would You Do with a Hundred MIPS on Your Wrist?,. 2001, IBM Research Division, Thomas J. Watson Research Center,
3. Matsucom, I., onHand/Ruputer. 2004: 1642 South Parker Road, Suite 212, Denver, Colorado, 80231 U.S.A.,
4. Narayanaswami, C. and M.T. Raghunath. Application Design for a Smart Watch with a High Resolution Display. in International Symposium on Wearable Computing. 2000. Atlanta, Georgia.
5. Kamijoh, N., et al. Energy Trade-offs in the IBM Wristwatch Computer. in 5th International Symposium on Wearable. 2001: IEEE.
6. Narayanaswami, C., et al., IBM's linux watch: The challenge of miniaturization. IEEE Computer, 2002: p. 33-41.
7. Fossil Inc. 2004, 2280 N. Greenville Ave., Richardson, TX 75082-4412, webguy@fossil.com, http://www.fossil.com/.
8. Timex Corporation. 2004, PO Box 310, Middlebury, CT 06762, http://www.timex.com/.
9. Bluetooth SIG, I., Bluetooth Core Specification. 2004, https://www.bluetooth.org/spec/.
10. Texas Instruments, MSP430x13x, MSP430x14x, MSP430x14x1 MIXED SIGNAL MICROCONTROLLER. 2004, Texas Instruments: Post Office Box 655303 Dallas, Texas 75265,
11. Mitsumi, BluetoothTM Module WML-C09. 2004, 2-11-2, Tsurumaki, Tama-shi, Tokyo 206-8567, Japan, http://www.mitsumi.com/.
12. Ren, A. and G. Maguire, Jr. An Adaptive Real-time IAPP Protocol for Supporting Multimedia Communication in Wireless LAN Systems. in International Conference on Computer Communication. 1999. Tokyo, Japan.
13. Manjunath, G., A Framework for Internet Content Adaptation. 2004, http://icap-server.sourceforge.net/.
14. Simunic, T., et al. Dynamic Voltage Scaling for Portable Systems. in Proceedings of the ACM Design Automation Conference. 2001.
15. Rosenberg, J., et al., SIP: Session Initiation Protocol, in IETF Request for Comments 3261. 2002,
16. Maguire, G., Jr., SmartBadge Version 4. 2004, http://www.it.kth.se/~maguire/badge4.html.
17 Hutterer, P., Smith, M. T., Piekarski, W,. Thomas B. H., and Ankcorn, J., Lightweight User Interfaces for Watch Based Displays. The Sixth Australasian User Interface Conference, Newcastle, Australia, Jan., 2005, (To appear)
18. Harvard Law, Really Simple Syndication (RSS) Specification. 2004, Berkman Center, http://blogs.law.harvard.edu/tech/rss.