Implementation of Smooth Streaming protocol through a generalized software framework

Miroslav Suša , Ilija Bašičević, *Senior Member, IEEE*

Abstract—Adaptive streaming is a technology for transmitting multimedia content over a network such as the Internet. This way the content is available at any time which has brought big changes. One of the many streaming technologies is Smooth Streaming. In addition to the transmission of content via one of the protocols, it is necessary to ensure its reproduction. In this paper, the implementation of the Smooth Streaming protocol within a single media player is presented. The implementation was performed through a generalized software framework, which will also be discussed. The role of the framework is to facilitate the integration of the remaining adaptive streaming protocols into the media player.

***Index Term*—adaptive streaming; content playback; Smooth Streaming;**

I. INTRODUCTION

The fourth industrial revolution brought many changes in terms of consuming multimedia content. When it comes to content transfer, the most important innovation is Streaming technology. Streaming makes it easier for users to access content whenever and at any time they want, which previous technologies could not provide. In the field of television, new technology is completely taking over the market from the traditional way of content broadcasting[1].

As a consequence of the development of a new way of data transfer, standards have emerged according to which this transfer will be performed. Some of the best known and most prevalent today are the MPEG-DASH and Smooth Streaming standards. In addition to the transmission of the content itself, it is necessary to ensure its reproduction.

This paper presents an implementation of the Smooth Streaming protocol, the creation of a generalized software framework for managing adaptive streaming protocols, as well as the integration of the software framework with the Smooth Streaming protocol and a media player for content playback.

The rest of the paper is organized in the following order: Section II discusses streaming technology and its types. Section III shows the architecture of the media player on which the work solution is implemented. Section IV defines a programmatic framework for managing adaptive streaming protocols. Section V discusses creating a library for the Smooth Streaming protocol. Section VI shows the architecture of the media player after applying the solution. Section VII deals with the validation of the solution and the results. Section VIII provides a conclusion on the work.

Miroslav Suša – RT-RK Institute for Computer Based Systems, Novi Sad, Serbia (e-mail: miroslav.susa@ rt-rk.com).

Ilija Bašičević – Faculty of Technical Sciences, University of Novi Sad, Trg Dositeja Obradovića 6, 21000 Novi Sad, Serbia (e-mail: ilibas@uns.ac.rs).

II. STREAMING

Streaming is a technique of continuous transmission of video and audio material via wired or wireless internet connection. Before the advent of streaming, playback of content from the Internet was possible in two ways. The first way is to upload the complete file to the device and only then is playback possible. Another way is to use a progressive download.

A. Progressive streaming

Progressive download allows you to play content while downloading it to your device [2]. Downloading is done regularly, which means that not only the selected part of the file can be downloaded, but the complete one. Any part of the downloaded content can be played as desired. The content transmitted in this way is of fixed quality and resolution. In other words, only one video file can be uploaded. Since different content resolutions require better or worse internet traffic, in a situation where the flow is poor, progressive downloads will often lead to transmission interruptions. In addition, the content will be displayed differently on different devices.

Higher resolution files take up more memory space, and the file transfer speed depends on the internet flow which tells us how much data the user can receive in a unit of time. If the flow is poor and the video has a higher resolution, part of it will not be able to be transmitted in its entirety and playback will be delayed. In addition to downtime, progressive downloads also cause the problem of presenting videos on devices with different screen resolutions. For example, when playing a video that is 720p resolution, on a screen with 1080p resolution, the image will be stretched and pixelated.

B. Adaptive streaming

Adaptive streaming is a streaming technology based on the HTTP (HyperText Transfer Protocol) protocol. The benefit of using HTTP technology is the unhindered passage through the firewall and NAT (Network Address Translation) devices that remap IP (Internet Protocol) addresses. In addition, the complete implementation of HTTP logic is on the side of the content seeker, which reduces the need for a continuous connection between the provider and the service provider.

Adaptive streaming, instead of a complete file, transmits and plays its parts for a few seconds [3]. We call such parts of a file its segments. Since the content is divided into segments, any part of it can be added as desired. Just before the segment expires, the next one to be played is delivered. After moving on to the next segment, the previous one is deleted, and the

process takes place until the complete content expires. This gives the impression of continuous playback of content. Information about all video and audio files and their segments, as well as details of the need for their transfer and playback can be found in the manifest file.

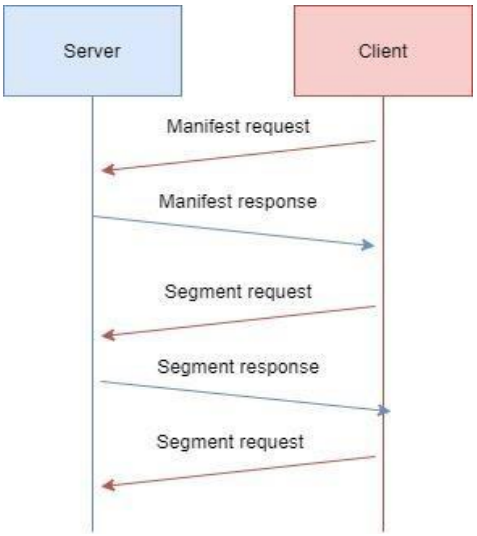

Fig. 1. File exchange between client and server during adaptive streaming

Adaptive streaming solves the problems of progressive download [4]. Files of different resolutions are created for the same content. Depending on the size of the client's screen, a file segment of the appropriate resolution is provided.

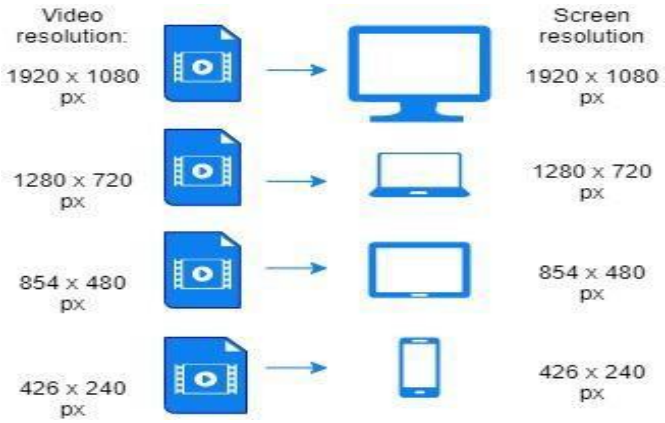

Fig. 2. Display video resolution selection for different screen resolutions.

In the typical communication scenario between a client and a server using adaptive streaming the client first sends a request for a manifest and receives it in response from the server. The client then sends requests for fragments which the server delivers.

III. MEDIA PLAYER ARHITECTURE BEFORE PROTOCOL IMPLEMENTATION

IWedia Player (IWP) is a library written in the C ++ programming language that aims to provide a high-level player interface. The player allows you to play audio and video content. It is used as a part of an application written for the Android platform and provides it with a user interface.

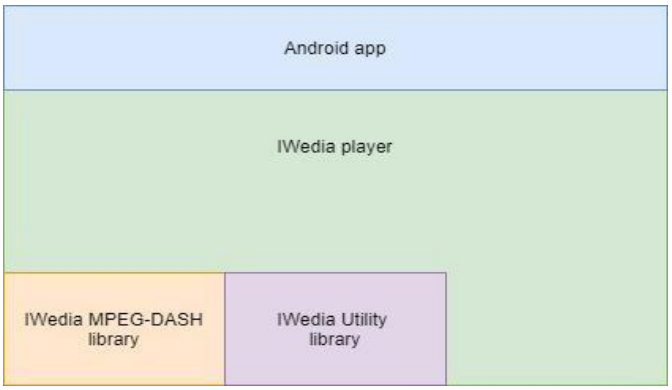

Fig. 3. Display of player architecture before implementing the solution.

In order for the player to enable the MPEG-DASH streaming protocol, an IWP-DASH library was created. In addition to defining the elements of this streaming protocol within the dash library, the logic for adaptive streaming has been implemented, which is closely related to the dash protocol. Components shared between other IWedia libraries such as the player and dash are housed in the IWedia utils library.

IV. SOFTWARE FRAMEWORK FOR MANAGING ADAPTIVE STREAMING PROTOCOLS

When it became necessary for the media player to support, in addition to the MPEG-DASH protocol other adaptive streaming protocols as well, the implementation logic had to be generalized and displaced from the dash library.

The goal of the adaptive streaming protocol management framework is to provide the media player with an interface through which to obtain content for playback. All protocols aim to transfer content and regardless of their complexity, we can see numerous similarities between them. Since MPEG-DASH is an official international standard, it is the most developed and provides the most opportunities during implementation [5]. All other standards can be viewed as its subset.

The software framework can be divided into four logical units that have a role in downloading manifests, segments, adapting to network conditions and creating an entrance to the library.

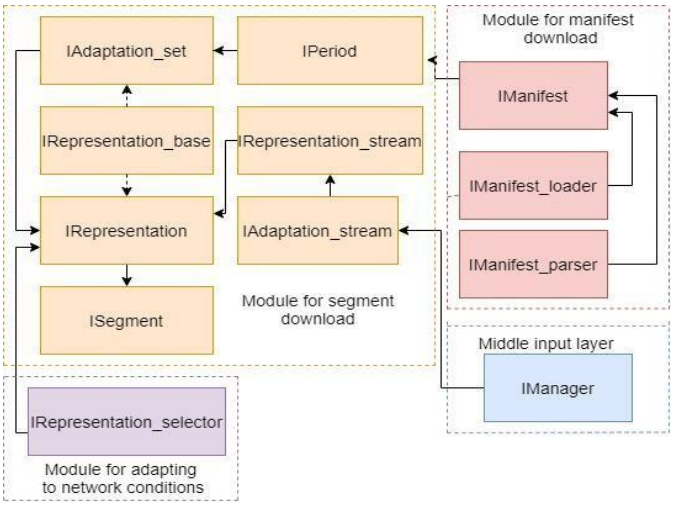

Fig. 4. Generalized software framework solution architecture.

A. Module for manifest download

Manifest, as mentioned, is the central document for gathering information on the content to be transmitted. It is available on the server along with the provided content. The type of manifest can be dynamic, if we broadcast live content, or static, if the content is downloaded on demand. To stream live content, the manifest needs to be delivered periodically because the content is constantly changing.

To provide a manifest, URI (Uniform Resource Identifier) from which it can be downloaded is required. The download is performed with the help of a previously implemented download class, which needs to be provided with data about the speed, number of attempts and download time of the manifest. As a result of successful delivery, a string with the contents of the manifest is obtained. After it is downloaded, the manifest is parsed.

At the level of libraries that implement the standard, it is necessary to implement interfaces that represent the manifest and the factory for creating the manifest.

B. Module for segment download

Within period elements of the manifest, that contain the initial time and duration of the content, there are adaptation elements. Their primary purpose is to provide information about the type of stream being transmitted. Inside the adaptation element are elements that represent the stream. They contain data on the flow rate required to download the stream in a certain quality as well as information on the segments that need to be downloaded. At the stream type level, a structure is created that will download the segments.

Libraries of specific protocols that implement the created interfaces define the form of stream representation as well as the form of segments. The representation form creates segments based on a given start time, carries information about the number of available segments, as well as the broadcast time period provided by the representation.

Segment download control is defined in this software framework. The time period in which the download will be performed is determined, the representation within which the segments will be downloaded is selected, the ordinal number of the next segment to be created is calculated and the creation of the segment is initiated.

C. Module for adapting to network conditions

The network adaptation module is key to performing adaptive streaming. Depending on the speed of the user's Internet flow, it is necessary to correct the representation of the stream being downloaded. The factor that influences the choice of representation, in addition to the flow rate, is the type of content that is downloaded. For the purpose of selecting a representation, a selector is created that stores all available representations and, based on the current flow and type of content, selects one of them to be played. After the download, the number of bits downloaded, as well as the time period required for the download, are forwarded to the flow rate meter. Based on the obtained parameters, the meter calculates the current flow rate that is available when initializing the next download.

Middle input layer

As part of the software framework, a manager has been created to manage the processes. Its presence is necessary in the media player that uses the library. The task of the manager is to initiate the loading of the manifest when it comes to streaming on demand or perform periodic loading of the manifest if a live broadcast is performed, as well as the interruption of these operations. In addition, the manager creates a program representation of the adaptation stream that has the ability to further manipulate stream representations and segments.

D. Software framework integration with the media player

In order to enable the reproduction of content by adaptive streaming protocols, it is necessary to integrate the software framework with media player. The integration is done by adding a component that has access to the software framework manager. In this way, the processes realized by the software framework are initiated, such as taking over the manifest and adding segments. The ultimate goal of process initiation is to obtain segments and prepare them for reproduction.

Within the media player, there is also logic for determining the adaptive streaming protocol that will be used, as well as factories that will, depending on the selected protocol, create a component that has access to the software framework.

V. SMOOTH STREAMING LIBRARY

The Smooth Streaming library is a C ++ implementation of the Microsoft Smooth Streaming protocol used by the IWedia player to play content. The library consists of "*ismc*" and *"abr"* modules. The *Ismc* part of the library was named after the extension of the Smooth Streaming protocol manifest client. Within this part, the manifest is parsed and elements and attributes representing the data collected by parsing are realized. *Abr* part of the library represents the implementation of a software framework for managing adaptive streaming protocols.

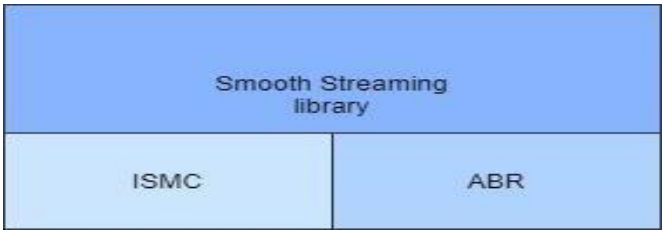

Fig. 5. Smooth Streaming Library Components.

A. Library creation

In order to implement the protocol, it is necessary to parse the protocol manifest and present its elements within the library. In addition, it is necessary to provide the types of content exchange messages defined by this transport protocol, which are: manifest request, manifest response, segment request and segment response.

1) Manifest request

A manifest request is sent to obtain a manifest containing all the necessary information to reproduce the content. In order to send this request, a URI to the manifest is required as well as information on which extension of the manifest file is

expected. Manifest extensions differ in whether it is a server manifest that has an *ism* extension or a client manifest whose extension is *ismc*.

2) Manifest response

The manifest response is obtained in the form of an *ismc* file with metadata related to the playback of the content. The file is a well-formed XML (Extensible Markup Language) and consists of the following elements: SmoothStreamingMedia, Protection, StreamIndex, QualityLevel and StreamFragment. All of the above elements are presented within the library as classes, and their correlations are clearly visible and described below.

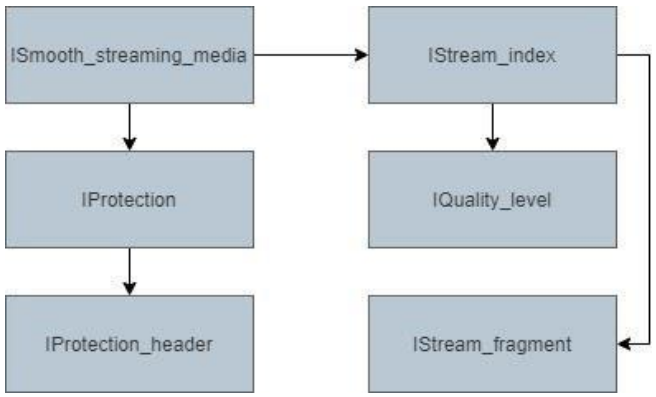

Fig. 6. Smooth Streaming library solution architecture.

a) SmoothStreamingMedia

SmoothStreamingMedia is a root element that contains all the other elements of the manifest. The direct descendants of this element are the StreamIndex and Protection elements. Its attributes carry information about the main and secondary versions of the manifest as well as whether the manifest describes live or on-demand content. Within the attribute, the duration of the content described in the manifest is also defined.

SmoothStreamingMedia is implemented within the library so that its creation requires URI of manifest as well as xml files in string format. The string is then parsed using the sub-element names and attributes listed as constants.

a) Protection

Protection is an xml element that includes the metadata needed to play protected content. It contains information on the unique identification of the security system used on the given content, as well as the encoded data that the system uses to enable the reproduction of the content to authorized users.

b) StreamIndex

StreamIndex is the most important element within a manifest because it contains metadata for playing a specific stream. This means that the element provides information about the type of content that is transmitted by a particular stream, that is, whether it is an audio, video or text stream. Based on the stream type, the availability of attributes within an element changes. Only in the case of video, there is information about the maximum available content resolution that is available, as well as the recommended playback resolution. The number of qualities, segments, as well as the duration of the stream are available within this element.

b) QualityLevel

The QualityLevel element carries metadata about the playback of a specific track within the stream. Depending on the type of stream in which it is located, its attributes differ. For video within the video stream, the required resolution attributes as well as parameters specific to a particular media format are required. When it comes to audio recording within the audio stream, in addition to the previously mentioned attributes, we also have data on the number of channels of the audio tape, sampling rate, sample size, limits for optimizing audio decoding and identification of media format used. The attributes that each record contains are those that carry information about the unique identification of each record and the download speed required to retrieve a particular record.

c) StreamFragment

The StreamFragment element contains metadata about a set of related segments in the stream. Its attributes carry information about the start time of the segment, its duration, the order in a series of segments as well as the possibility of repetition. For a segment to be valid, it must contain either a duration attribute or a start time attribute. A series of segments is called adjacent if the start time of any segment, with the exception of the first, is equal to the sum of the start time and the duration of its predecessor.

3) Segment request

A segment request is created to retrieve the desired segment from the server. To create it, it takes URI to the desired segment, its bitrate, the name of the stream within which the fragment is located, the start time of the desired stream, as well as the type of response that the client expects from the server.

4) Segment response

A segment response is a response that is received after sending a request to obtain a segment. The answer can be complete or partial. If the answer is complete it contains media and segment metadata, while partial responses contain only media or metadata.

B. Framework implementation

In the Smooth streaming library it is necessary to implement two of the four modules of the framework and they are: Module for manifest download and Module for segment download.

2) Implementation of the module for manifest download

As mentioned earlier, it is necessary to implement the manifest factory interface as well as the manifest interface.

a) IManifest interface

IManifest methods gather the necessary information that each manifest should have, namely: whether the manifest is live or on-demand, the duration of the manifest, the minimum time required to load the manifest, as well as adding the manifesto period. All data can only be obtained by parsing the manifest.

b) IManifest_factory

The manifest creation factory contains only one method that instantiates a class that implements the IManifest interface. It forwards manifest uri and the contents of the manifest that is necessary to parse.

3) Implementation of the module for segment download

Module for segment download defines the necessary logic to supply the parsed element data, as well as the logic for creating segments.

a) IRepresentation

Interface methods obtain, from the QualityLevel element, data described in the part of the paper with the same name. All data is present in the node and is very easy to obtain.

b) IAdaptation_set

The IAdaptation_set interface is composed from set of methods that retrieve data from the Stream_index element of the library. All methods return the present attributes or sub-elements

of a given element.

c) ISegment

This interface is defined by a set of methods for retrieving the Stream_fragment element attribute with the exception of the get_uri method. The get uri method calculates the uri to a given segment that is different from the manifest uri

d) IRepresentation.

The role of the IRepresentation interface is to create segments, add the total number of segments, add the duration of all segments and find the segment with a given index

The number of segments is obtained when initializing the class of this interface by going through all segments and taking into account their repeat attribute which tells how many times a given segment is repeated.

During the process of calculating segments, the total duration of all segments can be easily obtained. The timestamp of the first and last segment is taken, or their length and repeat tag if the timestamp is not available.

A segment with a given index is supplied by going through all available segments, taking their duration and repeat attribute, calculating the index of each segment and returning the resulting one. The limitation of this method is to pass an index that is not less than zero and that is not greater than the total number of segments.

VI. MEDIA PLAYER ARHITECTURE AFTER PROTOCOL IMPLEMENTATION

By removing the definition of adaptive streaming protocol from the dash library and generalizing it, a software framework for managing adaptive streaming protocols is obtained. This makes it easier to use and add new adaptive streaming protocols such as the Smooth Streaming protocol. In addition, their integration with the player is facilitated.

Fig. 7. Display of player architecture after solution implementation.

VII. TESTING

A. Description of the test environment

The environment for testing of this solution comprises Smooth Streaming content that can be accessed via the network, an application for playing content, as well as an Android P development board on which the application will be launched.

The content playback application is written in the Java programming language for the Android platform. It provides a simple user interface from which content playback can be controlled, and uses the IWedia player library interface for playback itself.

Content preparation, implementation of the Smooth Streaming library and software framework, as well as their integration with the media player are described in the previous chapters. The Android P development board connects to the same network from which the prepared content is available to it, and the Android application is installed and launched on it. With this step, the test environment is ready and testing can begin.

B. Testing procedure

After installing the Android application on the board and launching it, you get access to the list of all available streams. Clicking on the desired stream starts playback. Playback can be interrupted, paused or restarted at any time.

C. Test results

As stated, by clicking on the desired stream, in this case on the stream belonging to the group of Smooth Streaming streams, playback starts. The start of content playback always takes place at a lower resolution until the user's internet flow is determined. After that, if it is determined that the conditions are met, playback continues at higher resolutions, which tells us that the SmoothStreaming protocol has been successfully implemented.

Selecting one of the available streams that are transmitted by other adaptive streaming protocols results in content playback. Successful playback start shows that the generalized software framework has been correctly implemented.

VIII. CONCLUSION

Within this paper, a solution for integration of Smooth Streaming standards for broadcasting content is described. Also, a generalized program framework has been implemented, which enables easier integration of the remaining standards.

The protocol library and software framework are written in C++. In this way, speed and flexibility are achieved. Like the media player in which they are implemented, they can be used on both Android and Linux platforms.

The integration framework in the media player enables easier integration of existing, as well as the protocols that may arise in the future which can be seen as a subset of the MPEG-DASH protocol.

ACKNOWLEDGMENT

On this occasion, I would like to thank my colleague Nikola Špirić on the provided support.

REFERENCES

[1] Nenad Lovcevic, Jelena Simic, Miroslav Dimitraskovic and Ilija Basicevic “Modul za prijem i obradu JSON komandi u programskoj podršci digitalnog TV prijemnika” 61st IcEtran conference in Kladovo, Serbia, June 5 – 8, 2017.

[2] Stefan Lederer, Christopher Mueller,Christian Timmerer, Hermann Hellwagner “Adaptive Multimedia Streaming in Information-Centric Networks” IEEE Network, November 2014.

[3] Ilija Bašičević, Nenad Lovčević, Nenad Šoškić, Milan Ačanski “Internet as Infrastructure for Digital Television” 62st IcEtran conference in Palić, Serbia, June 11 – 14, 2018.

[4] Rubem Pereira, Ella Pereira “Dynamic Adaptive Streaming over HTTP and Progressive Download: Comparative Considerations”, IEEE 28th International Conference on Advanced Information Networking and Applications Workshops (WAINA) - Victoria, Canada, May 1 2014

[5] Sunho Seo; Younghwan Shin; Jusik Yun; Wonsik Yang; Jong-Moon Chung “Adaptive high-resolution image transmission method using MPEG-DASH” International Conference on Information and Communication Technology Convergence (ICTC) Jeju, Korea (South), October 18-20, 2017.