

Pseudo-Streaming to PC and iPhone/iPad Clients for Video Surveillance System

Phuong Nguyen Bui*, Hiroshi Yamamoto*, Katsuyuki Yamazaki *

*Electric Electronic Information Department, Nagaoka University of Technology, 1603-1 Nagaoka Niiagata Japan
yuanfang@stn.nagaokaut.ac.jp

Abstract— So far video surveillance systems can only stream video data inside a firewall/NAT, which limits the video access to users inside the firewall/NAT and prevents other users on the Internet from reaching these videos. In this paper, we propose a new video streaming system which provides pseudo-streaming of video through firewall using HTTPS. In this system, a camera deployed inside a firewall/NAT records and sends video files to a video server placed on the Internet using HTTPS which is allowed by every firewall. Then, the server saves the files to the storage device and writes the file paths into the database. When a user requests watching those files, the server transcodes the original file into an appropriate format that corresponds to the user's device and bandwidth. All what users need to watch these videos is a browser of PC or iPhone/iPad, this enables non-IT people to use this system easily and effectively. By this means, recorded video files are sequentially sent to clients' terminals (i.e., PC and iPhone/iPad), creating a live pseudo-streaming experience.

Keywords— HTTP, live streaming, security, video surveillance, iPhone/iPad

I. INTRODUCTION

Video surveillance systems are being used widely. Especially those that can be used to observe factories, hospitals, offices and private houses from distance in real time are highly noticed. However, existing video surveillance systems have problems such as high price, difficulty in installing applications on clients to view video files, tricky setting for normal users without ICT expertise. Furthermore, because cameras are deployed inside a firewall/NAT, they cannot send data to users on the Internet. Moreover, according to a recent study [5], data traffic of smartphones and tablets is predicted to exceed 58.3% of the entire global data traffic in 2016, which means future users will not only use their PCs but also use their smartphones and tablets to view recorded videos at anywhere in anytime.

Since current video surveillance systems on the market have not been able to fulfill the above problems and demands, the objectives of this research are: to develop a method to stream video from LAN to the Internet by traversing firewall/NAT and to enable users to view recorded videos on smartphone and tablets devices.

II. RELATED WORKS

A. Existing Video Surveillance Systems

Video surveillance systems are being made by many manufactures like Panasonic, Bosch, Sony, Logitech, Levana, Lorex, etc. Bosch, Panasonic and Sony cameras can send H264/MPEG4 videos or JPEG pictures to its owner. Some cameras have functions such as face and motion detection, zooming, alarm, etc. Lenava and Lorex have the baby care, pet care camera systems which use wifi to transmit video data from camera to monitor in only a small area. Even though different in names, these cameras share the same video transmitting method: the video servers (a camera itself may be considered as a video server) are deployed inside firewall/NAT, which prevents clients on the Internet from watching the recorded videos.

Logitech security cameras, on the other hands, send recorded videos to servers placed on the Internet, so the clients can use PC or handheld devices to watch these videos. However, since the cameras use a unique protocol instead of HTTPS to send video data, data transmission can be blocked by strict firewall settings.

B. Problems of the Existing Systems

In general, existing camera systems on market (Panasonic, Bosch, Sony, Logitech, etc.) have the following problems: using RTP (Real Time Protocol) or RSTP (Real Time Streaming Protocol) to send video data [1] and requiring player applications using Flash or Java installed on the clients to watch those videos. In detail, when streaming video from camera devices deployed on LAN of a company or a university network to the Internet, a firewall/NAT should be appropriately configured so that RTP packets can be sent out. For normal users who do not have expertise on information and communication technology, configuring the firewall to allow RTP packages or installing the player applications can be very challenging.

In addition, along with the widespread of smartphones (e.g., iPhone) and tablets (e.g., iPad), more and more people want to view the streamed video on these devices, which has not been deployed on the current video surveillance systems.

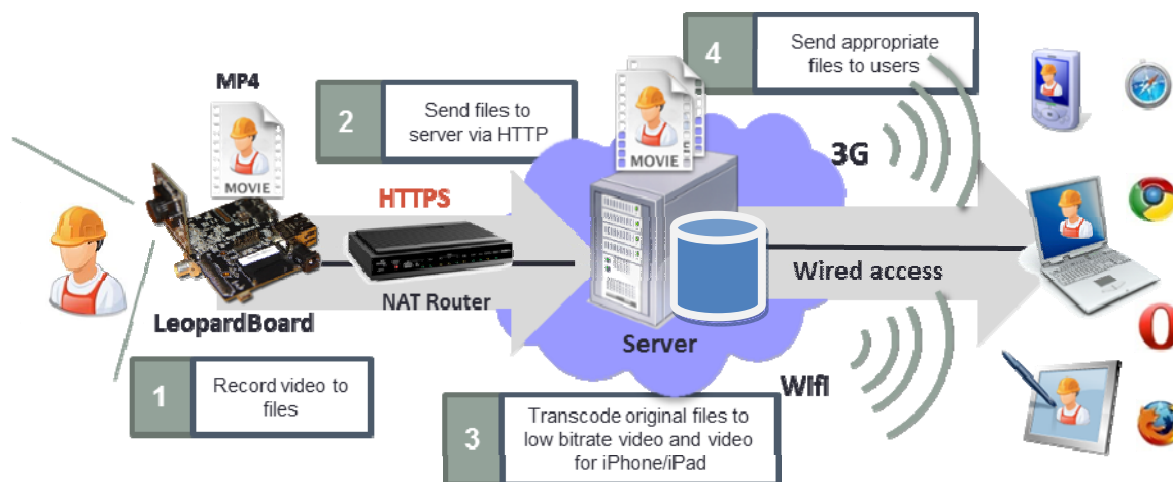


Figure 1. Overview of the proposed system

III. PROPOSED SYSTEM

A. An Overview of the Proposed System

In order to solve the above problems, we propose a new video streaming system shown in Figure 1.

Firstly, a camera (placed in factory, office, hospital, etc.) records a 5 seconds video and saves it to an MP4 file (H.264/AVC+AAC+MP4), which can be played by any browser (e.g., Internet Explorer, Firefox, Google Chrome, Safari, etc.). In order to achieve high performance at ultra-low cost, we use Leopard Board as the camera device. Leopard Board is an embedded system which is open source, small form factor and designed to process video effectively. In addition, HTTPS, a secure protocol widely accepted by all firewall, is used to send recorded video files from the Leopard Board to the video server.

After receiving the video files from the camera, the video server saves the files to storage devices and writes the file paths into the database for later access. It also creates thumbnail pictures of the videos for the users to look over the video content before downloading.

When the users demand on accessing those files, the server sends a small sample file and measures the downloading speed to detect whether the users are connecting to fast or slow networks. The server also reads the HTTP header of the request to acquire what types of device are being used. Then, the video server transcodes (converts) the original video files into an appropriate file format for both the clients' devices and the connected networks.

There are 4 types of video files. The first one is a high quality MP4 file for playing on PCs which connect to the high speed networks (broadband). The second one is another MP4 file with low quality, prepared for PC on slow networks (3G, modem). There is also an MPEG-TS file for streaming video to iPhone/iPad connecting to wifi, and another MPEG-TS file for iPhone/iPad device connecting to 3G (see Table 2).

In the next section, we will explain streaming method for PC and that for iPhone/iPad in detail.

B. Streaming Methods

1) *Pseudo-Streaming to PC*: As shown in Figure 2, pseudo-streaming to PC can be achieved simply by means of placing a new video content with the same file name on the server every 5 seconds, while the browser on the device automatically refreshes every 5 seconds to re-download and auto-play that video file. When the request is sent to the server, firstly, the server checks the network speed to determine which file type it should transcode. Then, it transcodes the original files into high quality video to send through broadband, or low quality video with small file size to send through narrowband (3G). As the result, when the user clicks a play button, a 5 seconds video is loaded and played on the device. After 5 seconds when it finished, the browser automatically refreshes, downloads new video, and plays again.



Figure 2. PC streaming method

2) *Pseudo-Streaming to iPhone/iPad*: The pseudo streaming method described in Section III.B.1 cannot be used for iPhone/iPad browser (Safari on iOS) because automatic refresh function has been disabled by Apple to avoid over-downloading in 3G cellular network. Therefore, we developed to utilize HTTP live streaming protocol [2] [4] prepared by Apple to stream video to iPhone/iPad. In the HTTP live streaming protocol (Figure 3), whenever the server receives

request for video file, it transcodes that video file into MPEG-TS file (.ts) and prepares a playlist file (.m3u8) which contains URLs of the encoded MPEG-TS files. Depending on whether the user is connecting to broadband network or 3G, the server transcodes the original video to the appropriate TS file. When the user taps “PLAY” button on the browser of iPhone/iPad, the browser downloads the playlist file from the video server, accesses the URLs and plays the video files. After finishing playing the video files, the browser accesses to the server again to read the updated playlist file and downloads new video files to play. If, by some reasons, the browser reads the playlist but finds no new video files, it waits for a certain amount of time before retrying. This delay is 0.5[s] for the first attempt, 1.5[s] for the second, and 3.0[s] thereafter.

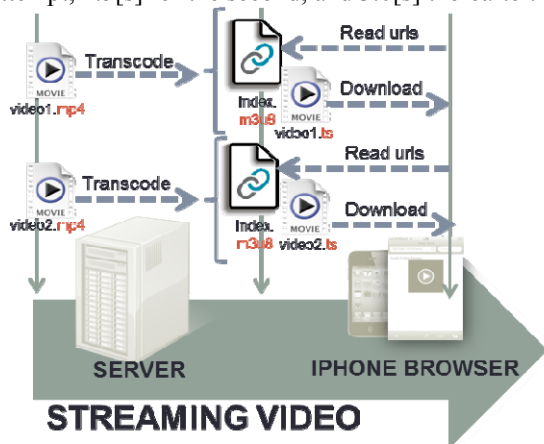


Figure 3. iPhone/iPad streaming method

C. Database

We install the database into our system to manage a large number of video files and to enable the user search for the requested file effectively. For the experimental evaluation, we design the system to work with 100 camera boards, so the database management system (DMS) should be fast enough to continuously process video files received from 100 camera boards. Besides, the DMS will periodically delete old video files which often cause the overhead problem for DMS and makes the system to slowdown. For that reason, we decided not to choose complicated DMS which is big, complex and has many unnecessary functions such as MySQL. We choose SQLite for its small, tidy and fast.

Figure 4 is the database structure of the system. The top main_tbl table manages camera board number, place and its related table. For each camera board in main_tbl table, there is LBXX_tbl table which is used to manage the metadata of each video sent by its camera board.

Figure 5 is the metadata information of videos written in the LBXX_tbl table. The first column is for video recorded time, the second is for video file saved path, the third is for video thumbnail picture saved path and the last one is prepared for storing data generated by extended features in the future.

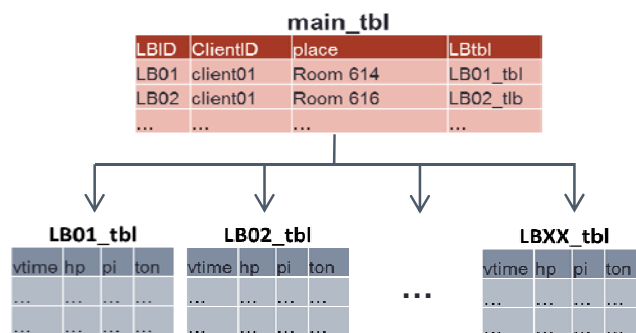


Figure 4. Database structure

vtime	hp	pi	ton
20120925_151505	Video/save/path	Picture/save/path	normal
20120925_151510	Video/save/path	Picture/save/path	scream
...	...	...	...

Figure 5. LBXX_tbl

IV. EVALUATION OF THE PROPOSED SYSTEM

A. Transcoding Performance

1) *Transcoding to PC-viewed format:* In order to maintain pseudo-streaming state, file transcoding speed has to be kept as high as possible. In current stage, as a 5-seconds-video-file is sent from the Leopard Board to the video server, excluded uploading/downloading time, the remaining time is less than 3 seconds for the transcoding process to take place. Therefore, we have evaluated video transcoding time to determine which transcoding format is the fastest and the most optimal. Transcoding process was run on the video server with its specification shown in Table 1. We use VLC[6] to transcode video format because it is a professional tool designed for video file processing.

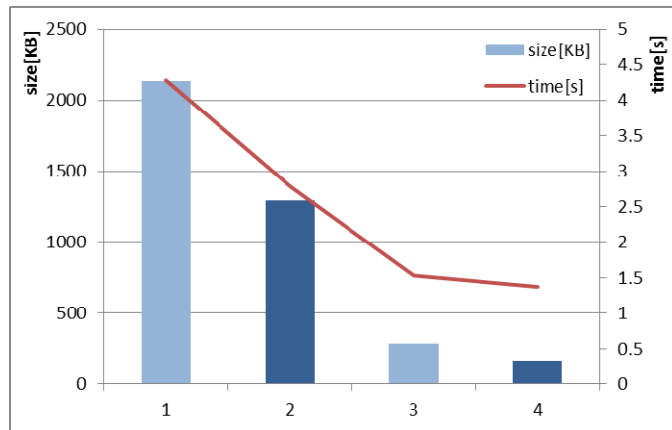
TABLE 1. VIDEO SERVER SPECIFICATION

Model	ProLiant DL120 G5
OS	Ubuntu 10.04
Processor	Intel Pentium Dual CPU E2160 1.80GHz
Memory	2GiB DDR2

For transcoding, we use H264/MPEG4 AVC which is a standard for video compression, and is currently one of the most commonly used formats for recording, compression, and distribution of high definition video. When compressing raw video into a video file, H264 marks a video frame as an anchor frame (or a frame to be compared with) and compares it with the other later frames to search for any difference between these frames in pixels. Only the anchor frames and different pixels between frames are recorded. Therefore, the number and quality of the anchor frames, and the complexity of the algorithm, which is used to calculate the difference between frames, will decide the transcoding time and video file size. [3]

Figure 6 shows the measurement result of the transcoding time. The bar is file size and the line is file transcoding time. A 320x240 frame is smaller than a 640x480 frame, and 15 fps

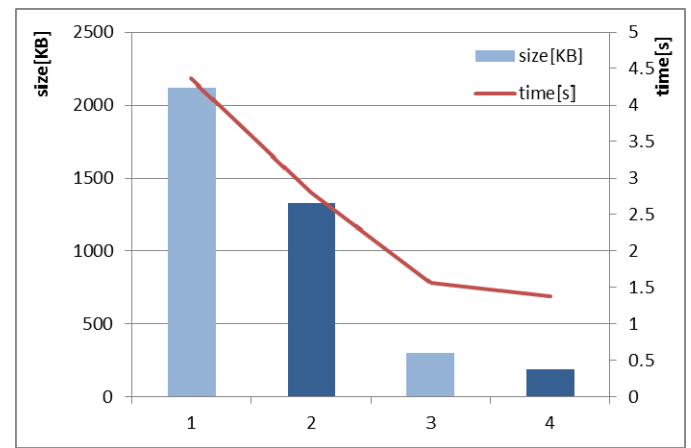
(frames per second) is less than 30 fps. As the frame becomes smaller and the fps becomes less, the number of pixel changed between frames is less and calculating speed becomes higher. All videos are encoded by using only I-frames to simplify the calculation process. Finally, among these 4 transcoding patterns, number 2 (size = 640x480, fps = 15) and number 4 (size = 320x240, fps = 15) are decided to be the best suited for our purpose. The 1.04MB high quality video for PC takes 2.4[s] to transcode and the 201KB low quality video for PC takes 0.25[s] to transcode. The high quality one is optimized for broadband and the low quality one can be watched smoothly in 3G. (see Table 2)



- 1 size = 640x480, fps=30
 - 2 size = 640x480, fps=15
 - 3 size = 320x240, fps=30
 - 4 size = 320x240, fps=15
- Figure 6.** PC Files Transcoding Performance

2) Transcoding to iPhone/iPad viewed format: As mentioned in Section III.B.2), in order for the iPhone/iPad browser to play video file, the transcoding process has to create a playlist file (M3U8 file) and media files (MPEG-TS files). When receiving a request from the user, the server starts VLC to convert MP4 files into TS files, then the code in the playlist file is regenerated with the new media file's URL. Depending on which network the users are on, the system generates the high quality video or the low quality one.

As shown in Figure 7, the transcoding time and file size is very alike to Figure 6. This is because they use the same codec (H264) and transcoding formats. Only the file container (MPEG4, MPEG-TS) is different. Thus, format number 2 and number 4 in iPhone/iPad transcoding are also the best ones.



- 1 size = 640x480, fps=30
 - 2 size = 640x480, fps=15
 - 3 size = 320x240, fps=30
 - 4 size = 320x240, fps=15
- Figure 7.** iPhone/iPad Files Transcoding Performance

TABLE 2. VIDEO FILE SPECIFICATION

	Original File	High Quality File for PC	Low Quality File for PC	High Quality File for iPhone/iPad	Low Quality File for iPhone/iPad
File Size [KB]	3760	1040	201	819	175
Format	MPEG-4	MPEG-4	MPEG-4	MPEG-TS	MPEG-TS
CODEC	AVC	AVC	AVC	AVC	AVC
Frame Size	640x480	640x480	320x204	640x480	320x240
Bitrate [Kbps]	5912	1350	250	1329	274
Frames/Second	30	15	15	15	15

B. Real-time Streaming Evaluation

To further optimize our system, not only transcoding process needs shortening but other processes do as well. Hence, we have measured the whole system processing time in real-time and got the result as below (Fig. 8). In Figure 8, three videos are continually being recorded, transcoded to high quality files and streamed to iPhone/iPad. Video processing steps include recording initialization, recording and creating a video file on Leopard Board, uploading to the server, transcoding that file on the server, and loading file to users device (iPhone/iPad). In these steps, uploading and downloading process depend on the file size and the bandwidth/congestion of the network. The time of initializing

and creating file process can be shortened by optimizing code and reducing redundant steps.

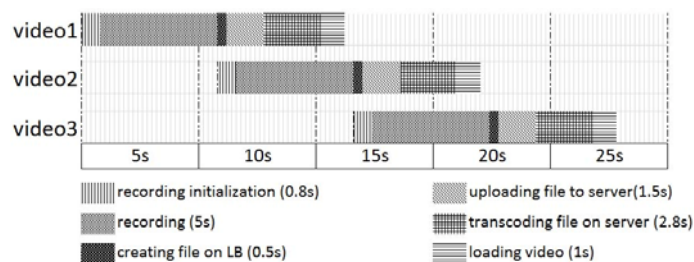


Figure 8. System processing time

C. Database performance

We evaluate our database performance by measuring database writing speed (unit: $\mu\text{s}/\text{line}$) when the server has no load and when it is transcoding files. As the result, database's writing speed when the server has no load is 36.0 [$\mu\text{s}/\text{line}$], and database writing speed while transcoding is 407 [$\mu\text{s}/\text{line}$]. This result shows that the DMS can process video files continuously received from 100 camera boards.

V. CONCLUSIONS

In this research, we have proposed a video pseudo-streaming system which uses HTTPS to stream video from the camera device deployed inside firewall/NAT to user terminals connecting to the Internet. PCs' browser can play these streaming video files by re-download and replay the video file every 5 seconds. iPhone/iPad browser, on the other hand, uses HTTP live streaming protocol to play video.

This system can stream video in a way that helps users to watch recorded video files at anywhere in anytime using their

browsers without changing the configuration of firewall/NAT router or installing any additional player application.

However, it still contains problems that need further improving. First, since video processing time is still rather long, there are problems such as silence gaps between videos or the same video file being played several times. Moreover, because recorded videos on Leopard Board are big (~3.6 MB/file), up to this stage of research, Leopard Board and server should be put in the same LAN to achieve the best streaming environment.

Therefore, in future study, we will focus on reducing processing time and speed up entire system so as to achieve the exact pseudo streaming through the Internet.

ACKNOWLEDGMENT

Our thanks to Mr. Watanabe and Mr. Ishizuki of BM&W Co. for helping us with this research.

REFERENCES

- [1] Stockhammer, T. 2011, "Dynamic adaptive streaming over HTTP -- standards and design principles. Proceedings of the second annual ACM conference on Multimedia systems", MMSys '11, ACM Press, New York, NY, 133-144. Feb. 2011, Available: <http://doi.acm.org/10.1145/1943552.1943572>
- [2] May, W. and Pantos, R. 2011 *HTTP Live Streaming draft-pantos-http-live-streaming-06* Apple Inc. <http://tools.ietf.org/html/draft-pantos-http-live-streaming-06#section-6.3>
- [3] Poynton, C. 2007 *Digital video and HDTV algorithms and interfaces*. Morgan Kaufmann publishers
- [4] Fecheyr-Lippens, A. 2010 *A Review of HTTP Live Streaming*. http://files.andrewsblog.org/http_live_streaming.pdf
- [5] Cisco White Paper: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2009-2016, http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-520862.html
- [6] VideoLAN - Official page for VLC media player, the Open Source video framework! Available: <http://www.videolan.org/vlc/>