The study of campus network traffic monitoring platform

Bing Xu*

Chongqing Three Gorges University, Wanzhou Chongqing

Received 1 January 2014, www.tsi.lv

Abstract

From the view of practical campus network traffic monitoring platform, one kind of solution based on the model of SNMP and NETFLOW network management frame was put forward to elaborate the designed overall structure of campus network traffic monitoring platform, data acquisition, traffic plotting and so on. Using Visual C++6 to design this platform, not only the key technology and methods for realizing the campus network traffic monitoring platform could be achieved, but also the network traffic monitoring and management should be completed. The implementation of this platform can efficiently monitor the network traffic.

Keywords: Campus traffic, Network traffic, Traffic monitoring, management platform, VC++6

1 Introduction

With the development of computer network and expansion of communication scale, the network management as an important technology has become essential factor in the construction of digital campus. Due to the constant expansion of network scale, increasing complexity and enhanced isomerism, a campus network system usually includes a number of subsystems, integrates a variety of network operating systems, and collects network application equipment from different manufactures. In addition, the campus network system needs to be supported by a lot of network application service software. Therefore, we need an efficient campus network monitoring and management platform to for effective management [1]. In addition, the implementation of campus network traffic monitoring and management platform has important significance in network traffic billing, network security, etc.

2 Design concept

Network traffic monitoring and statistical analysis are important constitutional parts of network management and maintenance process. In recent years, with the constant expansion of the network scale, the increasing complexity of network and constantly emerging of new network services, the issues that understand and accurately describe the characteristics of internet network traffic and network behaviour model have become increasingly outstanding [2]. Moreover, the good management contributed by traffic monitoring and statistical analysis has become more and more important. Mainly based on the characteristics of network traffic and foundation of network, the construction of traffic monitoring platform should be established.

3 Solution

In order to timely view the network traffic, net resources and the application program, and improve the network performance, the solution based on the model of SNMP and NETFLOW network management frame was developed. In addition, the information acquisition system based on the Windows platform was explored and realized. According to the needs from network management, the provided information service could determine a certain self-interested domain, router and its port, traffic of fixed IP address.

4 Design and implementation

4.1 OVERALL STRUCTURAL DESIGN

Based on the collected and analysed user requirements from network traffic monitoring software, the features were summarized below:

1) Capture the network interface data packet as much as possible, set the card to promiscuous mode and then capture the data packet;
2) Resolve the content of data package, and analyse its protocol type, source address, acquisition time and so on;
3) User-defined monitor according to different requirements from users through given address range, related package had specific agreements, etc.;
4) Yield the monitoring results, which should contain real-time traffic charts, lists and so on;
5) Record the logbook for later analysis;
6) Detect and analyse some common attacks;

The platform written by VC++6.0 mainly had three modules which could be divided into data acquisition and display module, traffic information statistical module,

* ***Corresponding author*** e-mail: cqwzxb888@163.com

and traffic drawing module. The schematic diagram of platform was shown in Figure.1.

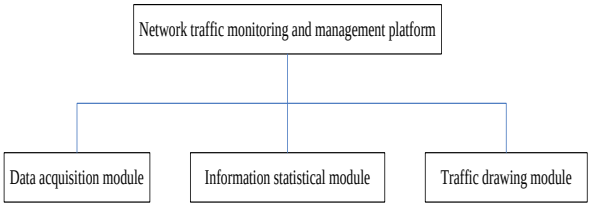

FIGURE 1 The component modules of the platform

Data acquisition module (*capture, analyse and display the network interface data*) is able to capture data based on user defined conditions, such as only monitor the data packets using TCP or UDP protocol, also monitor the data packets that related the IP addresses users want to pay close attention to, complete the data packets logging, enhance the flexibility of the system; meanwhile, judge some common attacked characteristics in the analysing process of data packets and send out warnings.

Information statistical module completes statistical functions, such as accounting the received number of data packets which could help IP finishing the statistics, the number of mistaken protocols among the received data, the number of messages regarding applying the transfers, the number of available routers in the list of routers, the number of discarded routers, the quantity of requiring organization/successful organization and so on, accounting ICMP required the number of messages including sending/receiving, exceeding the TTL, redirecting, timestamp request/reply and so on (the IP assistant function is used)

Traffic drawing module displays the total network traffic, input traffic, output traffic, instantaneous traffic value and maximum traffic value; acquires the related data through accessing the network performance data in the registry, displays these data by traffic diagram.

4.2 DESIGN OF SPECIES IN THE COURSE OF ACQUIRING DATA PACKET

Both overall function and work process of the system should be obtained based on the analysis and design as above. In addition, a series of operations including establishing, binding, setting work mode, creating thread, receiving data and so on, would take place in the process from editing socket to finally acquiring data. In order to solve every step in the process, the relationships of species regarding acquiring data were designed. The schematic diagram was shown in Figure.2.

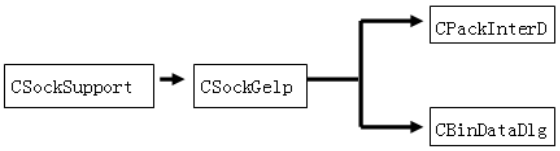

FIGURE 2 Relationships of species in the course of acquiring data packet

As shown above, like CSockSupport, CSockHelper, CPackInterDIg, CBinDataDIg and other species, different species were made up of different processing parts. In addition, these species contained relevant operational approaches.

4.3 MODULE OF DATA PACKET ACQUISITION AND ANALYSIS

4.3.1 Function specification

This functional module was mainly composed of CSockSupport, CSockHelper, CPackInterDIg and CBinDataDIg. These species would be described in detail as follows.

CSockSupport: mainly responsible for checking whether Socket was suitable for Version 2.0, embedded WSAStartup was able to start Socket;

CSockHelper: mainly achieved these activities such as acquiring the information structure from local computer, creating Socket, binding, establishing, creating thread, and all of the methods from data reception to protocol analysis. Detailed processes were shown in Figure.3.

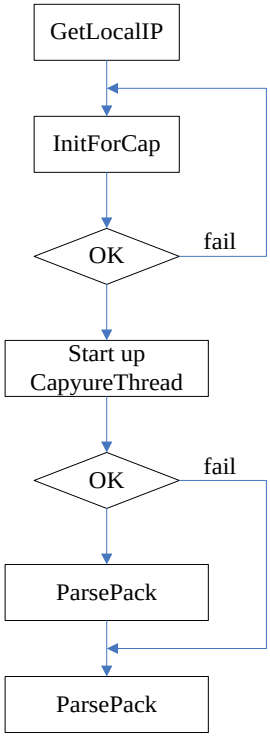

FIGURE 3 Flow chart of data packet acquisition

GetLocalIP implemented the method of acquiring the local address. The information structure of host computer was defined by LPHOSTENT lphp. The acquired process was completed by gethostname (szLocname, MAX_HOSTNAME_LAN) and gethostbyname (szLocname). The first parameter was used to deposit the buffer of local computer's name, and the second one was the length of buffer [3]. Finally, IP address was converted into "." address with inet_ntoa.

4.3.2 Creating socket, binding, setting up operational mode and creating thread

StartCapture could finish the creating socket, binding, setting up operational mode and creating thread. The detailed processes were shown as below:

m_sockCap = socket (AF_INET, SOCK_RAW, IPPROTO_IP);//create socket

bind (m_sockCap, (PSOCKADDR)&sa, size of (sa));//bind

setsockopt (m_sockCap, SOL_SOCKET, SO_REUSEADDR, (char*)&bopt, sizeof (bopt));//set operations

setsockopt(m_sockCap, IPPROTO_IP, IP_HDRINCL, (char*)&bopt, sizeof(bopt));// set operation

WSAIoctl (m_sockCap,SIO_RCVALL,&dwBufferInLen,size of (dwBufferInLen), dwBufferLen, size of (dwBufferLen), &dwBytesReturned, NULL,NULL);//promiscuous mode

m_hCapThread = CreateThread (NULL, 0, CaptureThread, this, 0, NULL);//start thread

The data reception was finished by a thread function named CaptureThread. After data reception, the resolution process took place unless the data in buffer zone had been converted into IP data [4]. The detailed process of acquiring protocol names was shown as follows.

```
for(int i=0; i<MAX_PROTO_NUM; i++)
        if(ProtoMap[i].ProtoNum==iProtocol)
            return ProtoMap[i].ProtoText;
        return "";
```

ParIPPack was able to resolve the data packet.

int iIphLen = size of(unsigned long) * (pIpheader->h_lenver & 0xf) //acquire the length of data packet
Resolution protocol was shown as below.

```
    switch(iProtocol)
    {
    case IPPROTO_TCP :
……
        case IPPROTO_UDP :
……
        case IPPROTO_ICMP :
……
default :…… }
```

The thread and socket could be closed by StopCapture as follows.

```
    if(m_hCapThread)
    {TerminateThread(m_hCapThread, 0); //terminate process
        CloseHandle(m_hCapThread); //close handle
        m_hCapThread = NULL;}
    if(m_sockCap)
        closesocket(m_sockCap); //close socket
```

CbinDataDIg was mainly responsible for depositing and displaying the acquired data. The targets of CbinDtaDIg and CsockHeliper created by CpackInterDIg were used to acquire, analyse, display and deposit data. Moreover, CpackInterDIg could edit these controls including setup conditions for acquiring and record daily work.

4.4 TRAFFIC DRAWING MODULE

4.4.1 Design specification

In the practical process of programming, one kind of interfaces supplied by windows system was able to access relevant data of network performance, such as flux. This module was divided into three sub functions that were the sub module that can access to the performance data (take charge of accessing registry and acquiring data), display sub module (draw the data in the window), and frame sub module (reflect and deal with message), respectively.

In this module, one function called RegQueryValueEX was used to access the registry [5]. This function would search relevant styles and data through the open registry keys and names. The prototype of this function was shown as follows.

LONG RegQueryValueEx(HKEY hKey, LPCTSTR lpValueName, LPDWORD lpReserved, LPDWORD lpType, LPBYTE lpData, LPDWORD lpcbData);

Function parameters: hKey was the default key assignment of registry; lpValueName was the name of key assignment which was needed to be searched for; lpReserved should be reserved unless it was NULL; lpType was the style of key assignment; lpData was the data that input/output received key assignment; lpcbData was the mark of buffer size used for input/output received key assignment [7].

In Windows 2003, when RegQueryValueEx was called, the returned data was not directly displayed as requested data if the hKey was set as HKEY_PERFORMANCE_DATA. Therefore, programs had to traverse the entire data block [6]. The logic structure in the data block was shown in Figure.4.

The searching process for ensuring performance data block was shown in Figure 4. It started from the performance data structure named PERF_DATA_BLOCK to PERF_OBJECT_TYPE. Then, the PERF_COUNTER_DEFINITION could find the site deviation through HeaderByteLength, which belonged to PERF_OBJECT_TYPE. Each unit of

PERF_OBJECT_TYPE called DefinitionLength had the ability to ascertain the structure of corresponding PERF_INSTANCE_DEFINITION. The structure of PERF_COUNTER_BLOCK was controlled by the structure of corresponding PERF_INSTANCE_DEFINITION [8]. The traffic diagram of this module in the course of running was shown in Figure 5.

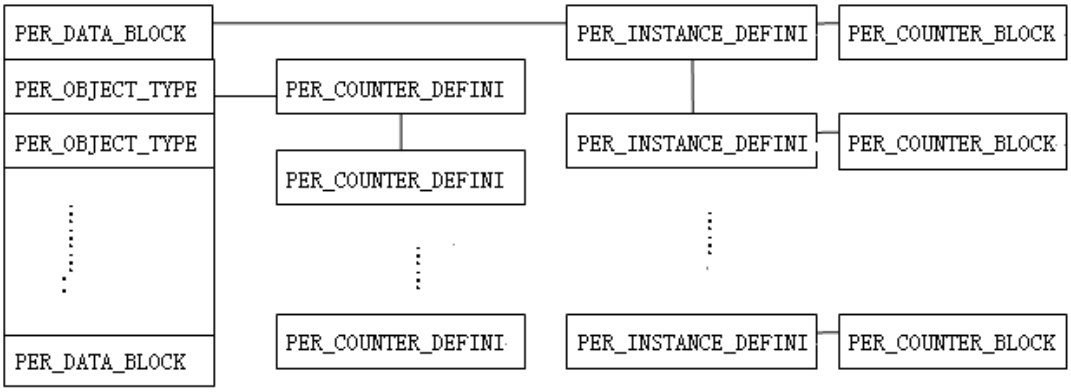

FIGURE 4 The logic structure of network performance data block in registry

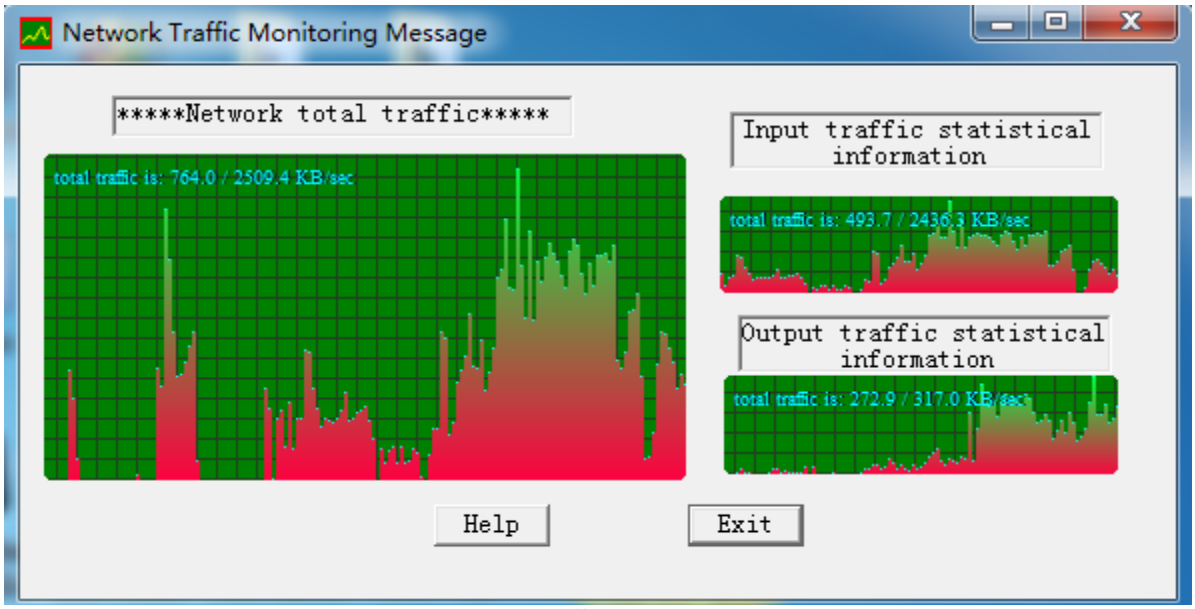

FIGURE 5 Traffic drawing diagram

4.4.2 The main codes in the traffic drawing module

//The function codes, which were defined by users were displayed by traffic when the network was in the connective state.

```
void MFTrafficButton::OnTimer(UINT nIDEvent)
{
    // TODO: Add your message handler code here and/or call default
    if(nIDEvent == NETTIMER)
    {
        #ifdef _I_HAVE_PLATFORM_SDK_INSTALLED_
        DWORD flag, reserved;
        BOOL erg;
        flag = 0;//INTERNET_CONNECTION_OFFLINE ;
        reserved = 0;
        TCHAR connectionname[1024];
        erg = InternetGetConnectedStateEx(
            &flag,  //OUT LPDWORD lpdwFlags,
            (LPTSTR)&connectionname,//OUT LPTSTR lpszConnectionName,
            1024,//IN DWORD dwNameLen,
            0//IN DWORD dwReserved
            );
        isOnline = erg;
#else
        isOnline = TRUE;
#endif
        // the current traffic was obtained
        double traffic = m_cTrafficClass.GetTraffic(SelectedInterface);
```

```
                DWORD totaltraffic =
m_cTrafficClass.GetInterfaceTotalTraffic(SelectedInterfa
ce);
                double delta1;
                double divisor =
(1000.0/(double)NETUPDATESPEED);
                delta1 = (double)(traffic * divisor) /
1024.0;
                CurrentTraffic.Format("current
traffic: %.1f KB/sec",delta1);

                // Should we recalculate the local
maximum per session or per display?
                if(useAdaptiveScale==TRUE)
                {
                        MaxTrafficAmount = 0.0;
                }
                // Shift whole array 1 step to left and
calculate local maximum
                for(DWORD x=0; x<TrafficEntries;
x++)
                {
                        TrafficStats[x].connected =
TrafficStats[x+1].connected;
                        TrafficStats[x].value      =
TrafficStats[x+1].value;
                        if(TrafficStats[x].value >
MaxTrafficAmount)
                                MaxTrafficAmount =
TrafficStats[x].value;
                }
                if(isOnline == TRUE)
                {
        TrafficStats[TrafficEntries].connected = TRUE;
        TrafficStats[TrafficEntries].value = traffic;
        if(TrafficStats[TrafficEntries].value >
MaxTrafficAmount)
                                MaxTrafficAmount =
TrafficStats[TrafficEntries].value;
                }
                else
                {
        TrafficStats[TrafficEntries].connected = FALSE;
         TrafficStats[TrafficEntries].value = traffic;
         if(TrafficStats[TrafficEntries].value >
MaxTrafficAmount)
                                MaxTrafficAmount =
TrafficStats[TrafficEntries].value;
                }
                double delta2;
                divisor =
(1000.0/(double)NETUPDATESPEED);
                delta2 = (double)(MaxTrafficAmount *
divisor) / 1024.0;
                MaximalTraffic.Format("maximal
traffic:%.1f KB/sec",delta2);
                AllTraffic.Format("total traffic is: %.1f
/ %.1f KB/sec",delta1, delta2);
        }
        // draw
        Invalidate(FALSE);
        CButton::OnTimer(nIDEvent);
}
```

References

[1] Phaal P, Panchen S, McKee N *Mon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks, RFC3176* September 2011

[2] *PPT 'sFlow & Benefits'* www.sflow.org Oct 2013

[3] Zseby T, Molina M, Duffield N, Niccolini S, Raspall F 2010 Sampling and Filtering Techniques for IP Packet Selection *draft-ietf-psamp-sample-tech-06.txt* February 2010

5 Conclusions

In the paper, from the view of campus network traffic monitoring platform, one kind of solution was put forward to elaborate the designed overall structure of campus network traffic monitoring platform, data acquisition, traffic plotting and so on. The database management technology and effective flexible warning system that could improve efficiency were applied into this platform. These technologies were not only able to record the time and site of fault, but also provide reliable information to analyse and solve the fault in time. Besides, the cost of platform development and operation was not high and the product was cost-effective. This platform was suitable for network of college and network with medium size.

[4] Dietz T, Claise B 2012 Definitions of Managed Objects for Packet Sampling *draft-ietf-psamp-mib-04.txt* February 18 2012

[5] Xu Peng, Qiong Liu, Sen Lin 2011 Internet traffic classification using support vector machine *Journal of Computer Research and Development* **46**(3) 407-14

[6] Xu Peng, Sen Lin 2011 Internet traffic classification using C4.5 decision tree *Journal of software* **10**(20) 2692-704

[7] Schroder C 2009 *Linux networking cookbook* (Translation Fen Liang) Nan Jing: Southeast University press 426-487*(in Chinese)*

[8] Kundu Dinangkur, Lavlu S M lbrahim *Cacti 0.8 network monitoring* UK: Packt Publishing Ltd 5-110

Authors

Xu Bing, born in February 10, 1976, Chongqing, China

Current position, grades: School of Computer Science and Engineering, Associate Professor and master supervisor in Chongqing Three Gorges University
University studies: Computer Science and Engineering in Chongqing University
Scientific interest: computer network and application
Publications: 3 Patents, 40 Papers
Research direction: computer network and application