

Remote UI protocols for home environment

Juha Leppilahti
Helsinki University of Technology
Juha.Leppilahti@hut.fi

Abstract

Increasing number of home appliances make the User Interfaces (UI) and usability of these devices challenging. In this paper we examine different remote UI protocols that enable unified and centralised remote controlling in home environment. The binary UI protocols aim at transferring the native UI with the native look and feel where the markup protocols transfer the description of the UI and hence allow unified look and feel that suits best for the displaying device. This paper examines Widex and UPnP protocols. UPnP is a big interoperability enabler that supports multiple remoting protocols. Widex on the contrary is still in early specification phase but has XML DOM based UI support and hence generic and flexible architecture for UI rendering. We also observed that although these two protocols are spreading and particularly UPnP is deployed in wide scale the support for device dependent protocols is still needed.

KEYWORDS: UPnP, Widex, Remote UI

1 Introduction

This paper discusses remote user interfaces in the home environment. Modern home devices are full of features but employing these features completely requires sophisticated methods of interaction with users. These devices traditionally are not delivered with multipurpose User Interfaces (UI) and monitors same way as computers, but instead these home appliances might only have a tiny screen and a simple remote controller. In addition to this, many homes may be equipped with dozens of devices all which have different user interface or operating logic. Hence, introducing a technical solution that could simplify usage of such devices or introducing a protocol that could unify UIs might make it possible also for ordinary people to enjoy benefits of modern entertainment technology more easily.

This study gives an introduction to solutions that try to unify the way interaction with different devices is handled and how interaction events can be transferred between devices which have been implemented by competitors. Paper introduces briefly different protocol types and highlights the essential features and key elements for the reader and thus can serve as an introduction to remote UI technologies of today. Reader can find an introduction to protocols like Widex, UPnP, SIP, VNC or UIF

The main focus of this paper is to describe Universal Plug and Play (UPnP) remote UI and Widget Description Exchange Service (Widex) operations and protocol behav-

ior. UPnP is especially important since it has been adopted by many major companies implementing home appliances. Next section gives an overview of recent standardization and development activities of remote UI protocols and lists the most common solutions available at the moment. Section three discusses different types of remoting protocols. Section four covers the inspection of UPnP remote UI case. Then section five discusses Widex and finally in section six we draw conclusions about remote UI technologies of today and discuss weaknesses and benefits of UPnP and Widex compared to different protocol types on the market.

2 Remote UI implementations and recent work

Remote user interface means a device or action can be controlled off-site or from a distance. It is also essential for user interface that the controller will receive some response to made requests i.e. feedback. Implementing such technology may not be easy but fortunately there are quite many solutions available. The solutions may be completely proprietary or based on open source code, open standards or patented technologies. Internet Engineering Task Force (IETF) standardization organization has a Charter Working Group which is developing Widex which is a generic solution of remote UI. Then there are big corporations like Microsoft which have participated to organizations dedicated to deliver ubiquitous computing solutions such as UPnP. In addition there are proprietary or patented technologies which may be used by licensee. Similarly the technical implementations may differ significantly from the others. The next sections briefly list different types of remote UIs.

2.1 UPnP Remote UI

UPnP technology suite provided by UPnP foundation is one of the most advanced and spread technology for home environment usage. UPnP suite allows dynamic service discovery in local area network and in addition it specifies services and service description for different purposes but still doesn't require any configuration effort. UPnP Remote UI (RUI) is a remote controller specification of UPnP and is covered in its own section later on. UPnP is a good example how companies can by co-operating come up with a technology that is both usable and supported by many different companies.[1]

2.2 Widex

Widex is a protocol developed by IETF to solve the remote UI problems. It is based on Nokia's Lightweight Remote Display Protocol (LRDP) and specifies service discover and remote controlling through different type of DOM objects which are passed over the network.[2] Widex and LRDP architecture and functionality is explained in dedicated section.

2.3 SIP

Session Initiation Protocol (SIP) is a protocol designed for signaling and is supported by many Voice over IP (VoIP) clients. IETF's SIP is also the official signaling protocol of 3GPP IP Multimedia Subsystem (IMS). Since SIP is open session signaling protocol it can also be used for media session controlling which makes it possible protocol for home media controlling. While the SIP hasn't been yet adopted to remote user interface implementations, the IETF community has still written a draft that explains the application interaction framework for SIP based interactions and thus remote UIs.[3]

2.4 VNC

VNC (Virtual Network Computing) is one of the most common remote controlling softwares for computer environment. It enables desktop sharing over the network by transferring raw image data to client and enabling device usage similarly to local usage. In VNC the client is made as light as possible and almost all work is done by the server. The server needs to check and notice changes on the screen and then send updates to the client. The client is then able to react and send keypresses or mouse movement back to the server.[4]

2.5 X Window System

X Window System also known as X11 or X is a protocol designed to display graphic over the network or on a monitor. X is commonly applied to Unix-systems as a graphical interface core but there're clients also for other operating systems. Employing X remotely requires operable X server also on client side which makes X a bit more complex choice when compared to other protocols. X can be compared to RDP.

2.6 RDP

Remote Desktop Protocol (RDP) is the Microsoft's solution for remote controlling of personal computers. It was deployed in Windows operating systems and is based on delivering Windows's native graphic calls over the network.[1] It is similar to VNC in a sense that all the same components that are seen in the local screen can be observed also from the remote screen as well. This also means that the protocol doesn't describe any abstract controlling components for client to interpret but contains exact definition of the UI. Because of the native API usage the RDP is only used with Microsoft Windows operating systems.

2.7 UI fragments

Philip's home remoting UI protocol UI Fragments (UIF) is markup protocol which is designed for the home environment. It specifies the typical UI elements needed for remote controlling in XML format and leaves the rendering on client's responsibility. This means that clients may implement the interface according to their own rendering rules. In addition UI fragments include a mechanism for partial interface updates to make the performance better.[5]

2.8 XRT2

Intel's Extended Remote Technology (XRT2) is a binary protocol developed for home environment usage. It is compatible with UPnP RUI and is designed to be scalable with different monitors. XRT2 uses binary coded predefined commands to communicate with client and server. The server decides how the client should render its display by sending the painting commands and images over the network.[6]

3 Protocol types

Remote UI protocols are roughly divided in two main categories. These categories comprise so called markup protocols and binary UI transfer. Binary UI transfer protocols are normally simpler than markup protocols since basically device just sends its native UI and events over the network. The markup protocols on the contrary express UI through formal expression which is then interpreted by the client.

3.1 Native UI transfer

Native UI transfers is a mechanism to display UI on client device. The UI outlook is specified by the server and the client merely acts as an dummy terminal or front end. In the simplest case server sends a bitmap screen capture of the user interface and the client shows it.

Employing Native UI transfer protocol is both simple and generic. The disadvantages however include problems with bandwidth usage, delays when updating the UI after a keyboard or mouse event, trouble with managing different screen resolutions or features, impossibility of transferring smooth UI transitions or animations and the lack of having the device dependent so called Look and Feel (LAF). However the majority of the remote UI implementations are based on Native UI transfer of some sort including such widespread protocols as VNC, RDP or X Window System.

3.2 Markup protocols

Markup protocols in contrast to binary UI transfer protocols have a completely different approach to remote UI problem. These protocols do not specify which kind of bitmaps or icons should be displayed but enable a generic description and framework for an UI. Through this framework markup protocols describe what kind of elements UI needs to have and how they should act. By transferring only the description, these protocols enable LAF which then can be adopted

from the controlling client or from the server depending on the way the UI is delivered.

If it's client's responsibility to render the UI, it means that client's features give boundaries to different UI effects such as animation, transition, resolution, instant feed back when pressing a button or for example having similar control buttons for all controlled appliances. Markup protocols implement remote UI by transferring the description of the UI. Mechanisms like Hypertext Markup Language (HTML), Extensible Markup Language (XML) and Scalable Vector Graphics (SVG) can be utilized for presenting the UI and updating it.

In this paper we examine the Widex and UPnP which are both markup protocols although the UPnP is not that much related to actual UI and suits better for enabling remote UI sessions.

4 UPnP remote UI

UPnP as a protocol framework provides the basic services which include service discovery, sessions, different service templates and interaction mechanisms. The actual usefulness is enabled through the service implementations which include the UPnP Remote UI for instance. Remote UI specification comprise device and service templates for both server and client sides of the protocol including features such as session management. Although UPnP remote UI alone doesn't explain how the actual UI should be transferred or presented it focuses on the UI sessions so that clients and servers can exchange the information needed for enabling remote UI. For example the devices might even use VNC for the actual remoting. One could wonder the benefit but it is easy to see that without the UPnP there wouldn't be any automation in remote UIs. UPnP enables well defined automatic service discovery and protocol negotiation and gives a possibility for UI clients to choose the remoting method they found the most usable and compatible. This section will cover the basic principles of UPnP as well as it will try to explain how the remoting UI could work through UPnP.

First we give a short introduction to UPnP and then explore the basic device architecture level by level. Third section examines the Remote UI server and client implementations leading us to the complete overview about the UPnP's applicability and distribution at the moment.

4.1 UPnP Overview

UPnP protocol framework consists of exact definition of interaction mechanisms that enable configuration free service discovery and usage in different networking environments. The UPnP benefits from the existing protocol solutions and employs Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) on Internet Protocol (IP) networks. In addition to traditional Internet protocols UPnP introduces simple Hypertext Transfer Protocol (HTTP) over UDP mechanism for lightweight service discovery message exchange. TCP is used for transferring device and service descriptions and payload over the network. In addition these descriptions are based on standardized XML templates

which implement some of the UPnP Device Control Protocols (DCP) such as Remote UI or Printer. Through these mechanisms UPnP enables a framework where the clients, namely control points, search for devices which offer services.

4.2 UPnP Layers

UPnP architecture is layered where the different layers are called steps. There are six steps overall which all are briefly introduced in this chapter. The information presented is based on reference [7].

The first layer, namely Step 0, handles the addressing. Addressing contains the basis of all UPnP communication as it makes it possible for computers to communicate over the IP network without any configuration effort. In addition to well-known dynamic addressing and naming concepts like DHCP (Dynamic Host Configuration Protocol) and DNS (Domain Name System) resolving, the UPnP introduces link local IPv4 addressing which is called Auto-IP. In addition the specification also explains how the host should deprecate addresses or check for addressing conflicts in network.

Furthermore the second layer, namely Step 1 or Discovery enables hosts to find each others. As the addressing only enables the communication the discovery specifies the way network is used to find other UPnP hosts. Basically UPnP uses multicast and unicast IP messages to communicate with other hosts. To be precise the multicast messages are utilized to request service advertisements (i.e. search services) or advertise hosted services for other UPnP hosts. In contrast, the unicast messages are in use when hosts start to exchange information. In general, the UDP is in use with multicast messages and when sending service announcements to specific host and then the TCP is in use when hosts transfer bigger data blocks. By advertising host can notify that it has joined the network and it has some services available. These service advertisements are in predefined format and always announce some service accepted by the UPnP foundation like for instance the Remote UI which is the most interesting service for this paper.

The third layer, namely Step 2 or Description layer provides more information about devices and their services. Although the knowledge about available services exist after the Discovery the Description gives the detailed information about them. The Description is based on XML-documents identified by the URL (Uniform Resource Locator) in the service advertisement. The devices which are interested about some specific services fetch the document. In UPnP single physical device may contain several logical devices which all provide different services for the UPnP network. To support such mechanism UPnP foundation has given templates and mechanisms that identify so called root device. UPnP device can then first get the root device description and after gaining knowledge about the included logical devices fetch the services for them.

Fourth layer known as Control or Step 3 implements the remote control functionality of the UPnP protocols. In practise the mechanisms is based on request-reply -protocol. The specification [7] compares the mechanism to remote procedure call:

Control messages are also expressed in XML using the Simple Object Access Protocol (SOAP). Like function calls, in response to the control message, the service returns any action-specific values.

The fifth, Eventing layer implements another kind of messaging mechanism which based on subscribing notifications. In practise the device which is interested in some specific state information, for example wheather a video is playing or not, might order events which are received every time the state changes on the server. Eventing layer defines messages for subscribe management and event templates. The eventing architecture is developed by IETF and called General Event Notification Architecture (GENA).

The final and the highest layer of the UDA (UPnP Device Architecture) is the Presentation layer that utilizes the services provided by the layers discussed in former paragraphs. Basically presentation can be seen as loading a HTML web page and then using the functionality of the web page for controlling or viewing the remote source.

4.3 Remote UI Server

UPnP Remote UI definition is divided to server and client side. Both sides are then divided to device definition and service definition which follow the UPnP templates. The Remote UI Server Device definition in practise includes a way to list information about the physical devices (e.g. manufacturer, model) but the most important information it offers are links to services, controls, eventing and presentation. These URLs can then be used to retrieve the actual Remote UI server service description from the device.[8]

Service specification then explains how the UPnP should give information about the UI and which protocols should be used for remoting. These actual remoting protocols namely out-of-band protocols in UPnP might be based on markup language or be completely vendor specific. Nevertheless UPnP enables session setup for such protocol as VNC, RDP or UI fragments through service template that has fields for protocol specific values. These values could include port numbers or other protocol specific and required information. A complete list of specified remoting protocols in the server services definition [9] can be found below.

- HTTP/HTML
- RDP
- VNC
- XRT2
- LRDP
- XHT
- SGXML
- UIF
- any vendor specific

To be precise UPnP RUI isn't a RUI implementation but a framework that enables remote user interfaces through other protocols. UPnP enables different devices to negotiate a compatible remoting protocol. Server specification also allows servers to find suitable UI renderers from the network by using the mechanism called *GetCompatibleUIs*. In addition these can be filtered to further enhance the search mechanisms.

4.4 Remote UI Client

Client has similar device template definition for enabling the service usage.[10] The client service in the opposite to server device and services definition concentrates more on the UI session management such as connecting and disconnecting to the server. In addition UPnP RUI client defines simple text messages to enable generalized message passing mechanisms that can be applied for notification without understanding the specific remoting protocol. The *ProcessInput* further on describes a generic method for providing input to RUI client. For example keyboard presses can be generated from the server side and send over the network and then RUI client would act the same way as if the user would press the key. The more advanced use cases of the UPnP RUI include moving, sharing a UI session.[11]

4.5 Conclusions

UPnP is based on XML based markup language but still it isn't a remoting user interface protocol. UPnP RUI is a framework and session protocol which cannot be thought as RUI protocol of any kind. The plain UPnP however is a remoting protocol although not a remote user interface protocol. Consider a UPnP controlled media player or firewall for example. The RUI specification concentrates on enabling Plug-and-Play behavior for third party protocols instead of specifying how the UI should be presented.

UPnP enables devices to find each other and find a compatible communication method but when the actual remoting takes place UPnP doesn't provide anything very useful besides the service availability announcements. However the generality and extendable specification of the UPnP allow many vendors to use UPnP as interoperability enabler with other devices.

5 Widex

Widex is a remoting UI protocol that uses markup language similarly to UPnP. The information presented in this chapter is based on LRDP[12] and Widex specification of the IETF[2][13]. LRDP is the ancestor of Widex and specifies the protocol.

An application layer protocol for a framework that enables accessing graphical user interface (GUI) applications remotely, so that devices with different form factors and UI capabilities can scale and adapt the exported UI to their local platform UI look and feel (LAF). Specified in the Extensible Markup Language (XML), the protocol

defines generic user interface (UI) operations and a mechanism for extending these operations for specific application needs.

In the following chapters we examine the Widex specification. First we give a short overview of the Widex and the next chapter then illustrates the basic functional model in the Widex. Further on the next chapter explains how the framework enables this model behavior. The third chapter concentrates on the objects utilized by the Widex for the interface updates and finally in the last chapter we examine the UI presentation through DOM objects. Conclusion will follow after that.

5.1 Architectural Overview

Widex is simple, the basic architecture contains only two entities, Widex Server (WS) and Widex Renderer (WR). Compared to LRDP, these are same UI Server and UI Client similar to UPnP's Remote UI Client and Remote UI Server. I.e. the WR displays the user interface for application running on WS. In addition to simple client-server architecture the content is packed into objects called Widex Objects (WO) which are utilized for transferring UI information for both directions. Objects may contain either complex or simple user interface data objects which are packed as XML DOM (Document Object Model), which enable flexible XML manipulations.

5.2 Model

Widex model is the basic definition of the Widex architecture. Framework specification [13] explains the framework model MVC (Model-View-Controller) usage the following way:

The model contains the core functionality and data. Views display information to the user. Controllers handle user input. Views and controllers together comprise the user interface. A change-propagation mechanism ensures consistency between the user interface and the model.

Then the framework specification also explains how the framework model is extended to use the transferred objects through so called updates and events which utilize network to bind the Controller and the View. The Controller in the Widex is located on the WS together with the Virtual View that maintains the state of View data in case the connectivity is lost. In contrast the actual View is always located on WR and updated by exchanging the XML DOM objects in the update and event actions.

5.3 Framework implementation

Similarly to UPnP the Widex framework implementation needs to take care of session discovery and session establishment. The other responsibility for the framework is to keep the views up to date on renderer and server and take care that the updates and events are transferred over the network.

Service discovery is not yet completed in specification which means that there isn't any Widex specific methods for

service discovery unlike in the UPnP. The framework however mandates that the service discovery is from the Widex point of view treated as a black box and hence can be any available method. Session setup is a bit more strictly specified and, according the framework specification[13], should negotiate DOM objects support level, device settings and Widex communication methods.

Synchronization and transport mechanisms take care about the session after establishment. Transport part of the Widex framework takes care of the object transfer while the synchronization part serializes events and updates which are presented as XML DOM objects.

5.4 Objects and Events

The key functionality of the Widex is hidden inside the DOM objects and developed together with W3C. And as the user interface is presented as XML DOM object and trees, also the updates are specified with these same mechanisms. The simple user interface model comprise only one XML DOM. The server side updates to the user interface are then executed with update object event, namely WO.Update, which is W3C specified REX (Remote Events for XML) object [14]. The complex RUI employs the same mechanisms but doesn't have the restriction of one DOM page but may contain several DOMs and scripted functionality.

The RUI side events, namely WO.Event, are on the contrary not updates to any DOM model but some events that user has caused on the renderer which need to be communicated to the server side. Furthermore application logic then interprets these events and user can see them as a device functionality. The WO.Events are packed as XML based EMMA (Extended Multi-Modal Annotations) format [15].

5.5 DOM objects

Widex displays handles and displays user interface as DOM objects. DOM is model to handle XML and HTML structure in formal way so that the HTML or XML is defined in hierarchical order and different entities can be modified or be updated separately. DOM specification enables easier programming of HTML or XML components by specifying an API to XML and HTML. This paper doesn't cover the DOM model further but those interested may explore the specification from the W3C site <http://www.w3.org/DOM/>.

5.6 Conclusions

Widex is not yet deployed or even developed completely but as the IETF aims to broad compatibility through utilizing different IETF protocols such as SDP or SIP or even to support protocols outside the IETF like UPnP we can assume that Widex will gain some users at some point. Widex is mostly developed by Nokia and is in very early phase. To be exact only the requirements and the framework standard documents have been started and there aren't any operable implementations yet.

6 Conclusions

In this paper two different type of RUI protocols were inspected. The binary UI protocols, which aim at transferring the native UI with the native look and feel, were left out but the markup protocols, that transfer the description of the UI and hence allow more unified UI and LAF that suits for the displaying device, were inspected with Widex. In addition we chose UPnP RUI as the second protocol although it doesn't take part on actual UI presentation. These markup protocols provide flexibility for clients over the binary UI protocols and thus the focus of the paper was to examine and compare these two protocols carefully from the session side.

For Widex we saw a very general idea of the protocol because of the early phase of development and thus were not able to compare the exact mechanisms. The focus of Widex was however clear and effectively aiming at precise UI definition through utilizing the W3C DOM specifications. The session discovery mechanism were supported but not defined same way as with UPnP which gives us possibility of employing UPnP together with Widex. Although it might sound strange these protocols could work together since we noticed that UPnP's remote UI supports LRDP which is the ancestor of Widex. Enabling Widex's generalized but highly defined UI mechanisms and UPnP's exact session and service discovery might prove to be a operable and flexible combination and might be an interesting topic for further studies.

However, after going through the study field we were left with the feeling that although IETF's Widex development is proceeding it might be a bit late. Several big vendors already have well defined protocols and are co-operating to generate a unified remoting protocol. In addition UPnP itself doesn't solve any remoting needs but seems to have a high push from the industry and hence probably will be a mandatory part of home devices in future and an important enabler when unifying connectivity of home appliances. Although Widex and UPnP were chosen for separate inspection in this paper it might be interesting to see how the UPnP will co-operate with these different protocols and which one of the actual RUI protocols will become the standard way for home device interaction.

References

- [1] Gertjan van Montfoort. Quality of Service for Remote UI University of Utrecht, Master of Science thesis in the Department of Computer Science, August 2005
- [2] V. Stirbu and D. Raggett. Widget Description Exchange Service (WIDEX) Requirements draft-ietf-widex-requirements-04 Internet-Draft, IETF Charter Working Group, January 2007 Available: <http://www.ietf.org/internet-drafts/draft-ietf-widex-requirements-04.txt> [Ref:20.3.2007]
- [3] Rosenberg J. A Framework for Application Interaction in the Session Initiation Protocol (SIP) draft-ietf-sipping-app-interaction-framework-05 Internet-Draft, IETF Session Initiation Proposal Investigation Working Group, July 2005
- [4] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood and Andy Hopper Virtual Network Computing In Internet Computing, IEEE Volume 2, Issue 1, Jan.-Feb. 1998 Page(s):33 - 38
- [5] Richard Verhoeven and Walter Dees Defining Services for Mobile Terminals using Remote User Interfaces In Consumer Electronics, IEEE Transactions on Volume 50, Issue 2, May 2004 Page(s):535 - 542
- [6] Intel XRT2.1 Binary Remoting Protocol, version 0.9 Intel, September 2004 Available: http://cache-www.intel.com/cd/00/00/23/41/234192_234192.pdf [Ref:18.3.2007]
- [7] Contributing Members of UPnP Forum. UPnP Device Architecture UPnP Forum, June 2000 Available: <http://www.upnp.org/resources/documents/CleanUPnPDA101-20031202s.pdf> [Ref:20.3.2007]
- [8] Mark Walker and Ku Bong Min RemoteUIServerDevice:1 Device Template Version 1.01 Contributing Members of the UPnP Forum, September 2004 Available: <http://www.upnp.org/standardizeddcps/documents/RemoteUIServerDevice1.0.pdf> [Ref:20.3.2007]
- [9] Jan van Ee and Mark R. Walker. RemoteUIServer:1 Service Template Version 1.01 Contributing Members of the UPnP Forum, September 2004 Available: <http://www.upnp.org/standardizeddcps/documents/RemoteUIServerService1.0.pdf> [Ref:20.3.2007]
- [10] Ylian Saint-Hilaire, Mark Walker and Ku Bong Min RemoteUIClientDevice:1 Device Template Version 1.01 Contributing Members of the UPnP Forum, September 2004 Available: <http://www.upnp.org/standardizeddcps/documents/RemoteUIClientDevice1.0.pdf> [Ref:20.3.2007]
- [11] Ylian Saint-Hilaire, Mark R. Walker and Jan van Ee. RemoteUIClient:1 Service Template Version 1.01 Contributing Members of the UPnP Forum, September 2004 Available: <http://www.upnp.org/standardizeddcps/documents/RemoteUIClientService1.0.pdf> [Ref:20.3.2007]
- [12] V. Stirbu, P. Shrubsole and J. Costa-Requena LRDP: The Lightweight Remote Display Protocol draft-stirbu-avt-lrdp-00 Internet-Draft, IETF Charter Working Group, February 2005 Available: <http://www.cs.columbia.edu/~{hgs}/rtp/drafts/draft-stirbu-avt-lrdp-00.txt> [Ref:20.3.2007]
- [13] V. Stirbu and D. Raggett Widget Description Exchange Service (WIDEX) Framework

draft-stirbu-widex-framework-03, IETF Charter Working Group, October 2006 Available: <http://tools.ietf.org/wg/widex/draft-stirbu-widex-framework-03.txt> [Ref:20.3.2007]

[14] R. Berjon Remote Events for XML (REX) 1.0 W3C Working Draft, October 2006 Available: <http://www.w3.org/TR/2006/WD-rex-20061013/> [Ref:13.3.2007]

[15] M. Johnston EMMA: Extensible MultiModal Annotation markup language W3C Working Draft, September 2005 Available: <http://www.w3.org/TR/2005/WD-emma-20050916/> [Ref:13.3.2007]