Proceedings of the World Congress on Electrical Engineering and Computer Systems and Science (EECSS 2015)
Barcelona, Spain – July 13 - 14, 2015
Paper No. 304

A Live Online Lecture System Using Adaptive Streaming Over HTTP

Lishuang Xu, Jiajun Wang
Shanghai Jiao Tong University, Continuing Education College
1954 Huashan RD, Xuhui District, Shanghai, China
lsxu@sjtu.edu.cn; jjwang@sjtu.edu.cn

Ruimin Shen
Shanghai Jiao Tong University, Department of Computer Science
1954 Huashan RD, Xuhui District, Shanghai, China
rmshen@sjtu.edu.cn

Abstract -This paper describes the design and implementation of a live online lecture system using adaptive streaming over HTTP. Our adaptive strategy as described in this paper is based on element stream-based HTTP streaming protocol where each element stream data is transferred as separate segment files via the network. Therefore the system allows the encoders and receivers to send and receive the element streams adaptively according to their device constrains and network capabilities. The specific adaptive strategy is proposed as adaptation logics in the form of flowchart in this paper.

***Keywords*:** http streaming, adaptive streaming, online lecture, streaming media

1. Introduction

Live online lecture systems are widely used in e-Learning, online training and web seminars. Being able to view online lectures from all kinds of clients including smart phones pose a great challenge to researchers and developers because of the limited resource and relatively unreliable network of the mobile clients.

In this paper, we present a live online lecture system using adaptive streaming over HTTP protocol, which is designed with the constraints of smart phones and mobile networks in mind.

This live online lecture system can contain two concurrent video element streams and also an audio stream. One video element stream shows what the lecturer is presenting on his/her PC's screen. Which we will refer to as slideshow element stream afterwards. The other shows video from the camera pointing at the lecturer, which we will refers to as camera video element stream afterwards. The system is able to switch among "slideshow element stream and camera video element stream" mode, "slideshow element stream only" mode, "camera video element stream only" mode, and even "audio only" mode. The system can change mode according to the resource and network status of the client.

The rest of the paper is organized as following. In section 2,we provide a survey of related works. In section 3, we describe the reasons we choose element stream-based http streaming protocol in our system. In section 4, we present the system structure, In section 5, we describe the specific adaptive strategies for the encode application and receiver application respectively in the form of flowcharts. In section 6 we share some implementation information. Finally we conclude with thoughts on future research directions.

2. Related Work

There are many online live lectures systems available, such as Cisco's WebEx(Web-1), Ctrix's Go2Webinar(Web-2), the open source Dimdim(Web-3) and a P2P streaming based PPClass(Weikai et al.,

2009). However, most of these systems only support PC version and haven't implemented the version that can run on smart mobile devices.

MLVLS (Carsten et al., 2009) is among the few mobile live lectures system available. It only provide client for Symbian phones. The major advantage of our system compared to MLVLS is that we use an enhanced HTTP streaming protocol and adaptive streaming technology base on the protocol to improve the performance and efficiency on mobile devices.

Cisco recently added iPhone and Android client support to their WebEx solution (Web-4),. But their iPhone and Android client does not support viewing the camera video element stream. And there is no information available about the underlying streaming protocol.

PPClass-M (Weikai et al., 2011) is the system adopting HTTP streaming protocol in mobile clients. But the system is not optimized by adaptive streaming related technology.

3. Element Stream-based HTTP Streaming Protocol

There are several reasons we adopt http streaming protocol in our online lecture system:

(1) Existing Internet infrastructures is designed for best-effort delivery of files, while what HTTP streaming does is to chop the media data into many small segment files and then these files are pulled by the clients with the HTTP protocol. So HTTP streaming has the advantages of fully exploitation of existing Internet infrastructures (e.g., caches, proxies, CDNs)(Christopher et al., 2011).

(2).HTTP streaming is based on the plain old HTTP protocol, which can traverse most NAT and Firewalls. While According to the design of RTSP protocol, a RTSP client should inform the server about the separate UDP port addresses the client will use to accept incoming RTP packets. If the client is behind a NAT or Firewall, such port addresses actually are internal ones and simply cannot be reached from outside (Weikai et al., 2011).

(3).HTTP streaming has better tolerance to the interim network problems frequently occurs in mobile networks. While In RTSP (RTP interleaving mode) and RTMP, each application-layer view session is bound to a long-lasting TCP connection which would be easily broken by relatively unreliable mobile networks (Weikai et al., 2011).

Due to above reasons, media streaming is nowadays more and more provided over-the-top (OTT) using HTTP streaming technologies. in order to further improve the performance and efficiency of media applications, Adaptive HTTP Streaming was proposed(Web-5), The basic idea is to chop the media file into segments which can be encoded at different bitrates or resolutions. The segments are provided on a Web server and can be downloaded through standard HTTP GET requests. The adaptation to the bitrate, resolution, etc. is done on the client side for each segment, e.g., the client can switch to a higher bitrates – if bandwidth permits – on a per segment basis. This makes the download behaviour of the client adaptive and dynamic to fit best for its given bandwidth. There are also other proprietary solutions from different companies like Microsoft's Smooth Streaming [Web-6], Adobe's Dynamic HTTP Streaming(Web-7) and Apple's HTTP Live Streaming (Web-8) which more or less adopt the same approach.

But the above adaptive approaches are not suitable for our system because our system is different from traditional multimedia systems in that our system needs to support two different video element streams to be transferred while traditional multimedia systems only support one video element stream to be transferred. Therefore we proposed an enhanced HTTP-streaming based streaming protocol called element stream-based HTTP streaming protocol in that media stream is decomposed into different element streams and every kind of element stream produces its own small segment files separately.

As figure 1 shows, we define stream data each short period of time the system produce as fragment. Each fragment includes different segment files for different element streams. for example, Fragment 1 contains ten seconds stream data which is composed of segment file for audio element stream, segment file for capture video element stream and segment file for slideshow video element stream.

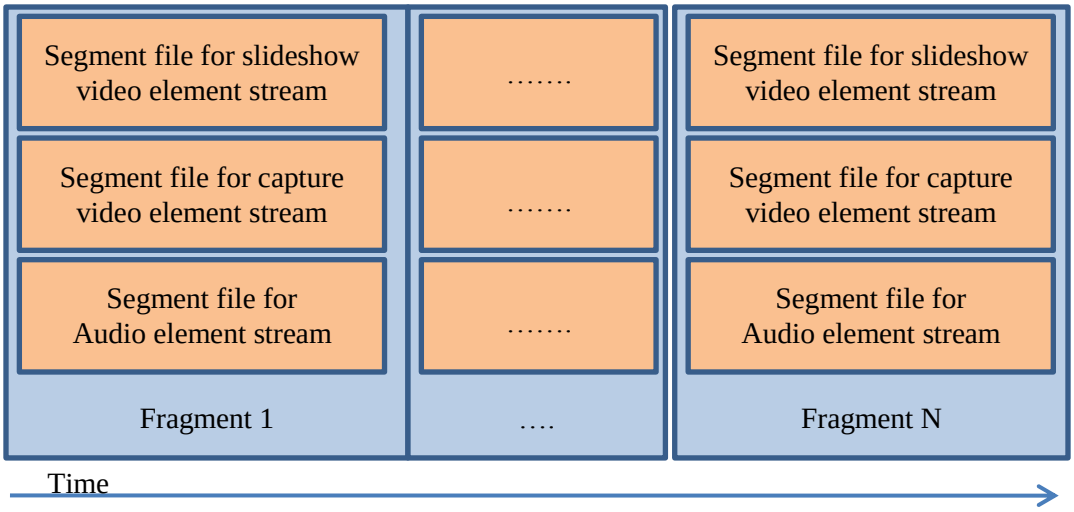

Fig. 1. The Relationship between fragment and segment files

4. System Structure

The system structure is illustrated in Fig 2. The encoder Application runs on the lecturer's PC. it sends element streams to the Mux server adaptively according to the needs and bandwidth permit. as figure 2 shows, the solid arrow in the picture means that the transfer of audio element stream is obligatory in the system, while the dashed arrow means that the transfer of slideshow element stream and the camera video element stream is optional.

The Mux Server is responsible for fragmenting the uploaded element streams into segment files. The Mux Server is also be responsible for generating an index file where the URLs of segment files for the latest N fragment durations are listed together with their timing information. This file is updated at the end of each fragmentation duration. Web server serves the segment files and the index file.

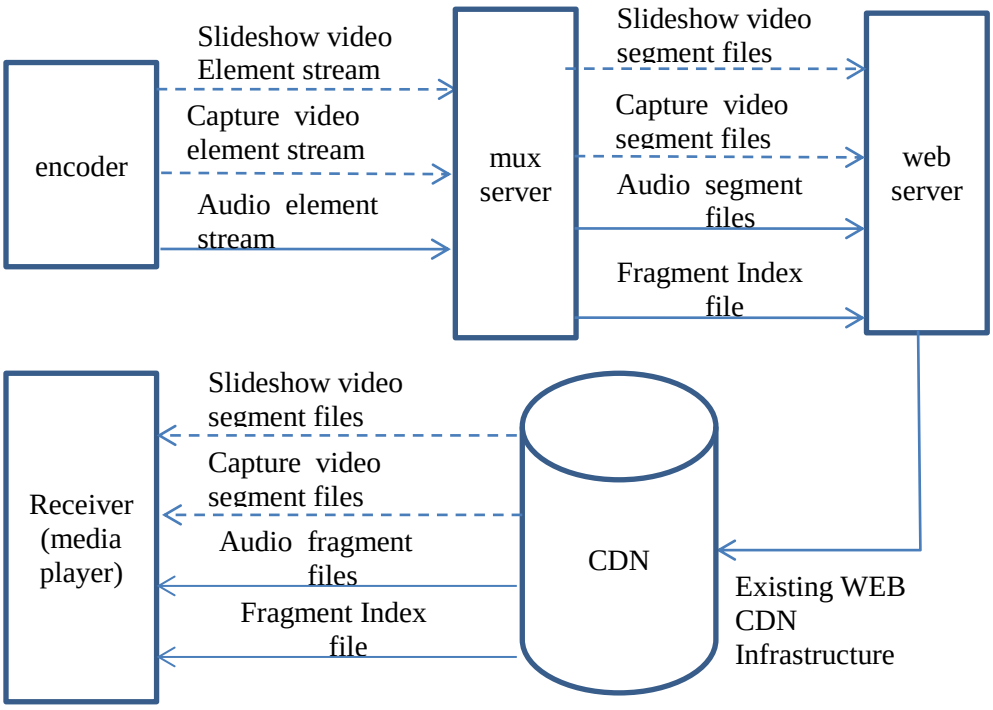

Fig. 2. The system structure

We leveraged the Web CDN service provided by ChinaCache to do client acceleration and load balancing.

The receiver (viewer) application runs on viewer's pc or smart phones. It can pull subset of segment files from web server. The specific adaptive strategy of the encode application and the receiver application goes to the next section of our paper.

5. Adaptive Streaming Over HTTP

Figure 3 shows the flowchart of the specific adaptive streaming strategy over HTTP for the encoder application. We firstly define priority value for each element stream (e.g. 0 for audio element stream. 1 for slideshow element stream, 2 for camera video element stream.). The smaller the priority value, the more important the element stream is. Lecturers should select element streams to be sent before the encoder starts. The selected element streams are added to the "sending list" which is a set for saving which element streams are to be sent to the Mux server.

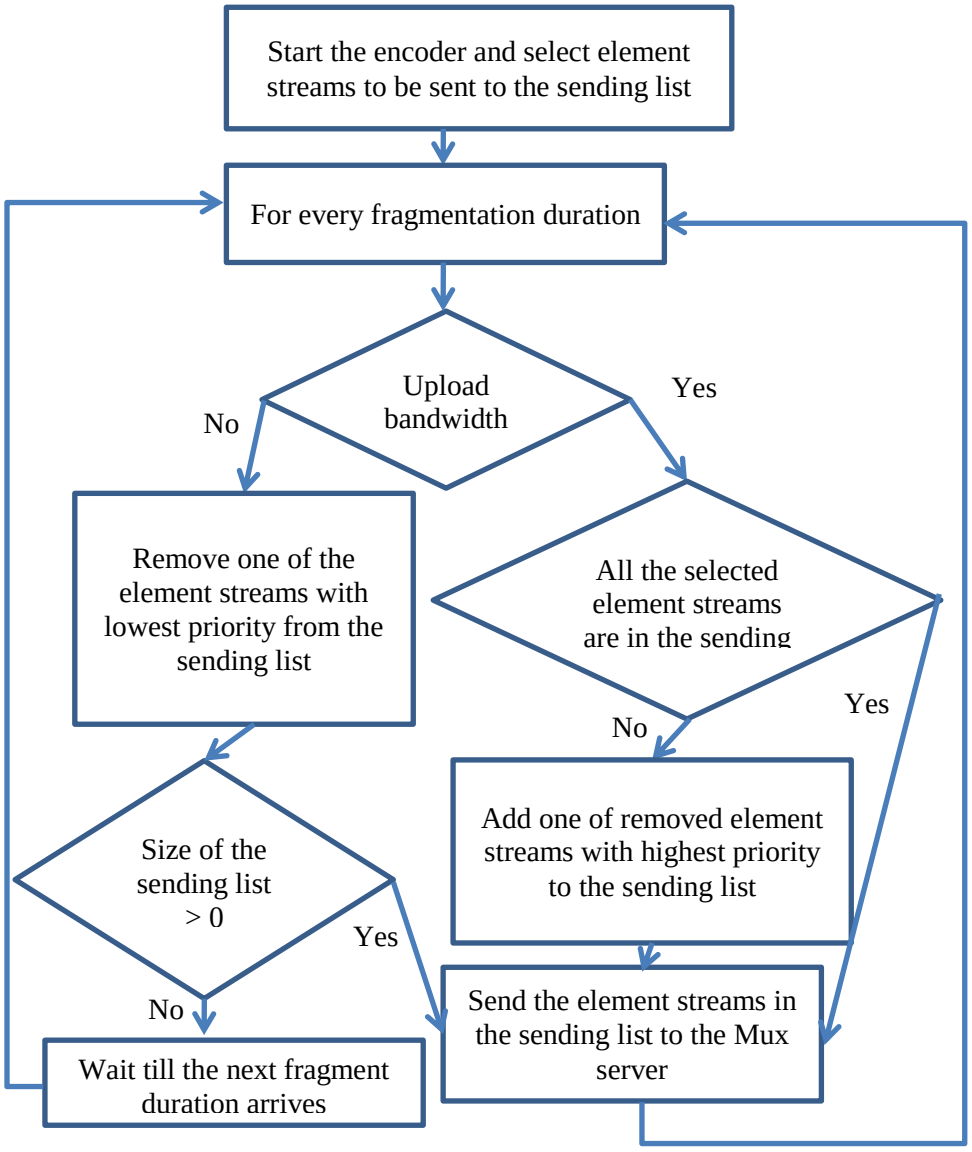

Fig. 3. The adaptive strategy for the encoder application

After the encoder application starts. for every fragment duration, the encoder checks it's upload bandwidth, if the bandwidth is enough, it will further check whether all the selected element streams are in the sending list, if so, all the selected element streams will be sent to the Mux server, otherwise the system will add one of the removed element streams with highest priority to the sending list and then all the element streams in the sending list will be sent to the Mux server.

If the bandwidth is not enough, the system will remove one of the element streams with the lowest priority from the sending list, and then further check whether size of the sending list is greater than zero. If so all the element streams in the sending list will be sent to the Mux server. Otherwise there are no element streams in the sending list and the application will wait till the next fragment duration arrives.

Figure 4 shows the flowchart of the specific adaptive streaming strategy over HTTP for the receiver application. Viewers should select element streams to be received after the receiver starts. The selected element streams are added to the receiving list (RL) and RL is assigned to the current receiving list (CRL) which is a set for saving which element streams are to be received. after that for every fragmentation duration the receiver pulls fragment index file form the web server, through parsing the index file we can retrieve the so-called "produced list" (PL) which is the information about which element streams are produced by the Mux server at current fragment duration. At this step we set CRL = CRL $\cap$ PL, which means the element streams which to be received is the intersection of CRL and PL.

Later on the receiver checks it's download bandwidth, if the bandwidth is enough, it will further check whether CRL equals to the intersection of RL and PL. if so, all the segment files related to the selected element streams in CRL will be pulled from the web server by the receiver, otherwise the system will add the element stream from (RL $\cap$ PL)-CRL with the highest priority to the CRL and then all the segment files related to the selected element streams in CRL will be pulled from the web server by the receiver.

If the download bandwidth is not enough. the receiver will remove the element stream with the lowest priority from the CRL and check whether size of the CRL is greater than zero, if so all the segment files related to the selected element streams in CRL will be pulled from the web server by the receiver. Otherwise there is no element streams to be pulled and the application will wait till the next fragment duration arrives.

6. Implementation

Till now the encoder application is implemented on Windows platforms. It consists of media capture module, media encoder module, adaptive streaming module and UI module. The UI module is implement with HTML/JS/CSS. All other modules are implemented in C++. The camera video element stream is encoded with H.264. The audio element stream is encoded with Speex. The slideshow element stream is encoded with SJSC+, an special type of video codec designed by us, which is optimized for compressing the image sequences captured from the computer screen (Chenping et al., 2010).

The Mux server application is implemented in python using the Twisted library. The server application can run on both Windows and Linux platform.

The receiver (viewer) application is implemented for Android-based smart phones. It consists of the adaptive HTTP streaming module, media decoder module, rendering module and UI module. The decoder module is implemented with C++ and invoked by other modules via JNI interface. All other modules are implemented in Java.

Figure 5 contains a screenshot of the receiver's application on android phone. We used a semi-3D theme for placing the two video windows in our "slideshow and camera" view mode. In the theme, the slideshow element stream window are rotated a small degree around Z-axis to give a sense of depth in space.

When the download bandwidth is not enough for the receiver to download all the three element streams segment files from the server, according to the adaptive strategy, the receiver will try not to pull the least important segment files from the server. As figure 6 shows, the application doesn't show the camera video element stream on the right window.

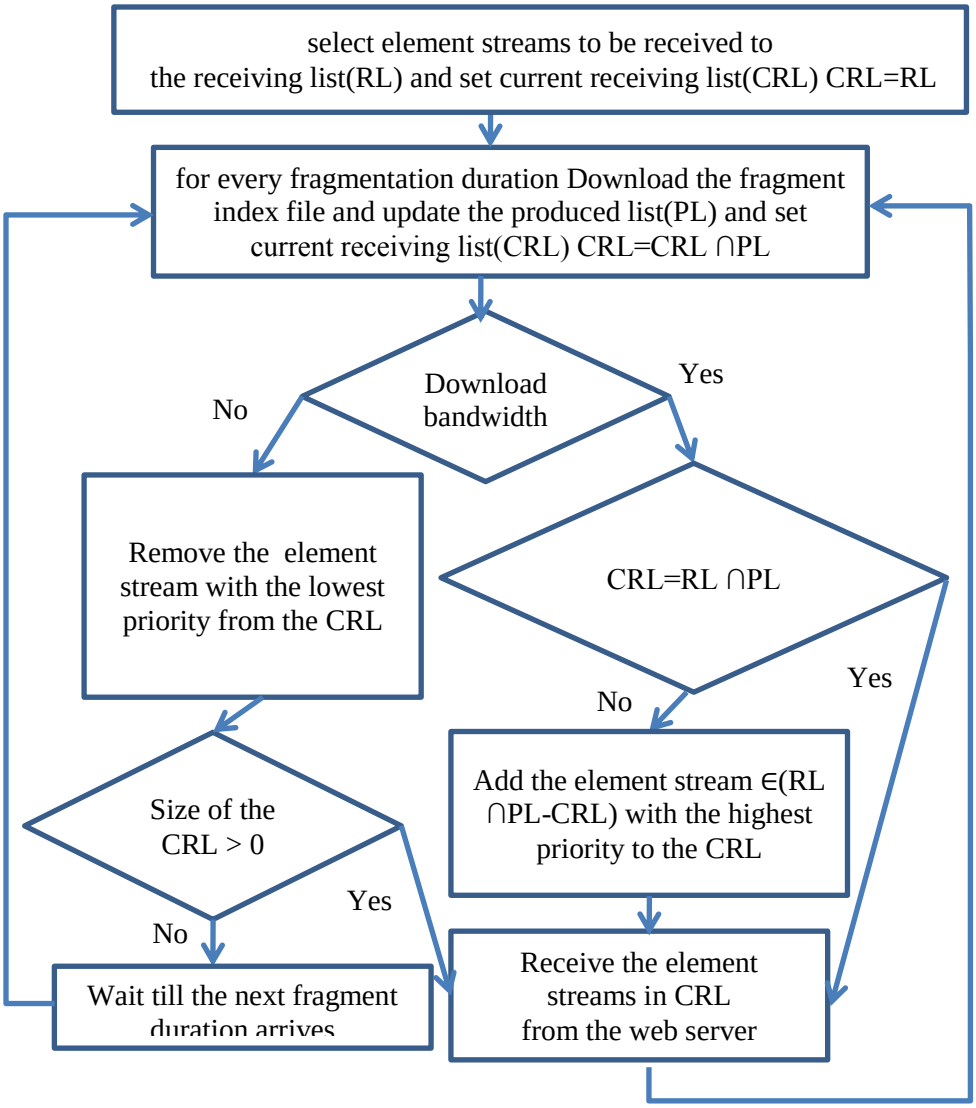

Fig. 4. The adaptive strategy for the receiver application.

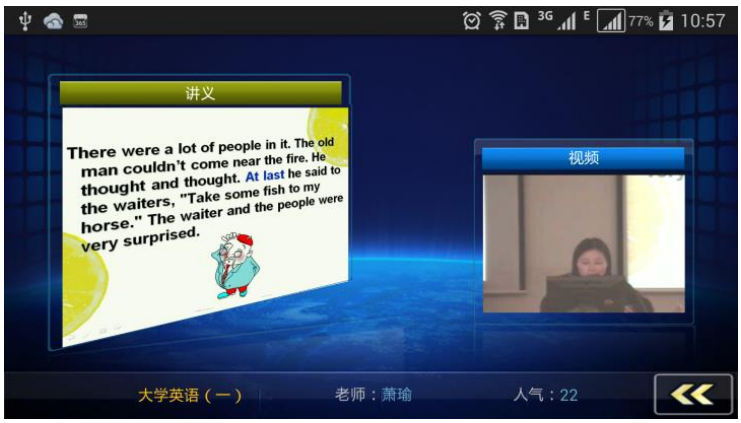

Fig. 5. "Slideshow camera video" view mode.

When viewer taps on the slideshow windows on figure 5 the application will switch to slideshow view mode as figure 7 shows. At this time there is no need to pull camera video's data from the server. Obviously not pulling camera video's segment files will improve the performance of the application and also save device's resources.

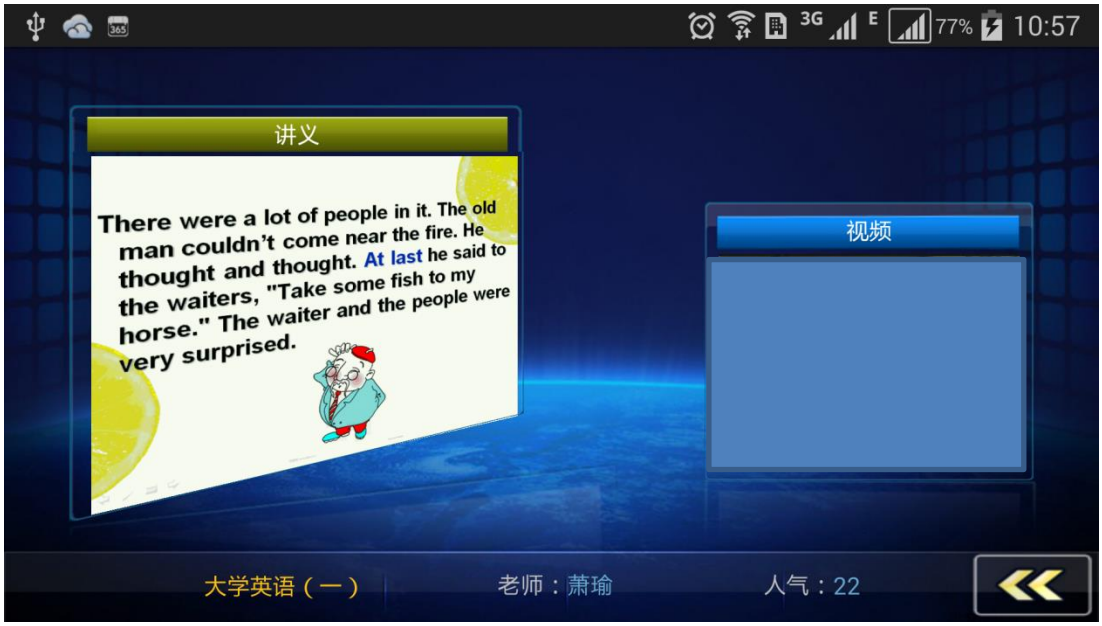

Fig. 6. "Slideshow camera video" view mode with camera video element stream not pulled due to bandwidth limit.

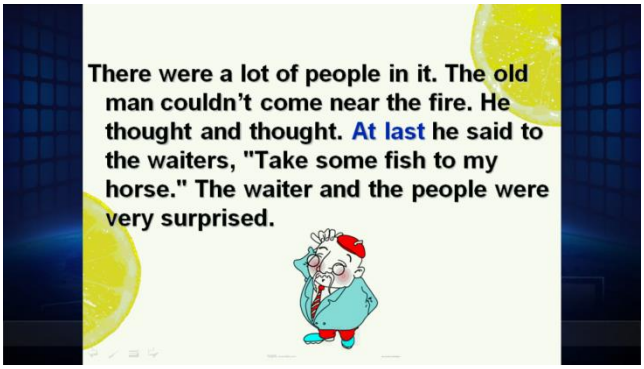

Figure 7. Slideshow view mode.

7. Conclusion

Our System is successfully deployed as a typical application for students at the Continuing Education College of Shanghai Jiao Tong University. By using the adaptive method described in the paper, users can get more comfortable viewing experience especially in the unstable network condition. We conducted a preliminary user study with our system. About 10 persons outside of our team were invited to use Android client to watch a typical live online lecture at the Continuing Education College of Shanghai Jiao Tong University. Then they are asked about questions related to the design of the user interface, the clarity and the smoothness of the lecture video and audio, etc. Most of the person gives positive response to these questions.

8. Future Directions

Our encoder application can only run on Windows platform before Windows 8. We are planning to port to Windows 8 and pad devices.

We are also planning to port our receiver implementation to iPhone and iPad.

References

Ulrich, C., Shen, R., Tong, R., & Tan, X. (2009). A Mobile Live Video Learning System for Large-Scale Learning – System Design and Evaluation. *IEEE Transactions on Learning Technologies*, 3(1), 6-17.

Lu, C., Xie, W., & Zhang, Z. (2010). An Enhanced Screen Codec for Live Lecture Broadcasting. *International Conference Audio, Language and Broadcast.*

Müller, C., & Timmerer, C. A VLC Media Player Plugin Enabling Dynamic Adaptive Streaming over HTTP. *Proceedings of the 19th ACM International Conference on Multimedia*, 723-726.

Xie, W., Zhang, Z., Lu, C., Wang, Y., & Shen, R. (2009). PPClass – A Classroom Lecture Broadcast Platform Based on P2P Streaming Technology. *Proceedings of International Symposium on Computing, Communication and Control*, Singapore, 482-488.

Xie, W., Lu, C., Zhang, Z., Lin, Y., & Shen, R. (2011). PPClass-M — A Live Online Lecture System Optimized for Mobile Access. *IEEE International Symposium on Broadband Multimedia Systems and Broadcast.*

Web sites:

Web-1: http://www.webex.com, consulted 3 March 2015

Web-2: http://www.gotomeeting.com/online/webinar, consulted 3 March 2015

Web-3: http://en.wikipedia.org/wiki/Dimdim, consulted 3 March 2015

Web-4: http://www.webex.com/products/web-conferencing/mobile.html, consulted 3 March 2015

Web-5: http://www.3gpp.org/DynaReport/26234.html, consulted 4 March 2015

Web-6: http://www.iis.net/downloads/microsoft/smooth-streaming, consulted 4 March 2015

Web-7: http://www.adobe.com/products/hds-dynamic-streaming.html, consulted 4 March 2015

Web-8: http://tools.ietf.org/html/draft-pantos-http-live-streaming-06, consulted 4 March 2015