Analysis and Design of an Online Data Collection Scheme Based on Campus Public Opinion Monitoring System

https://doi.org/10.3991/ijet.v16i11.23321

Zhiliang Xia, Zhijian Xiao
Wenzhou Polytechnic, Wenzhou, China
Zhejiang Dongfang Polytechnic, Wenzhou, China

Cong Zheng (✉), Xiaoling Zhang
Zhejiang Dongfang Polytechnic, Wenzhou, China
zhengcong1107@126.com

Abstract—The network data collection module is critical to the monitoring system of the public opinion on campus. To optimize the effect of network data collection, this study carried out comparative analysis on several different network data collection schemes, and formulated a data collection and filtering scheme based on the Libnids library. To improve the data collection performance, a multi-process pseudo-distributed processing scheme was prepared under multi-core CPU and the Linux framework. Finally, the proposed scheme was proved feasible and valuable in application based on campus public opinion monitoring system.

Keywords—Public opinion monitoring, data collection, Libnids library, pseudo-distributed processing

1 Introduction

The monitoring of public opinions on the Internet is a process of using data collection, classification, integration, and screening technologies to process various network information to form real-time statistical reports on Internet hot topics, dynamic network information, and netizen opinions, etc. [1]. As the Internet is providing us with an environment in which the information is open, diverse, fast, and convenient, some unhealthy or even illegal network information, such as pornography, violence, terrorism, fraud, and reactionary, are flooding college campuses, bringing serious harms to the physical and mental development of young students. In view of the dual effects of Internet, now more people have realized that measures must be taken to monitor and control the "double-edged sword" of the Internet [2].

In China, a few research institutions featuring "Internet public opinion" have been set up in recent years. In October 1999, the Public Opinion Research Institute of Tianjin Academy of Social Sciences was established, it is the earliest and the only research institute under the name of "Public Opinion Research" in China for quite a long time. On December 24, 2008, Renmin University of China and the public opinion research

center under the Founder Group jointly established the "Renmin University-Founder Public Opinion Monitoring Research Base", which aims to carry out research on the early warning systems of internet public opinions, the coping mechanisms of internet public opinions, the content integration and value development of network information, and the public opinion monitoring teaching and experiments, etc. [3]. In June 2011, the School of Journalism and Communication of Tsinghua University and the Youxun Times (Beijing) Network Technology Co., Ltd. jointly established the Tsinghua-Youxun Public Opinion Laboratory, which integrated the academic research resources of Tsinghua University and the market experience of Youxun to carry out research, teaching, and training works in the field of public opinion monitoring. After that, many other universities have also successively established research centers to conduct research on new media and Internet public opinion.

In this field, the main research achievements made by related research institutions at home and abroad are [4]:

1. China Public Opinion Network (http://www.yuqingz.com/). The system is a professional public opinion monitoring and data research platform developed by the Public Opinion Research Laboratory of Fudan University; it provides public opinion monitoring and analysis services covering the entire network.
2. Goonie (http://www.goonie.com.cn/). The system supports multiple webpage formats, multiple character set encodings, and content collection, extraction, and recognition of the entire Internet; it has multiple functions such as recognizing sensitive topics and hot topics of current affairs, tracking public opinions of topics, making brief summary automatically, analyzing opinion trends and emergency incidents, and alarming, counting, and reporting Internet public opinion incidents, etc. [5].

At present, domestic studies on the campus public opinion lack the support of a subject system, they are scattered and unsystematic. Current campus public opinion monitoring systems generally have low efficiency in network data collection, they lack an effective packet filtering mechanism and the system cost is high, which cannot meet the requirement of the high-speed network environment.

In view of these problems, this paper first conducted a requirement analysis on the network data collection module of the campus public opinion monitoring system, and compared a few common network data collection technologies and their pros and cons; then, targeting at the application requirements of campus public opinion monitoring system, this paper gave an in-depth analysis and determined to realize data collection and filtering based on the Libnids library; meanwhile, out of the consideration of the data collection module efficiency, this paper started from the process affinity under multi-core CPUs and analyzed the possibility of pseudo-distributed processing under multi-core CPUs; finally, this paper gave the complete design of the processing scheme, which is of certain application value for the design of the campus public opinion monitoring system.

2 Performance Requirement of Data Collection

The campus public opinion monitoring system monitors the network information within a certain range, such as a local area network. When it detects illegal, harmful, or sensitive information, the system will lock the IP address or account that sent the bad information, and then uses more accurate monitoring methods to monitor the special target. Regarding these features, since the campus public opinion monitoring systems generally perform small range monitoring, under normal conditions, the system's requirement for timeliness could be met [6]; in case of larger monitoring ranges, since it will require to accurately detect certain information in the massive network information and sometimes restore some bad information, the requirement on the network data collection and transmission is often very high, and the prerequisite for accurate detection and restoration is that the data packets won't get lost.

Through the above requirement analysis, it is not difficult to conclude that the improvement scheme of the system's network data collection module should at least satisfy the following aspects:

1. The network data collection module is separated from the data analysis module

The data collection module and the data analysis module should be separated. The data collection module only needs to acquire network data, reassemble data packets, write the reassembled data packets into files, and save these files into a certain folder. The data analysis/restoration module automatically reads data from this folder, and analyzes and restores the data. In this way, although the data analysis and restoration are relatively lagging, real-time and accurate data packet capture are ensured, which has greatly reduced the packet loss rate.

2. The data collection module only performs bottom-layer preliminary filtering

It's not that the collected data could be used directly without processing, the data collection module will perform preliminary filtering on the data [7], for example, among the collected TCP packets, only the POST data packets of the HTTP protocol are retained, other data packets will be filtered out. This process is performed at the bottom layer of the protocol, which can effectively reduce the volume of reassembled data and increase the data packet capture rate of the data collection module.

3. Pseudo-distributed processing method

After the data collection module is separated from the data analysis module, in order to further improve efficiency, under the condition of not increasing the hardware cost, we could consider to perform pseudo-distributed processing under multi-core CPUs [8].

3 Analysis and Comparison of Network Data Collection Schemes

3.1 Raw socket collection scheme

The Linux system provides a variety of network socket interfaces, which can be used for the network communication of host computers. The three most common socket types are the stream socket (SOCK_STREAM), datagram socket (SOCK_DGRAM), and raw socket (RAW_SOCKET) [9]. Stream sockets are mainly applied for network services with connections, namely the TCP service applications; datagram sockets are mainly used for network services without connections, namely the UDP service applications; and raw sockets enable users to better control the network data packets, and perform functions at the bottom layer; using raw sockets, applications such as network sniffer, denial of service attack DDOS, and IP spoofing could be achieved.

Raw sockets provide three functions that TCP and UDP sockets do not have, namely:

1. Using raw sockets, the reading and writing of ICMPv4, IGMPv4 and ICMPv6 packets could be achieved.

For example, the Ping program uses the raw sockets to send ICMP echo request and receive ICMP echo reply. The daemon mrouted used for multicast routing also uses raw sockets to send and receive IGMPv4 packets. The above-mentioned functions also allow applications constructed using ICMP or IGMP to be processed completely as user processes without having to add too many kernel codes. For example, the router discovery daemon is constructed in this way, and it processes two ICMP messages (router advertisement and router request) that the kernel does not know at all.

2. Using raw sockets, the reading and writing of special IPv4 datagrams could be achieved, and the kernel does not process the IPv4 protocol field of these datagrams [10].

Most kernels only process datagrams with values of 1 (ICMP), 2 (IGMP), 6 (TCP), or 17 (UDP), but the protocol fields might be other values, and RFC1700 lists all the values. For example, the OSPF routing protocol does not use TCP or UDP, instead, it directly uses IP and sets the protocol field of the IP datagram to 89. Therefore, since these datagrams contain protocol fields that are completely unknown to the kernel, the gated daemon that implements the OSPF protocol must use raw sockets to read and write datagrams, and this also applies to IPv6.

3. Using raw sockets, user-defined IPv4 header could be constructed, so that users can construct their own TCP or UDP packets

Raw sockets can directly interact with the data link layer which is in the system kernel, while the other two types of standard sockets are in the upper layer of TCP and UDP protocols in the system network protocol stack, as shown in Figure 1:

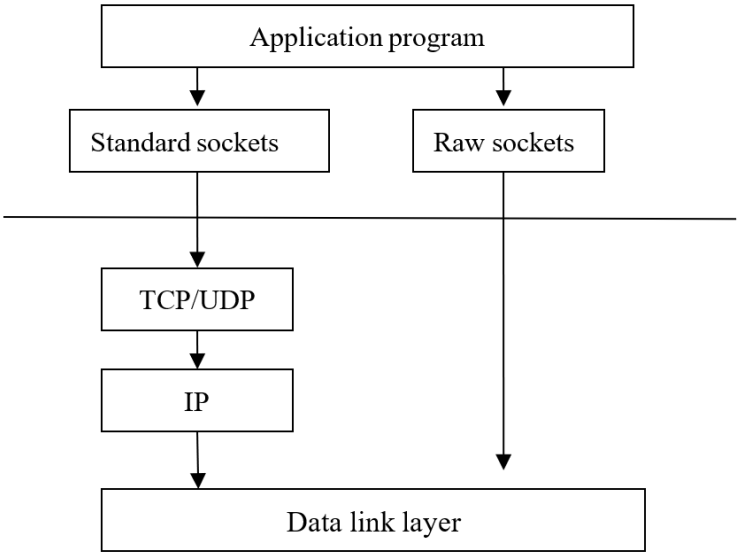

Fig. 1. Application program of standard and raw sockets

Therefore, when network card is set to the promiscuous mode, through the application programmed by RAW_PACKET, all the network data packets captured by the network card can be obtained.

However, the work efficiency of SOCK_RAW is relatively low, it does not provide core-based buffering and packet filtering mechanism, if a large amount of data in the core is directly transmitted to the application process, the system monitoring speed will be greatly reduced and the system overhead will be very large. In addition, raw sockets do not have filters for devices, IPv4 packets from any device will be transmitted to the socket interface, and the data from the device that is not of interest to the application must be processed and discarded by itself, therefore, SOCK_RAW is not suitable for high-speed network environment.

3.2 DLPI collection scheme

Data Link Provider Interface (DLPI) defines the services provided by the data link layer to the network layer; it is a standard interface between users and providers of Data Link Services (DLS), and its realization is based on the STREAMS mechanism of the UNIX system [11]. DLPI implements the Logical Link Control (LLC, ISO 8802/2) and the ISO Data Link Service Definition (DLSD, ISO 8886) at the kernel level.

The data link service allows users to use DLPI to directly access the data link layer, such as obtaining the status of the data link layer, or sending or receiving the raw frame of the data link layer, without the necessity of knowing the protocol of the data link layer. As long as the data link layer service provider supports the standard DLPI interface, users can use the DLPI interface to access the data link layer regardless of

whether the data link layer protocol is the Ethernet protocol or other protocols. When the application program accesses the data link layer, it only needs to turn on the device and use the DL_ATTACH_REQ of DLPI to request to attach it to DLPI. To improve efficiency, it generally needs to push in two stream modules: pfmod (packet filtering in the kernel) and bufmod (buffer the data passed to the application process) [12]. The specific working conditions are shown in Figure 2.

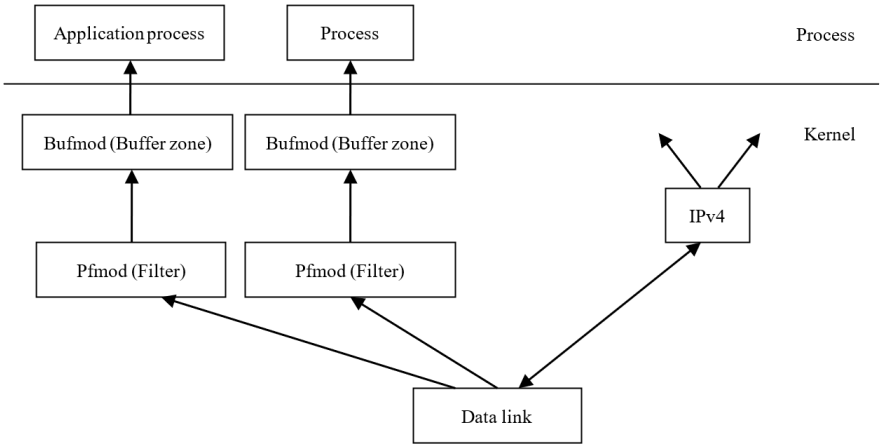

Fig. 2. Interception of packets using DLPI, pfmod, and bufmod

DLPI can be independent of specific network protocols, it can access the data link layer by sending and receiving stream messages. Therefore, users do not have to to know specific service provider information to use the data link layer services through DLPI, such as checking the working status of the network adapter, or receiving and sending the original network data packets. However, DLPI would cause greater system overhead when the data packet lengths are different, which cannot meet the requirements of high-speed network environment with diversified application protocols.

3.3 Libpcap collection scheme

Libpcap is a network data packet capture function library that provides packet capture mechanism for implementation-independent access operating systems, it runs on the Linux/UNIX platforms, and was developed by the Lawrence Berkeley National Laboratory Network Research Group; using Libpcap, the data link layer could be accessed directly [13]. Libpcap also supports the BPF (Berkeley Packet Filter) under Berkeley kernel, the DLPI under Solaris 2.x, the NIT under SunOS 4.1x, the SOCK_PACKET socket interface of Linux, and other operating systems, and tcpdump uses it as well. This library provides a consistent API interface and program framework for user-layer data collection applications.

The entire function flow of data packet collection based on Libpcap library is shown in Figure 3.

In order to improve the efficiency of data packet collection, generally, the filtering step of Libpcap is placed in the kernel, namely to adopt the kernel-level filter BPF [14]. The process of generating BPF rules is: first, the user uses the syntax specified by the Libpcap function library to compile the filter rules in the string form; then, the pcap_compile() function compiles the user-edited rules into BPF codes that can be recognized by the function library; finally, the pcap_setfilter() function loads the filtered BPF codes into the Linux system kernel. In this way, the BPF works in the system kernel and filters the data packets according to the configuration of the user.

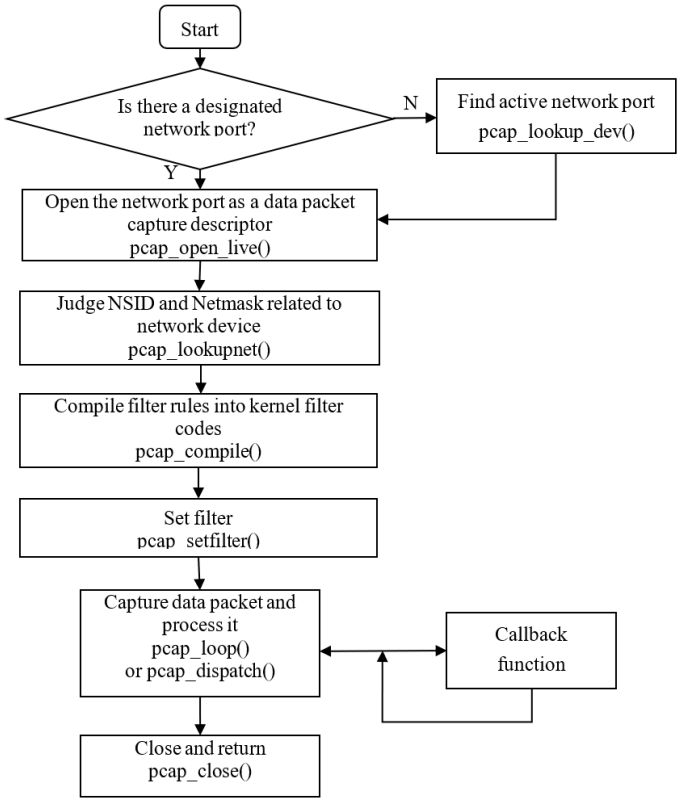

Fig. 3. Flow of data collection based on Libpcap

BPF is a kernel packet capture architecture, which enables applications under Linux to use a highly-optimized method to read the data passing through the network adapter [15]. BPF is mainly composed of two parts: Network tap and Packet Filter. Its working principle is shown in Figure 4.

When network card receives a new data packet, the network card driver will first make a copy of the data packet before delivering it to the system kernel and passing it to the BPF in the operating system kernel, where the BPF filters the data packet according to the filtering rules. If a data packet needs to be discarded, the network card driver will immediately return from the interrupt and prepare to receive a new data

packet; if the data packet needs to be accepted, the driver will transmit the data packet to the system kernel protocol stack, and then return.

If Libpcap is adopted for data collection, after a data packet is collected, it writes the data packet into the user space (namely the hard disk), and then uses other programs for protocol reassembling. This requires to perform disk reading and writing one more time, and a separate reassembling program is required as well, therefore, it doesn't use a separate Libpcap library to collect data, but adopts the embedded libnids library and the libpcap library with the data reassembly function [16].

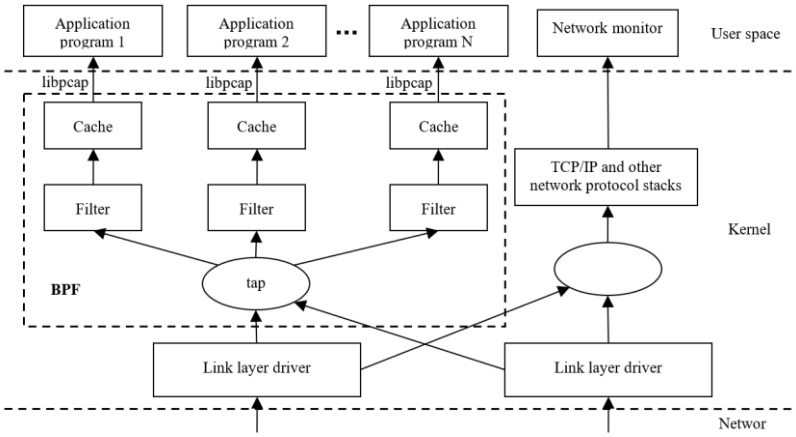

Fig. 4. Structure of BPF

3.4 Libnids data packet collection and reassembly technology

The full name of Libnids is the Library Network Intrusion Detection System, it is a professional programming interface for detecting network instruction, it was developed based on the two C function interface libraries libnet and libpcap, since general-purpose functions required for NIDS development were encapsulated in it, it has the function of data packet capture; moreover, Libnids also has the TCP data stream reassembly function, it can also analyze the various application protocols based on the TCP protocol. Therefore, with the help of the libnids interface function library, NIDS developers do not need to write other bottom-layer network processing programs, but only need to focus on implementing the functions of NIDS.

The entire function process of data packet collection and reassembly using the libnids function library is shown in Figure 5.

Libnids is a professional programming interface developed for network intrusion detection. It can efficiently complete the functions of IP fragmentation and TCP data stream reassembly. With this feature of Libnids, emphasis could be laid on the analysis of application data without caring about the implementation of the bottom-layer reassembly. Therefore, in the system, the collection and reassembly of data packets are implemented using the API provided by libnids.

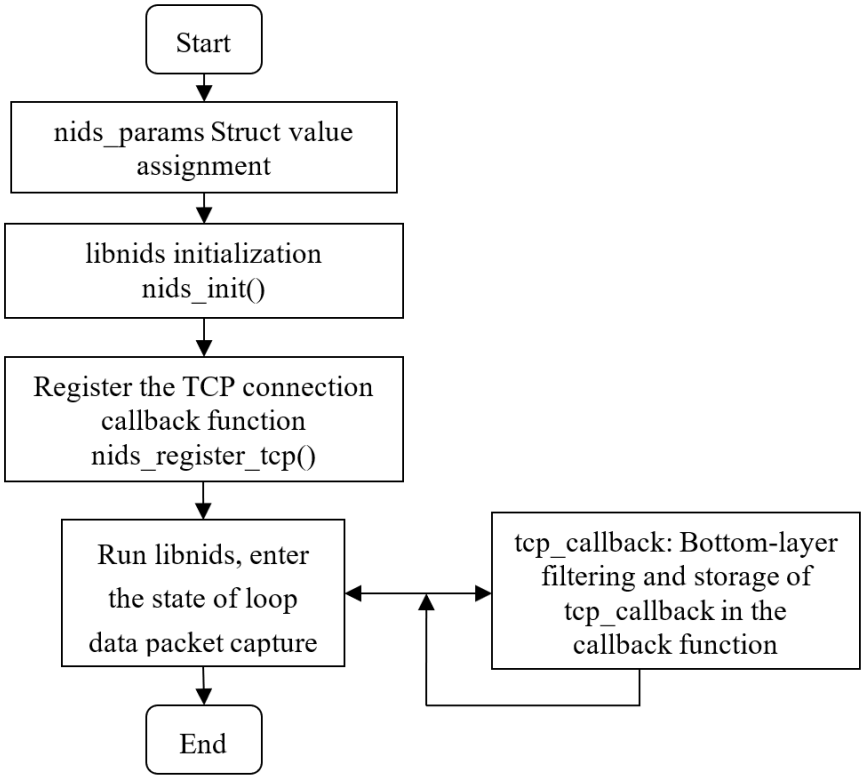

Fig. 5. Libnids data collection process

4 Design of the Network Data Collection Module

4.1 Module function

The network data collection module adopts the API architecture of the libnids library. The main function is to capture the network data packets at the exit of LAN, perform preliminary data packet filtering according to the bottom layer protocols, and store TCP data streams in the corresponding folder with a certain format for the data analysis module to read.

4.2 Module flow

The flow of the network data collection module is shown in Figure 6.
The Libnids library functions used in this module are described as follows:

- Void libnids _ip(void(*)) /*Define callback function, receive normal IP data packets, and process the IP data packets in the callback function*/
- Void nids_register_ip_frag(void(*)) /*Register a callback function that can detect all IP data packets; before using the libnids _ip function, it needs to call this function to register*/
- Void libnids_tcp_callback(struct tcp_stream *tcp_ns, void **tcp_param) /*Process the callback function of TCP link data, tcp_ns refers to all information in the tcp connection, its type is the tcp_stream type data structure; parameter tcp_param refers to the connection parameter information that needs to be transmitted, it may point to the private data in the TCP connection*/
- Void nids_register_tcp(void(*)) /*Register the callback function of TCP stream; before using the callback function libnids_tcp_callback, it needs to call this function to register*/.

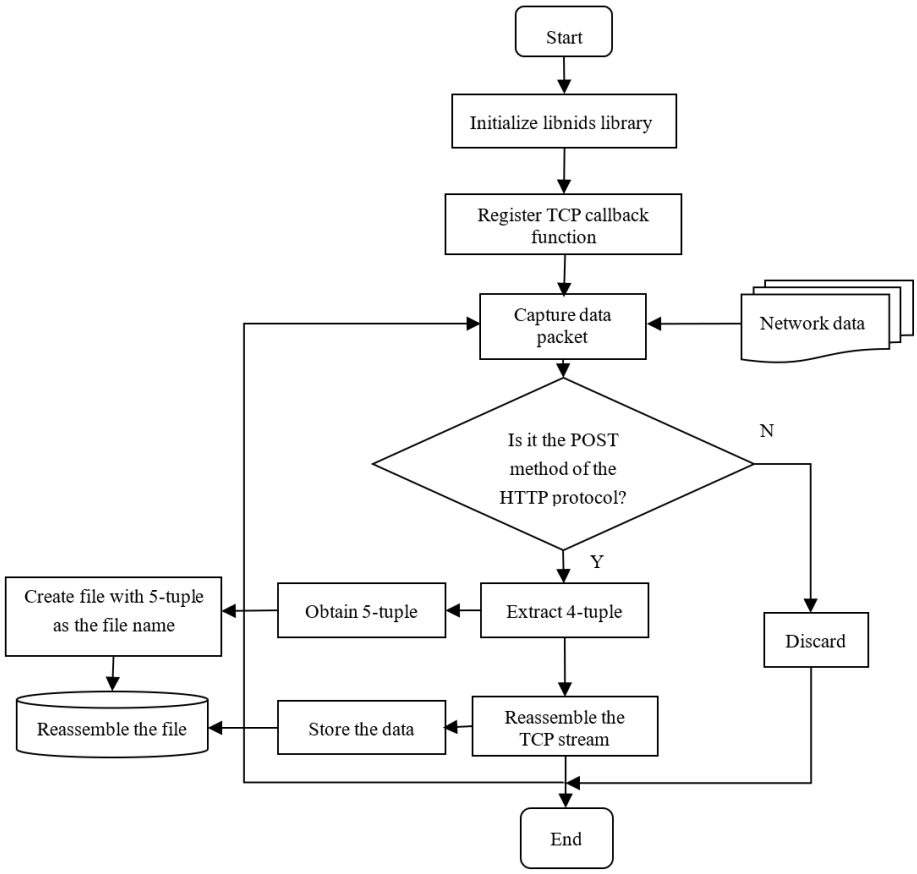

Fig. 6. Flow of the network data collection module

4.3 Module data structure

Before reassembling the TCP data stream, the four-tuple of the TCP stream needs to be extracted, namely the source IP address, source port, destination IP address, and destination port.

In this paper, the 4-tuple, together with the time when the TCP stream is transmitted, constituted a 5-tuple, thereby uniquely determining a TCP stream. The five-tuple definition is shown in Table 1:

Table 1. Define the 5-tuple

typedef struct _tcp_tuple5 { in_addr s_ip; unsigned short s_port; in_addr d_ip; unsigned short d_port; unsigned long time; } tcp_tuple5_info;	 s_ip is the source IP; s_port is the source port; d_ip is the destination IP; d_port is the destination port; the time of TCP stream reassembly

After the 5-tuple is extracted and the TCP stream is stored and reassembled, the file is renamed using the 5-tuple, in this way, file name repetition could be effectively avoided especially in the case of large data volume.

The tree structure of file storage is shown in Figure 7:

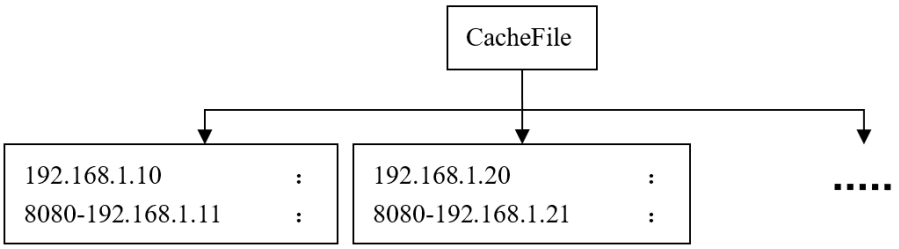

Fig. 7. Storage structure of reassembled data files

5 Design of Pseudo-Distributed Processing Scheme for System with Multiple CPUs

To improve the efficiency of the network data collection module, we can consider to separate the data collection module and the data analysis/restoration module of the campus public opinion monitoring system, this can greatly improve the efficiency of data collection. Since now most computers are dual-core or have more than two CPUs, in order to further improve the efficiency of the data collection module, this paper studied binding specific CPU for program process under multi-core CPUs to construct the system into a pseudo-distributed system without increasing the hardware cost.

5.1 SMP (Symmetrical Multi-Processing) load balance in Linux kernel

When a task is created in a SMP system, it will be placed in a given queue of CPU processing. Under normal circumstances, we cannot determine whether the task is short-term or long-term. Therefore, the initial allocation of CPU tasks is likely to be unsatisfactory.

In order to maintain the balance of task loads between CPUs, tasks can be migrated from CPU with heavier task loads to CPU with lighter task loads. The process scheduler of the latest Linux kernel version provides this function. Every 200ms, the processor will check whether the load of each CPU is balanced; if it is not balanced, the processor will perform a task balancing operation between the CPUs. Figure 8 shows a screenshot of the top-1 command under Linux, according to the content in the box in the figure, the system had balanced the workload:

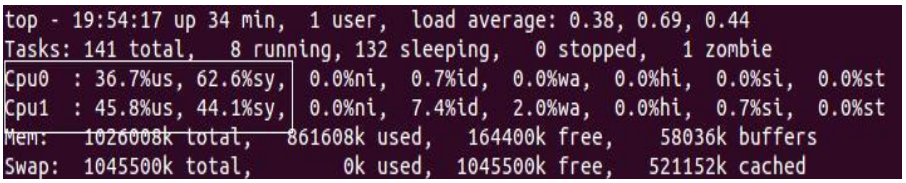

Fig. 8. A screenshot of SMP load balance in Linux kernel

However, when the Linux kernel scheduler performs load balancing, there is a negative effect that the CPU cache is cold for newly migrated tasks, that is, the data needs to be re-read into the CPU cache, which will inevitably cause a certain waste of resources; moreover, the CPU occupation of other processes will inevitably affect current working processes on this CPU [17].

In conclusion, in this system, if the Linux system's SMP load balance is adopted, the data analysis/restoration module will have certain impact on the work efficiency of the network data collection module; considering that the CPU occupation of other processes will affect the efficiency of the network data collection process of the system, the proposed system architecture in this paper adopted a distributed structure.

5.2 Multi-core computer pseudo-distributed processing scheme

In the proposed system, after the network data collection module and the data analysis/restoration module are separated, if the two modules are run on the same host, the data analysis module will have an impact on the operating efficiency of the data collection module; if we just simply increase the number of CPUs, when the system is busy (this situation is bound to happen), situations such as data cold copy will occur, but the efficiency increment is not much, therefore, the distributed system architecture should be considered.

If two hosts are used to run the two modules separately, then we'll need two host computers, when the two hosts are communicating, the host with the data collection module needs another separate network card to communicate with the host with the

data analysis module, and thus causing resource wastes. Therefore, in view of the utilization of hardware resources and the system performance requirements, the process of each module is bound to a separate CPU for execution, in this way, a processing method similar to the distributed processing is realized, since it's not the real distributed processing, but somehow has certain advantages of distributed processing, it is called the pseudo-distributed architecture.

Under the Linux architecture, binding a process to a specific CPU for execution is achieved by changing the CPU affinity level of the process. CPU affinity is the tendency of a process to run on one (or a few) CPU(s) as long as possible without being migrated to other CPUs; greater CPU affinity means less times of migration of processes between CPUs, and this indicates that the system load is smaller, which is exactly what we want.

The Linux kernel process scheduler has a feature called the soft CPU affinity. Although it enables the processes to have a certain CPU affinity, still it would cause task migration. In the Linux kernel, another mechanism is provided, namely to let developers program to realize hard CPU affinity, in this way, the application program can explicitly specify on which CPU(s) will the process run.

In Linux kernel, all processes have a data structure called task_struct. This data structure is very important for many reasons; among which, the most relevant to affinity is the cpus_allowed bitmask. This bitmask is composed of n bits, corresponding to n logical processors in the system one-to-one. A system with 4 physical CPUs can have 4 bits. If these CPUs have hyperthreading enabled, then the system has an 8-bit bitmask.

Therefore, if a given bit is set for a given process or task in the program, then this process or task can run on the specified CPU and will not be migrated to other CPUs. Therefore, if a process can run on any CPU and can migrate between CPUs according to requirement, then the bitmasks are all 1 [18]. In fact, this is the default state of the process in Linux. The APIs of Linux kernel provide several methods to allow users to view and modify current bitmask:

1. Sched_set_affinity() (modify bitmask)
2. Sched_get_affinity() (view current bitmask)

Through these two APIs, the hard affinity of the process can be modified; in this way, the requirements of binding process to CPU for execution and conducting pseudo-distributed processing could be achieved.

6 Conclusion

This paper studied the network data collection technology and compared the pros and cons of the raw socket collection scheme, DLPI collection scheme, Libpcap collection scheme, and Libnids collection scheme; then, the paper concluded that the data packet collection and reassembly method provided by the Libnids library is the most effective data collection scheme for the proposed system; moreover, this paper analyzed the system requirements and hardware conditions, and researched the multi-

process distributed processing technology and schemes under the condition of multi-core CPUs and Linux architecture, which can improve the data collection performance of the system to a certain extent, and has certain application values for the campus public opinion monitoring system.

7 Acknowledgement

The 2020 College Visiting Scholars "Teacher Profession Development Project" of Zhejiang Province: Research on the Deep Integration Path of Virtual Reality (VR) and Art Major Training Teaching in Vocational Colleges (FX2020084).

The 2021 Zhejiang Province Collaborative Education Project: Research on the Training of Technical Talents for New Technology Application under School-Enterprise Cooperation Mode - Taking the Digital Intelligence Manufacturing Professional Group as an Example (242). The 2020 Zhejiang Province Higher Education Teaching Reform Project: "Professionals (Group) + Studios" the Four-In-One Innovation and Entrepreneurship Exploration and Practice (jg20190913).

8 References

[1] Zhang, Z. (2015). Design and application of information network security audit and monitoring system. Journal of Henan Science and Technology, (16), 6-8. http://dx.doi.org/10.3969/j.issn.1003-5168.2015.16.002

[2] Zheng, C. (2018). Network information collection and restoration system based on knowledge base. PC Fan, (8), 26. http://dx.doi.org/10.3969/j.issn.1672-528X.2018.08.025

[3] Wang, X.H. (2019). Research and implementation of computer network security monitoring system. China New Telecommunications, 21(6), 165. http://dx.doi.org/10.3969/j.issn.1673-4866.2019.06.136

[4] Li, F. (2017). Design and research of network security audit and monitoring system. Computer and Information Technology, 25(6), 48-50, 63. http://dx.doi.org/10.3969/j.issn.1005-1228.2017.06.013

[5] Zhuang, Y., Wei, P. (2016). Research on bad information filtering system and its designing method. Journal of Chongqing University of Science and Technology (Natural Sciences Edition), 18(2), 111-113. http://dx.doi.org/10.3969/j.issn.1673-1980.2016.02.030

[6] Palshikar, G.K., Kale, MS., Apte, M.M. (2007). Association rules mining using heavy itemsets. Data & Knowledge Engineering, 61(1), 93-113. https://doi.org/10.1016/j.datak.2006.04.009

[7] Yan, X.W., Jiang, Y.C. (2001). Fuzzy data mining. Mini-Micro Systems, 22(4), 504-506. https://doi.org/10.3969/j.issn.1000-1220.2001.04.032

[8] Zhang, J. (2021). Distributed network security framework of energy internet based on internet of things. Sustainable Energy Technologies and Assessments, 44, 101051. https://doi.org/10.1016/j.seta.2021.101051

[9] Lu, K., Wang, H. (2016). Research and implementation of network monitoring system based on network data packet. Journal of Anhui University of Science and Technology (Natural Science), 36(5), 41-45. https://doi.org/10.3969/j.issn.1672-1098.2016.05.009

[10] Wang, L.H., Yang, Y.B., Zhao, Y. (2017). Network intrusion detection method based on data mining. Journal of Information Securyity Research, 3(9), 810-816. https://doi.org/10.3969/j.issn.2096-1057.2017.09.006
[11] Zhang, L.M. (2018). Research on high-speed network data capture and design of monitoring system. Nanjing University.
[12] Li, D.M., Zhou, Z.M. (2002). DLPI in UNIX environment. Intelligence Command and control system and Simulation Technology, (2), 58-59.
[13] Wang, Q.G., Liu, Z.Y. (2018). Design and implementation of wireless network monitoring system based on Libpcap. Cyberspace Security, 9(4), 92-96.
[14] Liu, Q. (2015). Analysis of Libpcap's BPF filter. Shandong Industrial Technology, (19), 239.
[15] Zhong, T., Liu, Y., Geng, J. (2005). Study on fast packet filter under network processor IXP2400. Journal of Computer Applications, 25(11), 2568-2570.
[16] Lu, B., Li, L. (2020). Network programming and development. Tsinghua university press.
[17] Xi, L., Jiang, C., Liu, J.H., Chen, Y.D., Yang, H.L. (2019). UNIX Programming Teaching Research Based on network development security topic guidance. (1) Proceedings of Heilongjiang Higher Education Society in an empirical study on the modernization of Higher Education, 164-168.
[18] Zhang, Q. (2010). Research on program performance optimization in multi core system. University of Science and Technology of China.

9 Authors

Zhiliang Xia (1979-), male, is a member of Wenzhou Polytechnic, Wenzhou, China; Zhejiang Dongfang Polytechnic, Wenzhou, China (xia441@sina.com). Master degree, associate professor, mainly engaged in the research on the information technology for vocational education.

Zhijian Xiao (1977-), male, is a member of Wenzhou Polytechnic, Wenzhou, China; Zhejiang Dongfang Polytechnic, Wenzhou, China (626091272@qq.com). Master, professor, mainly engaged in the research on the information technology for vocational education.

Cong Zheng (1983-), female, is a member of Zhejiang Dongfang Polytechnic, Wenzhou, China (zhengcong1107@126.com). Master, lecturer, mainly engaged in the research on computer networks and computer information systems.

Xiaoling Zhang (1993-), female, is a member of Zhejiang Dongfang Polytechnic, Wenzhou, China (zhangxiaoling@sina.com.cn). Master, assistant teacher, mainly engaged in the research on computer networks and computer graphics.

Article submitted 2021-03-18. Resubmitted 2021-04-20. Final acceptance 2021-04-23. Final version published as submitted by the authors.