ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 5, Issue 2 (Apr. - Jun. 2017)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2017)

Design of High-Speed UDP Data Receiving and Processing Based on WinPcap

[I]Guoping Chen, [II]Bing Chen, [III]Tianzhen Wang, [IV]Bowen Sun
[I,II]College of Optoelectronic Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[III,IV]College of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing, China

Abstract
This paper presents a design method of high-speed UDP data receiving and processing based on WinPcap, analyzes the data capture process of WinPcap technology, uses multithreading technology to design the system,the multithreading communication mechanism based on Qt is introduced.In this design, thread synchronization is carried out by using mutex, circular queue, signal and slot mechanism, and the key code of the program is given. It provides a common system flow for radar signal processing based on network.

Keywords
WinPcap; UDP; Qt; Multithreading; Data Transmission

I. Introduction

Gigabit Ethernet is based on decades of advances in computer networking and Ethernet Technology, Due to the adoption of many technical specifications of the original Ethernet standard, Gigabit Ethernet is fully compatible with the previously used Ethernet. Gigabit Ethernet has the remarkable characteristics of high efficiency, high speed and high performance. Now it has developed into the mainstream LAN technology[1]. With the continuous development of computer and network technology, we can transmit radar data to the computer for compression, decompression and other pretreatment, and use network to transmit radar signals in real time. The network protocols commonly used in network transmission are UDP protocol and TCP protocol, Compared with TCP, UDP takes advantage of low CPU resources and high network utilization, and supports unicast, broadcast and multicast transmission modes simultaneously. If the network is running well and needs to continue to transmit large quantities of data, the UDP protocol is more suitable for the transmission of radar signals. This system is radar imaging system, after collecting echo data from ADC, the data is encapsulated as UDP data packet and sent to the host computer, in general SOCKET programming, a large amount of data is transmitted over the network, and packet capture in SOCKET can cause severe packet loss, this system uses WinpCap technology to capture packets on the network, WinPcap (windows packet capture) was proposed and implemented by Fulvio Risso and Loris Degioanni who are Italians[2]. WinPcap technology can effectively reduce data packet loss rate[3]. After the host computer accepts the data, it uses multi thread programming technology to process the data, At present, many well-known operating systems support multithreaded (structured) processes such as Solaris 2.x, Mach 2.6, OS/2, WindowNT, Chorus, and so on. Many computer companies have introduced their own thread interface specifications, such as IEEE's multithreaded programming standard, POSIX 1003.4a, it is believed that multithreaded technology will be used more and more widely in software development[4].

II. The basic structure of the Winpcap

WinPcap is a packet capture library derived from Berkeley packet capture library, which is used to intercept and filter the data on Windows platform. The basic structure of the Winpcap is shown in figure 1, It consists of three modules: Netfroup Packet Filter,NPF; Packet.dll; Wpcap.dll[5].

First module is the kernel part (NPF) that filters the packets, delivers them untouched to user level and includes some OS-specific code (timestamp management) as well. Its function is to filter data packets and to transmit the data packet to user mode module infact.

Second module, packet.dll, is created to provide a common interface to the packet driver among the Win32 platforms. In fact, each Windows version offers different interfaces between kernel modules and user-level applications: packet.dll deals with these differences, offering a system-independent API. Programs based on packet.dll are able to capture packets on every Win32 platform without being recompiled.

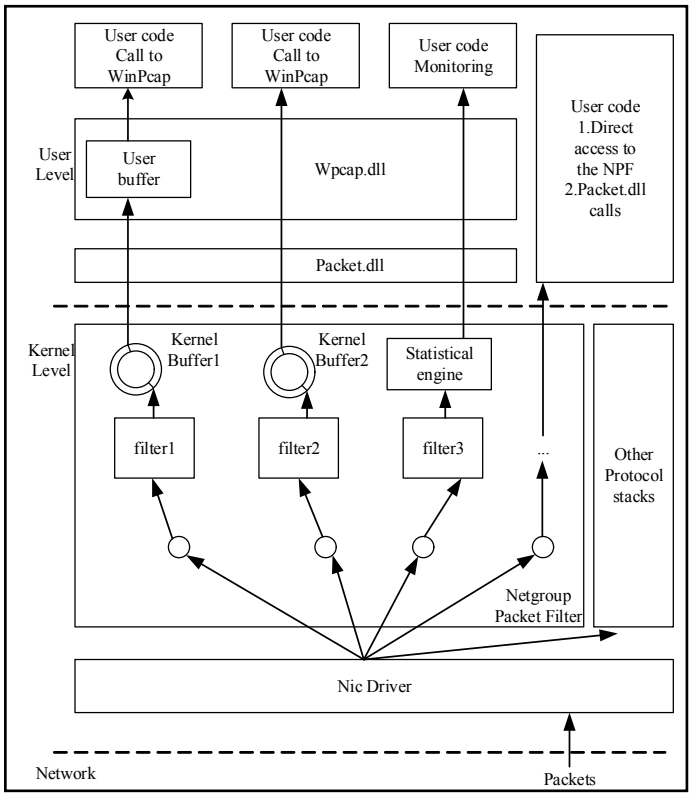

Fig. 1: The basic structure of the Winpcap

Third module, WPcap.dll, is not OS-dependent and it contains some other high-level functions such as filter generation and

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

user-level buffering, plus advanced features such as statistics and packet injection. Therefore programmers can have access to two types of API: a set of raw functions, contained in packet. dll, which are directly mapped to kernel-level calls, and a set of higher level functions that are provided by WPcap.dll and that are more user-friendly and more powerful. The latter DLL will call the former automatically; a single "high-level" call may be translated in several NPF system calls. Programmers will normally use WPcap.dll; direct access to packet.dll is required only in limited cases[6].

III. Multithreaded mechanism in Qt and Thread synchronization

1. Multithreaded mechanism in Qt

Qt toolkit provides multithreading. In a multithreaded Qt application, the application runs in its own thread and the processing takes place in one or more other threads[7]. the thread - related class is platform independent QThread class, the QThread class provides a variety of ways to create, run, and manage threads. Threads are overloaded by the run function of the QThread class.

2. Thread synchronization

a. Mutex

Mutex is a thread synchronization mechanism used to protect multithreaded shared resources. The use of mutex guarantees that only one thread is allowed to access the critical section at the same time.

b. Producer-Consumer

Two threads executed in the same process address space are called the producer thread and the consumer thread, the producer thread produces the item, and then puts the item in an empty buffer for the consumer thread to consume. The consumer thread gets the items from the buffer and then releases the buffer. When the producer thread produces the item, if there is no empty buffer available, then the producer thread must wait for the consumer thread to release an empty buffer. When consumer threads consume objects, if the buffer is empty, then the consumer thread will be blocked until new items are produced[8].

IV. System Architecture

The workflow diagram of the system is shown in figure 2 .The MainWindow thread provides the interface for human-computer interaction as the primary thread, DataReceive thread perform high-speed data reception, DataRecombination thread is responsible for data parsing and reorganization,and the DataProcessing thread processes the parsed data.A memory is shared between two threads as a circular queue for thread synchronization.The DataRecombination thread communicates with the DataProcessing thread by using Signals and slots mechanisms.Multi thread programming technology can effectively improve the efficiency of the system and the response speed of human-computer interaction interface.

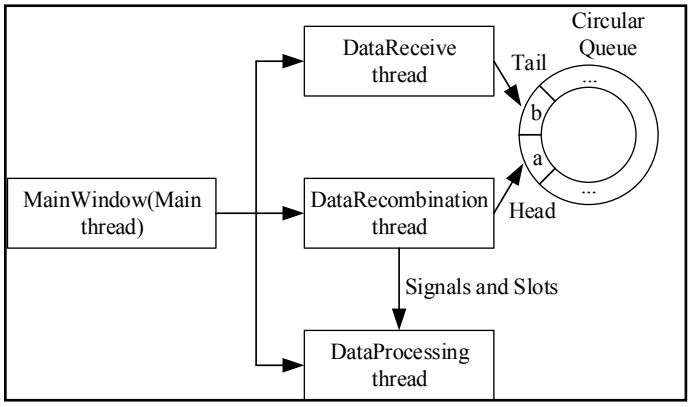

Fig. 2: The workflow diagram of the system

V. Introduction to Child Thread

1. DataReceive thread

DataReceive thread perform high-speed data reception, and as a producer, writes the received data to the tail pointer of the circular queue,.The DataReceive thread is defined as follows:

```
class DataReceive: public QThread
{
public:
DataReceive (QObject *parent);
~DataReceive ();
int init();
protected:
void run() Q_DECL_OVERRIDE;
PacketQueue *queue;
};
```

int inti () initializes the WinPcap, with the following steps:

a. get a list of attached network adapters:

```
if (pcap_findalldevs(&alldevs, errbuf) == -1)
{
fprintf(stderr, "Error in pcap_findalldevs: %s\n", errbuf);
return -1;
}
```

b. opening an adapter

```
if ((adhandle = pcap_open_live(d->name,
        65536,
        1,
        1000,
        errbuf
)) == NULL)
```

c. capturing the packets

```
while((res = pcap_next_ex( adhandle, &header, &pkt_data)) >=
0)
{
if(res == 0)
/* Timeout elapsed */
continue;
/* save the packet on the dump file */
pcap_dump((unsigned char *)dumpfile, header, pkt_data);
}
```

The captured packets are placed in the tail pointer of the circular queue, and the implementation code for the circular queue is as follows:

```
class PacketQueue
{
public:
```

```
uchar * queBase;
int usedCount;
int front;
int tail;
QMutex mutex;
QWaitCondition bufNFull;
QWaitCondition bufNEmpty;
public:
PacketQueue();
bool init();
};
```

Construct a circular queue of size n, when DataReceive thread receive a packet, usedCount plus 1, protect usedCount by using mutex, determine the size of the usedCount to identify the empty and full state of the circular queue and the read and write data of the circular queue are controlled by waiting conditions bufNFull and bufNEmpty. If usedCount is equal to N, stop writing data to the circular queue, and wait for the bufNFull.wakeAll () letter to continue writing data to the tail pointer of the circular queue.

2. DataRecombination thread

DataRecombination thread is responsible for data parsing and reorganization, The key code is as follows:

```
class DataRecombination: public QThread
{
Q_OBJECT
public:
PacketQueue *queue;
DataRecombination ();
bool init();
void run();
signals:
void sendMessage(QString message);
};
```

When the DataRecombination thread starts running, it constantly detects the value of the usedCount. If usedCount is 0, the data is stopped reading from the circular queue, and wait for bufNEmpty. wakeAll () to continue reading data from the head pointer of the circular queue and parsing the data. Each time a packet is parsed, sendMessage (QString, message) sends the signal to the DataProcessing thread, and the DataProcessing thread begins to process the data.

3. DataProcessing thread

DataProcessing thread processes the parsed data, the key code is as follows:

```
class DataProcessing: public QThread
{
Q_OBJECT
public:
DataProcessing ();
void run();
public slots:
void getMessage(QString message);
};
```

In the DataProcessing class, a slot called void getMessage (QString message) is defined to receive the semaphore. The signal and slot mechanism is the core mechanism of Qt that allows programmers to bind objects that are not related together, and implements communication between different objects, The connection of the signal and slot is realized by the connect () function:

```
connect(DataRecom0, SIGNAL(sendMessage(QString)), DataProcessing0, SLOT(run(QString))).
```

VI. Conclusion

This paper presents the process of receiving and processing high speed UDP data in project, the project uses WinPcap to acquire UDP data, and uses cyclic queue and multi thread design to improve system efficiency, Actual test shows this design can receive and process the UDP data packets at the 100M transmission rate.

References

[1] *Chunjiang Wen. Design of Transmission Between FPGA and PC Based on PCIe and Gigabit Ethernet [D]. Xidian University, 2014.*
[2] *Lu X, Sun W, Li H. Design and research based on WinPcap network protocol analysis system[C]// International Conference on Computer, Mechatronics, Control and Electronic Engineering. IEEE, 2010:486-488.*
[3] *Li H, Lu X. Research on the Capture Technology of Data Package based on Winpcap[J]. Network Security Technology & Application, 2010.*
[4] *Bin Luo, Xianglin Fei. Study and application of multithread technology [J]. Journal of computer research& development, 2000, 37(4):407-412.*
[5] *Xiaoyuan HU, Shi H. Analysis and Application of WinPcap System[J]. Computer Engineering, 2005, 31(2):96-98.*
[6] *Risso F, Degioanni L. An Architecture for High Performance Network Analysis[C]// Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on. IEEE, 2001:686-693.*
[7] *Blanchette J, Summerfield M. C++ GUI Programming with Qt 3[M]. Prentice Hall PTR, 2004.*
[8] *Byrd G T, Flynn M J. Producer-consumer communication in distributed shared memory multiprocessors[J]. Proceedings of the IEEE, 1999, 87(3):456-466.*