A Novel Real-time Video Transmission Approach for Remote Laboratory Development

http://dx.doi.org/10.3991/ijoe.v11i1.3167

Ning Wang[1], Xuemin Chen[1], Gangbing Song[2] and Hamid Parsaei[3]
[1]Texas Southern University, Houston, USA;
[2]University of Houston, Houston, USA;
[3]Texas A&M University at Qatar, Doha, Qatar

***Abstract*—Remote laboratories are an inevitable necessity for Internet enabled education in Science, Technology, Engineering, and Math (STEM) fields due to their effectiveness, flexibility, cost savings, and the fact that they provide many benefits to students, instructors and researchers. Accordingly, real-time experiment live video streaming is an essential part of remote experimentation operation. Nevertheless, in the development of real-time experiment video transmission, it is a key and difficult issue that the video is transferred across the network firewall in most of the current remote laboratory solutions. To address the network firewall issue, we developed a complete novel solution via HTTP Live Streaming (HLS) protocol and FFMPEG that is a powerful cross-platform command line video transcode/encoding software package on the server side. In this paper, a novel, real-time video streaming transmission approach based on HLS for the remote laboratory development is presented. With this new solution, the terminal users can view the real-time experiment live video streaming on any portable device without any firewall issues or the need for a third party plug-in., This new solution also significantly benefits remote laboratory development in the future.**

***Index Terms*—Remote Laboratory; HTTP Live Streaming; HLS; FFMPEG; Across Firewall.**

I. INTRODUCTION

As the cost of setting up and maintaining the physical equipment rises, engineering laboratories are becoming an obstacle for school administrations. A good solution for this is the use of remote laboratories that allow students to perform real experiments over the Internet. Remote laboratories, also known under the name of online laboratories, remote workbenches, etc., have found widespread acceptance during the last decade [1][2]. Over 20 years ago, Stanford University introduced a new concept to learning: the use of video for the distribution and broadcast of classroom lectures [3]. Then, in the following few years, remote control of experiments and equipment over the Web was an idea that was being explored. Tools were also becoming available for remote control of instrumentation using network communication, and several demonstrations of camera control and data acquisition, as well as simple experiments have been made [4][5]. In the last decade, with the fast improvement and development of Internet technology, video broadcasting and the Web have merged and remote experiments are now available over the Internet to students, instructors and researchers anywhere and anytime. Consequently, the laboratory setting has been globalized.

Remote laboratories offer a range of benefits that can significantly improve pedagogical success: 1) they adapt to the pace of each student; 2) an experiment may be concluded from home if the time available at the lab was not sufficient; 3) it can be repeated to clarify doubtful measurements obtained at the lab; 4) the student may improve the effectiveness of the time spent at the lab by rehearsing the experiment beforehand; 5) safety and security are improved as there is no risk of catastrophic failure [6]. Remote laboratories are often used in Control, Robotic and Mechatronic education to illustrate theoretical principles and deployment methodologies [7]. Remote laboratories can be of many kinds and our previous works mainly focused on remote laboratories where the users accessed physical equipment through the Internet for remote experimentation purposes. From the point of view in the software architectures, remote laboratories architects have no choice but to build a middleware allowing remote clients to connect to the local computer that handle the device.[8] The novel unified framework was designed and developed to allow the set up of a distributed network of online experiments that works in any Internet browser without the need of any extra plug-ins [9]. And then, we also deliver the improved unified remote laboratory framework, which uses a Comet solution via Node.js and its Socket.IO package to improve the control commands and experiment data transmission performance. Nevertheless, real-time experiment live video streaming is an essential function for remote control experiment development, and normally it is also a very important interface for the remote experiment controllers and viewers [10].

Currently, among the many common solutions of the remote laboratory, the real-time video transmission approach in remote laboratory implementation almost used the special network port and Virtual Private Network (VPN) to transfer the video streaming. Accordingly, there is still an essential challenge of remote laboratory development which is real-time video transmission across a network firewall. Consequently, our improved solution for the real-time experiment video transmission across a network firewall is an essential improvement of the remote laboratory development, which will extend into the future.

II. PREVIOUS WORKS AND METHODS SEARCH

In our previous works, we developed the unified remote laboratory framework [10] and also resolved some challenges of developing a cross-browser and cross-device Web user interface for improvement of the unified framework [11]. This framework was used to implement the remote control engineering experiments in the last year. As an example, the new Smart Vibration Platform (SVP)

remote experiment, which was used to teach students in materials engineering courses, and this remote control experimentation were aimed to offer students hands-on experience through the Internet on structural vibration control using Magneto-Rheological (MR) or Shape Memory Alloy (SMA) braces under the novel unified remote laboratory framework. In order to improve the control commands and experiment data transmission performance of the unified framework, a Comet solution via Node.js and its Socket.IO package was implemented on the server side, and a new Web socket protocol, which lets the experiment communicate with Socket.IO was created for the workstation [10]. With this improved unified framework, the terminal users can remotely conduct the experiment by using any portable device without installing any plug-ins. However, we still faced an across the network firewall issue and the real-time experiment video was transferred via port 1026, which is the special network port in this improved framework of the remote laboratory. In order to resolve this essential real-time experimentation video transmission issue, we need to be looking for a new approach to transfer the real-time video via network port 80. Meanwhile, all Web contents of the experimentation are also transferred to the terminal users via the same network port. Therefore, the goal of this novel approach is to solve the real-time experiment video transmission across the network firewall issue and the Flash plug-in in Web browser issue we faced in the remote laboratory development.

Nowadays, with the fast development and improvement of network technology, a novel, reliable and free video steaming transmission protocol, HTTP Live Streaming (HLS), is more and more popularly used for real-time video transmission across a network firewall. HLS is an HTTP-based media streaming communications draft protocol implemented by Apple Inc. Apple used this protocol on September 1, 2010 to stream their iPod Keynote event live over the Internet, and on October 20, 2010 to stream their 'Back to the Mac' Keynote event live over the Internet as part of their QuickTime and iOS software [12]. Currently, there are more and more giant software companies' solutions to support HLS, such as Adobe Flash Media Server (Adobe FMS), Microsoft IIS Media Server, Google Android Honeycomb, etc. Since it requests to use only standard HTTP transaction, HLS is capable of traversing any firewall or proxy server that lets through standard HTTP traffic. Unlike UDP-based protocols (such as Real-time Transport Protocol (RTP)), HLS also allows content to be delivered over a widely available Content Delivery Network (CDNs).

In this paper, we will present the detailed working process and technical architecture regarding our novel real-time video transmission approach proposed for the remote laboratory development. There are three main parts in our new solution. These are HLS protocol, FFMPEG, which is a powerful cross-platform command line video transcode/encoding software package, is used as the video encoder working for HLS to cut the real-time experiment video into many little segments, and Segmenter software package, which is the cost-free Video streaming segments software package

III. PROPOSED APPROACH

The basic purpose of the new approach is to improve the remote laboratory video transmission function. In order to achieve this goal, we proposed the new real-time video transmission solution via HTTP Live Streaming protocol in combination with FFMPEG, which is a powerful cross platform command line video trans-code / encoding software package. For addressing the network firewall and flash plug-in issue, our new approach works by breaking the overall real-time experiment video stream into a sequence of small HTTP-based file downloads. Each download contains one short chunk of an overall potentially unbounded transport stream. As the stream is played, the client may select from a number of different alternate streams containing the same material encoded at a variety of data rates, allowing the streaming session to adapt to the available data rate. Briefly, the client (normally proposed Web browser) loads a playlist file of these video segments which was created following HLS protocol on the Web server (we proposed Apache Web Server). The content of this playlist are short clips residing on a Web server. In order to keep the real-time video transmission and playing smoothly, we regularly proposed 10 seconds for one short video clip. So for the server, we need to create and maintain a playlist file and the short segments of the real-time video stream. Therefore, the sequence of small HTTP-based file downloads is transferred via HTTP Live Streaming protocol, and we used a FFMPEG software package to break the overall real-time experiment video stream into short segments. Finally, the real-time experiment video segments are transferred via HLS protocol through network port 80 and the real-time video will be reassembled in the Web browser and shown to the end users.

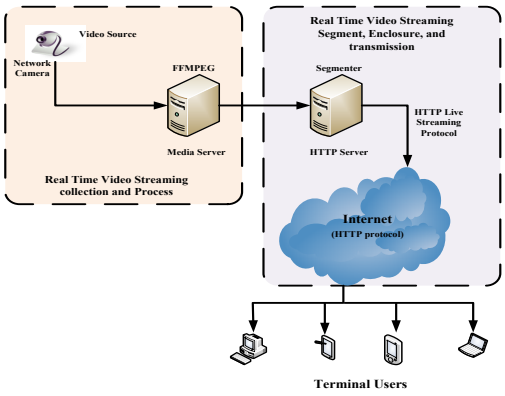


Figure 1. The working process of the new real-time video transmission approach

Figure 1 illustrates the working process of the novel real-time experiment video transmission approach. It is very clear that we not only need to setup the FFMPEG software package in media server side, but we also need to compile and install the Segmenter software package and configure the Apache Web server in HTTP server side. We build up the real-time experiment video processing and transmission environment, then the new solution will be implemented to resolve the real-time video transmission problem mentioned above.

IV. ARCHITECTURE OF THE NOVEL SOLUTION

The novel real-time experiment video transmission approach has been developed on the novel, good and free video steaming transmission protocol, HTTP Live Streaming, which is a HTTP-based media streaming communications draft protocol implemented by Apple Inc. As depict-

ed in Figure 2, it consists of the following basic entities: HTTP Live Streaming protocol (HLS), FFMPEG and Segment software package.

A. Streaming protocol Selection and HTTP Live Streaming protocol introduction

For the real-time experiment video transmission, we need to select a stable transmission protocol that performs well. Currently, there are three popular streaming transmission protocols which are widely used, Real-time Message Protocol (RTMP), HTTP Live Streaming (HLS) and Real-time Streaming Protocol (RTSP). As depicted in Table 1, we can know the detailed correlation between these three main streaming transmission protocols. Based on our requirements, we need to look for a real-time video transmission protocol which is stable, free, and has good capability across a network firewall. Consequently, after comparing, it is the best selection for us that the HTTP Live Streaming protocol (HLS) will be used in our new solution.

HTTP Live Streaming is a way to send video and audio over HTTP protocol from a Web server via network port 80 to client software (Web browsers and other client applications) on desktops, laptops or to other portable devices (including IOS-based portable devices, Android-based portable devices, Win8-based portable devices, Smart phones, etc.). Accordingly, the HLS Protocol mainly consists of three parts which are the server component, the distribution component and the client software. The server component is responsible for taking input streams of media and encoding them digitally, encapsulating them in a format suitable for delivery, and then preparing the encapsulated media for distribution. The distribution component consists of standard Web servers and they are responsible for accepting client requests and delivering prepared media and associated resources to the client. For large-scale distribution, edge networks or other content delivery networks can also be used. The client software is responsible for determining the appropriate media to request, downloading those resources, and then reassembling them so that the media can be presented to the terminal users in a continuous stream. The HLS protocol is also an Internet-Draft protocol submitted by Apple, Inc to the Internet Engineering Task Force (IETF), and the Internet-Drafts are working documents of the IETF, its areas, and its working groups [13].

Figure 3 shows the HLS protocol standardized basic configuration, which was defined by Apple, Inc [13]. As illustrated in Figure 3, there are four main critical parts of the process, which are video and audio source collection, video and audio streaming encoder module, media segment module and video and audio files distribution. In our remote experiment development, the video source is the real-time experiment video, which is input from the network camera directly and is the H264 format video. Currently, we only need to collect the video streaming without audio streaming to show the whole procedure of remote experimentation for the terminal users. As it fulfills our project requirements and is easy to maintain and improve, we prefer to use the FFMPEG software package as the encoder to process the experiment video, and we also use an open source free segment software package to break up the video into the little video clips. For the Web server setup, there are some good Web server software packages which are Nignx Web server and Apache Web server. Because in our previous unified framework development we used the Apache 2.0 Web server software package on the server side, as an improvement of the unified framework we implement the new video streaming transmission approach based on the Apache 2.0 Web server.

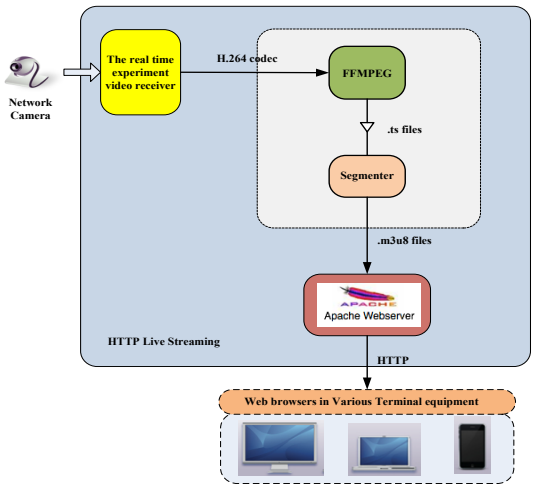


Figure 2. The architecture of the novel real-time video transmission solution.

TABLE I.
MAIN VIDEO STREAMING TRANSMISSION PROTOCOL COMPARISON

	RTMP	***HLS***	***RTSP***
Name	Real-time Message Protocol	Http Live Stream	Real-time Streaming Protocol
Upper Protocol	TCP / HTTP	HTTP	RTP / RTCP
Mode	C/S	B/S	C/S
Main Force	Adobe	Apple	Microsoft
Client request.	Flash supported Browser HTML5 supported Browser	HTML5 supported Browser	Players
Video request.	FLV, F4V	MP4	NA
Server Request.	FMS Server/Flash Server/Red 5	Normal HTTP Server	RTSP Streaming Server
Real-time Live Request.	dedicate Encoder Flash Media Encoder	a. media encoder(H.264 & AAC) into MPEG-2 streaming b. streaming segmenter from MPEG-2 streaming to segments for live c. file segmenter from file into segments for on-demand	related with server
Play format Request	FLV/F4V file divided into F4f media data file f4x index file	.Ts media data file, .M3u8 index file	related with server & player
Delay	Dependence	5~30 sec	< 2 sec
VCR	Support with high precision	Support with high precision	Support with normal precision
NAT traversal	Good	Not Good	Good

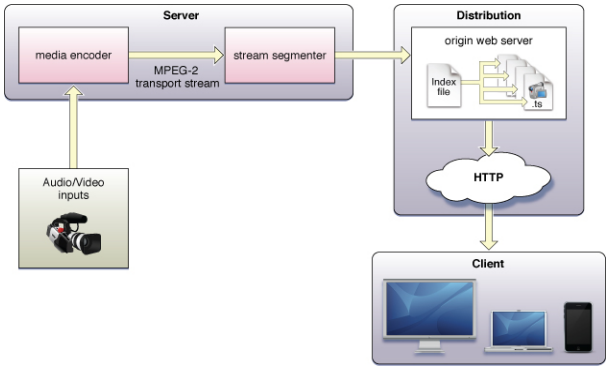


Figure 3. HLS basic configuration.

B. The FFMPEG for the real-time video encoder

Normally, the encoder is used to produce a MPEG transmission stream. For large audio and video producers (such as television, radio, etc.), it isn't a problem and they usually have hardware-based encoders that output exactly this. As there is a limited budget and the purpose of this experiment is to provide an approach that is easily extended and maintained in future, we need to be looking for a good and stable software solution. Consequently, a great open source software solution will be our objective. With this goal, we prefer to select the FFMPEG as the video encoder in our solution.

FFMPEG is a complete, cross-platform command line software package which can record, convert and stream audio and video. It includes libav-codec which is the leading audio/video codec library, and it is also the leading multimedia framework which is able to decode, encode, trans-code, Mux, deMux, stream, filter and play video and audio source files that humans and machines have created and collected. The FFMPEG not only can encode the streams in all required Codecs (such as HE-AAC, MP3, MP4, etc.), but also output these elementary streams in MPEG-TS. The FFMPEG software package is free software licensed under the LGPL or GPL depending on your choice of configuration options and it supports the most obscure, ancient formats up to those that are cutting edge. No matter if they were designed by some standards committee, the community or a corporation.

FFMPEG is a very fast video and audio converter that can grab from a live video and audio source (such as a network camera), and also can convert between arbitrary sample rates and resize video on the fly with a high quality poly-phase filter. Normally, the FFMPEG software reads from an arbitrary number of input files which can be regular files, pipes, network streams, grabbing devices, etc., and writes to an arbitrary number of output files which are specified by a plain output filename. In principle, each input file or output file can contain any number of streams as different types (such as video, audio, subtitle, attachment, data, etc.). The streams which were allowed numbers and types can be limited by the container format. Therefore, as a general rule, FFMPEG options are applied to the next specified file. Consequently, when we used the FFMPEG operation commands to process video, it was very important that the FFMPEG operation commands input must be ordered. And you can also have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file. As Figure 4 illustrates, we can know the detailed working process of the FFMPEG software package

Figure 4. FFMPEG Trans-Coding process.

FFMPEG calls upon the libavformat library (containing deMuxers) to read input files (our input files are the real-time experiment video streaming files) and get packets containing encoded data from these files. When multiple input files are coming, FFMPEG tries to keep them synchronized by tracking the lowest timestamp on any active input stream. Afterwards, Encoded packets are passed to the decoder. The decoder produces uncompressed frames (for example, raw video, PCM audio, etc.) which can be processed further by filtering. After filtering, the frames are passed to the encoder, which encodes them and outputs encoded packets again. Finally, those are passed to the Muxer, which writes the encoded packets to the output file.

Example 1 shows the command line which is one of the FFMPEG commands that were used in our new video transmission solution to collect the real-time video resource file from the network camera device

ffmpeg -i "rtsp://10.3.52.36/axis-media/media.amp?videocodec =h264&streamprofile =svp" -vcodec libx264 -subq 1 -g 250 -qmin 10 -qmax 51 -i_qfactor 0.71 /var/www/html/Hls_Test /Record. mp4

Example 1: FFMPEG command line

Previously, we used to output two video files and constantly switch them to achieve the real-time monitor remote experiment process. So we also developed an automatically executing command file using shell in the Linux server.

C. The real-time video segment Module Selection and implementation

In the architecture of our new solution, we need to segment the video which was output from the FFMPEG software package to the short clips (regularly proposed 10 seconds). Therefore, we need to be looking for one software package for this purpose. Currently, there are some good segment software packages for selection, and the Segmenter is a good candidate for us. The Segmenter is a software package which splits a transmission stream into chunks and then updates a playlist file with these chunk files. Most Segmenters are not open source. Consequently, they are difficult to improve and maintain in the future. As an example, Apple's Segmenter is a good solution and a cost-free Segmenter, but it is a binary program made for the Mac and does not run very well on the Linux server. It is not suitable for our new solution, since our approach needs to implement the segment function on the Linux server.

By the above description, the Free and Open Source Segmenter solution which was written by Carson McDonald, is the best selection for us up to date. It can run well on a Linux server, and we can freely download the source codes of this solution from his blog [14].

The segment software package, which is used in our novel solution, is based on the open source Segmenter solution and we modify some parts to match our requirements for the real-time experiment's video transmission. Figure 5 shows the detailed working process of the segment software packages in our new solution.

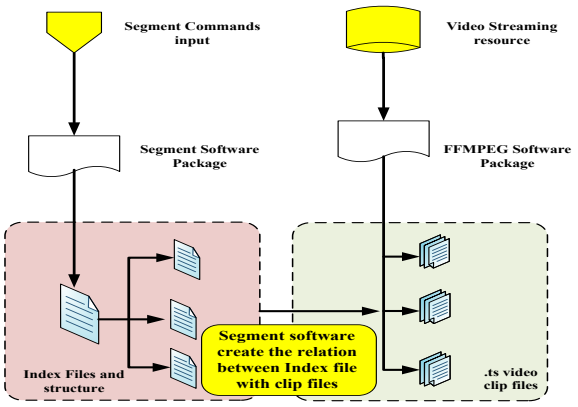


Figure 5. Segment software package working process.

The Following command line is an example used to create a stream from a video file created with the above FFMPEG command split into 10 second intervals:

```
Segmenter Record.ts 10 Record Record.m3u8 http://vr-lab.
engineeringtech. tsu.edu/ Hls_Test/
```

Example 2: Segmenter command line

D. Web Server configuration and Video transmission

For the real-time video stream transmission via network port 80, we need to configure the Web server that used the Apache 2.0 HTTP server software package to match the HTTP Live Streaming protocol requirement. Actually, Apache is probably not the best choice for delivering files via HLS protocol. However, our previous tasks were already finished on the Apache Web server and there's a low possibility that we can change the Web server for our current project status. Therefore, we decided to go with the very flexible and customizable Apache instead of the Nginx or Lighty Web servers, which are better suited for HLS.

```
AddType application/x-mpegURL .m3u8
AddType video/MP2T .ts
```

At this point, we should have a set of files that represent the stream and a stream definition file. Those files can be uploaded to a Web server at this point, but there is another important step to take that ensures they will be downloaded correctly and that is setting up mime types. There are two mime types that are important for the streaming content that needs to be added in the Web server configuration files.

So we need to add the following two line content in the Apache Server configuration file, httpd.conf, in order to let the Apache Web server support the HLS protocol and files.

```
.m3u8 application/x-mpegURL
.ts video/MP2T
```

We can create the corresponding folder to manage the m2u8 files on the Web server side. Client side, we only need to add the HTML code or use JWPlayer.js functions in the Web pages, and then the real-time video will be shown in the Web page for end users. Here, it is necessary to point out that the browser must support HTML5, otherwise we need to use the JWPlayer.js to support the HLS video. So far, most of the popular newest browsers (such as, Safari, Chrome, IE10, Firefox, etc.) almost support the HTML5.

V. THE REALIZED NOVEL VIDEO TRANSMISSION SOLUTION

A. The Realized real-time experimentation video live streaming with HLS across network firewall

In order to achieve the purpose of the real-time experimentation live video streaming with HLS protocol across a network firewall, we create two video files in http server side. We then write a short script to automate the run command file, which runs in Linux server side to record the video from the network camera, and then automatically switches to transfer these two video files with HLS protocol to the end user. The switch time slot was set to 1 minute, which also means that one video file time slot is 1 minute.

Currently, there is an issue where we cannot achieve continuous video playback through the smart phone because the hardware configuration is not high enough. And when we used the low configuration smart phone to load the video stream, the video playback is not smooth. However, this problem is not found in the IPhone and other high configuration smart phones.

B. Technical Characteristics of the improved unified remote laboratory framework

The architecture of the new video transmission solution, which is described in the previous section, has been implemented in the new Smart Vibration Platform (SVP) remote experiment, which was developed based on the improved unified remote laboratory framework. As the improved unified framework based on the Web 2.0 Technology, we built the whole framework directly on top of database, HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and jQuery/jQuery-Mobile JavaScript libraries. The PHP, which stands for Hypertext Preprocessor and database driven data-streaming solution, eliminated the need for the client side LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) plug-in. For the whole improved unified remote laboratory framework, we mainly use three technologies for the system implementation, which are LabVIEW to Node.js technology for experiment data transmission and experiment equipment control commands, the novel video transmission approach based on HLS protocol for real-time system monitoring, and Mashup technology for user interface implementation. Table II depicts the detailed technical characteristics of the whole improved unified remote laboratory framework.

C. Sample Paradigm of the new solution

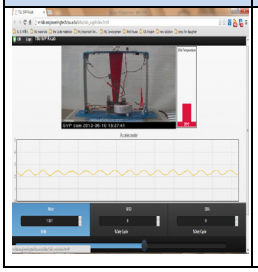
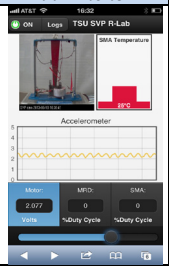
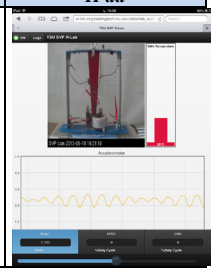
Table III illustrates the comparison of the sample paradigms, which show the terminal user interface, which was developed under the improved unified frameworks in different terminal devices.

This user interface can be run in any terminal equipment without the installation of any plug-ins or software. As an example, we used the remote Smart Vibration Platform (SVP) experiment to depict the experiment operation method from the user interface. When the switch on the left corner of the Webpage is on, the user can move the slide bar to control the motor rotation speed. The accel-

TABLE II.
MAIN VIDEO STREAMING TRANSMISSION PROTOCOL COMPARISON

Level	*Name*	*Technology/Protocol/Software package*	*Remark*
1.	Client – User Interface	Mashup technology, JavaScript	
2.	Data Protocol	Socket.io/Web socket	
3.	Real-time experiment video Transmission	HTTP Live Streaming Protocol/FFMPEG/Segmenter software package	
4.	Server - Web Service	Node.js (V0.8.8), JSON , MySQL	LtoN
5.	Experiment Server	LabVIEW (V8.6)	

TABLE III.
SAMPLE PARADIGM OF THE NEW SOLUTION COMPARISON IN DIFFERENT DEVICES

Sample Paradigm in Desktop	*Sample Paradigm in IPhone*	*Sample Paradigm in IPad*
		

erometer output, which indicates the platform vibration, is shown in the middle of the Webpage. And the real-time video is displayed on the top of the Webpage. With our novel transmission solution, the real-time experiment video can be transferred to different terminal interfaces across the network firewall.

Obviously, the remote laboratory system based on the improved unified framework is able to take full advantage of the real hardware and real instrumentation utilization. With the new real-time experimentation video transmission approach, the improved remote experiment provides better control performance in a wider range of terminal equipment.

VI. CONCLUSION

In this paper, we presented a new, real-time experiment video transmission approach to solve the across network firewall and flash plug-in issue, which were not solved in the unified framework for remote laboratory development. In our new solution, we not only introduced the current popular stream transmission technology, HTTP Live Streaming Protocol, to the remote laboratory development, but also successfully used this new protocol to improve a remote SVP experiment and solve the real-time experiment video across network firewall issue. As an example, we used this new solution in the improved remote SVP experiment development. The end users can view the remote SVP experiment by Web browsers, which must support HTML5 anywhere through the HLS protocol without any plug-in. Consequently, this novel real-time experiment video transmission solution gives the unified framework prodigious improvement.

REFERENCES

[1] Ma, J., Nickerson, J. V., “Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review”, ACM Computing Surveys, Vol. 38, No. 3, Article 7, September 2006.

[2] Gille,. D., Nguyen, A. V., Rekik, Y., “Collaborative Web-Based Experimentation in Flexible Engineering Education”, IEEE Transactions on Education, Vol. 48, No. 4, November 2005.

[3] Gibbons, J.F. et al. "Tutored videotape instruction: a new use of electronics media in education" Science, Vol.195, no.4283, pp. 1139-46, 18 March 1977. http://dx.doi.org/10.1126/science.195.4283.1139

[4] Barry P., "Virtual Laser Lab - The Future of Distance Learning is Here!", Instrumentation Newsletter, Volume 10, Number 4, Winter 98-99.

[5] Barrie, J.M., and Presti, D.E., "The World Wide Web as an Instructional Tool", Science, Vol. 274, 18 October 1996 http://dx.doi.org/10.1126/science.274.5286.371

[6] Ferreira, Sousa, Nafalski, Machotka, Nedic (2010). "Collaborative learning based on a micro-Webserver remote test controller", iJOE International Journal of Online Engineering vol. 5, no. Special Issue, pp. 18-24 1861-2121, August 2009

[7] Tzafestas, C.S., Palaiologou, N., Alifragis, M., “Virtual and Remote. Robotic Laboratory: Comparative Experimental Evaluation”, IEEE. Transactions on Education, vol. 49, no. 3, p.360-369, August 2006 http://dx.doi.org/10.1109/TE.2006.879255

[8] Gravier. C, Fayolle. J, Bayard. B, Ates .M, and Lardon .J, “State of the Art About Remote Laboratories Paradigms—Foundations of Ongoing Mutations DIOM Laboratory,” iJOE, vol. 4, no. 1, pp. 18-25, Feb. 2008.

[9] Olmi, C., Cao, B., Chen, X. and Song, G., "A Unified Framework for Remote Laboratory Experiments", In Proceedings of ASEE Annual Conference & Exposition, Vancouver, BC, Canada, June 26 - 29, 2011.

[10] Chen, X., Osakue, D., Wang, N., Parsaei, H., Song, G., “Development of a Remote Experiment under a Unified Remote Laboratory Framework” in Proceedings of the World Congress on Engineering Education 2013. H.R. Parsaei and K.S. Warraich, eds.

[11] Omli, C., Chen, X. and Song, G., "A Framework for Developing Scalable Remote Experiment Laboratory", In Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2011 (E-Learn 2011), pp. 2045-2050, Honolulu, Hawaii, October 18-21, 2011.

[12] Foresman, Chris., (July 9, 2009). "Apple proposes HTTP streaming feature as protocol standard". Ars Technica. Retrieved 2009-07-10.

[13] Roger, Pantos (Apple Inc.), William May, Jr (Apple Inc.) "the IETF Internet-Draft of the HTTP Live Streaming specification. " Internet Engineering Task Force, Online, Retrieved October, 2012

[14] McDonald, Carson “An open source segmenter– written by Carson McDonald” Subversion (SVN) online, Submitted: June 28, 2009.

AUTHORS

Ning Wang is with Department of Computer Science, Texas Southern University, Houston TX, 77004 USA (n.wang4648@student.tsu.edu).

Xuemin Chen. is with Department of Engineering Technology, Texas Southern University, Houston TX 77004, USA (e-mail: chenxm@tsu.edu).

Gangbing Song is with Department of of Mechanical Engineering, University of Houston, Houston TX, USA (e-mail: gsong@uh.edu).

Hamid Parsaei is with Department of Mechanical Engineering, Texas A&M University at Qatar, Doha, Qatar (e-mail: hamid.parsaei@qatar.tamu.edu).

This material is based upon work supported by the Qatar National Research Fund under Grant No. NPRP 4-892-2-335.Submitted 06 September 2014. Published as resubmitted by the authors 25 January 2015.