<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>Rapid prototyping of networked A/V CE appliances</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Corcoran, Peter; Callaly, Frank</td>
</tr>
<tr>
<td><strong>Publication Date</strong></td>
<td>2006-05-15</td>
</tr>
<tr>
<td><strong>Publisher</strong></td>
<td>IEEE</td>
</tr>
<tr>
<td><strong>Link to publisher's version</strong></td>
<td><a href="http://dx.doi.org/10.1109/EURCON.2005.1630199">http://dx.doi.org/10.1109/EURCON.2005.1630199</a></td>
</tr>
<tr>
<td><strong>Item record</strong></td>
<td><a href="http://hdl.handle.net/10379/1588">http://hdl.handle.net/10379/1588</a></td>
</tr>
</tbody>
</table>

Some rights reserved. For more information, please see the item record link above.
Rapid Prototyping of Networked A/V CE Appliances

Peter Corcoran, member IEEE and Frank Callaly

Abstract — In this paper the design and rapid prototyping of networked A/V CE appliances is described. Practical implementations of three types of appliance are described: (i) a streaming A/V server; (ii) a streaming A/V client device and (iii) a networked A/V transcoding device. It is shown how these devices can be used to provide both a robust and scalable multimedia solution, which can be adapted for use in many applications. A review of known techniques for realizing the required multimedia functionality is given and some methods of reducing the development time of these prototypes are also discussed.

Index Terms — Multimedia, Video & Audio Streaming, Home Networks, and WiFi (802.11).

[1] INTRODUCTION

Next generation home networks will be based on TCP/IP networking technology. However there are additional layers of software infrastructure present in a network. These additional software layers are required to support “configuration-free” networking of devices and to facilitate and support device interoperability. Recently there has been much interest in adding additional A/V functionality to home networking protocols to simplify the development of complex networked A/V CE appliances. In this paper we examine the development of such A/V appliances, and describe how these devices can be used together as a distributed A/V delivery system.

Probably the best known home networking middleware is Universal Plug & Play (UPnP) [1] technology which is incorporated in the latest versions of the Microsoft Windows OS. Other well known middlewares include the Java-based JINI [2] technology supported by Sun Microsystems, the HAVi [3] home gateway infrastructure supported by an industry consortium, the Rendezvous [4] (aka Zeroconf) technology supported by Apple computers and the Salutation technology from IBM. As UPnP has been more broadly adopted and documented than other current home networking standards it was decided to initially focus on UPnP compliant implementations for this paper.

For the purposes of this paper, we focus on three principle types of networked A/V CE appliance namely:

i. A/V streaming servers, known as MediaServers in UPnP parlance; these are devices which provide access to media content. The content may be derived from a number of sources including live terrestrial or satellite TV, removable or fixed storage devices (e.g. DVD, HDD) or other networked A/V servers (e.g. Internet servers). These devices allow the user to browse the available content so that items of interest can be easily located. They also provide an interface for streaming clients to access the content.

ii. A/V streaming clients, which are generally display devices, these are known as MediaRenderers in UPnP and are charged with the task of converting and/or filtering the received A/V stream to optimize presentation on a display console.

iii. A/V transcoding appliances, which convert a data stream from one format to another; for example it might be necessary to convert a high resolution MPEG2 stream to a low resolution MPEG4 stream for display on a handheld device. This type of device may also be used as a buffer, so that a user can be pause or rewind a live TV stream, even if this functionality is not supported by the originating streaming server.

Note that many practical A/V appliances will combine aspects of all of the above “building block” devices. During the design process, however, it is very useful to be able to focus on several simpler devices and subsequently illustrate how they may be combined into more complex devices.

[2] DEVELOPMENTS IN CONSUMER A/V NETWORKS

Part of the driving force behind the proliferation of A/V CE devices in recent times has been the developments in the field of multimedia compression. The development of multimedia compression standards by the ITU-T and MPEG groups, and in particular the MPEG-4 [5] and H.264 (aka MPEG-4 part 10) [6] standards, means that content can now be streamed over networks which previously could not handle A/V streams. This increased compression also allows for greater storage of media content, and so consumers have become accustomed to devices which can store large amounts of A/V data.

Another development which must be considered, is the widespread replacement of 11Mbps 802.11b hardware with 54Mbps 802.11a/g technology. The convenience and portability
that high-speed wireless networking provides has opened up many possibilities which were, until now, completely impractical. However, it should be noted that wireless networks are more susceptible to packet loss and other inconsistent behavioural aspects, this must be considered in the design of streaming A/V devices.

To put these developments into context, A DVD quality multimedia stream encoded with MPEG-2 would be in the order of 8Mbps but an 802.11b network could only provide a practical throughput of around 5-6Mbps. Now, a stream of similar quality encoded using MPEG-4 uses approx. 2-3Mbps and an 802.11g network can give throughput of 20Mbps. This allows for a number of streams on a network, and with the use of 108Mbps 802.11a/g “turbo” or the forthcoming 802.11n standard, the number of streams that a network can handle will rapidly increase.

Users can be slow to adopt these devices as there can be a fear of the perceived complexity of using networked devices, and many have had negative experiences of trying to configure devices to interact reliably. Many consumers still use devices in a standalone configuration, even though they can act as network devices. It was decided early on that one of the main design goals would be to make these devices as reliable as possible, and as user friendly as possible.

[3] Device Design

The server device prototypes were run on standard desktop PCs. It was felt that in many cases a home PC is the place where most of the media content will be stored, and so this would be a suitable platform. For the clients, low power VIA Mini-ITX motherboards were used, these include hardware MPEG-2 acceleration, which is used by the clients whenever an MPEG-2 stream is received, otherwise the stream is decoded in software.

Early on in the development of these devices, it became clear that there was a need for a very flexible programming platform in order to achieve the type of rapid prototyping desired. However some sections of the devices, in particular the media codecs, would need to be implemented at quite a low-level in order to use the available resources efficiently. It was decided that the most suitable solution was to use the Python [7] programming language with low-level sections written in the C programming language and wrapped as Python extensions.

Python is a very popular high-level programming language, the Python interpreter is implemented in C, which makes it quite easy to extend so that C libraries and routines can be called from Python code. The syntax and the data structures available in Python make it very suitable as a prototyping language.

Fig 1: Block Diagram of Streaming A/V Server Device

A number of openly available libraries were used in device development. Intel's UPnP library [8] was used to as the basis for the middleware components of the devices. A number of openly available media codecs were also used to encode/decode content, in particular the libavcodec and libavformat libraries from the ffmpeg [9] project were used, Developing new codecs for each of the supported media formats would be quite impractical when prototyping, as it would require a large development effort. The video display engine was based on the freely available Xine [10] video engine.

A number of reusable software components were developed specifically for these devices. An RTSP [11] video on demand (VOD) server was developed which is used by server devices to export content to clients. Using RTSP, a TCP connection is initially established between the client and the server, this connection is used to setup and control the stream. The A/V stream is then sent over a separate UDP connection, As the UDP data is being sent over a wireless network, the server will attempt to use the most robust format that the client can accept. This may be RTP packets or MPEG transport stream packets.

A specialized HTTP server was also developed, this server can be used in a similar manner as the RTSP server. It is used by clients that do not support the RTSP protocol. Both of these servers read A/V streams from loadable access modules. Each access module allows streams to be read from a particular source, e.g. hard drive, video capture card, CD, DVD. This means that they can be extended to read from new sources as they arise.

It should be noted that the transcoding device appears on the network to be separate Media Server and Media Renderer devices. This is very close to its actual make up, as the transcoding device contains many elements of both the Media Renderer and Media Server devices. However it cannot access multiple sources of media, nor can it render media on a display device. In many situations a transcoder device will be on the same physical device as a Media Renderer, so they will use the same hard disk drive. By choosing this architecture the
complexity is divided into manageable sections within each device.

To demonstrate the functionality of these devices, a control interface was developed, which can be run on a number of platforms including many PDAs. This control interface includes a stripped down UPnP library, designed specifically for this task. The control interface includes a GUI, which allows a user to browse content on any available server, and display it on any available client.

In Fig 4(a), a scenario is shown where TV programs are saved for later viewing. Here, the incoming analog, digital or Internet TV stream is saved to the hard disk. For this to happen, the user must instruct the Media Server to record the stream at the required time, through the UPnP control point. Recording can also be set to happen at regular intervals, e.g. if the same program is to be recorded each week. Media items from removable storage such as CD or DVD, can also be transferred to the hard disk in this manner.

The recorded items are now automatically added to the list of content at the media server. Making them available to be viewed in the same manner as items in Fig 4(a). Content items stored on the Media Server may be deleted by a UPnP control point on the network. Items can also be set for automatic deletion after a set period of time has passed.

In order to reduce the required storage space, the recorded streams may be transcoded to an alternative video format, such as MPEG-4 video. This will also reduce the bandwidth required when streaming. For transcoding to take place, the transcoder is selected as a 'Media Renderer' by the control point. The transcoder connects to the Media Server and begins to receive and transcode the required stream. The transcoding parameters may be set through its UPnP interface, although usually the default settings will suffice. The transcoder will now store the stream on its hard disk for later viewing.
In Fig 4(c), the devices are set up as a Personal Video Recorder. This scenario extends that shown in Fig 4(b). Here a stream available at a Media Server is selected to be played by the transcoder, this stream then becomes available through the transcoder's UPnP Content Directory Service. From here it will be accessed by a Media Renderer.

**Fig 4(c). Personal Video Recorder (PVR)**

If the Media Renderer chooses to pause playback of the stream, it will continue to be buffered by the transcoder. The Media Renderer can also seek backwards or forwards in the stream depending on the current state of this buffer. This allows for pausing and seeking in live streams.

**Fig 4(d). Handheld Rebroadcast Appliance**

In Fig 4(d) a setup is shown where the Media Server and transcoding devices act to rebroadcast a stream in a format suitable for display on a handheld device, such as a PDA, a Sony PSP or a mobile phone. This case is similar to that shown in Fig 4(c). However, now the transcoding parameters need to be set to suit the handheld device. These settings include the encoding format, quality/bitrate, temporal and spacial resolution. The settings may be saved as a profile at the control point, and so easily reused at a later date. The stream in the new format is available to the handheld from the transcoder through its HTTP and RTSP servers. The transcoder can support multiple streams, though this is limited by the available network bandwidth, and by the hardware used for the transcoding device.

**[5] Conclusions**

It has been shown here how it is possible to rapidly develop reliable networked AV devices, which use network middleware technologies to advertise their capabilities. These devices can take advantage of reliable software which is freely available, in order to speed up the prototyping process. It has also been shown that choosing the correct design tools, and an architecture which facilitates reusable software components is critical to efficient design.

A number of different usage modes for these devices have been described. It has been seen how the basic devices described can be used together as the building blocks for complex and scalable media delivery systems. The distributed nature of these systems makes them very flexible and easy to maintain.

**[6] Acknowledgment**

This research was funded by Enterprise-Ireland's Commercialization Fund, Technology Development Phase.

**[6] References**