Research on HD video transmission of Smart Home based on Linux System

Shanhui Yin[1, a], Chunhai Zhang[1, b], Hao Liu[1,c], Yanxiu Sheng[1,d], Xi Wang[2,e] and Zhiqiang Wei [1, f]

[1]College of Information Science and Engineering, Ocean University of China, Qingdao, China

[2]Central Research Institute of Haier Group, Qingdao, China

[a]yinshanhuiouc@163.com,[b]chai@ouc.edu.cn,[c]liu.hao@msn.com,[d]shengyanxiu@ouc.edu.cn, [e]wangxi@haier.com, [f]weizhiqiang@ouc.edu.cn

Keywords: video transmission, socket, multithread, intelligent home

Abstract. With the rapid development of computer network technology and multimedia technology, in the intelligent home field, digitalization, network, multimedia and intelligent will become the inevitable trend of development. In the context of convergence of three networks, intelligent household demand for video transmission technology of HD is more and more high. On the basis of comparative study of the current domestic and international video transmission technology, applying technology of computer network, technology of processing of digital image and combining with the actual needs of the intelligent home system, designing and implementing a set of intelligent home HD video transmission system based on Linux.

Introduction

Since the world's first intelligent building in the United States, the United States of America, Canada, Europe, Australia and Southeast Asia and other economically developed countries have put forward various intelligent home solutions. Intelligent home are widely used in the United States, Germany, Singapore, Japan and other countries. In 1998, "Asia 98 of household electrical appliances and Consumer Electronics International Exhibition" is held in Singapore. It introduced the intelligent home system of the Singapore model. Intelligent home experiences home electronic, home automation, WISEHOME of America and SMARTHOME of Europe.

In resent years, video application systems are used more and more widely. The transmission of video information on the network has become the focus of attention. The socket is one of the network communication application program interfaces. It is the most popular and it can be used conveniently to transmit data within a LAN. This paper discusses how to realize video transmission problem in LAN, on the base of the introduction of multithread and network programming, combining with video transmitting system developed by the author.

Fundamentals

Network programming interface. In the network communication, the most commonly used protocol is TCP/IP protocols. It has become a fact standard. But the TCP/IP does not standardize the application program interface, the development of network communication applications can not communicate directly with the TCP/IP core, but with the network application programming interface. Linux Socket programming interface is a most widely used network programming interface under Linux environment.

Socket, shared memory, pipeline and DDE are actually the mechanism of inter-process communication. Socket provides a method for communication between processes and the inter-process communication is extended to the network environment. Linux Socket provides full support for TCP/IP.

Socket is basically divided into two categories: the stream socket (sock-STREAM) and a datagram socket (sock-DGRAM). The stream socket or connection-oriented sockets, corresponds to the TCP protocol in the TCP/IP, provides connection oriented, reliable, not repeated data flow service. A datagram socket in the TCP/IP, corresponds to the UDP protocol, provides the transfer function based on the datagram and is not guaranteed to be reliable, orderly, and not repeat. This paper focuses on the programming method of stream socket, a datagram socket can refer to other literature.

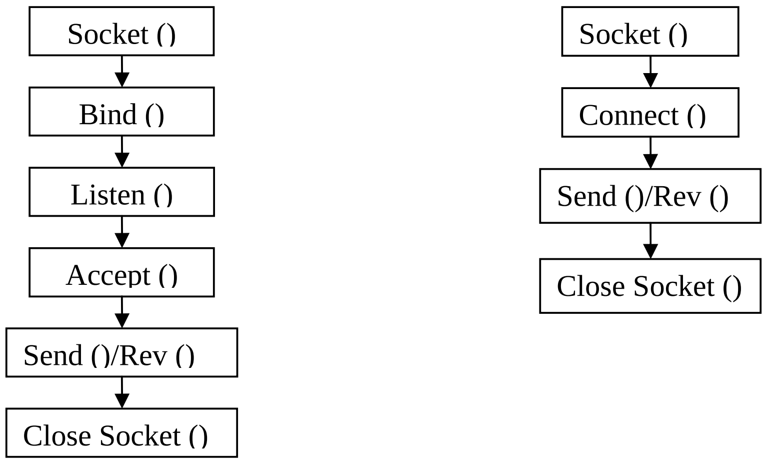

Fig. 1 The calling process of function of connection socket

As can be seen from Fig. 1, the server is the left part and the client is the right part. Firstly, the server starts. Establishing a socket by calling the socket (), and then calling bind () in order to connect the socket with the local network address (including the host address and the port). Then calling listen () in order to listen and setting the length of the request connection queue for limiting the number of arrangement for requesting and the default value is 5. Afterwards, calling accept () in order to receive connection. The client can call connect () to establish socket connect with server. Once a connection is established between the client and the server, the send () and the recv () can be called to send and receive data. The sender and the receiver are in the same position and there are no distinction points. Finally, the closesocket () can be called to close the socket.

Multithreading. The thread is the basic entity that the system distributes the time of CPU. It also is the smallest unit that the code is executed. In fact, only one thread is permitted to run at the same time. CPU switches frequently on all threads. Because time slice of each thread is very small, this makes the system operates multiple programs at the same time.

Linux is an operating system that supports multithreading, multitasking, but its kernel's achievement is different from other operating systems'. For example, WINDOWS NT defines threads independently. Therefore, the basic scheduling unit is thread, which increases the complexity of kernel and scheduling program. Linux defines thread as "thread of execution context". In fact, it is another execution context of the process. And the Linux basic scheduling unit is still process, its kernel only need to separate processes and only need a process/thread array.

Linux is an operating system that conforms to the IEEE POXIS standard. The thread is divided into: user threads and kernel threads. User threads are abstract concepts in full user level, multiple control flow unit of processing program that user uses, it does not need the support of the kernel and can be achieved in user' program. Thread's operation, fore example creation, synchronization, scheduling and management, can be implemented by the functions of the thread library of POXIS in Linux.

Part implementation of video transmission system

System architecture design. This system adopts connection-oriented stream socket and is based on client / server model. At the same time, one server can transmit video data with multiple clients. And this system includes two kinds of channel, namely the system channel and data channel. The system channel is responsible for creating and managing data channel. One client binds a system channel and the system channel can bind several data channels.

The data channel is responsible for the transmission of video data, and the data channel is controlled by the system channel. The way that sends video data includes three kinds. Respectively: no-buffer type, flow-buffer type, block-buffer type. The no-buffer type is that the client sends the video data to the server directly and does not have buffer; flow-buffer type is that the client put the video data to the flow buffer, then the thread for sending video data sends the data to the server, the thread for receiving video data receives the data and put it to flow buffer, the server can read the data when need. This way of processing data regards video data as stream. Because video is divided into frames, so the system provides a way that sends a block data to the server, the client sends a block data to block buffer, thread for sending sends the block data to the server, thread for receiving receives the data and put it into the block buffer, the server can read it when need.

……

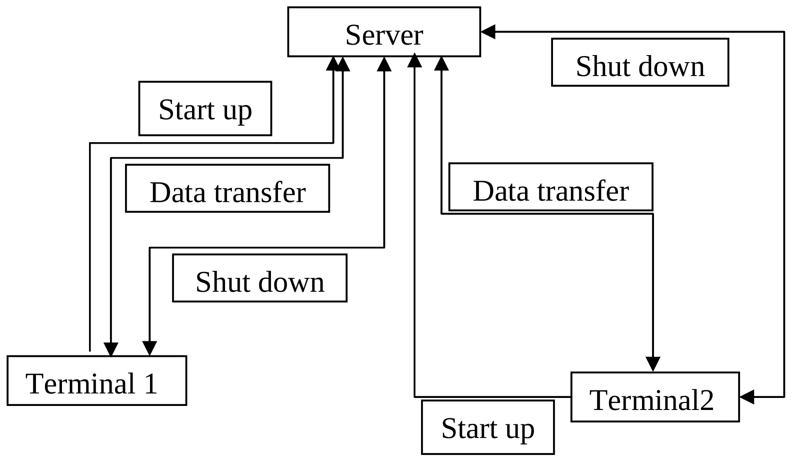

Fig. 2 System flow chart

According to Fig. 2, firstly, the client starts up the transmission. Then the transmission can carry on. When the transmission is over, the client or the server can shut down the transmission. At the same time, the server can transmit video data with multiple clients.

Port. There are several types of channels in this system, for instance, a client transmits video data to server, and the client has a system channel, several data channels. Every channel has a socket, so each channel has a port, the port is random.

Buffer. For the transmission of data, each data channel has its own buffer for sending data and buffer for receiving data, the type and the size of the buffer is chosen by the sender and receiver.

Video transmission. Creating and closing channel.Firstly, the server starts up, and then the server monitors a port. Monitoring the port is to establish system channel with client and this port needs to be specified. When a client requests a system channel, the essence is to establish communication of socket with server, and the server will produce different sockets for different clients. At the same time, the server starts up a thread, the thread is established for receiving client's command for creating data channel. If the client wants to transmit data, it must create data channel. The client sends the command "DCCREATE +\r\nID:+space+channel id+\r\n\r\n". The server receives the command and analytic, then monitoring a random port, waiting for the connection of the client. After the server monitors success, it sends command to the client, and the command form is "STATUS+space+return-code+\r\nPORT:+space+boundport+\r\nID:+space+channel id+\r\n\r\n". The client establishes connection with the server, according to the port. At the same time, the client and the server both starts up two threads, one for sending, one for receiving, and they set the way of transmission and the size of buffers through the callback function. When the way of transmission is non-buffer type, the receiver needs to set callback function for receiving video data.

When the transmission of video data is completion, the data channel should be closed first, and then the system channel is closed. The client or the server sends command to close the data channel, and the command is “DEND+space+0000+\r\n”. The client or the server sends command to close system channel, the command is “CTENDEND+\r\n”. When the server or the client receives the commands, the threads of data channel for sending and receiving are closed, the thread of system channel for receiving is closed. At the same time, the threads of the client’s data channel or the server’s are closed.

Three data transmissions are as follows.

When the style is no-buffer type. The sender sends video data. The form is “DATA+space+length of data\r\n+data”, the most data that can be sent once is 1KB, if the data is more than 1KB, the data will be divided into several parts and be sent several times.

When receiving, first, the thread for receiving receives the head of data to make sure the size of data that need to receive. Then the thread for receiving receives the video data and calls the callback function in order to provide video data to the receiver. And the head of data is “DATA+space+length of data\r\n”.

When the style is flow-buffer type. The sender puts the video data into the buffer for sending. The thread for sending finds the data and then sends the data. The form is “DATA+space+length of data\r\n+data”. The most data that can be sent once is 1KB. When the data is more than that, it will be divided into several parts and be sent.

The receiving thread reads the head of data to make sure the size of data for receiving. The thread put the video data into buffer for receiving in order to be read by the receiver.

When the style is block-buffer type. The sender put the whole video data into sending buffer. If the block size is larger than 1KB, sending thread will send the data several times, the most data that can be sent is 1KB every time. The form of first part of data that is sent is “DBLS+space+length of data+\r\n+data”, the form of last part of data that is sent is “DBLE+space+length of data+\r\n+data”, the form of others is “DATA+space+length of data+\r\n+data”.

The receiving thread receives the video data, according to the agreement, and the whole block data can be merged, and the block data is put into receiving buffer in order to be ready read by receiver.

Conclusion

The communication of multiple threads helps to improve real-time of applications. Based on introduction of network programming and multithreading, the realization of video transmission system in LAN is discussed. The Socket programming and multithreading is used to achieve a set of video transmission system, and the environment is Linux.

Table 1 Test result of the system

Transmission type	Lowest rate	Highest rate	Average rate
No-buffer type	8.0[MB/s]	9.0[MB/s]	8.5[MB/s]
Flow-buffer type	9.0[MB/s]	10.0[MB/s]	9.5[MB/s]
Block-buffer type	9.5[MB/s]	11.0[MB/s]	10.25[MB/s]

According to the table, the lowest rate is 8MB/s, and the transmission type is no-buffer type. The highest rate is 11MB/s, the transmission type is block-buffer type. The average rate is between 8.5MB/s and 10.5MB/s, and it basically meets the needs of the video transmission.

References

[1] W. Richard Stevens, UNIX Network Programming, Prentice Hall PTR, 1988.1.

[2] Kay A. Robbins, Steven Robbins, UNIX System Programming, Prentice Hall, 2003.6.

[3] Kevin W. Fall, W. Richard Stevens, TCP/IP Illustrated, Volume 1 (2nd Edition), 2011.9.

[4] Ellis Horowitz, Sartaj Sahni, Susan An, Fundamentals of Data Structures in C, Sillicon Pr, 2007.8.

[5] Robert Sedgwick, Algorithms in C, Parts 1-4, Addison-Wesley Professional, 1997.9.

[6] Samuel P. Harbison, Guy L. Steele, C: A Reference Manual (5th Edition), Prentice Hall, 2002.2.

[7] Brian W. Kernighan, Dennis M. Ritchie, The C Programming Language, Prentice Hall, 1989.3.

Acknowledgement

This work was supported by Technology development and Cooperation (No.20120465); Doctoral F und of Ministry of Education of China (No.2010013211009).