

Implementation of Continuous HTTP Live Streaming using Playback Position Request Mechanism in Heterogeneous Networks

MinSeok JANG*, Hyeontak OH*, Jinhong YANG*, Jun Kyun CHOI *, Keuneun KIM **, Ilkwon CHO **

* Department of Electrical Engineering, KAIST (Korea Advanced Institute of Science and Technology)

** NIA (National Information Society Agency)

chaiojang@kaist.ac.kr, hyeontaek @kaist.ac.kr, sunupnet@kaist.ac.kr, jkchoi59@kaist.edu,
kekim@nia.or.kr, ikcho@nia.or.kr

Abstract— Wide spread of smart devices makes high demand on multimedia service using these devices. Especially, demand on continuous service between 3G/4G network and Wi-Fi network are increasingly high. Also, multimedia services are combined with web technology to make HLS(HTTP Live Streaming)and HDS(HTTP Dynamic Streaming) in web convergence era as most of services are converged to web. In this paper, we implement continuous seamless multimedia service using HLS. In HLS, a video is truncated to fragments of the constant length; these are contained to MPEG-TS and transmitted over HTTP. Periodically requests of these fragments and playing them continuously enable streaming service. If fragments are served with multiple rates, these enable link adaptation to select fragments of adequate rates variably. In playing an on-demand HLS video using built-in application of iPhone and android phone, the video is usually reset to play from the beginning when connection changes to Wi-Fi to 3G/4G. Due to disconnect the video stream and make a new stream, and due to drop playback position in changing streams, the video is reset when the connection changes. To prevent this, we implement web browser based player. The player remember the fragment number in playing and, request the next fragment instead of the first one when the network connection changes. This does not reset the video and continues the multimedia service.

I. INTRODUCTION

With the proliferation of mobile and portable devices, there is an imminent need for network to allow enhanced video and multimedia services[1]. Especially, high resolution devices such as iPad and android tablets lead needs for high quality multimedia services on these devices and draw a huge amount of mobile multimedia traffic. The average volume of video traffic on mobile networks has risen by 10 percentage points since 2011 - up to 50% from 40%. In certain networks, video traffic is up to 69% of total traffic [2-3].

However, networks are sometimes unstable and smartphone frequently changes its network interfaces between 3G/4G network and Wi-Fi network. Smartphone users usually use their own Wi-Fi networks for high speed and low charges, but they have poor connections when they move around due to short range of Wi-Fi. Also, it has a few meters of coverage in high attenuation indoor environment.

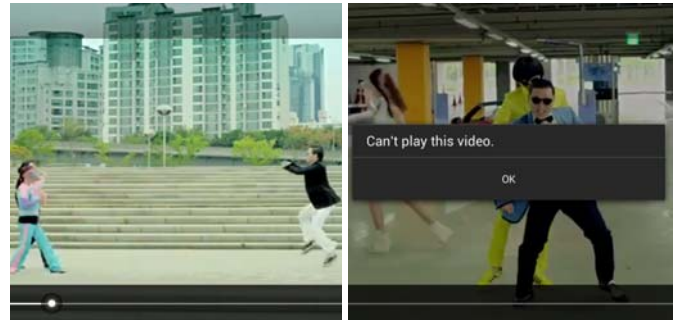


Figure 1. Normal playback (left) and the playback error of built-in multimedia player of Android 4.0 when network interface changes from WiFi to 3G (right)

In these frequent changes of network interfaces, multimedia stream usually suspended or even lose its connection. As shown in Figure 1, built-in players of iOS and Android are suspend and terminated when network interfaces changes in playback of video sources from the web.

To solve this problem, we implement web-based multimedia player based on standard web technologies; HTTP Live Streaming [4-5], JavaScript [6-7], and HTML5 Video Events and API [8-12].

II. PROPOSED ARCHITECTURE

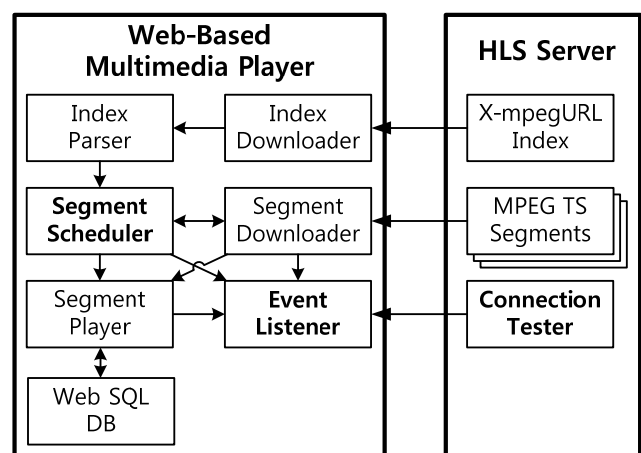


Figure 2. The architecture of Web-Based Multimedia Player

In this section, we propose the architecture of continuous player for HLS on heterogeneous network. The player is based on the standard HTML5 tags, JavaScript, and built-in video player of mobile web browser.

HLS server offers index files that containing URL of mpeg segments. Index Downloader downloads this index and Index Parser parses this to JavaScript object.

Segment Scheduler gets URLs of mpeg segments then organizes download and playback schedule based on Event Listener. This scheduler is a substitute of scheduler of built-in video player since that of built-in one makes error and disconnect HLS when networks interfaces changes.

Event Listener receives various events from downloader, player, Connection Tester, etc. It detects download and playback completions of segments. Also, it detects whether the player is online or offline.

Connection Tester of HLS Server returns connection information to Event Listener. The player based on pure web technologies of mobile web browser cannot access its own connection information including IP and bandwidth. So, Event Listener queries to Connection Tester to get these information in critical events.

Web SQL DB keeps track of playback information and other logs for resuming playback, error recovery, and maintenance.

III.IMPLEMENTATION

This section covers details of implementation and demo scenario of Web-Based Multimedia Player. Its Major parts are A) player, B) downloader, C) event listener, and D) connection tester. We implement the player based on the built-in web browser and built-in video player of Andriod 4.0. We use Adobe Flash Media Server 4.5 for HTTP Streaming Server and jQuery library.

```
$(segments).each(function(n){
    $('<video />', {
        id: 'segment_'+n,
        src: host + segments[n],
        preload : 'none',
    }).css('display','none').appendTo("video_player");
});
```

Figure 3. Code for initiating player

In initializing the player, HTML Video objects of each segment are created and hidden without preloading.

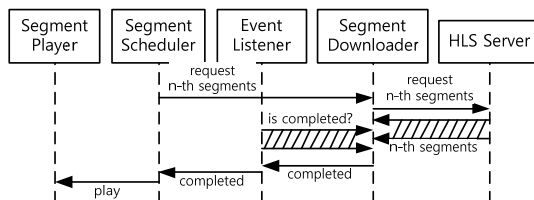


Figure 4. Overall procedure of download and play of n-th segment

In figure 4, overall download and play procedure are described. The n-th segment is requested by Segment Scheduler then, Segment Downloader start to download and Event Listener operates its own loop to manage the download. The segment downloaded is played after Segment Scheduler receives completion messages from Event Listener.

```
$('#segment_'+n)[0].addEventListener('stalled', function(){
    //code to reset download and to notify it to the scheduler
}, false);
$('#segment_'+n)[0].load();
```

Figure 5. Code for download n-th segment

For downloading, load method is called. Then built-in web browser downloads the segments. If download is suspended for more than three seconds, download is reset.

```
[Loop]
    if(video[n].duration == video[n].buffered.end(0)){
        //code for Event Listener to notify download completion
    }
[/Loop]
```

Figure 6. Code for notifying Event Listener to complete download of n-th segment

To manage download of each segment, Event Listener periodically checks downloads in the loop using setInterval() method of JavaScript.

```
$('#segment_'+n)[0].addEventListener('ended', function(){
    //code for Event Listener to notify the end of playback of n-th segment
}, false);
$('#segment_'+n).css('display','block')[0].play();
```

Figure 7. Code for play n-th segment with Event Listener for notifying end of the playback

In the play of a segment, Event Listener is added to the player to detect the end of the playback of the segment to play next segments.

```
var is_online = false;

window.addEventListener('online', function(e) {
    if(!is_online){
        //codes to query to Connection Tester
    }
}, false);

window.addEventListener('offline', function(e) {
    if(is_online){
        //codes for connection loss
    }
}, false);
```

Figure 8. Code for sensing connection information

As web browsers cannot detect its own connection information except whether they are online or offline, we

implement Connection Tester to collect connection information including its own IP address and bandwidth of the connection.

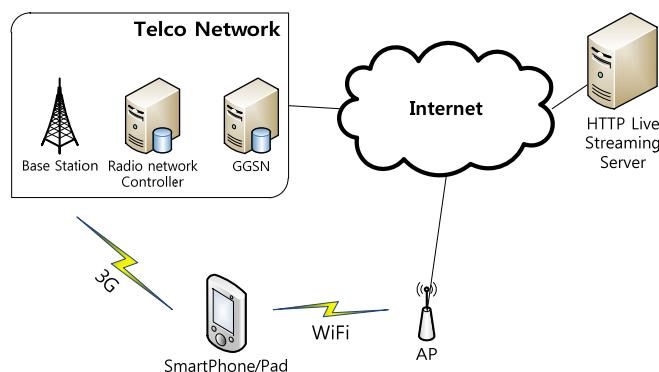


Figure 9. Demo scenario of Web-Based Multimedia Services in Heterogeneous Networks

We tested our player in the demo scenario. At first, smartphone is connected to WiFi AP. As we turn off the AP, the player detects network connection and plays segments in the buffer. After reconnection to 3G, the player downloads and plays next segments again.

IV. RELATED WORKS

A. HTTP Live Streaming

HTTP Live Streaming is an HTTP-based media streaming protocol implemented by Apple Inc. In HLS, a video is truncated to fragments of the constant length; these are contained to MPEG-TS and transmitted over HTTP. Periodically requests of these fragments and playing them continuously enable streaming service. At the start of the streaming session, it downloads index (m3u8 file) containing the metadata for the various sub-streams which are available.

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-ALLOW-CACHE:NO
#EXT-X-VERSION:2
#EXT-X-TARGETDURATION:8
#EXTINF:8,
gangnamstyle.mp4Frag0.ts
#EXTINF:8,
gangnamstyle.mp4Frag1.ts
#EXTINF:8,
gangnamstyle.mp4Frag2.ts
#EXTINF:8,
gangnamstyle.mp4Frag3.ts
#EXTINF:8,
...
#EXTINF:8,
gangnamstyle.mp4Frag30.ts
#EXTINF:4,
gangnamstyle.mp4Frag31.ts
#EXT-X-ENDLIST
```

Figure 10. An Example of index file (m3u8 file) of HLS

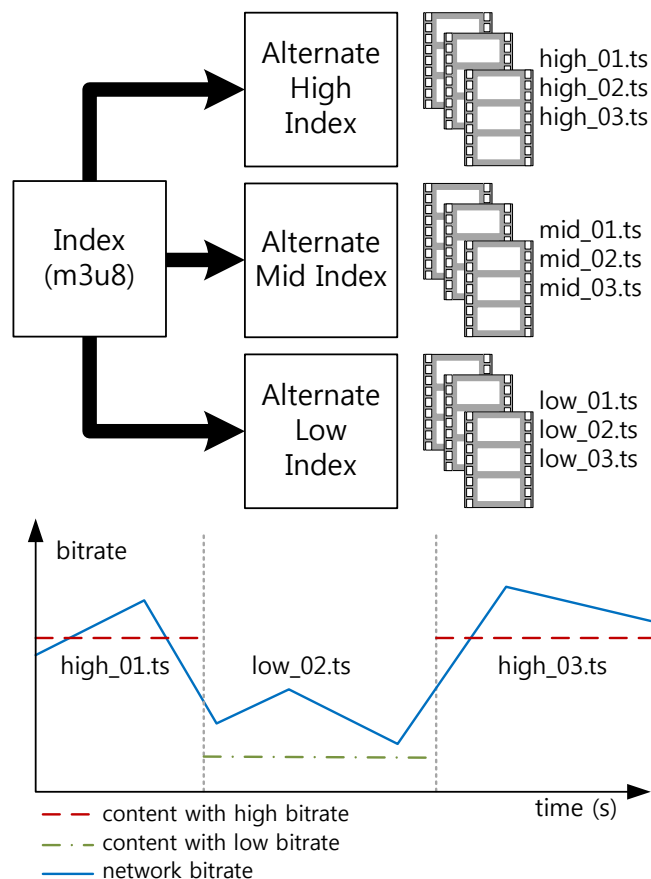


Figure 11. An Example of index file (m3u8 file) containing various bitrates and adaptive bitrate control

As we shown in figure 11, HTTP Live Streaming support adaptive bitrate streaming. It detects user's resources – bandwidth and CPU capacity - in real time and adjusts the quality of video stream. It prepares segments with multi bitrate and request adequate segments to continue the video stream in network congestions. However, built-in player lost its playback information and does not recover the stream when network disconnects. That's the reason to implement our web based multimedia player.

V. CONCLUSIONS

In this paper, we implement Web-Based Multimedia Player using pure web standard technology and HTTP Live Streaming. We tested the player between WiFi to 3G connections and works with few delay in connection change.

In future research, we will study multimedia services on smart phone with multiple NICs (Network Interface Cards). Android and iOS kernel do not allow enabling multiple NICs simultaneously, in other world, it allows single data connection. So, 3G/4G connection is disconnected when you try to use Wi-Fi and vice versa. In further study, we will modify android kernel to allow enabling multiple NICs simultaneously, then implementing HLS request through multiple connection before dropping one connection.

ACKNOWLEDGMENT

This work was supported by the IT R&D program of MKE/KEIT. [KI001822, Research on Ubiquitous Mobility Management Methods for Higher Service Availability] and [10039160, Research on Core Technologies for Self-Management of Energy Consumption in Wired and Wireless Networks].

REFERENCES

- [1] Cisco, Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2011–2016, 2012
- [2] Bytemobile, Inc, Mobile Analytics Report, 2012, Available: http://www.bytemobile.com/news-events/2012/archive_230212.html
- [3] Brandon Butler, Network World, Video now accounts for half of mobile data traffic, 2012
- [4] IETF, HTTP Live Streaming – Internet Draft, 2010, Available: <http://tools.ietf.org/html/draft-pantos-http-live-streaming-05>
- [5] Adobe Flash Media Interactive Server 4.5 <http://www.adobe.com/products/flash-media-interactive.html>
- [6] JavaScript Tutorial, W3CSchool.com, 2012, Available: <http://www.w3schools.com/js/>
- [7] jQuery, 2012, available: <http://jquery.com>
- [8] Philippe Le Hégarret, HTML5 Video Events and API, W3C, 2011, Available: <http://www.w3.org/2010/05/video/mediaevents.html>
- [9] Mario Fischer, HTML 5 Video – DOM Attributes and Events, 2010, Available: <http://www.chipwreck.de/blog/2010/03/01/html-5-video-dom-attributes-and-events/>
- [10] Microsoft, Using HTML5 video events, 2012, Available: <http://msdn.microsoft.com/library/hh924822>
- [11] HTML5 Video, Wikipedia, 2012, Available: http://en.wikipedia.org/wiki/HTML5_video
- [12] HTML5 Video, W3Schools, 2012, Available: http://www.w3schools.com/html/html5_video.asp