Implementation of an Interactive IPTV System

Leonard Busuttil
Department of CCE
University of Malta
Msida, MSD2080, Malta
leobusuttil@gmail.com

Reuben A. Farrugia
Department of CCE
University of Malta
Msida, MSD2080, Malta
reuben.farrugia@um.edu.mt

ABSTRACT
The television has always been a profit-making service providing a variety of entertainment, alongside with educational channels. In time, cable and satellite TV have provided a wider range with regards to broadcasted channels. The Internet Protocol Television (IPTV) is the Next Generation Networking (NGN) application, providing new exciting opportunities for both service providers and subscribers. IPTV consists of the implementation of multimedia content, delivered over an IP network. It combines web and video streaming services, together with interactive based control. Video payload is transmitted using the Real-time Transport Protocol (RTP), that initiates and controls the video streams.

This paper presents the implementation of an IPTV system, where a server is used to store different video streams. The server listens to client requests for particular content. The server was designed to communicate with various client applications such as laptops, mobile devices and set-top boxes. The set-top box was developed using the Digital Video Processing board which is based on the DaVinci platform and was programmed to provide the same interactivity which is available on normal personal computers. The developed system was tested and subjectively evaluated and has further proven the feasibility of the IPTV system.

Keywords
Computer Networks, Internet Protocol Television, Real Time Streaming, Video Streaming

1. INTRODUCTION
Internet Protocol Television (IPTV) is the delivery of digital video content using the public Internet Protocol (IP) infrastructure [3]. IPTV is a new technology, which offers several advantages over traditional TV broadcasting, including interactivity and integration of TV and web applications. Cable and satellite TV providers operate by broadcasting television channels at known frequencies, enabling subscribers to tune television sets or set-top boxes to the desired channel. Both cable and satellite television, operate very similarly with respect to the use of a tuner to select the frequency for a particular channel.

One major advantage of the IPTV infrastructure is that only a single video stream flows in response to a request, thus minimizing the bandwidth requirement. IPTV can therefore be considered to represent a software-based pull-push technology [3]. Pull refers to a request for a particular video stream by the subscriber. The request is received by the IPTV provider, which receives the request and Pushes the video stream from the server to the requesting client. All this is done using the Internet infrastructure.

This paper presents the implementation of an IPTV Video on Demand system. The system includes a server hosting video content to be streamed and a number of clients ranging from mobile devices to High Definition – TV (HDTV). The server listens for client requests and passes the control to the Real–Time Streaming Protocol (RTSP) [6] server which distributes the video content to the intended client using the Real-time Transport Protocol (RTP) [5]. The client software on laptop and mobile devices was developed using C++ programming language while the set-top box client was developed using the Digital Video Processing board [2]. The board is Linux based and was programmed using the C programming language. The board supports both RTSP control signals and RTP video streams and incorporates an H.264/AVC [1] decoder. The decoded stream can then be outputted through the High Definition Multimedia Interface (HDMI) output port to an HD–TV.

The following sections are structured as follows. Section 2 describes the developed video on demand system. It gives the detail behind the implementation of each individual block including the Video on Demand server in sub-section 2.1 and the Video on Demand client in sub-section 2.2. The testing and results are presented in section 3 while the final comments and conclusion are delivered in section 4.

2. PROPOSED IPTV INFRASTRUCTURE
Fig. 1 illustrates an abstraction of the blocks needed to implement the IPTV system. Given that the system must work on both PC/mobile clients and the Set–top box clients, two different types of clients were implemented. The Server and the PC Client were programmed using the C++ programming language, while the Set-top box was programmed using the C programming language. The C++ was chosen for both server and laptop/mobile clients of its efficiency.

The implementation represents the concept given a uni-cast IPTV system. Each one of the clients is directly connected to the server, hence the server is directly responsible for every connection. By using such technology, the server will have a limit on the number of users that can be connected at once. Factors affecting this limit are the processor speed of the server and the bandwidth available for the server. More detail about each building block of the IPTV system is provided in the following sub-sections.
2.1 Video on Demand Server

The Video on Demand server was programmed to listen to a particular port number for client requests. This allows the user to deliver the requests to the right process. The Video on Demand server is further responsible in managing the video content and thus the following commands can be used to remotely control the content stored on the server:

- **Add stream** – requests the user to input details regarding the stream including name, description thumbnail path and also movies. This information is stored in a database so that it is available for client requests.

- **Remove stream** – is used to remove a specific video stream and all relevant information from the database. The stream configuration file is first parsed in order to check if the stream exists. If the stream forms part of the database, it is then removed along with all the data related to it.

- **List stream** – parses the configuration file and lists all the available video streams.

If the Video on Demand server receives a request for a particular stream, the server will consult the content within the list to identify the available video streams. If the video is available, then the server launches the VLC RTSP server [9] through command line. This process is hidden to the user and the server will run in the background unnoticed. A check is done again in order to check if this execution of the external RTSP server is successful, if not the user will be notified and the flow is sent back to the main menu. A thread is created in order to handle sockets, which in turn handle the client requests. If the thread creation is successful, the initiation of the server is also successful.

The Video on Demand server adopts the TCP socket to listen to a particular port number for client requests. As soon as a connection is received, it changes state to accept client, so as to negotiate a connection between the server and the client. After the connection is accepted, the handleClients() function is called in order to receive requests from the client. At the stage, the socket will wait until data is received. When the data from the client arrives, data is analyzed for verification of the command. During the implementation, the server accepts three types of commands, which are **GET**, **DESC** and **IMG**.

- **GET command** – indicates that the particular client needs a delimited list of the streams available. This is done by first traversing the database in which the streams are stored. When the names are retrieved, they are assigned to a buffer and between each stream a delimiter is placed in order for the client to be able to parse such data.

- **DESC command** – indicates that the requesting client needs the description of every movie contained within the database. Therefore, the server has to parse the configuration file and extract the description of every movie contained within the database.

- **IMG command** – the server will upload all the thumbnails of the streams contained within the database to the client.

The server application continues to handle clients and monitors the RTSP server, until a shutdown command is requested by the user. In the event of a shutdown, the process that handles the RTSP server is terminated. If the termination is successful, the socket is then closed and the state of the server is set to OFF.

2.2 Video on Demand Client

The proposed system contains two implementations of the Video on Demand Client i.e. one for mobile/laptop devices and another one for the Set–top box. The interfaces of both clients are the same, however the specific implementation is significantly different.

In both cases, the Video on Demand client will send a **GET** request in order to get the list of video streams available at the server. The server will deliver a delimited list containing the requested information using the TCP protocol, as illustrated in Fig. 2. Once all the list is available, the client requests for additional information through the **DESC** command followed by an **IMG** command to retrieve the thumbnails. This information is then displayed on the screen as shown in Fig. 3. The user is then able to choose from the list provided and the respective stream is displayed.
After the VoD connection is set up, an RTSP client is then launched to be able to communicate with the RTSP server located in the Video on Demand server.

Figure 3: Client display using the Set-top box

2.2.1 Digital Video Processing Board

The DVPB-HD board is a product developed by eInfochips [2], in order to provide rapid development platform based on Texas Instruments’ TMS320DM6467 DaVinci Technology. The board is designed in order to target applications such as Video Encode, Video Decode, Video Surveillance, IP Video Phone, IP Net Cam and also Advanced IP Set-top boxes. It includes several interfaces including Video/Audio Input/Output, 1000 Mb/s Ethernet, RS232, RS485, USB 2.0, UART/IrDA/CIR and more. Fig. 4 depicts the commonly used devices that the development board supports and most of all the rich set of inputs and outputs it provides.

Figure 4: DVPB-HD board (from [2])

This processor is made up of two cores, the high performance TI’s TMS320C64x+ Digital Signal Processor (DSP) and the Reduced Instruction Set Computer (RISC) processor ARM926EJ-S. The principal of having two cores in one processor is suitable for applications were the Operating System (OS) is executed over the RISC processor, whereas heavy mathematical calculations are executed by the DSP. This development board is also designed in order to be able to output High-Definition video. It supports Y–Pb–Pr (Component) and HDMI output. These two technologies are used as a source of HD output. The robust processor is able to decode and even encode such HD video content as well as sound. The integration of the Ethernet port makes this board the right candidate for our project, and will be used as the Set-top box in our implementation.

The DVPB-HD board is managed by the MontaVista Embedded Linux OS [4]. The MontaVista Linux is a cut-down version of the Linux open-source OS, which is capable of running on the above-mentioned processor, integrating the drivers needed for the smooth operation of the OS. With the aid of the embedded OS, developing applications for this development board becomes easier, since one can use the same methodology as if programming is being done for a Linux OS.

Because of the Linux environment, programming for this kind of board has to be done on a Linux platform. Compilation was possible by the use of make files, which in them integrate the Software Development Kit (SDK) offered by Texas Instruments [8]. On top of this, another SDK developed by eInfochips provides development in order to be able to use all the drivers to drive the output peripherals as mentioned before. The SDK provided by eInfochips mainly provides a set of Application Programming Interfaces (APIs) in order to make it easier for the developer to develop applications for DVPB-HD [2].

3. RESULTS

Bandwidth usage is considered as one of the most important factors that affect IPTV systems. Traffic analysis was done so as to provide a satisfactory level of quality and to design a system that is scalable for many users. This analysis also helps IPTV system providers in order to predict which technology the infrastructure will be designed with, to be able to provide adequate bandwidth to each customer. One has to keep in mind that IPTV systems can be deployed alongside an Internet connection and also a VoIP service, thus one cannot afford to use up the entire bandwidth by the IPTV system.

In this implementation a unicast system was implemented, where each user is directly connected and served by the server. This means that with every client connected to the system, the bandwidth to serve such client increases accordingly. Fig. 5 shows the throughput graph for a Standard-Definition stream of length 360s shown in red, along with a throughput graph for a High-Definition stream shown in blue. As can be seen from the graph, the average throughput imposed on the system with such a stream is of 3.5Mbps. On the other hand a High-Definition stream had an average throughput of 5.2Mbps. This clearly indicates that the High-Definition stream is more bandwidth hungry than the Standard-Definition one.

The server was then loaded with more clients in order to analyze how this addition in streams affects the bandwidth. Fig. 6 shows the bandwidth increase when the server was loaded with seven streams. The steps-like increase in the beginning, depicts each user connecting to the server and the usage increase imposed. The step size increase in bandwidth shows that with each user streaming with a Standard
Definition, bandwidth increases by 3.5Mbps. Thus verifying the expected bandwidth increase, since the implementation works on a unicast system. The bandwidth has a direct proportional relationship with the streams, thus with every newly connected stream, bandwidth utilization increases by a further 3.5Mbps. These figures give a clear indication for the planning of such implementation.

In order to ensure that the proposed system provides a satisfactory Quality of Experience, a subjective test of the system was considered. The subjective test consisted of 30 non-experts and was initiated with only one stream on the server, and the candidates were asked to share their opinion about the quality produced from the stream. The candidates were not aware that during the test, load was being increased on the server, so as not to bias the test subjects. After their first impressions were recorded, the server was loaded gradually until the majority of the bandwidth was utilized. The test subjects were asked to point out and report about any degradation visible, compared to the previous tests.

Results were recorded, and analyzed as shown in Table 1. The majority of the test subjects state that the quality was excellent and that they did not experience any quality degradation. The test subjects were also given the opportunity to use the Video on Demand system to test the interactivity of the system. A very positive answer was recorded from all the participants about the interaction. It was stated that the response time was satisfactory with regards to interaction. It was also pointed out that the fact that a user can access any movie at any particular time is something that everyone will be looking forward to. The ability to pause a stream during playback was positively acknowledged as well.

### Table 1: Subjective Test Results

<table>
<thead>
<tr>
<th>Test</th>
<th>Excellent</th>
<th>Good</th>
<th>Fair</th>
<th>Poor</th>
<th>Very Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Load</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Load</td>
<td>99%</td>
<td>1%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Interact.</td>
<td>98%</td>
<td>2%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

4. **COMMENTS AND CONCLUSION**

This paper has presented the implementation of an IPTV system capable of streaming multimedia content over heterogeneous networks. The developed system was made up of a Video on Demand server and two different Video on Demand clients. The first type of client represents mobile devices and laptops, for which software was developed using the C++ programming language. The second type of client represents the Set–top box which was developed using a Digital Video Processing board. The system was tested and the subjective tests provided in the results section confirm that the users did indeed experience a good quality of experience. In particular, it was noted that the users did like the interactivity functions provided by the system.

The current IPTV system adopts the unicasting paradigm, which transmits a different stream to different users. However, the required bandwidth can be substantially reduced by adopting multicasting instead, where a stream is delivered to a group of users. Another improvement is to adopt H.264/SVC [7], which improves the scalability of the IPTV system enabling the transmission of one unique stream to heterogeneous devices.

5. **REFERENCES**