Streaming media live broadcast system based on MSE

Heyao Chen, Jiali Bian
Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing University of Posts and Telecommunications, Beijing, 100876, China
chyxyjh@163.com

Abstract. In this paper, a streaming media live broadcast system is designed and initially implemented based on MSE (Media Source Extensions API). This paper selects HTTP-FLV as the server transport protocol and implements an HTML native FLV player. The streaming live broadcast solution achieved has better performance, lower power consumption and excellent user experience than the traditional Flash Player live broadcast solution.

1. Introduction
With the development of Internet, the video live broadcast industry has become an important part of the daily entertainment life of people. Therefore, technical research on the field of streaming media has become a major technology hotspot in the Internet industry. The traditional web live solution packages media data in the FLV container format, transmits data through the RTMP protocol, and play through the Adobe Flash Player on the browser. As time goes by, Flash Player’s high CPU usage, high power consumption and poor performance has become increasingly prominent. Based on the Media Source Extension API proposed into HTML5 recently, this paper implements a FLV streaming media live broadcast system under HTML5 platform, which effectively improves the performance and user experience on streaming media live broadcast.

2. Key technologies for streaming media live broadcast system

2.1 FLV
FLV is a container file format used to deliver digital video content. Due to its clever design and simple format, FLV is suitable for streaming live broadcast scenarios. Its transmission and encapsulation performance have greater advantages than MP4 formats.

2.2 HTML5 video player
The <video> tag proposed into HTML5 provides HTML with native video play capability. Before HTML5 video player’s appearance, browser play video relied on support from third-party platforms such as Flash Player. Compared with the Flash Player, HTML5 video player has the advantages of good performance and low power consumption. However, HTML5 video player can only play one file and only guarantee support of the MP4 format in common browser environment. Obviously, HTML5 video player provides a better video play platform than Flash Player, but cannot assume the task of providing a live streaming environment independently.

2.3 MSE-Media Source Extensions API
MSE is a specification proposed by W3C on November 17, 2016 and is currently compatible with mainstream browsers. This specification allows JavaScript to dynamically build media streams for <audio> and <video>. In a word, based on MSE, we can control the play of the HTML video player, not limited to playing only one file, but can be streamed.

2.4 HTTP-FLV
HTTP-FLV is an HTTP-based streaming IO transport FLV technology. The implementation of HTTP-FLV is that the server does not add content-length field in HTTP header when replying to the client's request for media data, so that the client considers that the data has not been transmitted over and keep on receiving data. In this way, the client can continuously receive the FLV data sent by the server, forming a streaming FLV transmission.

3. System structure
The system is mainly composed of video collection source, streaming media server and HTML5 FLV streaming player. The video collection source is responsible for collecting video data, compressing the video data, and then pushing the video data to the streaming media server. The streaming media server is responsible for transporting streaming media data to the HTML5 FLV streaming player. The HTML5 FLV streaming player is responsible for decoding and playing the streaming media data, and presenting the live broadcast to the user.

3.1 Video collection source
The video collection source is mainly responsible for two tasks, video collection and video coding. Because video encoding and decoding can greatly influence system performance, the streaming media server is designed to only transfer the encoded video data. Therefore, the video collection terminal must complete the video encoding work before pushing the video data to the server. This system selects H.264 as the video coding standard, and the video collection source needs to compress the encoded video data through the H.264 encoder and then push it to the streaming media server through RTMP or other protocols.

3.2 Streaming media server
The streaming media server is based on HTTP-FLV. It establishes a streaming FLV transmission service and continuously transmits the streaming media data from the video collection source to the browser client for playing by the HTML5 FLV player. The connection protocol between the server and the live client mainly includes RTMP, HTTP-FLV, and HLS. Among them, HTTP-FLV has the advantages of small delay and high performance. What's more, media data transmitted through HTTP-FLV can be read by browser JavaScript running environment. Therefore, this system selects HTTP-FLV as the server transmission protocol.

3.3 HTML5 FLV streaming player
The HTML5 FLV streaming player is the core of the system. The native HTML5 player has better performance and lower power consumption than the Flash Player, but it cannot decode the FLV format video, nor is it suitable for streaming video live scenes. Therefore, based on the Media Source Extensions API, this system converts the FLV streaming media data into the ISO BMFF (Fragmented
MP4) segment that can be decoded and played by the HTML video player, realizing the real-time play of the FLV stream, and designing better performance and user experience than the Flash Player.

This paper focuses on the implementation of the HTML FLV streaming player, which will continue to be discussed below.

4. HTML5 FLV streaming player
The main idea of the HTML FLV streaming player is to convert the FLV data stream transmitted by the server into an ISO BMFF (Fragmented MP4) segments and then send it to the native HTML video player to realize the live streaming effect. The implementation of this function can be summarized into the following four processes:

• Initiating a Fetch request to the streaming server in the browser JavaScript runtime environment.
• The server establishes an HTTP-FLV transport stream for the Fetch request from the browser, and continuously sends the collected real-time FLV data to the browser.
• The browser JavaScript script performs demux on the newly received FLV data through the demuxer module, and then encapsulates it into ISO BMFF (Fragmented MP4) that can be played by HTML video through the remuxer module.
• Based on MSE, the latest video clips are submitted to the HTML5 video player to play, and the effect of streaming live broadcast is realized.

Figure 2. HTML5 FLV streaming player.

The implementation of the demuxer module is a major difficulty in this system. The demuxer module is responsible for extracting the audio and video data encapsulated in the FLV, and the extracted data is then sent to the remuxer module for encapsulation. For a FLV streaming data, first check the signature in its FLV Header to confirm that the correct FLV data has been received. Each piece of data in the FLV is called a tag. Then check the tag header in each tag, determine the number of bytes in the tag data part according to the datasize field, and then assign different types of tags to different data extraction methods according to different types (audio, video and script data) to extract data. The extracted data can be submitted to the remuxer module for packaging.

In summary, the HTML5 FLV streaming player implemented by this system only performs demux and remux on the transmitted FLV streaming media data, only adjusts the container packaging format.
of the video, and does not change the video encoding. The highly time-consuming tasks such as video decoding and encoding are not executed. In that way, we manage to reduce the performance pressure on the browser.

5. System test
A test system is implemented to carry out system testing. The test system uses Open Broadcaster Software to record the screen as a video collection source, and implements an HTTP-FLV streaming server based on Go. After loading the HTML5 FLV streaming player on the Chrome browser and configuring the streaming server's host, we can see a smooth, clear live view. In order to test the performance of the HTML FLV streaming player implemented by the system, we tested this module on PCs of different specifications. The test results are shown in the table. We can see the HTML FLV streaming player performs well on a variety of PCs.

| Hardware Configuration | CPU | Memory |
|------------------------|-----|--------|
| PC1 Core i5-2400+4GB   | 44% | 76%    |
| PC2 Core i5-3210M+8GB  | 35% | 51%    |
| PC3 Core i7-8550U+16GB | 17% | 30%    |

6. Summary
This paper proposes a streaming media live broadcast system based on Media Source Extensions API. A native HTML FLV player has been implemented to keep FLV as container format in streaming media transmission. The system can better solve the shortcomings of the traditional Flash Player live streaming solution, and the HTML-based streaming player solution has good cross-platform and can be widely applied to Web service scenarios. Since the system is still in the testing phase, the functionality provided is limited, and more features will be added later to enhance the user experience.

References
[1] W3C (2016) Media Source Extensions. https://www.w3.org/TR/media-source/.
[2] Zakas, N. (2012) Professional JavaScript for Web Developers. Wrox Publishing, Birmingham.
[3] Akidau, T. (2018) Streaming Systems. O'Reilly Media, Sebastopol.
[4] Huang, Q. (2016) Analysis of Video Transmission Technology Based on HTML5. Information system engineering, 09: 49–50.
[5] Adobe (2008) Video File Format Specification Version 10. https://www.adobe.com/content/dam/acom/en/devnet/flv/video_file_format_spec_v10.pdf