2012 International Conference on Image, Vision and Computing (ICIVC 2012)
IPCSIT vol. 50 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V50.24

A Dynamic Forecast Load-balancing Algorithm for High-speed Network Instruction Detection System

Yu Ying[a,*] and Deng Qidong[b]

[a] College of Computer Science,Sichuan University,Chengdu, China
[b] 2nd Squadron, Suining Detachment, Chinese People's Armed Police Force,Suining, China

Abstract. The high-speed data flow network always makes a serious performance bottleneck on intrusion detection system, leading to the detector of parallel intrusion detection system load unevenly. Packets cannot be fast through the detection system, the processing time tends to make the network more congested and packet loss rate surged high. To solve this problem, a dynamic forecast load balancing scheme for high-speed network intrusion detection system is bring forward. The program calculate the load of detector at some point by dynamic forecast algorithm and the detector can migrate spontaneously or through the control center to ensure that the load between the various detectors are basically the same. In the meanwhile, the load balancing based on connection strategy can effectively maintain the attack evidence. The simulation results proved the feasibility and effectiveness of the program. It can effectively balance the load, reducing the system loss rate and node processing delay.

Keywords: load balancing; load transfer; connection remains; queue; network intrusion detection system

1. Introduction

As the rapid development of network technology and applications, the network traffic and bandwidth is fast increasing. Capabilities of single-node intrusion detection system (IDS) are limited. The IDS cannot handle the large flow of network data, so it was replaced by a number of IDS sharing network traffic at the same time. But how to distribute traffic among multiple IDS, load balancing problem in the NIDS leads to the question, it became one of the bottlenecks restricting system performance. Therefore, the research of load balancing for network intrusion detection systems in a high-speed network environment has become a major research. [1]

In 2002, Christopher puts forward a parallel intrusion detection system model which is based on the rapid network development of the first time. [2] The system uses the Scatterer, Slicer and Reassembler, a three-tier architecture, which brings a new design method for the following load model. Charitakis and his partners figured out the dispatcher on the local cache and early filtering can improve the performance of parallel intrusion detection system [3], but the local cache may cause the packet to reach the detector in different order with the original order, the early filter detector will not see all the data packets. Judd, proposed a traffic classification algorithm based on the program by a variety of flow indicators. [4] However, while it reduces the high false alarm rate, but cannot improve the performance of NIDS. In their experiments, the prototype system's processing capacity is only 20Mbps.

In the high-speed network load balancing approach must be integrated into several elements: (1) keep dynamic load balancing between each detector during a long time, (2) maintain the attack evidence, (3) less time and space complexity of the algorithm, (4) less load pressure and burden from the feedback of each load balancing algorithm for detecting engine 5 the robustness when facing the event of invasion attack and

* Corresponding author.
E-mail address: shadow4460@163.com.

so on. This paper took the above factors into account, in the case of connection-based, dynamically balanced the load on the detector. The system's overall processing rate has been effectively improved. Experiments show that the model under high-speed networks has better load balancing effect.

2. Parallel Instruction Detection System Model

Figure 1 shows the architecture of parallel instruction detection System. Control center assigned data packets to the appropriate detector based on a connection-based distribution strategy after it receives the network data packets. Control center can regularly collect data to understand the load information of each detector and control the load on each detector through a scheduling algorithm so as to reach the detector on the effect of load-balancing.

Network attacks can be divided into single-link attack and multi-link attack. Since all packets distributing is based on connection, so for the single-link attack, the evidence of all the attacks exists in one connection, it will not lost evidence as long as we sent packets that belong to the same connection to the same detector. For multi-link attacks, according to the model, detection depends on the connection of the packet and the load of the detector, each detector can still detect the attacks independently. [5]

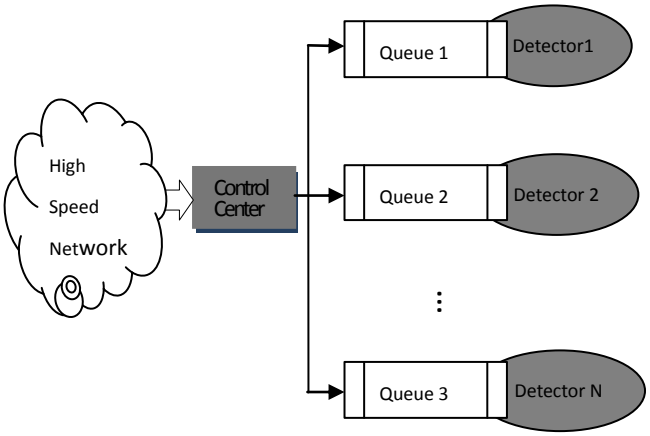

Figure 1. Architecture of parellel instruction detection system

3. Data Distribution Strategy

As the high-speed network environment, the length of the packet queue of each detector should be a primary consideration in the load.

3.1. Calculation of Detector Load

We use the following prediction algorithm to determine the load status of the detector.

Definition 1 we use the end of the first scheduled time as beginning, gather statistics of the packet queue in the 0 ~ ε period of time. Assume P packets into the queue; Q packets out of the queue; the queue has N packets during ε times; p_i is the bytes of the No.i data packet into the queue; q_j is the bytes of the No.j data packet out of the queue; n_k is the bytes of the No.k data packet in the queue.

In 0 ~ ε period of time, the input bytes of the queue is $B_{in} = \sum_{i=1}^{P} p_i$; the output bytes of the queue is $B_{out} = \sum_{j=1}^{Q} q_j$; system processing capacity per time unit is

$$D = (B_{out} - B_{in})/\varepsilon = (\sum_{j=1}^{Q} q_j - \sum_{i=1}^{P} p_i) \Big/ \varepsilon \quad (1)$$

The bytes in queue at time ε is $B_\varepsilon = \sum_{k=1}^{N} n_k$, the average package size is $\bar{b} = \frac{B_\varepsilon}{N}$, the queue length is $l_\varepsilon = N$.

According to above information, predict that the byte size of queue at time t is $B_t = B_\varepsilon + D(t-\varepsilon)$, the queue length is

$$l_t = B_t/\bar{b} = \left[B_\varepsilon + D(t-\varepsilon)\right]/\bar{b} = N\left[1+\left(\sum_{j=1}^{Q} q_j - \sum_{i=1}^{P} p_i\right)(t-\varepsilon)\Big/\varepsilon\sum_{k=1}^{N} n_k\right] \quad (2)$$

Definition 2 S_t is the ratio of data packet length that takes up of the total length of the detector queue at time t. n is the total length of the queue, then

$$S_t = l_t/\eta \quad (3)$$

Definition 2 L_t is the load of detector at time t, said that

$$L_t = k_1E_t + k_2M_t + k_3S_t \quad (4)$$

Where, E_t is the detector CPU occupancy rate at time t, M_t is the detector memory occupancy rate at time t, k_1, k_2, k_3 are respectively the dynamic adjustment factor of E_t, M_t and Q_t, and

$$\sum_{i=1}^{3} k_i = 1 \quad (5)$$

At high-speed network, k_1 accounts for the main size, CPU and memory load has little effect on the share, and taking into account the complexity of the algorithm, we can make $E_t = E_\varepsilon$, $M_t = M_\varepsilon$.

The coefficient should be determined by experimental analysis and comparison based on specific traffic environment.

3.2. Identification Method of Connection

Connection-based load balancing can effectively maintain evidence on one detector, which can improve the detection rate. A data packet quintuple <*PN*, *SIP*, *SPT*, *DIP*, *DPT*> can distinguish between two communication nodes of a connection, where *PN* is the protocol number, *SIP* and *DIP* are the source IP address and destination IP addresses, *SPT* and *DPT* are the source port number and destination port number.

The establishment of TCP connection needs a "three-way handshake," termination of the connection need to exchange FIN and ACK data packets. After a simple analysis we can find, two SYN packets began a connection, while two FIN packets terminate a connection. Therefore, when there are two SYN packets, we can say that connection is into the active state, and when there are two FIN packets, the connection enters the closed state. In the TCP protocol, if there is an exception, a RST packet is used to terminate it, so once a RST packet is found, the connection also enters the closed state. For data packets of connectionless protocols (UDP, ICMP, etc.) or TCP protocol does not complete the connection, we use record item “connection time” in the table to delete the connection record in time. Each connection has three states, 0 for off, 1 for intermediate state, and 2 for connected, a SYN leads the state plus 1, a FIN leads the state minus 1, a RST leads the state minus 2. [5]

Algorithm uses two tables. Table T is used to map a TCP connection to the detector, handling a set of TCP connection detector indexes. Index numbers start at 1, 0 means no distribution of any detector to deal with this group of TCP connections. Table U are used to map a set of UDP connections to detectors, each element of U is the detector index. Algorithm is described as follows:

```
if (packet type! = TCP | | packet type! = UDP) {
   forward the packet to the next sequence detector
}else {
    h = XOR (PN, SIP, DIP, DPT)
    if (packet type == TCP) {
       if (T [h]. sensor == 0) {
            // Forward the packet to the next detector s
            T [h]. Sensor = s
       } else {forward the packet to the T [h]. sensor}
       if (set SYN packet) {
           T [h]. State = T [h]. State +1
       }
       else if (set FIN packet) {
```

```
            T [h]. State = T [h]. State-1
        }
        else if (set RST packet) {
            T [h]. State = T [h]. State-2
        }
        if (T [h]. state == 0) {
            T [h]. Sensor = 0
        }
    } else {
        if (U [h] == 0) {
            // Forward the packet to the next detector s,
            U [h]. Sensor = s
        } else {
            forward the packet to the U [h]
        }
    }
}
```

3.3. Load Balancing Strategy

In this model, you can categorize detector load into light, medium and overloading 3 statuses. Based on threshold model, the definition of the detector load $Lt \geq \alpha M$ for the overload condition, $Lt \leq \beta M$ for the light load condition, $\beta M < Lt < \alpha M$ for moderate state (where α is the overload factor, β coefficient for the light load, M is the load capacity).

When the detector load $Lt \leq \beta M$ at time t, it represents the load is light and the detector can be a recipient of the load transfer; when $\beta M < Lt < \alpha M$, it represents detector load is moderate, all of the tasks currently in the local node can be digested, but do not allow the detector to receive the load transfer; when $Lt \geq \alpha M$, said detector of heavy load, will stop in task allocation, load to be migrated.

In this paper, a two-stage scheduling strategy of marking distribution and balanced scheduling is adopted to achieve rapid distribution of data packets and dynamic load balancing between detectors. Detector load is divided into light load, medium and overloading 3 states. Tasks can be digested in the local in light or moderate state, the queue is allowed to receive data packets, but overloaded detectors have much pressure, need to pause a while to receive. Initial state of detectors are light load, the load balancing based scheduling phase changes, combined with dynamic predictive model to markup queue status, guide the implementation of marking distribution.

The system will queue each detector arranged by serial number, save the queue sequence that is last distribution for a connection, and save the identified state for each queue. Light and moderate queue identified as available, overloaded queue marked as suspended. When receiving a new connection, use Round-Robin algorithm and select the detector queue of next Sequence, if the queue is in available state, the new connection can send packets to the queue for processing; otherwise continue to find the next queue. Using the polling method, need not to real-time control the queue load, just to understand the current state of the queue as to reduce the complexity of the algorithm and improve the system of data packet reception. Algorithm is as follows:

```
P_DISPATCH(Message msg)
index=Sequence
while(true)
index=(index++)%QUEUE_SIZE
if(Queue[index].state as available)
transfer(msg,Queue[index])
```

At dynamic load balancing stage, to achieve load balancing between the packet queues, the queue automatically apply for scheduling or control center schedule the queue in time. Queue apply for an scheduling indicates the packet queue calculate it's information in time ε and predicts the load at time t in the load calculation model, if the judge is reported to the center of the queue overload, the center collected the packet queue load and balance scheduling; center time scheduling indicates after the calculation, did not find the queue overload, the centre took the initiative to acquire queue load in the time T, schedule to achieve load balancing among queues. The overloaded queue marking and load transfer strategy realize the load

scheduling and load balancing in system [6]. It supports dynamic load balancing, and it improves the overall system performance.

When any of the predicted detector queue is overloaded, they reported to the control center. The control center collect all the detector load Lε and Lt, Lt judge the detector load status, the detector according to the state marked as overload, moderate and light-load, overload detector does not accept packets input, until the state is Marked as moderate or light load so far. Marked by the overload can avoid data packets in the overloaded detector queue continue to increase, leading phenomena such as the input data packet drop occurs. Meanwhile, according to the current load Lε, the control center sort the queue from overloaded to light load. Transfer load between the heaviest and the lightest load queue, the data packets that in excess of the mean load of the two will be transferred from the heaviest load queue to the lightest load queue. Migration is successful; two queues are basically in a moderate state of the load. System continues to perform this step until all the overloaded queues are completely transferred. After load balancing, the control center tag the system time to zero, start the next scheduling cycle again. Algorithm is as follows:

```
ACTIVE_SCHEDULE (Queue [] Q)
Collect Lε and LT of all Q
for each q in Q
if (LT is overloaded) mark state suspended
else mark state available
Sort Q by Lε asc
i = 0, j = QUEUE_SIZE-1
while (Q [j] overload)
Q [j] to Q [i] overload migration
i ++, j --
```

In a T period of time after last scheduling, the load are calculated and predicted through a number of ε time. Overloading is not found and the system enters center time scheduling. The purpose for center time scheduling is to balance load between light and moderate detector queue to increase load of the light load queue even some zero packet queue. So the system efficiency and utilization are improved. After load balancing, the control center tag the system time to zero, start the next scheduling cycle again

4. Experiment

In order to prove the validity of Load balancing algorithm of this article, we build a system simulation platform based on the Architecture of Parallel Instruction Detection System as shown in figure 1. The system uses multi-core network processing platform OCTEON CN38XX series designed by security processor vendor Cavium Networks, and its basic configuration is as follows: 16 cn-MIPS64 processor, 550MHz frequency, 4G memory, eight 1000Mb / s Ethernet ports, 250G hard disk. The core test platform, distributed as follows, core0 runs control center, core1 ~ core5 run five detectors, each detector on the same SNORT load configuration.

In this experiment, the MIT Lincoln Laboratory for 99 years on Tuesday, the fourth week of TCPDUMP flow data as a parallel NIDS test data, the use of high-speed playback TCPREPLAY simulated high-speed network environment. Through repeated experiments, the platform has been the most suitable experimental parameters: dynamic adjustment coefficient k 1 = 0.81, k 2 = 0.08, k 3 = 0.11; detector light load is 10%, 80% overload, active scheduling time interval ε = 1000 ms, center regularly scheduled intervals T = 10 000 ms. Select the system state at a time beginners, 10ms interval for the data packets input, task completion time shown in Figure 2.

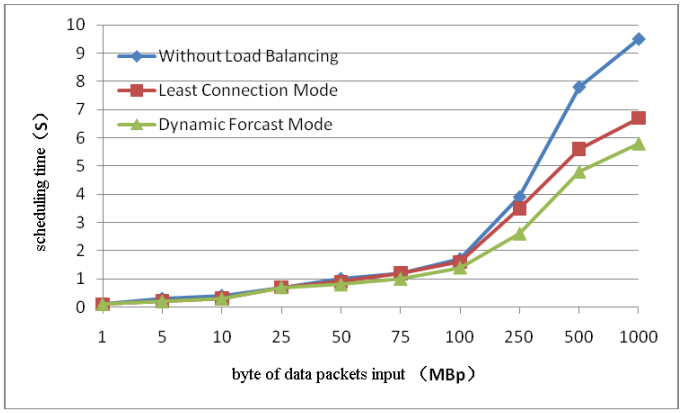

Figure 2. Comparison of processing time

Can be seen, when the number of bytes of input data packets are less (PACKET_SIZE ≤ 25), all of the detectors are in light load condition. The minimal connection model of traditional programs and a dynamic forecasting model of the system programs are not scheduled on the detector. The results of three kinds of strategies are similar; increasing in the input data packet, the state of detectors are moderate (50 ≤ PACKET_SIZE ≤ 100), the dynamic prediction model of program scheduling on the queue for load balancing, and since then the detector load is moderate, and does not meet the minimum connection threshold scheduling model, the least connection scheme did not schedule anything, so the best treatment effect is dynamic prediction model; further increasing the input data packet, part of the detector gradually into the overload state (PACKET_SIZE ≥ 250), the minimum connection model programs and dynamic prediction model programs have begun scheduling between detector, can be seen from the data two kinds of programs are better than the scheme that do not use balanced scheduling, in which the prediction model program a little better in relative performance. At the same time as the program for a more fine-grained packet based on byte size load forecasting, the load state accuracy is higher.

5. Summary

In this paper, we discussed the elements in the design and implementation for parallel NIDS load balancing algorithm, established a dynamic prediction model, used prediction algorithm to predict the load of the detector, and reduced packet processing time through the coordination of two-stage scheduling between the detector load in the system. Simulation results show that the proposed scheme can effectively reduce the packet processing time under the high-speed network intrusion detection system and improve the load balance level. The Load Balance Scheduling Algorithm reduced the burden of each detector .Load balancing, attack evidence to maintain, high efficiency also been assured. The algorithm is good coordination of these elements .the load balancing capacity is much better than the traditional-based algorithm for the least connection. Taking into account the experimental parameters of the algorithm is more dependent on experience and experimental environment; the next step work will focus on adjustment so as to have better stability and robustness.

6. References

[1] Tian Ye, Zhang Yujun. Multithreading for load balancing in Network Intrusion Detection System[J]. Microelectronics & Computer, 2006,23(3) :65-69

[2] Christopher K, Fredrik V, Giovanni V, Richard K. Stateful intrusion detection for high-speed networks//Proceedings of the IEEE Symposium on Security and Privacy. Washington,USA, 2002: 285~293

[3] Charitakis I, Anagnostakis K, Markatos E. An active traffic splitter architecture for intrusion detection//Proceedings of the 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems. Orlando, USA, 2003: 238-241

[4] Judd J D, McEachen J C, Michael J B, Ettlich D W. Network stream splitting for intrusion detection//Proceedings of the 11th IEEE International Conference on Networks. Sydney, Australia, 2003: 525-530

[5] Lai Haiguang, Huang Hao, Xie Junyuan. PABCS: A Traffic Partition Algorithm for Parallel Intrusion Detection)[J]. Chinese Jounal of Computers, 2007，30(4)：555-562

[6] Ji Zhihui, Ni Hong, Liu Lei. Research and Application of Dynamic Message Queue[J].Computer Engineering, 2009,35（8）：31-37

[7] Wang H, Xie H, Qiu L,et al. Cope: traffic engineering indynamic networks [J].ACM SIGCOMM Computer Communication Review, 2006,36(4):99-110.

[8] Kandula S, Katabi D, Sinha S. Dynamic load balancingwithout packet reordering[J].ACM SIGCOMM ComputerCommunication Review, 2007,37(2):51-62.

[9] Levy H, Zlatokrilov H. The effect of packet dispersion on voice applications in IP networks[J].IEEE Transaction on Networking, 2006,14(2):277-388.

[10] Rossi P S, Romano G, Palmieri F,et al. Joint end-to-end loss-delay hidden Markov model for periodic UDP traffic over the Internet[J].IEEE Transaction on Signal Process, 2006, 54(2):530-541