Journal of Software Engineering and Applications, 2014, 7, 975-982
Published Online November 2014 in SciRes. http://www.scirp.org/journal/jsea
http://dx.doi.org/10.4236/jsea.2014.712085

Scientific Research

Applications of a Streaming Video Server in a Mobile Phone Live Streaming System

Chongyu Wei*, Honglin Zhang

Qingdao University of Science and Technology, Qingdao, China
Email: *qustwcy@163.com

Received 30 August 2014; revised 25 September 2014; accepted 21 October 2014

Abstract

This paper describes the entire process of completing a multicast streaming media server and applying it to a cellphone live streaming system. By using the RTSP protocol, the streaming media server controls the connection of video capture client, data stream reception and processing as well as playback interaction. The media server can publish real-time data from a data acquisition terminal or a historical data file in the server to the Web so that users on the network can view all the videos through a variety of terminals anytime and anywhere, without having to wait for all the data downloaded completely. The streaming media server of the system is developed based on an open-source streaming media library Live555. The developed server can run on a Windows or a Linux system. The standard RTSP/RTP protocol is used and the video media format is H264. The paper mainly introduces the design of a streaming media server, including data processing for real time, designing a data source, the implementation of multi-acquisition end and multi-player operation, RTSP interaction and RTP packaging, and the setting up of the data buffer in the server. Experiment results are given to show the effect of the system implementation.

Keywords

Live Streaming, Streaming Media, Live555, RTSP, RTP, H264

1. Introduction

The ever-increasing data rate in wireless network provides a good condition in video transmission, and the video-delivery technology and its applications in wide band radio network have been widely researched [1]-[3]. The cellphone live streaming system uses mobile phones to complete video data acquisition. Since the widespread use of smart phones, the video real-time shooting and displaying become simple, low cost and suitable

*Corresponding author.

How to cite this paper: Wei, C.Y. and Zhang, H.L. (2014) Applications of a Streaming Video Server in a Mobile Phone Live Streaming System. *Journal of Software Engineering and Applications*, **7**, 975-982.
http://dx.doi.org/10.4236/jsea.2014.712085

for general public use.

The core technology of a cellphone live streaming system is the streaming media technology. The key platform in the system is a streaming media server, whose function is selecting the media data from a mobile phone, storing them in a data base, packing and transferring the data to a user terminal for play-back. This paper describes the system design and implementation.

2. The Integrated Design of the Cellphone Live Streaming System

The live system, using the C/S architecture, can be divided into two parts of the server and client subsyem as shown in Figure 1. A client may be a smartphone or a personal computer. A smartphone can be used as a video data acquisition front-end and a play back end respectively, while a PC can only be a play back terminal. The server subsystem is comprised of three servers, a streaming server, a Web and a data base. The Web server designed using the B/S three-tier architecture is used for Keepping the server side and a client connected, releasing resources to the client, providing an operator an access entrance. The working principle of the interaction between a client and the server-side is shown in Figure 2.

3. Design and Implementation of Streaming Media Server

The streaming server is implemented by transplanting the Live555 open-source library. Since the data transmission in Live55 is based on the file on demand, the data structure, abstract data source and their interface must be modified for streaming media use. The pertinent data structure in the streaming server modified is given in Figure 3.

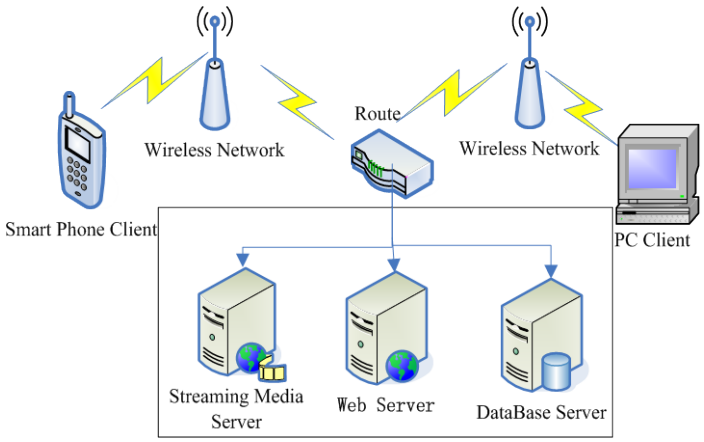

Figure 1. Block diagram of figure the cellphone live streaming system.

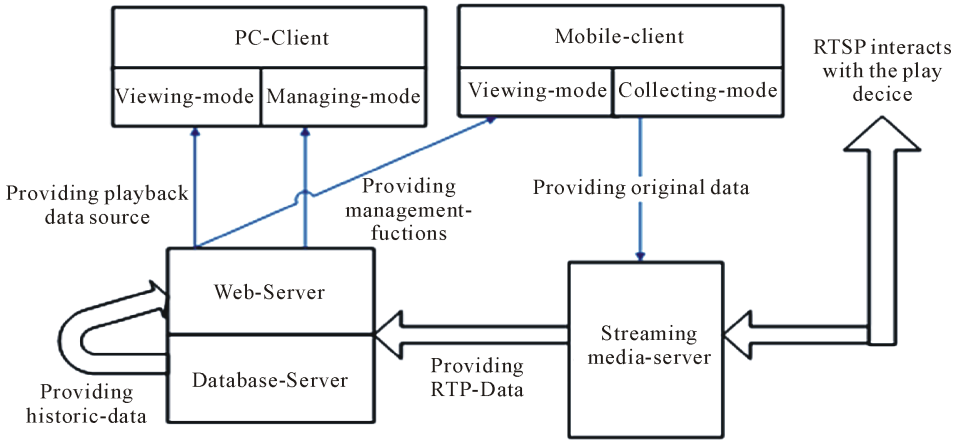

Figure 2. The operation principle of the server subsystem.

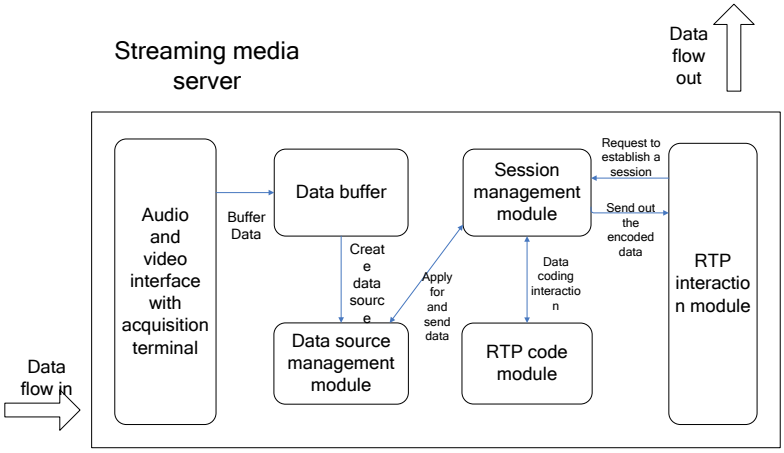

Figure 3. Streaming media server internal module structure.

The coded video data transmitted by UDP (user datagram protocol) on to Live555 will be first received by the server from a specified port and save them in a data buffer. Then, a corresponding abstract data source is set up in a data management modul. Later perations on the saved data in the server will all be through the source. When a play end wants to connect the server, the RTSP interaction modul will first sends some messages of request. After parsing and certification, session management modul will be requested to set up a new session and a sink for the new data consumer. Then, later operations on transmitting data in the server will all be completed through the sink [4] [5].

3.1. Real-Time Processing

To implement real-time processing, a buffer is set up in the streaming server. The data in the buffer keeps freshing so that the data sent to a client end for playing back is the latest. Since the video encoding format is H264 in the system, the play back operation of the video streaming is designed to start from I frame (key frame) [6]. The SPS and PPS needed for decoding are also be sent at the begging of video together with I frame. These two points must ensure to complete real-time video processing.

H264/AVC encoding is divided into two levels, video coding (VCL) and network abstraction level (NAL). The latter is responsible for formatting data and provides the header information [7]. For each data frame, a NAL element contains a head mark and frame data. In the actual H264 data flow, see **Figure 4**, there is a 00 00 01 separator in the front end of every NAL element. The begging of PPS is 00 00 00 01 68. The label I frame is 00 00 01 65. The first data frame from the encoder is for PPS and SPS, then I frame and non-critical frame. Every once in a while the encoder will output an I frame. The flow chart for video real-time processing is shown in **Figure 5**.

Audio real-time processing is easier than video. Since audio data have no key frame, only the audio head is needed to save. When the buffer is full, intercept the latest of several frames and move to the buffer beginning. Afterwards, the buffer gets fresh audio data again.

3.2. Design of the Server Data Source

In original Live555, only “ByteStreamFileSource” is internal used as the data source to read. The “ByteStreamFileSource” is an abstract data source of file data. Its interface for reading data is “doReadFromFile()”, in which a series of parameters are defined and call the fread (fTo, 1, fMaxSize, fFid) to read the data file. Therefore, to implement real-time video play another data source must be formed to replace “ByteStreamFileSource”. In this paper an “UDPVideoSource” is created with a creating function “createnew”.

3.3. Implementation of Multiple Acquisition Terminals and Clients

When connected to multiple acquisition terminals, the system needs to save the video data from every terminal and broadcast in real-time channel simultaneously. Multiple clients can also choose to playback the video from a

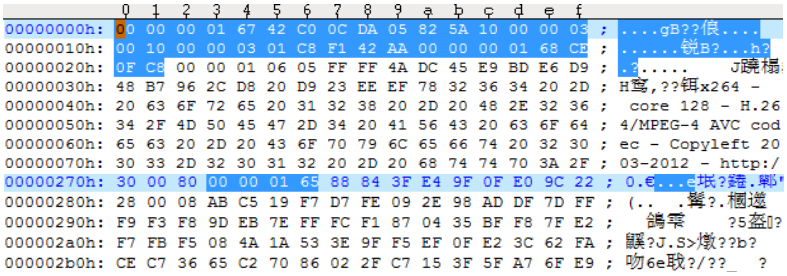

Figure 4. H264 Video data format.

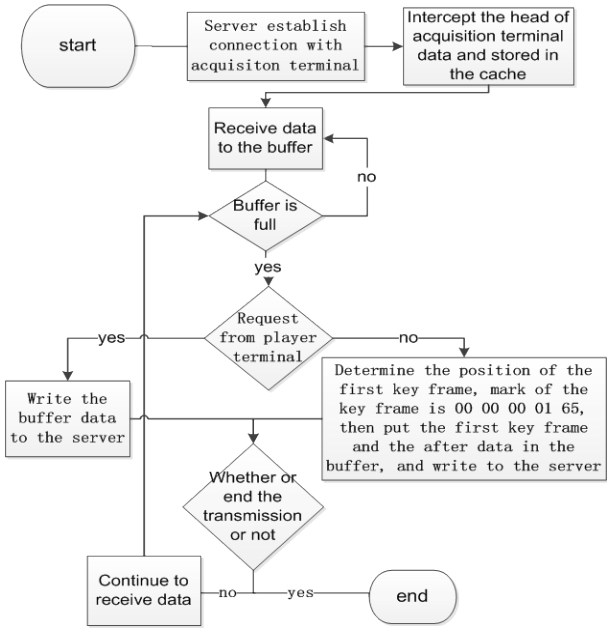

Figure 5. Flow chart for real-time processing.

corresponding acquisition terminal. It is required to allocate the audio and video interfaces between the server and acquisition terminals, and to handle multithreads. In addition, handling of the client session and task scheduling are also needed.

There are two solutions to implement the connection of multi-client. One is to use negotiate port and multithreads. In this solution, a TCP negotiate port is established first. Then, an acquisition terminal connects a negotiate port and applies to the server for a data transfer port. After obtaining the information about the acquisition terminal and verified, the system allocates a separate thread for every one terminal to save their name, identifier, status and corresponding data buffer. Each acquisition terminal is also allocated a separate UDP port for data transmission. The second solution is using a select model and single thread. To use this solution, the system uses a non-block way to listen to a set of socket ports (FD_SET) and then circularly check every FD_SET. When a port output data, the system starts processing.

Due to a non-blocking way is used, in which the system does not need to open multiple threads to receive data, the second solution can reduce system overhead. However, because of the high requirement of real-time video, when multiple acquisition terminals connected, a single thread processing cannot satisfy the requirements of system performance. On the other hand, although the first solution needs more system overheads, but every acquisition terminal uses a separated thread and the system can have good data independence and real-time performance. Moreover, the way of using a negotiate port explicitly publishes the only port connected to the server. When multiple acquisition terminals connect the same port, this solution can avoid the port is occupied. Therefore, the second solution is adopted in this paper.

The implementation of multiple terminal connections in the system is based on the RTSP service mechanism

in Live55. In the main thread of Live555, a command "env->taskScheduler().doEventLoop()" performs all the event loop. Command "DynamicRTSPServer" dynamically creates "RTSPServer", which then establishes the RTSP server. In the "RTSPServer" an "RTSPClientSession" is defined to deal with every single client session. Through task scheduling, the server can handle every task given by "RTSPClientSession" and ensure multiple clients operating simultaneously.

Afterwards, a "socket handler" is added into the task scheduling, and the system can initialize the "socket handler" and connects a terminal.

When a play back client sends a "Describe" request, the server creates a "ServerMediaSession" to show this play task. Each data stream is marked by a "SubSession".

3.4. RTSP Interaction and RTP Packaging

In Live555, the encoding modul of RTP is responsible for data packaging and transferring. The flow chart of RTP data transmission is shown in **Figure 6**.

3.5. Design of Server Session

The function "Session" is used to handle the requests from a play back client and establish a connection between a data source and a consumer "sink". When the "RTSPServer" receives a request "DESCRIBE", it finds a corresponding "ServerMediaSession" and then call the "ServerMediaSession::generateSDPDescription()". The "generateSDPDescription()" does a ergodic calling of "ServerMediaSubsession" used in the "ServerMediaSession" and obtains all the "sdps" from each "Subsession" through "subsession->sdpLines()". Finally synthesizes all the "sdps" into a complete "SDP" and then return. The flow chart is shown in **Figure 7**.

When obtained all the "sdps" from all the "Subsessions", the server creates an "H264VideoStreamFramer" and an "H264VideoRTPSink" repectively in the "Session" used for data connection. "H264VideoStreamFramer" connects to "H264VideoSource" and then to "UDPVideoSource", while "H264VideoRTPSink" also connects

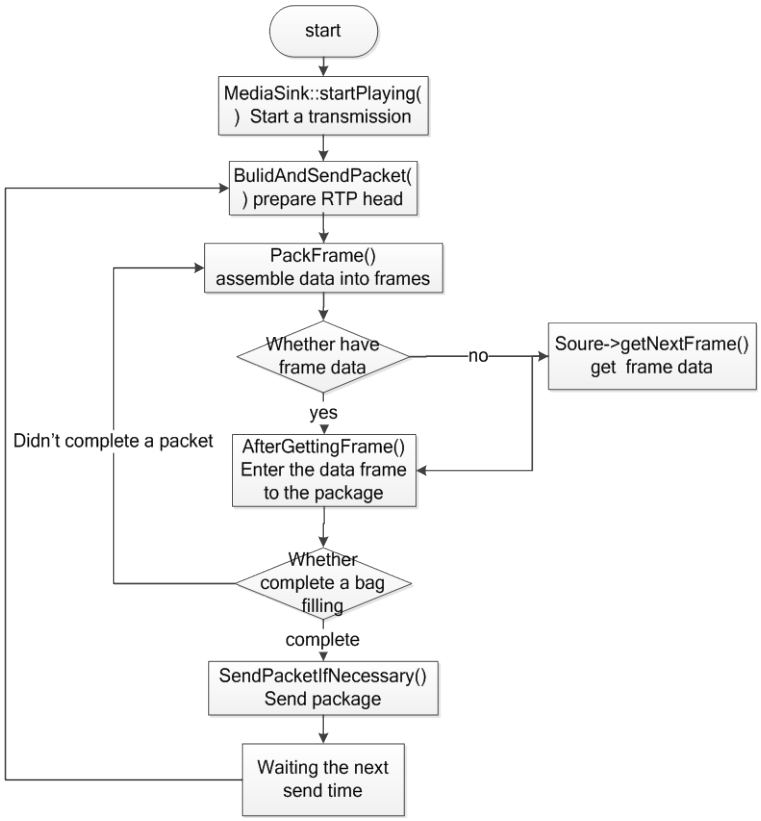

Figure 6. Flow chart of RTP data transmission.

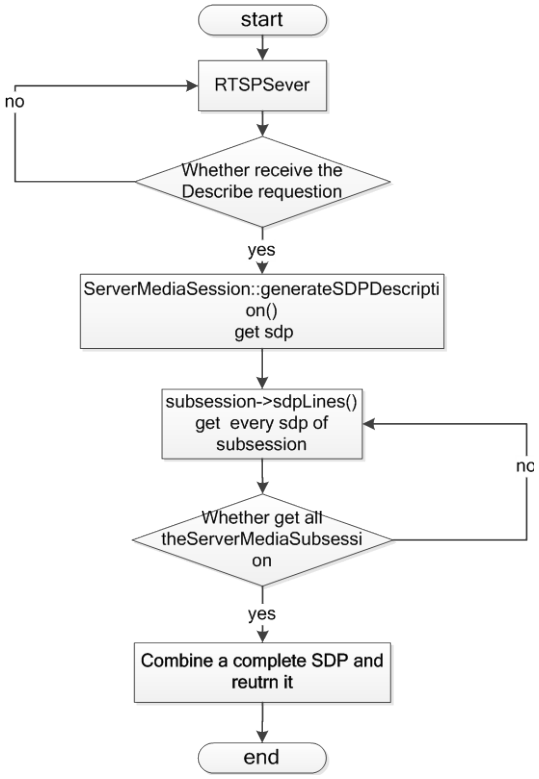

Figure 7. Flow chart for obtaining "sdp".

the final data consumer object—play back client.

3.6. Server Buffer Setting

The receiving data buffer is the first piece of memory area in an acquisition terminal. The buffer is defined by

unsigned char v_buf[V_BUF_SIZE];

unsigned char a_buf[A_BUF_SIZE].

The buffer size is difined by

const int V_BUF_SIZE = 200,000;

const int A_BUF_SIZE = 20,000.

On the processing of key frames (I frames), parameter "Iframe_num" impacts the transmission delay significantly. Either its oversize or undersize will degrade the system's real-time performance.

The buffer "fMaxSize" in "StreamParser" of Live555 is also an important influencing factor to system delay performance and its size should also be seting up appropriately. In this paper, the "Iframe_num" used is 3 and the "fMaxSize" is 30,000 bytes.

4. Experimental Results

Information displayed after start-up of streaming media server is shown in **Figure 8**. When a data acquisition client connects, the server allocates an UDP port for data transmission and registers the information and data source of the client. The server saves the data from the acquisition client and at the same time output the client name and ID in the server queue. Outputing information of the server is shown in **Figure 9**. When a playback client connects, the server parses the request and sends video data in real time to the client. Outputing information is shown in **Figure 10**. The video data can be played by live video website, mobile client and standard video player. The effect of the live web pages is shown in **Figure 11**. Mobile live broadcast with a PC and two handset clients is shown in **Figure 12**. When multiple acquisition terminals are used, a playback client can select one of them to display and the others will be stored in database as history video data for future browsing. The

```
LIVE555 Media Server
        version 0.74 (LIVE555 Streaming Media library version 2012.07.03).

This Live555 version extends by HuangShuai, QUST, 2014.
Support multiple real-time devices input and transmition and adjust some paramet
ers of real-time live video
Any question: sanyechong12@gmail.com.

Play streams from this server using the URL
        rtsp://192.168.1.178/<filename>
where <filename> is a file present in the current directory.

You can connect server TCP port 2100 to allocate a UDP port.
The real-time data transfers by UDP port 1888 to 2000.
(We use port 8000 for optional RTSP-over-HTTP tunneling, or for HTTP live stream
ing (for indexed Transport Stream files only).)
MySQL DataBase ready.
```

Figure 8. Server starting information.

```
LIVE555 Media Server
        version 0.74 (LIVE555 Streaming Media library version 2012.07.03).

This Live555 version extends by HuangShuai, QUST, 2014.
Support multiple real-time devices input and transmition and adjust some paramet
ers of real-time live video
Any question: sanyechong12@gmail.com.

Play streams from this server using the URL
        rtsp://192.168.1.178/<filename>
where <filename> is a file present in the current directory.

You can connect server TCP port 2100 to allocate a UDP port.
The real-time data transfers by UDP port 1888 to 2000.
(We use port 8000 for optional RTSP-over-HTTP tunneling, or for HTTP live stream
ing (for indexed Transport Stream files only).)
MySQL DataBase ready.
Here comes a real-time video collection client, name: luan, id: 0
```

Figure 9. Acquisition terminal connecting information.

```
LIVE555 Media Server
        version 0.74 (LIVE555 Streaming Media library version 2012.07.03).

This Live555 version extends by HuangShuai, QUST, 2014.
Support multiple real-time devices input and transmition and adjust some paramet
ers of real-time live video
Any question: sanyechong12@gmail.com.

Play streams from this server using the URL
        rtsp://192.168.1.178/<filename>
where <filename> is a file present in the current directory.

You can connect server TCP port 2100 to allocate a UDP port.
The real-time data transfers by UDP port 1888 to 2000.
(We use port 8000 for optional RTSP-over-HTTP tunneling, or for HTTP live stream
ing (for indexed Transport Stream files only).)
MySQL DataBase ready.
Here comes a real-time video collection client, name: luan, id: 0
A paly client connect!
```

Figure 10. Play end connecting information.

Figure 11. Play back of live web pages.

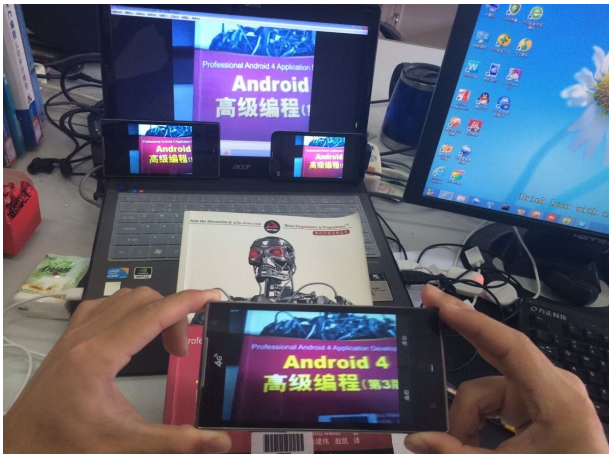

Figure 12. Mobile broadcast effect with 3 clients.

transmission delay is tested using a second chronograph in a smart phone. When testing 20 times, the average delay is 2.41 s.

5. Summary

The streaming server is developed based on an open source media library Live555 and can run on the Windows or Linux system. In wireless network such as 3G, WLAN and WAN, the live broadcast system based on the server can interconnect expediently smartphones, the streaming server and PC clients. The system has good versatility and compatibility because RTSP/RTP protocol is used for real-time transmission. The transmission delay from data acquisition to play back terminal is about 2.41 s.

References

[1] Wang, Z.H., *et al*. (2014) *Journal of System Simulation*, **26**, 556-561.

[2] Wang, W.L., *et al*. (2013) *Computer System Applications*, **22**, 161-167.

[3] Li, Y.Z., *et al*. (2013) Research on Mobile Streaming Media Rate Adaptive Method Based on Wireless Network Environment. *Microcomputer & Its Applications*, **32**, 54-56.

[4] Mao, Y.F., *et al*. (2011) Research and Design of a Web DVR Based on RTSP. *Computer Engineering and Design*, **32**, 2523-2526.

[5] Li, X.L., *et al*. (2011) Design and Implementation of Wireless Video Surveillance System Based on RTP/RTCP, RTSP. *Video Engineering*, **35**, 89-92.

[6] Zeng, B., *et al*. (2008) Design and Implementation of Performance Measure Tool for Stream Media Applications Based on Active Measurement. *Journal of Computer Applications*, **28**, 832-836.

[7] Gong, H.G., *et al*. (2005) Research Advances in Key Technology of P2P-Based Media Streaming. *Journal of Computer Research and Development*, **42**, 2033-2040. http://dx.doi.org/10.1360/crad20051201

Scientific Research Publishing (SCIRP) is one of the largest Open Access journal publishers. It is currently publishing more than 200 open access, online, peer-reviewed journals covering a wide range of academic disciplines. SCIRP serves the worldwide academic communities and contributes to the progress and application of science with its publication.

Other selected journals from SCIRP are listed as below. Submit your manuscript to us via either submit@scirp.org or Online Submission Portal.