UACEE International Journal of Computer Science and its Applications - Volume 2: Issue 3 [ISSN 2250 - 3765]

Flash on a Set Top Box to Enhance Interactivity

Vinod Kumar N
Department of Computer Science and Engineering
B.M.S. College of Engineering
Bangalore-560 019, India
vinod_k1001@yahoo.com

Lakshmikantha H.A
Department of Computer Science and Engineering
B.M.S. College of Engineering
Bangalore-560 019, India
kanth.cbp@gmail.com

G Varaprasad
Department of Computer Science and Engineering
B.M.S. College of Engineering
Bangalore-560 019, India.
varaprasad555555@yahoo.co.in

***Abstract*—Before pay systems and digital television, all that consumers needed to watch television was a standard television set that they could buy anywhere. In the 1980s this simple model began to change. The advent of cable and satellite television required consumers to connect their TVs to an additional device like a set top box to receive the signals. This paper discusses about the usage of flash in the application layer of the set top box architecture. Adobe flash is bandwidth friendly vector animation originally designed to create animation for web. It is extended and incorporated into a set top box to enhance the interactivity by improving the look and feel of the user interface. The flash user interface is developed with object oriented script language called action script.**

Keywords—Settop box, adobe flash, user interface, called action script.

I. Introduction

The transition to digital television has created a new dimension in the way people use television. Personalized and interactive services attract viewers into the television experience. Interactive TV is a broad term that includes any service which enhances the TV viewing experience [1]. Interactive TV ranges from an attractive user interface to adding a new functionality to the TV. Great content is no longer enough to keep platforms and network operators competitive. Today, STB's are marketed to a new generation of demanding, technology conscious consumers. These consumers are also sophisticated TV viewers, expecting more personalized services, detailed program guides and the content that interests them available on-demand. These consumers want more than a passive TV viewing experience. They want to be involved. They want to play along with game shows; experience sports as if they are in the stadium, make impulsive purchases of content or merchandise etc. When the above services are provided through a better user interface, the interactivity between the viewer and the TV will be increased further. Flash being a technology that dominated the web is incorporated into the set top box to build a better application layer.

The paper is organized as follows. Section 2 describes about the Adobe flash lite player which is a lighter version of the flash player. The architecture of the flash set top box is discussed in Section 3. Section 4 gives an overview of the flash engine that supports the flash applications. Section 5 explains some of the performance considerations for flash applications. Section 6 gives the results. Conclusions and future enhancements are given in section 7.

II. FLASH LITE PLAYER

Adobe flash lite player is a lighter version of the Adobe flash player. The Flash lite player is specifically intended for mobile phones and other electronic devices [2]. This player is further extended and incorporated into a STB. Flash lite is a development technology implemented at the client-side or user interface layer. It will operate on devices with a lower hardware specification. It uses vector graphics to minimize file size and create files that save bandwidth and loading time. Flash is a common format for games, animations, and GUIs. Figure 1 shows the support that is extended by the Flash lite player. Flash lite cannot be considered as an embedded operating system, it is a technology for developing applications that run on an embedded operating system.

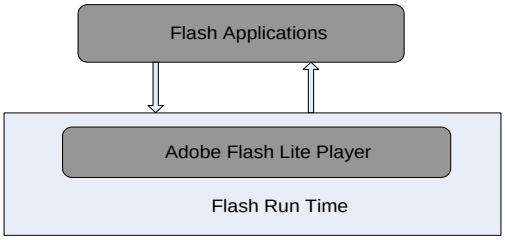

Figure 1. Adobe flash lite player.

III. SET TOP BOX FLASH ARCHITECTURE

The proposed architecture of the STB is shown in figure 2. This architecture follows layered architecture [3] and discusses about the porting of flash applications on the set top box. The flash applications may be a flash electronic program guide or flash games. The flash engine will host the flash lite player and it will provide a runtime environment for the flash applications. The flash engine will adhere to the guidelines of the platform service layer i.e only one engine will be active at any instant of time. The API's from the flash application layer will be implemented in the adaptation layer. The Flash application extracts services from the below layers through Flash Application Program Interface (FAPI). It is an API specification in scripting language called action script.

The adaptation layer is introduced in order to make the application developed using FAPIs to execute on top of

middleware platform. FAPIs are only the API specifications and hence it needs to be implemented in the adaptation layer. The adaptation layer is responsible for implementing all FAPIs classes in native language, creating and maintaining equivalent native object for every FAPI object created in application layer and provide a unique handle for every object. For every FAPI implementation there should be a reference to identify the FAPI object context. In order to achieve this, whenever a FUAPI class object is created in the application layer, corresponding native object (instance of a structure) also created in the adaptation layer. The adaptation layer provides a unique handle to every created object to identify FAPI object context. This handle shall be associated with FUAPI class object using the binding layer. The middleware abstracts the hardware and the OS of the STB to the above layers. Middleware is responsible for collecting the raw data that is received from the service provider, process it and make that content available to the application. The services provided by the middleware include memory management, access to the MPEG decoder/demultiplexer, TCP/IP return channel, graphics support, data tables, and conditional access and TV services.

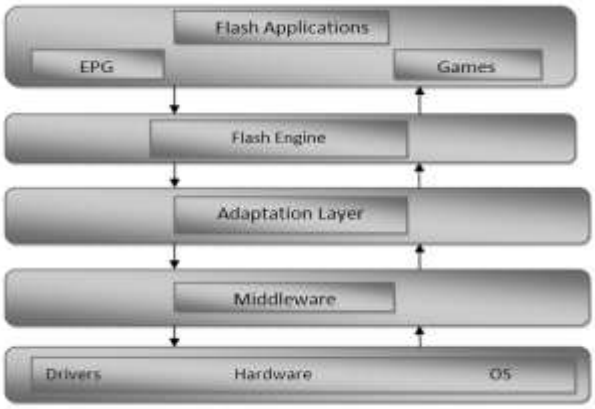

Figure 2. Set top box flash architecture.

IV. FLASH ENGINE

The component design of the flash engine is shown in figure 3. Only one engine of the STB will be active at any instant of time. The flash engine is similar to the EPG engine and the browser engine with respect to the engine initialization, start-up, shutdown, execution and event handling. System related requests like memory, timers, event management etc are made directly to the core of the middleware. Graphics related queries like surfaces, fonts, rendering etc are made to the graphics component of the middleware. Service information, tuning and other parameters are exposed via action script interfaces and are routed through the middleware.

A. Adobe Flash Lite Engine

The Adobe flash lite engine exposes two types of interfaces.

SI: - To be implemented by the host that will be triggered by the flash engine.

FI: - It is the player interface to the host. The host invokes this interface from port layer as well as from the flash engine client module.

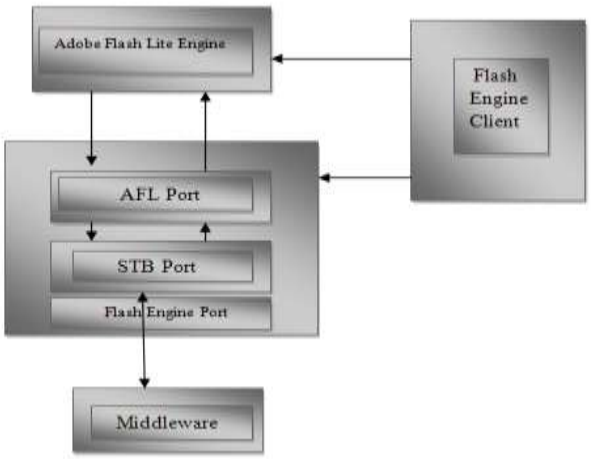

Figure 3. Flash engine components.

B. Flash Engine Port

This port module is multilayered and consists of AFL porting layer and STB porting layer. The AFL porting layer implements the SI interfaces required by the flash engine. This layer maintains all the information specific to Adobe flash lite and calls the STB port layer to get the actual functionality done. The STB port layer is a reusable port layer for the core and the graphics of the middleware. It can be used by any third party engines. It abstracts the STB specific implementation and communicates with the core and graphics. The major components of the flash engine port are.

Graphics: - This module is responsible for implementation of graphics (fonts, frame buffer, hardware rendering). It interacts with the surface and font memory module, which makes call to the graphics component.

URL: - This module implements the URL APIs to load files from HTTP, file etc. It interacts with the core and the load status s notified via FI interface.

Memory: - The memory module is responsible for the implementation of the memory related APIs of AFL. Runtime memory allocation during the initialization of the flash engine will take place via this module. It has specific APIs for memory allocation from the system heap as well as from the contiguous memory device for image decoding and bitmap caching.

Sound: - This module is a pass through module to middleware for playing MP2. The porting layers will expand for MP3 implementation.

Timer: - This provides current system time and UTC time in millisecond.

Control: - This module will log the traces in action script and is responsible for error management.

Image: - This module is aimed to support asynchronous image decoding for the images formats such as PNG, BMP, etc that are not supported by flash engine on runtime.

C. *Flash Engine Client*

This module is primarily responsible for initialization and start up of the flash engine, termination of flash engine, event handling and invokes the respective functions in the AFL port layer.

D. *Communication Flow*

SI: - This flow is from the Adobe flash lite engine to the porting layer, which in turn makes a call to the middleware to provide the relevant functionality.

FI: - This flow is from the porting layer back to AFL to notify events from the middleware components related to asynchronous request raised by the flash engine calls. These calls are the replies to the SI calls.

V. PERFORMANCE CONSIDERATIONS FOR FLASH APPLICATIONS

Flash applications on PC are faster and the same application crawls on the STB. This is mainly due to the processor power. Hence, we cannot take an application written for PC and try to run it on the STB as it will expect same performance. This is due to the fact that flash applications written for PC do not consider the memory and performance. The following guidelines can be taken into consideration during the flash application development.

Stage Size: - The stage size (width and height) of the flash document should be set to match the device resolution. This will avoid scaling, which impacts performance.

Hidden Movie clip:- Use bitmap caching option on the movie clip instance of a vector graphic which is being used in animations and in recurring drawings. This has an impact on memory and hence care should be taken while cashing vector. Also frequently changing vectors (inside an animation) should not be cached as it will impact performance due to cache creation and destruction inside the animation window.

Memory versus Performance: Caching of movie clips and objects improves performance but increases memory and hence this trade off is at memory availability and hence left to the user discretion.

Animations in Timeline: Maximize the animation definition in timelines rather than doing everything in code as it betters and improves performance.

Objects in memory: Load only the objects and movie clips, which are required for that instance. After usage, unload unwanted movie clips and objects so that memory will be freed.

Movie clip Behavior: It is easier to manage and update actions in the movie clips itself rather than in an external AS file.

XML: Avoid loading and parsing XML files as it consumes processor and affects performance. If external data is required, it is better to use name-value pairs.

Explicit Function Definition: Functions must be defined explicitly as they are more efficient that anonymous functions.

Math and Floating Point: Minimize the use of math functions and floating point values. Pre-calculate the math functions and store the values in array. The array access will be faster that runtime calculations.

First frame initializations: Even though pre-loading the content in the beginning helps, it directly impacts memory and start-up time. Space the content throughout the movie so that movieclip and other objects are created and initialized when they are used.

VI. RESULTS

Flash can be successfully ported on to the STB. The interactivity can be further improved by enhancing the look and feel of the user interface. Also interesting flash games can be integrated into the application layer of the STB to attract more viewers. The usage of flash increases the response time of the STB by 20%. The figure 4 shows the EPG on the TV screen.

Figure 4. Flash Electronic Program Guide.

VII. CONCLUSION AND FUTURE ENHANCEMENT

Flash is a proprietary technology of Adobe. As Flash is used in the User Interface(EPG) there is a need for packages that support flash. Flash is not open source, it has to be purchased from Adobe. Hence, this adds to the cost of the project. After flash is used in the application layer of the STB the interactivity can be taken to the next level. Many companies can take advantage of porting flash on to the STB to increase the number of viewers. Further, as flash is widely used in web and supports streaming, the STB can be connected to the internet and view online videos and

download flash game. This enhancement is possible with the availability of required infrastructure in the STB.

References

[1] Dobbie, W, "Interactive electronic programme guides", IEE Half-day Colloquium in Navigation in Entertainment Services 1998,pp. 1-5.

[2] http://en.wikipedia.org/wiki/Adobe_Flash_Lite , June 2011.

[3] Rudolf Jaeger, "Set-Top Box Software Architectures for Digital Video Broadcast and Interactive Services". IEEE International Conference in Performance, Computing and Communications 2001, pp. 287-292.

[4] Minhong Yun, Jaeho Lee, Woosik Kim, Sunja Kim, "UI Player Framework for Mobile Devices" The 9th International Conference on Advanced Communication Technology 2007, pp.151-154.