A Flexible Architecture of Real-Time Audio Transmission to Heterogeneous Devices for Surveillance System

Yu-Tung Cheng, Heng-An Lin, Yun-Jaw Yeh
Telecom Laboratories
Chunghwa Telecom Co., Ltd.
Taoyuan, Taiwan, ROC
ytcheng, heng_an, jackey@cht.com.tw

***Abstract*—This paper presents a software architecture for audio transmission to network cameras while integrating multiple types of cameras from different manufacturers under a web-based surveillance system. Adding audio streams in surveillance system will enhance the capabilities for remote monitoring. This architecture offers a plug-and-play service of audio transmission to cameras over the network for numerous users. It not only simplifies but also facilitates the integration process of heterogeneous devices.**

Keywords—Surveillance system; audio transmission; network camera

I. INTRODUCTION

With the widespread use of the Internet and the advances of information technologies, web-based systems for delivering information or services become more and more prevalent. Anyone at anytime from anywhere could access such systems via a web browser over the Internet or other network connections. No need to install software packages locally or worry about whether the latest version. A surveillance system for remote management is no exception. A web-based remote security system model was discussed in [1].

Behind a web-based surveillance system, a recording and streaming server is required to receive video streams from network cameras, do recording jobs and deliver streams to clients. The existence of server can overcome the constraint on maximum number of streams retrieving from the same network camera simultaneously. Moreover, the bid of government project for intersection surveillance usually has to integrate the original cameras due to cost considerations or the need to have different functional network cameras in response to different environmental conditions. As a result, supporting different types of network cameras from different manufacturers is necessary. Topic about architecture to support heterogeneous devices for video surveillance system had been addressed [2].

However, the aforementioned system pays no attention to the value of audio in surveillance. With audio supported network cameras, two directional audio streams provide the abilities to listen and talk respectively over the network. The former is audio transmitted from cameras and the latter is audio transmitted to cameras. By audio streaming from cameras, sounds captured in remote sites can be utilized to monitor anomalous sound level, namely, trigger an alarm when sound level reaches the pre-defined specification. What's more, received audio could be applied to audio recognition, such as scream and gunshot detection and localization [3], then steer PTZ (Pan/Tilt/Zoom) cameras to the source of audio for recording, and notify the police of a crime, etc. By audio streaming to cameras, the administrator could speak to people around cameras which are attached with speakers to issue some warnings or broadcast public announcement and so on. A combination of both audio streaming directions is so-called two-way audio communication. A novel method for two-way audio-video communication was introduced [4].

We try to add audio streams in the existing surveillance system. Audio stream from a device can be retrieved easily over RTP (Real Time Transport Protocol)/ RTSP (Real Time Streaming Protocol) [5] on its own or accompanied by video stream. RTSP is a standard protocol for streaming control and RTP is in charge of media transmission over the transport layer. If a camera doesn't been implemented as a RTSP server, the developer SDK (Software Development Kit) provided by manufacturer is required for stream retrieval. With regard to audio stream to one device and even several heterogeneous devices in surveillance system will be the main topic in this paper.

This paper is organized as follows. Section II discusses audio transmission to devices under manufacturer solutions. Section III describes sending audio under the proposed surveillance system architecture. We will address a framework to integrate heterogeneous devices with flexibility. Finally, conclusions of this paper are drawn.

II. AUDIO TRANSMISSION TO DEVICES UNDER MANUFACTURER SOLUTIONS

A network camera with a built-in web server makes us access it remotely over the network. Through an Internet Explorer web browser, the ActiveX component is automatically downloaded when you connect to the network camera for the first time. The ActiveX component provided by device manufacturer is used for the playing of real-time video or audio. Sending audio to an audio-supported device is usually

allowed. In addition to the web browser, another convenient user interface of transferring audio data is remote software developed by camera manufacturer. The remote software lets administrator manage several cameras at the same time. Viewing, monitoring and recording are often available. Two-way audio communication may be provided under the premise of a device with audio capability.

No matter which user interface is used, the core works of audio transmission to devices is similar. It consists of audio capture module、audio encoding module and audio transfer module. Audio capture module needs an audio input device which is attached to a computer, such as a microphone, to capture analog audio sound and convert it into digital data. Then encode the captured audio PCM (Pulse-Code Modulation) data to the compressed one according to the specified audio encoding algorithm which the network camera supports. Next transfer the compressed audio data to the device and it must follow the specified audio transmission protocol implemented by the network camera. The process and modules of audio transmission from a client to one device is displayed in Figure 1. The arrows indicate the audio stream direction.

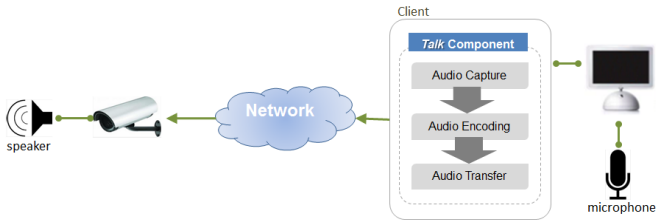

Figure 1. Audio transmission to one device using manufacturer solution.

Despite the fact that the web client and the remote software are solutions of audio stream to one device, it's annoying to satisfy the requirement of heterogeneous devices integration. Since every device manufacturer has their own specific client solution and isn't compatible with others. Using different web clients or remote software for audio transmission to different devices is tedious. This situation is plotted as below in Figure 2.

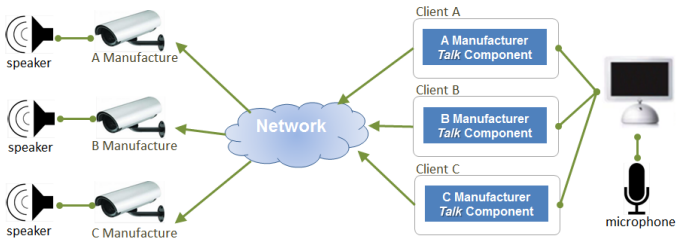

Figure 2. Audio transmission to heterogeneous devices using manufacturer solutions.

III. AUDIO TRANSMISSION TO DEVICES UNDER SURVEILLANCE SYSTEM

When heterogeneous devices require to be considered, efforts should be put on a unified portal and a flexible mechanism for audio transmission. A unified portal is to solve client inconvenience. A flexible mechanism is to relieve the burden of developers. So we will propose an architecture for audio transmission with the integration of heterogeneous devices under a web-based surveillance system.

General speaking, certain firewall environment and different network segments is common in a large-scaled surveillance system for the sake of security. We can't transmit audio from a client to any devices directly. Hence one or more relay servers are required. The overall architecture is shown in Figure 3.

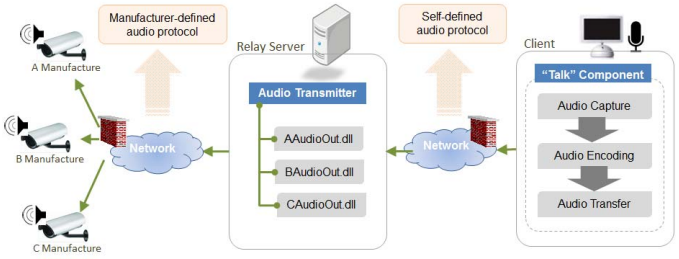

Figure 3. Audio transmission to heterogeneous devices with a web-based surveillance system.

A. The Client Design

On the client side, a "talk" component wrapped in the form of ActiveX is developed to be responsible for audio capture, audio encoding, and audio transfer. These three modules remain unchanged as before. It provides the identical interface to handle audio transmission for all the heterogeneous devices. It's worth noting that audio streams between clients and a relay server are transmitted by the self-defined protocol in audio transfer module.

The packet header format in Figure 4 is defined so as to determine what kinds of information transmitted and the amount of information received. Each packet header starts with an "&" sign for marking the beginning of packet. It follows by one byte packet type listed in Figure 5 and then two bytes on behalf of the length of each packet data.

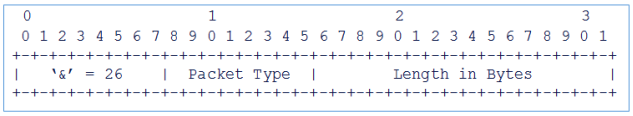

Figure 4. Packet header format.

abbrev.	name
REQ	request
ADO	audio data
BYE	goodbye
BUSY	device busy
ERR	error

Figure 5. Packet types.

- REQ：A request message from a client to a server is composed of JSON (JavaScript Object Notation) format, including authentication code and device information array. The device array is all of the devices you want to talk to. That means audio broadcast to multiple cameras is possible in such a structure. The device information comprises device IP, model ID for distinguishing brand model, codec type, account and password, etc. The model ID is a key parameter in deciding an algorithm of audio transmission protocol involving runtime dynamic linking on the server side.
- ADO：After a request packet is sent successfully, the client will prepare a continuous flow of compressed audio data and transmit to the server.
- BYE：Indicates the end of audio transmission while the client stops talking.
- BUSY：A server always holds a device busy list. If the specified device audio out is occupied, this packet will be sent to the requesting client.
- ERR：When error happened, this packet will be sent, for example DLL load failed.

B. The Server Design

A relay server is a listen server which allows clients to connect to it. It receives packets defined above, parses them and does the corresponding processing. The main functionality is to receive the audio streams from clients and then forward to the specified devices. Later we will work on encapsulating a specific audio transmission protocol to a plugin-like module mounted automatically for a relay server.

In object oriented programming, open-closed principle [6], which means open for extension but closed for modification, is emphasized for easy maintenance. We should extract variable features and attempt to isolate from others. In this case, the variation mainly comes from the newly integrated devices. Each device manufacturer has its own specific transmission protocol for transferring an audio stream to one device. It's reasonable to divide the whole program into server and each individual device. The server code keeps unchanged as much as possible for closed principle, whereas adding a standalone module for the specific audio transmission of a new device brings about easy extension.

With further details, encapsulating each algorithm makes them be selected at runtime and interchangeable easily is the essence of strategy pattern. From this point of view, an interface class is a channel of communication for a relay server and device audio out classes. Each audio transmission protocol for each heterogeneous camera device must be a concrete class implementing the related operations. For instance, the `IAudioOut` is an interface class which defines common behaviors that all audio transmission algorithm supports. Please see the following diagram in Figure 6. We ignore the parameters of `PutAudioSample()` which should be an audio data block for simplicity. The strategy classes `AAudioOut`, `BAudioOut`, and `CAudioOut` implement their specific behaviors for their own audio transmission protocol. The relay server is only aware of `IAudioOut` interface class and doesn't care about the concrete object.

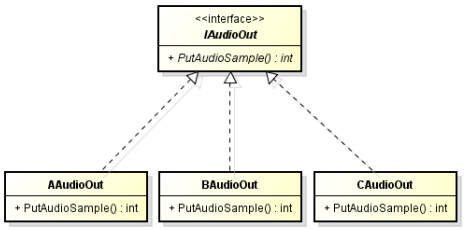

Figure 6. A class diagram of encapsulating different audio transmission algorithms.

However, a relay server inevitably involves the creation of an audio out class using decision making statements to deal with a multi-way branch, like if-else or switch-case structures in Figure 7. The model ID information can be acquired from the REQ packet representing a specific type of camera. In this example, the user-defined model ID 101, 102 and 103 represent audio transmission algorithms for manufacturer A, B and C respectively. Depending on which value in model ID, the corresponding audio out instance will be created.

```
IAudioOut* pAudioOut = NULL;

switch(nModelID)
{
  case '101':
    pAudioOut = new AAudioOut();
    break;
  case '102':
    pAudioOut = new BAudioOut();
    break;
  case '103':
    pAudioOut = new CAudioOut();
    break;
}
```

Figure 7. A switch selection statement for audio transmission algorithms.

Adding a newly integrated device class will lead to modify the code of decision making. And project rebuilding is required. In order to decouple strategy classes from the relay server that uses them, two actions could be taken. One is to use a factory class of creational pattern to encapsulate the source code of object creation. Putting object creation process together makes maintenance easier and less bug prone. Figure 8 shows the role of the simple factory in class diagram. The other is to make program modular.

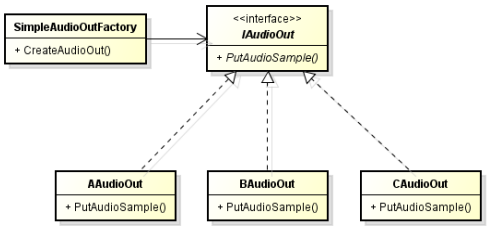

Figure 8. A class diagram of adding a factory class.

Here DLLs (Dynamic-Link Library) are introduced to encapsulate each device audio out module for the purpose of runtime dynamic linking. Implementing each audio out class that inherits from an abstract interface `IAudioOut` is the same as usual. The difference is that we export a factory function that creates an instance of a subclass audio out in DLL shown below.

```
class IAudioOut
{
  public:
    virtual int Initiate(char* pszIP, char* pszUser, char* pszPwd) = 0;
    virtual int PutAudioSample(char* pSample, int iSize) = 0;
    virtual void Release() = 0;
};

extern "C" AUDIOOUT_API IAudioOut* CreateAudioOut();
```

Figure 9. The C++ code snippet of the header file in DLL.

The Windows API `LoadLibrary()` can load a DLL module for the calling process by specifying the file name of the DLL. There is a one-to-one relationship between a model ID and its corresponding DLL name. Thus we create an INI configuration file to record them. In this way the factory class in a relay server would load a DLL by referring to the model ID in the configuration file, retrieve the address of the exported factory function to get the instance of the subclass via `GetProcAddress()`, call the class member functions in it, and do the audio transmission job. As long as we have the need to integrate a new audio transmission algorithm of a newly added device, just inherit from `IAudioOut` and write it in a DLL. A relay server can load algorithms dynamically from DLLs, and plugin different modules as required.

Besides plug architecture, a relay server also has other characteristics described as follows.

- Broadcast: The relay server is capable of audio broadcast to multiple devices at the same time. It's useful to warn someone invading restricted areas.
- Device access control: The server maintains a device busy list. If a client attempts to transfer an audio stream to an audio out occupied device, the request will be rejected.
- Scalability: Sometimes more than one layer of relay server is needed under certain network environment. The first layer of server receives packets from clients and just relays them to the second layer of server. A DLL `RelayOut` which realizes the `IAudioOut` interface takes the responsibility of transferring packets to another server. It's easy to scale out under this architecture. Figure 10 illustrates the overall view about a scalable architecture.

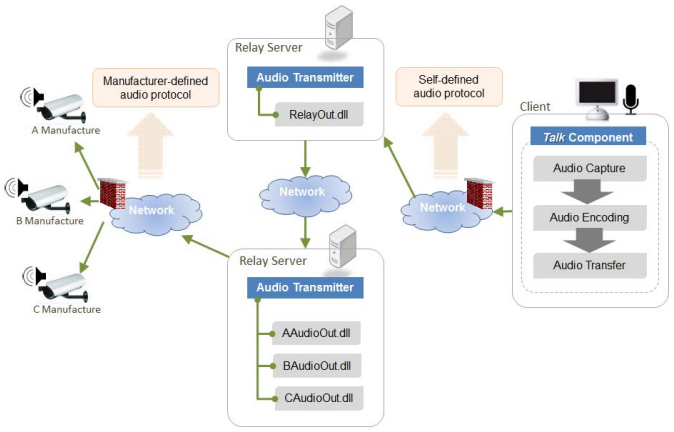

Figure 10. A scalable architecture of audio transmission for heterogeneous devices with a web-based surveillance system.

IV. CONCLUSION

As a result of the requirement of heterogeneous devices integration in large-scaled surveillance system, we demonstrated a flexible and scalable architecture to facilitate the integration process applied to audio transmission to devices.

On the client side, it's beneficial for users to have a unified portal of surveillance system while facing various network cameras. A "talk" component including audio capture, audio encoding and audio transfer module had been implemented. On the server side, strategy design pattern and DLL runtime dynamic linking work together that makes different algorithms corresponding to different audio transmission protocols become plugin-able and scalable.

Additionally, audio transmission to devices should be under access control with high level security in case of misuse which triggers the disturbances. One-time password would be a more secure authentication code which deserves to be considered.

REFERENCES

[1] Wunnava, S.V. and De La Cruz, M., "Web based remote security system (WRSS) model development," Southeastcon 2000. Proceedings of the IEEE, pp. 379-382, 2000.

[2] Mikki, M.A., Kyungroul Lee, Wansoo Kim and Kangbin Yim, "Architecture to support heterogeneous devices for video surveillance system," Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on, pp. 16-19, 2011.

[3] Valenzise, G., Gerosa, L., Tagliasacchi, M., Antonacci, E. and Sarti, A., "Scream and gunshot detection and localization for audio-surveillance systems," Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on, pp. 21-26, 2007.

[4] Kamat, S.P. and Behera, T.R., "A novel method for two way audio video communication," Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on, pp. 943-946, 2009.

[5] Real Time Streaming Protocol (RTSP) RFC 2326.

[6] Bertrand Meyer, Object-Oriented Software Construction. Prentice Hall, 1988.