Design of high performance firewall based on multi-core network development platform

Wei-wei ZHANG [1, a] , Ming-gang DONG [1, b]

[1] College of Information Science and Engineering, Guilin University of Technology,Guilin 541004, China;

[a]304880546@qq.com, [b] d2015mg@qq.com;

Keywords: Firewall; Multi-core; Network development platform; High performance

Abstract. The rapid development of network applications has brought great challenges to the design of the firewall. In order to improve the processing performance of the firewall, the design method of high performance firewall based on RMI multi-core network development platform is proposed. The hardware design and software design of the multi-core network development platform are discussed in detail. The effectiveness and feasibility of the design scheme are verified through experiments.

Introduction

The firewall technology is one of the main technical means to maintain the network information security [1]. With the rapid growth of the network traffic, the performance of the firewall and the design of high performance firewall have been put forward higher requirements.

Network processor is a kind of special purpose processor for network. Several representative network packet classification algorithms are implemented on the multi-core network processing platform and FPGA platform [2]. Xun etc. Use the SR-IOV specification of high performance data transmission characteristics and the filter function to receive the receipt, so that the virtual domain directly interacts with the network card and improve the performance of the I/O network which ensure the security of the firewall [3]. The basic functions of IPv6 firewall are implemented on the multi-core platform, and the high performance NAT64 function is implemented on the multi-core platform, which realizes the high speed interconnection between IPv4 and IPv6 [4]. However, the implementation of these functions has not been fully utilized in the performance of multi-core network platforms.

This paper mainly introduces the design of high performance firewall based on XLR732 processor. It is high performance firewall based on multi-core network platform which described and analyzed from three aspects of hardware platform, software design and performance analysis.

Hardware platform

Hardware firewall uses RMI multi-core network platform. Its processor is XLR732 that Central Processing Unit (CPU) has 8 hard cores that each core has four threads. Each thread can be performed separately on the task, so XLR 732 is considered to be a 32-core processor. Level-2 level cache is 2MB. XLR Roadmap clocked up to 1.5GHz. XLR is programming with FMN loop message and IPI (thread interrupt) to achieve inter thread communication [5]. The main functions of XLR732 RMI multi-core and multi thread platform can be used as network accelerator, which can be programmed to receive packets, analysis packet, manage packet flow and verify messages. The hardware components of the hardware platform are: DMA, Parser (high level programmable message parsing, support layer 3IP and layer 4 TCP/UDP verification and verification), Director Packet (responsible for packet flow management), Packet Distribution Engine(load balancing for the completion of multiple threads on packet distribution).

Packet processing flow: (1) Distribution of DMA, Buffer uses a pre allocated Buff storage message. (2)At the same time: the network interface to receive the message for the analysis and classification. (3) Based on the result of the classification, select the corresponding thread (in order to process the message), and the network interface sends the FMN message to the specified thread

(message arrival). (4)The network interface and the thread uses the packet descriptor to complete the packet reception, the message creation, the packet forwarding (to other threads, SAE or the network interface) (5) After the completion of the packet processing, the network interface is automatically released Buffer (in fact, the release of the message descriptor) via the FMN information.FMN is a 64 bit, low delay, one-way, message based on ring message network. XLR nuclear and IO devices transmit information through the FMN.

Software design

Firewall software design is a software system which supports the high performance of firewall hardware, and the software part of the firewall is mainly divided into the high efficiency data forwarding, the data packet inspection and the deep packet inspection.

Efficient Packet Forwarding. XLR732 supports the following three kinds of processor scheduling: Round Robin Scheduling, Fixed Cycle Scheduling and Priority Thread Scheduling. XLR732 network platform provide the thread scheduling flexibility and provide different applications with different thread scheduling.According to the Amdahl's law, the speedup of the parallel computing on multiple processors, the computing tasks can be executed parallel in the part of positive correlation, and the computing task must be executed serially negatively correlated. The formula is as follows:

$$S(n) = \frac{\mathrm{T}(1)}{\mathrm{T}(n)} = \frac{\mathrm{T}(1)}{\mathrm{T}(1)(B+\frac{1-B}{n})} = \frac{1}{B+\frac{1-B}{n}} \tag{1}$$

S(n) as the speedup ratio, N as the number of CPU, B is a serial processing algorithms in proportion. In the multi-core CPU platform, there are two factors that influence the B in the formula, which is the spin lock and Cache. In order to reduce the value of B further, need to further optimization on multi-core application design.

XLR single hard core structure has a separate 32KB of the primary instruction buffer and 32KB size of the data cache area support, and has the atomic instruction operation support. The hardware architecture is suitable for that separate application processes are running on the different hardware core which can realize the hardware core process of parallel. Thus, the application can be parallel running in the RMI network platform at the same time.

Through the reconstruction of the counter and Buff part of the code counter nuclear original extension for each CPU counter, the design of the structure members of each CPU can modify their counters. The counters are mapped to a different Cache-line. Modifying the Buff memory allocation, in order to increase the memory overhead for memory access efficiency is higher, in order to improve the efficiency of data processing forwarding. At the same time, the configuration of RMI applications in the msgring.cfg file, enable the RMI multi-core platform 32 Core CPU simultaneously. The following configuration method:

```
 bucket "cpu_0_0" {
     size 32;
     "tx_stn_cpu_0" 4;
}
```

As well, using multi thread polling processing messages from FMN, can avoid the message loop blocking, so as to improve the performance of multi-core processing.

Data packet verification filter. Conventional filtering technology has a filter for the IP address of the data packet and the access type of the data packet, which this data packet filtering method for data packets is less. The application of RMI platform based Trie tree algorithm realize the data packet splitting, restructuring, transfer and filter design verification. Based on the realization of software developing RMI Trie mainly realizes a method for 32 bit wide static routing table lookup and data packet forwarding path.

According to the Trie algorithm to construct the routing table based on the tree structure. The routing table includes the source IP, source MAC, destination IP address and destination MAC. According to the Trie query algorithm, query and scheduled the same rules of the data packet

structure, which determine whether the string matching with the IP header. According to the matching of the routing table information, determine the next-hop address.

XLR732 RMI platform using two 10G Ethernet XGMAC ports realize three layers of data processing and forwarding algorithm. Each XGMAC uses 63 idle Buffer to receive and forward packets. The basic algorithm of three layer data packet processing is as follows: (1) Extract the IP version number and compare with 0x45, if maching jump to the next step, or jump to the end. (2)Extract the IP survival cycle (TTL), verify that TTL, if TTL > 0 next, otherwise TTL = = 0, skip packet forwarding. (3) IP destination address from the packet. (4) Matching the 28K entries in the trie data structure with the longest prefix. (5)If matched, the destination IP hits the routing table, obtains the next hop IP address from the routing table. (6)If matched, the destination IP hits the routing table, obtains the next hop IP address from the routing table. (7)TTL IP value decreases. (8) Computing the IP packet's verification. (9)In the receiving link to send packets, the CRC package for the MAC package is available for this data packet. (10) Returns the buffer pool of the MAC buffer, using the automatic memory management feature.

The RMI multi-core network platform that combined with the Trie tree algorithm and the three layer data processing and forwarding algorithm, complete the work of a data packet's verification and filtering, which intercepting and filtering the packets through the RMI network multi-core platform.

Deep packet inspection. Pattern matching is the most basic technology for the detection of packet depth. Pattern matching can be achieved by three ways which is the software pattern matching, the TCAM pattern matching and the ASCI module matching. Among them, the flexibility of the software approach is high, and with the development of multi-core CPU, the performance of rapid increase is becoming the mainstream of the future.

The software design of the deep packet detection algorithm based on RMI network multi-core processing platform is as follows: (1) Receive data packets from the 10G Ethernet MAC interface. (2) 4 each adjacent 2 bytes are converted to an integer and 0x01010101 is added to them. (3) Send descriptor format to the same MAC transmission. (4) Using the automatic memory management function， and go back to the buffer zone for this packet.

Performance analysis

We download the program on the RMI network platform use the super terminal. RMI network platform send data packets through the host for high performance forwarding, detection, verification. After 63 packets which through RMI of data packets are analyzed, Network traffic research: The average time of the GMAC0 network interface to the host network interface of the RMI network is 0.332ms, as shown in figure 1. The average time of the RMI network interface to the GMAC0 interface of the network interface to the SPI0 interface is 0.109ms, as shown in figure 2. The average time of the RMI network interface to the GMAC0 interface of the network interface to the SPI1 interface is 0.106ms, as shown in figure 3.

In the RMI network platform, different network interface packet forwarding efficiency is different. GMAC0 network interface to the SPI0 network and SPI1 network port packet forwarding time is significantly lower than the GMAC0 network interface to the host's packet forwarding time. Packet forwarding time of data packets through the network platform improve the 0.226ms.

Summary

The firewall is an important tool of network security, but the performance of the firewall has become the bottleneck of the efficiency of the network. This paper presents a design scheme of high performance firewall based on RMI multi-core network platform. Research and introduction of hardware design, high efficient packet forwarding, data packet's fast detection and verification, and experimental tests are carried out under the existing experimental conditions. The feasibility and effectiveness of the scheme are verified.

```
[root@localhost ~]$ ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: icmp_seq=0 ttl=64 time=0.506 ms
64 bytes from 192.168.1.100: icmp_seq=1 ttl=64 time=0.308 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=64 time=0.301 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=64 time=0.347 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=64 time=0.319 ms
64 bytes from 192.168.1.100: icmp_seq=5 ttl=64 time=0.285 ms
64 bytes from 192.168.1.100: icmp_seq=6 ttl=64 time=0.303 ms
64 bytes from 192.168.1.100: icmp_seq=7 ttl=64 time=0.367 ms
64 bytes from 192.168.1.100: icmp_seq=8 ttl=64 time=0.337 ms
64 bytes from 192.168.1.100: icmp_seq=9 ttl=64 time=0.333 ms
64 bytes from 192.168.1.100: icmp_seq=10 ttl=64 time=0.306 ms
64 bytes from 192.168.1.100: icmp_seq=11 ttl=64 time=0.274 ms
64 bytes from 192.168.1.100: icmp_seq=12 ttl=64 time=0.368 ms
64 bytes from 192.168.1.100: icmp_seq=13 ttl=64 time=0.320 ms
64 bytes from 192.168.1.100: icmp_seq=14 ttl=64 time=0.312 ms
64 bytes from 192.168.1.100: icmp_seq=15 ttl=64 time=0.332 ms
64 bytes from 192.168.1.100: icmp_seq=16 ttl=64 time=0.375 ms
64 bytes from 192.168.1.100: icmp_seq=17 ttl=64 time=0.317 ms
64 bytes from 192.168.1.100: icmp_seq=18 ttl=64 time=0.331 ms
64 bytes from 192.168.1.100: icmp_seq=19 ttl=64 time=0.317 ms
64 bytes from 192.168.1.100: icmp_seq=20 ttl=64 time=0.312 ms
--- 192.168.1.100 ping statistics ---
21 packets transmitted, 21 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.274/0.332/0.506/0.046 ms
```

Fig. 1 GMAC0 network interface to the host network packet forwarding

```
[root@localhost ~]$ ping 192.168.1.102
PING 192.168.1.102 (192.168.1.102): 56 data bytes
64 bytes from 192.168.1.102: icmp_seq=0 ttl=64 time=0.267 ms
64 bytes from 192.168.1.102: icmp_seq=1 ttl=64 time=0.095 ms
64 bytes from 192.168.1.102: icmp_seq=2 ttl=64 time=0.101 ms
64 bytes from 192.168.1.102: icmp_seq=3 ttl=64 time=0.109 ms
64 bytes from 192.168.1.102: icmp_seq=4 ttl=64 time=0.107 ms
64 bytes from 192.168.1.102: icmp_seq=5 ttl=64 time=0.094 ms
64 bytes from 192.168.1.102: icmp_seq=6 ttl=64 time=0.107 ms
64 bytes from 192.168.1.102: icmp_seq=7 ttl=64 time=0.094 ms
64 bytes from 192.168.1.102: icmp_seq=8 ttl=64 time=0.109 ms
64 bytes from 192.168.1.102: icmp_seq=9 ttl=64 time=0.093 ms
64 bytes from 192.168.1.102: icmp_seq=10 ttl=64 time=0.106 ms
64 bytes from 192.168.1.102: icmp_seq=11 ttl=64 time=0.093 ms
64 bytes from 192.168.1.102: icmp_seq=12 ttl=64 time=0.105 ms
64 bytes from 192.168.1.102: icmp_seq=13 ttl=64 time=0.097 ms
64 bytes from 192.168.1.102: icmp_seq=14 ttl=64 time=0.106 ms
64 bytes from 192.168.1.102: icmp_seq=15 ttl=64 time=0.093 ms
64 bytes from 192.168.1.102: icmp_seq=16 ttl=64 time=0.105 ms
64 bytes from 192.168.1.102: icmp_seq=17 ttl=64 time=0.093 ms
64 bytes from 192.168.1.102: icmp_seq=18 ttl=64 time=0.108 ms
64 bytes from 192.168.1.102: icmp_seq=19 ttl=64 time=0.096 ms
64 bytes from 192.168.1.102: icmp_seq=20 ttl=64 time=0.106 ms
--- 192.168.1.102 ping statistics ---
21 packets transmitted, 21 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.093/0.109/0.267/0.036 ms
```

Fig. 2 GMAC0 network interface to the SPI0 packet forwarding

```
[root@localhost ~]$ ping 192.168.1.103
PING 192.168.1.103 (192.168.1.103): 56 data bytes
64 bytes from 192.168.1.103: icmp_seq=0 ttl=64 time=0.247 ms
64 bytes from 192.168.1.103: icmp_seq=1 ttl=64 time=0.105 ms
64 bytes from 192.168.1.103: icmp_seq=2 ttl=64 time=0.094 ms
64 bytes from 192.168.1.103: icmp_seq=3 ttl=64 time=0.106 ms
64 bytes from 192.168.1.103: icmp_seq=4 ttl=64 time=0.094 ms
64 bytes from 192.168.1.103: icmp_seq=5 ttl=64 time=0.103 ms
64 bytes from 192.168.1.103: icmp_seq=6 ttl=64 time=0.093 ms
64 bytes from 192.168.1.103: icmp_seq=7 ttl=64 time=0.106 ms
64 bytes from 192.168.1.103: icmp_seq=8 ttl=64 time=0.093 ms
64 bytes from 192.168.1.103: icmp_seq=9 ttl=64 time=0.103 ms
64 bytes from 192.168.1.103: icmp_seq=10 ttl=64 time=0.093 ms
64 bytes from 192.168.1.103: icmp_seq=11 ttl=64 time=0.106 ms
64 bytes from 192.168.1.103: icmp_seq=12 ttl=64 time=0.092 ms
64 bytes from 192.168.1.103: icmp_seq=13 ttl=64 time=0.104 ms
64 bytes from 192.168.1.103: icmp_seq=14 ttl=64 time=0.094 ms
64 bytes from 192.168.1.103: icmp_seq=15 ttl=64 time=0.106 ms
64 bytes from 192.168.1.103: icmp_seq=16 ttl=64 time=0.094 ms
64 bytes from 192.168.1.103: icmp_seq=17 ttl=64 time=0.104 ms
64 bytes from 192.168.1.103: icmp_seq=18 ttl=64 time=0.093 ms
64 bytes from 192.168.1.103: icmp_seq=19 ttl=64 time=0.104 ms
64 bytes from 192.168.1.103: icmp_seq=20 ttl=64 time=0.094 ms
--- 192.168.1.103 ping statistics ---
21 packets transmitted, 21 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.092/0.106/0.247/0.032 ms
```

Fig. 3 GMAC0 network interface to the SPI1 packet forwarding

Acknowledgements

This paper has been supported by the National Natural Science Foundations of China under grants No. 61563012, 61203109, Guangxi Natural Science Foundation No. 2014GXNSFAA118371, and foundations of Guilin University of Technology.

References

[1] Feng Dengguo, Zhao Xianfeng. Introduction to information security [M]. Beijing: Electronic Industry Press, 2014.

[2] YaXuan Qi, Li Jun. High performance network packet classification theory and algorithms [J]. Journal of computer, 2013,2 (36): 408-421.

[3] Zhongkai Xun, Huang Hao, Jin Yincheng. Computer engineering design and implementation of virtual machine firewall based on SR-IOV [J]., 2014,5 (40): 154-157.

[4] Fang TianJi. Implementation and performance optimization of IPv6 firewall with multi CPU implementation [D]. Hefei: Chinese Academy of Sciences, 2014

[5] RMI. 2007PM_XLR-Family_Program_Manual_v3.40.pdf,2006.