URCP: Design and Implementation of a Protocol to Support the Single Task Multiple Interface Model of User Interaction

A. Donnelly & E. Barnstedt

Department of Computer Science
Trinity College Dublin,
Dublin,
Ireland
Alexis.Donnelly@cs.tcd.ie, Erich.Barnstedt@cs.tcd.ie

June 1999

We present the design of URCP (Universal Remote Console Protocol) a high-level, media-independent protocol for interacting with a smart building and URCP-supporting appliances within it. The protocol supports multiple user interface modalities and provides important economies by separating the user interface from the interface between console and appliance/building. We also discuss the principles of Universal Design which we believe are of interest to user interface designers. We also evaluate our design in the light of these principles. Finally, we report on a prototype implementation based on infra-red built to assess the protocol. Despite flaws in the supporting platforms available, URCP appears efficient and well suited to its purpose.

1.0 Introduction

Recent work on wearable computers [1], [2] and [3] has demonstrated the productivity benefits of wearable computers in applications such as aircraft maintenance [14], navigational assistance [13] and vehicle inspection. The benefits derive from natural, intuitive transparent interfaces (e.g. head-mounted displays, touch pads), the inherent mobility of these machines and the way in which they can augment reality. Problems with such machines are their size and weight and the relatively short powerpack lifetimes associated with them [12]. Despite more recent advances, some shortcomings remain that still prove cumbersome in some work environments or make the wearer socially distinctive. One solution to the portability problem is to devote scarce mobile computing resources to the provision of a mobile interface to a larger stationary computing resource nearby. Wireless-enabled personal digital assistants (PDAs) may become sufficiently powerful to fulfill this role, though their present user interfaces are not as natural as the systems described above.

In this paper we describe a protocol, Universal Remote Console Protocol (URCP), which we have developed to support such a mobile console. We envisage the console being used principally to interact with a smart building and with appliances within it which also support the protocol. The console acts as a "thin client" with the building and appliances playing the role of servers. The console supports interactions aptly described by the metaphor "plug, play and unplug".

In section 1 we motivate the “Single Task, Multiple Interface” theme behind URCP and we briefly present the associated principles of Universal Design. In section 2 we present URCP itself and follow with a description of our prototype implementation in section 3. URCP is evaluated in section 4 where we refer to other work and sketch future plans and we conclude in section 5.

1.1 Single Task, Multiple Interfaces

We have traditionally used a single user interface to accomplish all computing tasks - first a command line interface using keyboard and terminal and more recently a graphical user interface using Windows, Icons, Menus and Pointer (WIMP). We are somewhat less used to accomplishing our computing tasks through an auditory user interface [10]. The advent of wearable computing offers the prospect of exploring a design space with multiple interfaces and the possibility of tailoring these interfaces to the task at hand or to respond to changing environments. Our URCP design is intended to support that exploration in the context of one task - controlling and interacting with a smart building and its various appliances using many interfaces.

One author has categorized various user interfaces by the perceptual load on the user generated by output from the interface and by the cognitive load on the user required to provide input to the interface [16]. The continua so produced are reproduced in Fig 1 below. Under this categorization, a WIMP interface is particularly demanding - an observation that bears out the experience of users of some wearable computers [12]

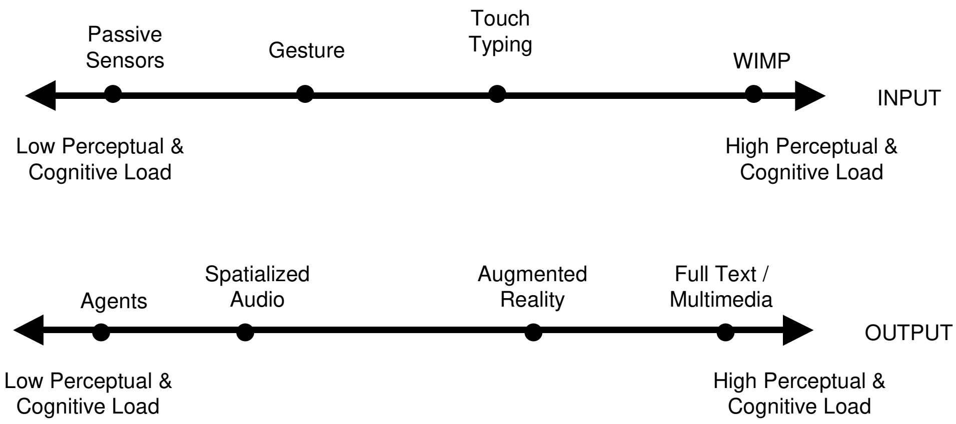

Fig. 1. Rhodes' Continua for User Interfaces (see [16])

A good user interface for these users might be a passive sensor such as a directional noise-cancelling microphone or a good voice synthesizer. For others, a head-mounted display that augments the physical reality around them may be appropriate. The choice of interface will depend ultimately on the task at hand, the user-senses devoted to it and the cognitive and perceptual loads generated by the interface. With this flexibility in mind, we have designed the URCP protocol to support several modalities of interaction with the appliances being controlled.

Taking a higher level view, the abstraction supported by the protocol - essentially that of invoking one of a number of methods on an object [4], [7] is maintained, despite the precise interaction modality employed. Indeed, our design achieves an important separation of concerns - the user interface can be tailored as required for the user and her environment, but the abstract interface with the appliance being controlled is unchanged. This situation is depicted in Fig 2 below. Two economies enabled by this design choice follow: manufacturers no longer have to provide many user interfaces to make their appliances usable in different environments - that task can be left to experts in user interface design. Further, the same tailored console can be used to interact with several appliances, provided they support the URCP protocol.

Indeed these economies are not lost on users with various physical impairments [15]. In passing, it is probably worth noting that some familiar productivity-enhancing appliances evolved from attempts to accommodate various physical impairments. The first typewriter was devised to support a Italian noblewoman who could not use the conventional writing instruments of her day. The efforts that led to Bell's invention of the telephone were devoted to turning speech into writing for his deaf wife - a problem we now know to be difficult! Despite nominal failure ìn this regard, the telephone went on to conquer the great physical barrier of distance.

Our initial inspiration for URCP came from an idea developed at the Trace R & D Center at the University of Wisconsin-Madison. The Universal Auxiliary Interface (UAI) [6] and Universal Remote Console Communication Protocol (URCCP) [5] sought the same separation of the user interface from the appliance interface to provide economic access to users with a variety of impairments. URCP extends that work and we report our experience with a prototype implementation in section 3 below.

1.2 Universal Design

Much of the work at the Trace R & D Center, including UAI and URCCP, is inspired by Universal Design – the goal of designing products and environments to be usable on equal basis by as many people as possible [8], [9]. This idea, common to many design disciplines led a working group of architects, product designers, engineers and environmental design researchers, to collaborate in producing a respected guide, based on seven principles, to universal design [9]. These principles may be applied to evaluate existing designs, guide the design process and educate both designers and consumers about the characteristics of more usable products and environments. We feel that these principles, which have guided our work, deserve more publicity throughout the computing community. We present them briefly here since they confirm much recent discussion and research on wearable computers reviewed above. Readers interested only in the details of URCP itself can safely skip on to section 2. While these principles are most immediately relevant to user interfaces they have been important to our work since a poor design choice at the heart of a communication system could prevent good design choices being made at either end.

1. **Equitable Use**
   This principle suggests that all user groups should be treated equally, e.g. same means of use for all where possible, equivalent means when not; no group should be stigmatized or segregated; the same provisions for privacy, security and safety equally available to all. The recent establishment of the Web Accessibility Initiative (WAI) under the W3C is a notable response to the exclusion experienced by web surfers who are visually impaired.

2. **Flexible in Use**
   The design should accommodate a wide range of preferences and abilities e.g.: right- and left-handedness, the user's (in)accuracy and (im)precision and pace; the design should provide choice in the method of use. This principle is particularly important in some work environments where the user-interface must accommodate work patterns that change over time. The WAI's emphasis on the use of style-sheets in web page authoring is a good example of a design choice which facilitates flexible rendering (and therefore flexible reading) of a web page.

3. **Simple and Intuitive Use**
   Use of the artifact should be easy to understand regardless of the user's experience, knowledge, literacy and language skills and concentration level. In the context of information systems this means remaining consistent with user expectations for the context and providing effective prompting and timely feedback. A relevant example would be recent browser manufacturers' efforts to provide hooks by which screen readers (used by visually-impaired users to browse the web) can replace or bypass the standard WIMP interface. Without the screen reader and other extensions a browser is cumbersome and sometimes impossible to use for a surfer who is visually impaired.

4. **Perceptible Information**
   The design should convey information easily despite difficult ambient conditions and varying sensory capacities. In the context of user interfaces this means exploiting contrast, redundant presentation of the same information and exploitation of varying modalities. Pertinent examples include the ALT text now required in IMG tags in HTML 4.0 and WAI's recommendation of style sheets already referred to.

5. **Tolerance for Error**
   The design should highlight hazards and minimize the consequences of accidental or unintended actions. For user interface work this means grouping similar actions together - frequently used options together, hazardous ones highlighted and shielded from accidental use. Examples include the facility to "undelete" or recover an a deleted file and confirmation of a request to delete an identifiably important file.

6. **Low Physical Effort**
   The design can be used efficiently, in comfort and with a minimum of fatigue. This means using neutral body positions, requiring reasonable operating forces and minimizing sustained effort and repetitive action. All these guidelines have clear application in the area of wearable computers. Familiar examples of their non-application include repetitive strain injury resulting from failure to use ergonomic keyboards, chairs and tables.

7. **Size and Space for Approach and Use**
   The design should provide clear, posture-independent lines of sight and hearing and similar accommodations for reach and manipulation. While clearly included with wheelchair users in mind, those with limited dexterity also benefit from application of this principle. Again, this principle has obvious application for wearable computers.

For the interested reader, [9] presents the consensus following the deliberations of a multi-disciplinary group all committed to universal design. The Norwegian publication, [8], is a thoughtful discussion of the definitions and principles of universal design in planning, architecture and product design together with the implications for the design process itself. Both are useful background material for the designers of user interfaces. We defer an evaluation of the URCP until after our description of the protocol below.

2.0 Design of the URC Protocol

The Universal Remote Console Protocol (URCP) is intended to facilitate a nomadic user, moving throughout a building, interacting with appliances located within the building and with computer systems which may themselves interact with or control the building. As explained above, several user interfaces can be supported using URCP according to users' preferences and/or environments (see Fig 2 below).

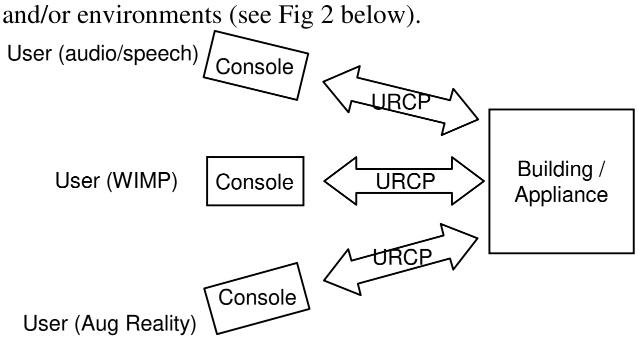

Fig. 2. URCP Enables multiple Interfaces to the same task

The protocol supports an interaction that we view as essentially object-oriented [4]. The controlled entity (a domestic appliance, a building, or a building management system) behaves abstractly as an object - exporting several methods which are the only way to manipulate the appliance. During the interaction, the interface may change resulting in some methods being temporarily unavailable. URCP supports this style of interaction by defining abstract means for requesting, activating and updating menus of available commands between console and appliance. As a consequence of the abstraction, the user-interface is now entirely separated from the console/appliance interaction. Since URCP is defined above the Session Layer of the OSI model its is not dependent on any particular communication medium, facilitating exploitation of the most appropriate underlying communication facilities for the mode selected.

The console acts as a client in the detailed interaction described below, initiating all communication while the appliance acts as server in response. Interaction can begin with the console broadcasting a Find message to which any active appliances reply with a DevInfo(n,tf,opt) message containing the device name, n, supported transmission format(s), tf, and available options, opt, which the appliance supports. Alternatively, should the console already possess this information, it can initially send a ConnReq(n,tf,opt) message requesting a connection to the specified appliance in the specified formats. The appliance responds with a Completed(S=1,ch=1) message containing a success flag and a changed flag if it is available for interaction. If not, a Completed(S=0) message containing a cleared success flag is sent. Simplified state transition diagrams for appliance and console can be found below in Figs 2 and 3 respectively. Transition edges in Fig 2 are labelled with the notation "Message Received / Message Sent", with an asterisk indicating the possibility of receiving more than one message. The protocol supports interactions via text, audio or graphics.

The console now sends a List message and the appliance responds with a Table message containing a list of available menu options. For each option, its name, type and value are sent. To execute a desired command, an Activate(MenuItem) message is sent by the console. In the case of an illegal menu item, the appliance responds with an IllegalCommand message. Otherwise, the appliance responds with a OneMoment(t) message informing the console that the appliance will be unable to respond for at least a further t units of time. Following command completion, a Completed(S, ch) message from the appliance indicates that the command has been completed. The S flag indicates the success or otherwise of the command and the ch flag, if set, indicates that the appliance has changed state and a new menu is now required. In the latter case, the console must now request a new menu via the List / Table interaction described earlier. The Completed message may carry an error code for use by the appliance in higher level protocols.

At any time when a menu item may be activated, the console may send a Disconnect message. The appliance responds with a Completed(S) message in which the S flag indicates whether the disconnection has been successful or not. Similarly, the console may send a Reset message at any time a menu item activation is allowed. The appliance responds with a OneMoment(t) message indicating that the reset is anticipated to take t time units. When the reset is finished, the appliance sends a Completed(S,ch) message indicating the success or otherwise of the operation via the S flag. Again, the ch flag indicates whether a new menu should be requested. We assume that after a reset the appliance will be returned to the state in which the initial menu is appropriate.

The state transition diagram for the console (see Fig 3) mirrors that for the appliance described above. Note that edges are labelled with Message Sent / Message Received – the reverse of the notation used for the appliance state transition diagram.

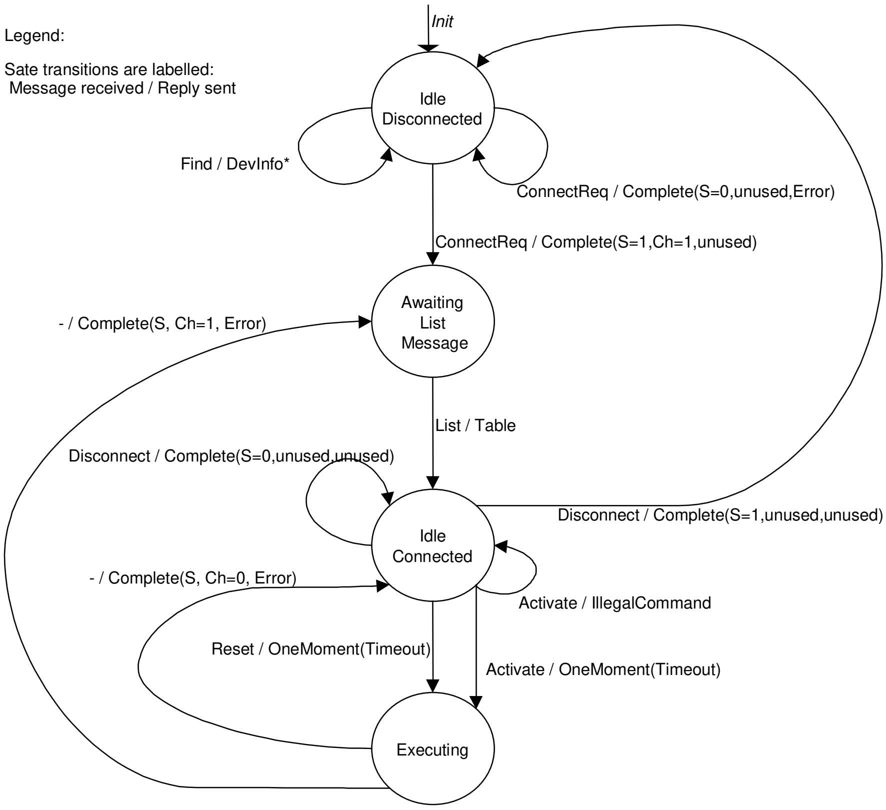

Fig. 2. Appliance State Transition Diagram.

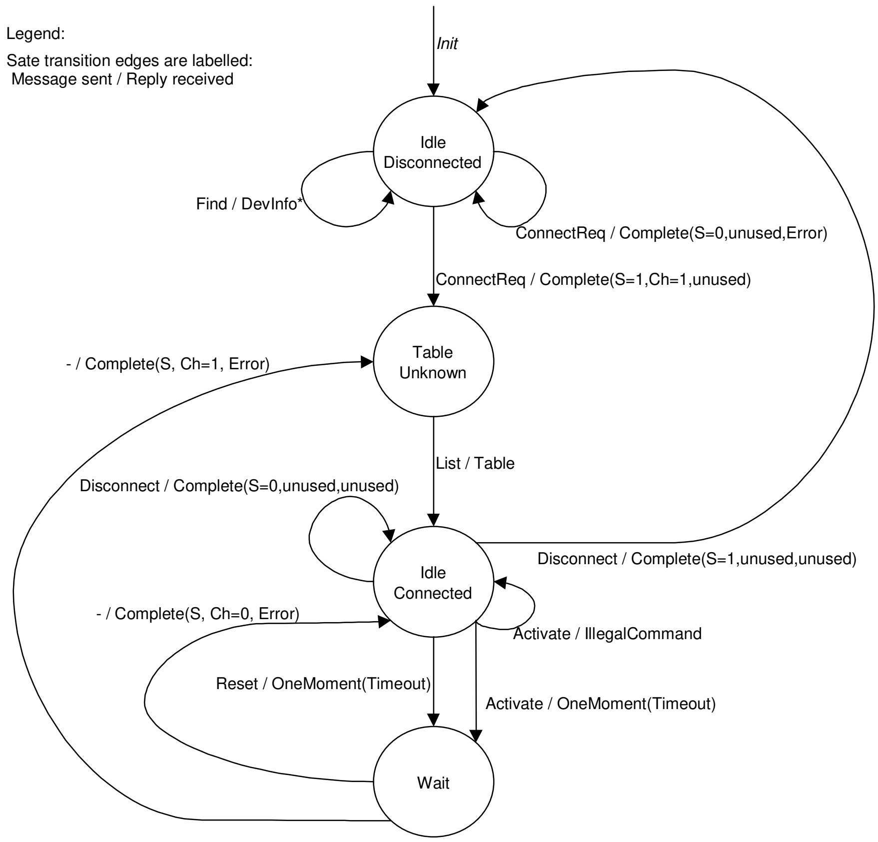

Fig. 3. Console State Transition Diagram

During the set-up of a connection, there is a negotiation of the choice of transmission format of which we (currently) envisage three - text, audio and graphical (i.e. icons). The format of the ConnReq message (see Fig 4 below) supports this process and facilitates exploiting optimizations in the underlying communication support.

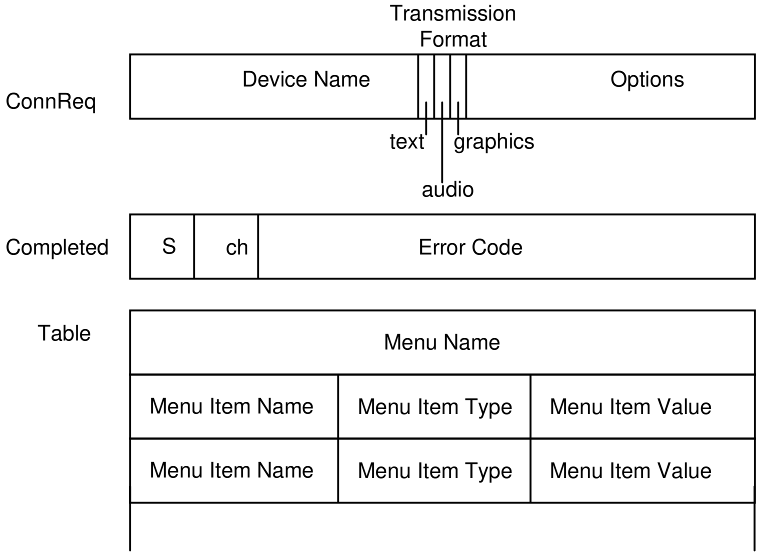

Fig. 4. URCP Message Formats

The options field in the ConnReq message is used to supply further details about the transmission format – e.g. if text then which language and/or character set are being used, or if icons are being used, the precise format. The Completed message provides space to return an error code if the S flag is cleared indicating failure. In the Table message, provision is made for the name, type and value of each menu item to be sent.

In this section we have presented the design of URCP including state transition diagrams and message formats. Our experience with a prototype implementation (see next section) indicates that the URCP is sufficient to facilitate interaction with supporting appliances.

3.0 Prototype Implementation

As part of a short feasibility study, we used readily available hardware and software to investigate whether URCP would support the building-interactions we envisaged. We implemented URCP on an infra-red communication medium using a Palm III PDA as a console. We simulated several domestic appliances above an IrDA-enabled Windows 98 PC (a CD player and a TV). A simple CD player was written in Visual Basic using the Windows Multimedia Control Interface (MCI) API.

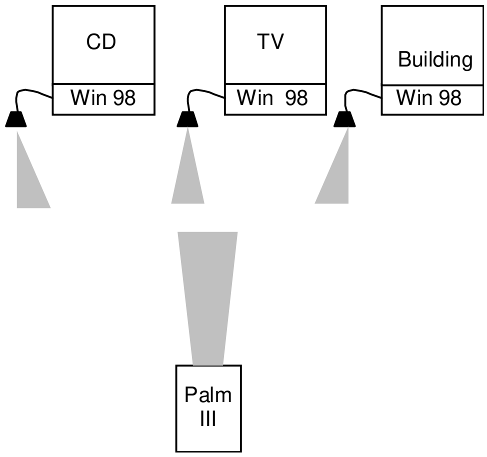

Fig. 5. Prototype Implementation for URCP

A similar front-end was written for a TV-tuner card inserted in another PC. A simulation of a building was written for yet another PC to experiment with the URCP Protocol in supporting a console for a smart building. All PCs were equipped with a small ActiSys IrDA transceiver. Windows 98 was used because the IrDA interface for Windows 95 did not appear to work [11].

One the console side, a standard WIMP interface was implemented on the Palm III together with a scan-and-select console which required the user to use only a single button. We also implemented a trivial encryption scheme with applications that require privacy in mind (e.g. ATMs).

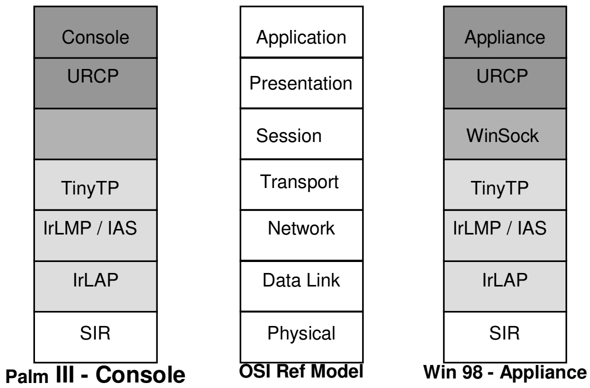

Fig. 6. Networking Architecture for Prototype Implementation

The networking architecture used is depicted in Fig 6. The physical layer, IrPhy or SIR (Slow IrDA) is a “lite” implementation of the IrDa standard intended for PDAs and PCs. It provides half-duplex, serial data transfers between line-of-sight aligned transceivers. The IrLAP (IrDA Link Access Protocol) data link layer is based on HDLC, and negotiates data rates during connection handshaking and incorporates simple flow control mechanisms. IrLAP provides a simple reliable bit stream connection between two devices. Further, IrLAP adds a discovery protocol that is used to enumerate other IrDA devices in range. It retrieves their 32-bit device address, their nicknames (e.g. "PalmIII") and a bitmask of hints that suggest the type of device (e.g. "PDA"). The IrLMP (IrDA Link Management Protocol) layer adds multiple session support to IrLAP. In addition, IrLMP includes a directory services protocol, called Information Advertising Service (IAS) to accommodate multiple session support. IAS stores application service names (defined by the application providing the service) and a corresponding Link Service Access Point Selector (LSAP-Sel) which is used to access the service remotely. The TinyTP protocol provides independently (per LSAP-Sel) flow-controlled transport connections (packet oriented) to overcome the problem of many multiplexed channels on one flow controlled medium as well as segmentation and re-assembly of large data buffers. It accesses the IrLAP layer directly, bypassing the Link Management Protocol. Usually, this is the protocol (together with IrLMP) which is accessible via the API. All these layers were already available on both the Palm III and Windows 98. In the latter case access to the TinyTP layer was via the Windows WinSock API.

On the Palm III side, an event-driven C program implemented the URCP and various simple consoles. URCP was implemented, again in C, as a Dynamic Link Library (DLL), on the Windows side while the appliances were implemented as separate Visual Basic applications. Following resolution of some problems posed by misleading documentation, implementation on both platforms was fairly straightforward despite the very different programming styles (Palm III applications must be written in an event-driven style).

4.0 Evaluation & Future Work

While our work on URCP does not directly involve the design of user interfaces, it is intended to support user interaction with a smart building and its appliances using several modalities. A poor design choice at the heart of a communication mechanism can obstruct good design choices at either end. In the spirit of the separation of logical and physical markup inherent in style sheets, we have tried to support the abstract essence of command interfaces. This includes identifiably tagging the relevant parts of traffic to facilitate flexible manipulation at both ends and its efficient passage through the communication infrastructure. We first examine how URCP supports or hinders the seven principles of universal design discussed in Section 1.2.

1. **Equitable Use**
   Different transmission formats, as indicated in the format of the ConnReq message, allow URCP to support differing interaction modalities.

2. **Flexible Use**
   This is principally the task of the user interface, but URCP supports a change in menu item type in mid-interaction (via an appropriate appliance command), assuming the appliance provides one. The current URCP design does not (yet) provide a means to identify traffic type and exploit possible underlying communication optimizations.

3. **Simple and Intuitive Use**
   Again the task is mainly a one for the user interface, but URCP faciltates prompt feedback by being efficient itself and facilitating optimizations in lower layers (with the exception of changing traffic type noted under principle 2).

4. **Perceptible Information**
   Again the user interface is mainly responsible here, but as URCP does not impose any constraint on the type or value of menu items, carriage of redundant presentations is possible and identifiable by the communicating parties.

5. **Tolerance for Error**
   URCP's Table message supports the grouping of related commands in one menu, and the typing tag could be used to identify menu items with hazardous consequences.

6. **Low Physical Effort**
   Not applicable

7. **Size and Space for Approach and Use**
   Not applicable, except that as our prototype shows, a light protocol such as URCP is unlikely to impose great demands on either communicating party's platform, so it should not hinder the comfort of a wearable computer or its interface(s).

While our work did not involve the construction of user-interfaces directly we have tried to ensure that the design of URCP supports and does not hinder such an effort, as illustrated above.

As might be expected with such a prototype, performance of the IrDA 1.0 link, especially over longer distances was somewhat disappointing. The objective of our investigation was to assess the functionality of the protocol not optimize its performance. We therefore restricted ourselves to readily available hardware. The Palm III console functioned (with fresh batteries) over distances of up to 3 m from the PC IR transceivers. Data rates at this distance were very low. To obtain the highest data rates, a separation of about 10 cm was required. Under these conditions, the highest data rate that could be obtained was 19.2 kbps. Instrumenting the software showed, as expected, almost no overhead in the URCP and application code. There were however long latencies (sometimes up to 5 seconds) during connection set-up and command execution. This appeared to be due to interference between neighbouring transceivers with the medium being continually reported busy. Further tests appeared to show that the transceiver was being continually polled by Windows software despite this behaviour being explicitly switched off in the Windows IrMonitor Control Panel application! Polling causes the transceivers to transmit an IR signal and the Palm III console and neighbouring PC transceivers pick this up.

Our experience indicates that despite the poor performance due to other predictable factors in the platforms initially available, URCP is an efficient protocol that appears capable of supporting interaction via a lightweight remote console. On both platforms URCP was implemented in about 1000 lines of C code. Our measurements indicate that measured latencies are due to the underlying protocol stack and errant behaviour of one of these (persistent polling on the Windows side). With better IrDA implementations becoming available and more powerful transceivers we anticipate that the difficulties we experienced will not reoccur. Data rates as high as 4 Mbps are promised for IrDA 1.1 that should comfortably permit further traffic over such a link.

Support for the multi-modal interactions we envisage might be provided using a Java-based platform and associated class libraries. We eschewed this choice since the class libraries were not fully defined and the first draft of the associated Jini architecture was not released. It is still far from clear that the supporting protocols (e.g. RMI) will adequately support the more demanding modes (i.e. speech). We also wish to minimise the demands on memory and powerpack lifetimes on the mobile console as far as possible. In the case of speech this means offloading the compute intensive task of recognition to other machines.

An alternative software architecture for a universal remote console might be based on an Object Request Broker (ORB) as in the CORBA standard. However, current ORBs are rather large applications for a handheld platform and such an architecture would require all appliances to export an object-oriented interface that could be encapsulated in a CORBA object. Some economy in code size could be achieved by restricting the services supported by a handheld

ORB and/or making it client-side only. In the applications envisaged for URCP, the requirement for a discovery type protocol or a naming service appears evident. Whether the CORBA approach is followed or not, this standard and its supporting protocols may be a source of inspiration for the development of URCP.

In the short term we intend to expand the complexity of our test applications further and port URCP to a radio-enabled PDA. This is to support our smart building project and to remove the requirement for aiming the console at another IR transceiver during interaction. Obvious candidate technologies include DECT, Bluetooth and a wireless LAN based on IEEE 802.11.

Developments planned for URCP include support for re-negotiating the transmission format in mid-interaction (perhaps due to a change in interface) in a way that permits exploitation of efficiencies in the lower protocol layers. Another task is to provide support for multiple simultaneous interactions from the same console so that the user can switch between different appliances without having to disconnect and re-connect. A longer-term suggestion is to build a console application that constructs agents and dispatches them to perform tasks asynchronously leaving results in a pre-arranged location for later retrieval.

5.0 Conclusion

In this report we have presented the design of URCP, a simple protocol to support multiple interfaces via a mobile console to a smart building and appliances within it. We have briefly discussed URCP in relation to seven principles of universal design. While these principles are most appropriate for interfaces themselves, we suggest that URCP efficiently supports and does not hinder the application of these principles to user interfaces serviced by URCP. We advocate that user interface designers consider these principles carefully. Our prototype implementation on readily available hardware justifies our confidence that URCP is simple and efficient and readily implemented on small portable platforms. Some disappointing performance results in the prototype are entirely due to flaws in the platform software available to us. We look forward to further work extending and porting URCP.

We have a proverb in Irish – "Tús mhaith leath na hoibre" - A good start is half the work.

We would like to acknowledge the help of several people and groups at TCD: Professor John Byrne for his encouragement and financial support, the Computer Architecture group for their tolerance while we "borrowed" some of their machines and the Smart Buildings Group for their continued support and encouragement. Thank you all.

References

[1] "Workshop on Interfaces for Wearable Computers", IEEE Virtual Reality and Artificial Intelligence Symposium. (see http://www.hitl.washington.edu/people/grof/VRAIS98/home.html)

[2] First International Symposium on Wearable Computers (see http://iswc.gatech.edu/iswc9798/wearcon97/)

[3] Second International Symposium on Wearable Computers (see http://iswc.gatech.edu/iswc9798/)

[4] "SIMULA Begin", G. Birtwhistle, O-J Dahl, B. Myhrhaug, K. Nygaard, Chartwell-Bratt, 1979.

[5] "Universal Remote Console Communication Protocol", G. Vanderheiden, Proceedings of the TIDE Congress – Improving the Quality of Life for the European Citizen, pp 368-373, Helsinki, June 1998, published by IOS Press, Amsterdam, 1998

[6] "Universal Auxiliary Interface Protocol", Working Paper, 15 July 1996, Trace Research & Development Center, University of Wisconsin-Madison,.(see http://www.trace.wisc.edu/docs/uaip/uaip.htm)

[7] "Object-Oriented Software Construction", B. Meyer, Prentice-Hall, 1997.

[8] "Universal Design: Planning and Design for All", F. Aslaksen, S Bergh, O. Bringa and E. Heggem, Norwegian State Council on Disability, Olso, Norway, 1998.

[9] "Principles of Universal Design", Version 2.0, April 1997, Center for Universal Design, North Carolina State University, 1995 (see http://www.design.ncsu.edu/cud/pubs/udprinciples.html)

[10] "Auditory Interfaces: Towards the Speaking Computer", T. V. Raman, Kluwer Academic, 1997.

[11] "Writing Ad-Hoc Point-to-Point Networking Applications using IrDA", Mike Zintel, Microsoft Press, 1997

[12]"Smart Clothing: The Wearable Computer and WearCam", S. Mann, Personal Technology, Vol 1., No 1., Springer Verlag, Berlin, 1997

[13] "Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring Urban Environments", S. Feiner, B. MacIntyre and T. Hollerer, Proceedings of the First International Symposium on Wearable Computers, pp 78-81, IEEE CS Press, Los Alamitos, CA, 1997,

[14] "Wearable Computing at Boeing", D. Mizell, Workshop on Interfaces for Wearable Computers, March 1998. (see http://www.hitl.washington.edu/people/grof/VRAIS98/home.html)

[15] "Wearable Computer Interfaces for People with Disabilities", K. Echt, Workshop on Interfaces for Wearable Computers, March 1998. (see http://www.hitl.washington.edu/people/grof/VRAIS98/home.html)

[16]"WIMP Interfaces Considered Fatal", B. Rhodes, Workshop on Interfaces for Wearable Computers, March 1998. (see http://www.hitl.washington.edu/people/grof/VRAIS98/home.html and "New Ways to Manage Information", M. Billinghurst and T. Starner, pp 57-64, IEEE Computer, Vol 32., No 1., January 1999.)