Self-Adaptive Frequency Scaling Architecture for Intrusion Detection System

Qiuwen Lu[1,2,4], Zhou Zhou*[,2,4], Hongzhou Sha[2,3,4], Qingyun Liu[2,4], and Hongcheng Sun[1]

[1] Beijing University of Chemical Technology, Beijing, China
luqiuwen@nelmail.iie.ac.cn, sunhc@mail.buct.edu.cn
[2] Institute of Information Engineering, Chinese Academy of Science, Beijing, China
{zhouzhou, liuqingyun}@iie.ac.cn
[3] Beijing University of Posts and Telecommunications, Beijing, China
shahongzhou@nelmail.iie.ac.cn
[4] National Engineering Laboratory for Information Security Technologies
Beijing, China

Abstract. Recently, Intrusion Detection Systems (IDS) have been deployed in the Internet for information security. Nevertheless, with the growing of Internet traffic, IDS becomes increasingly complicated and consumes much more energy. Existing studies concentrate on saving energy, but do not adapt to the change of network traffic. In this article, we propose a new method to adjust the frequency of IDS's devices' main processors automatically based on the prediction of the network traffic. It calculates optimal frequency scaling operation sequence via an internal sandbox model, so as to achieve energy saving purposes. Experiments show that our method can save 60% power consumption generally.

Keywords: Green IDS, Sandbox Model, Dynamic Frequency Scaling

1 Introduction

As the Internet traffic is growing up rapidly over time, the number of intrusion detection systems (IDS) devices in a trustworthy service is increasing [1]. Massive equipment run at their full rate all the time which consume large amounts of electrical energy. It causes huge waste of energy and brings the opportunity for the energy reduction since current link utilization is 30% in average and below 45% in peak [2]. This causes a huge waste of energy and bring the opportunity for reducing the energy consumption.

A lot of power management approaches have been proposed for substantial conservation in energy consumption, and can be classified into two categories: *smart standby* and *dynamic power scaling*. *Smart standby* methods maximize energy conservation by letting them fall asleep during the gaps of packets. However, these approaches may block the service called "network presence" that need to maintain network connectivity [3]. *Dynamic power scaling* approaches adapt the capacities of devices dynamically according to the actual or predicted traffic [4]. Some algorithms of them [5] switch frequency using fixed parameters given by users, which cannot adapt to the variational

* Corresponding author

network traffic. Therefore, it is too complex for users to use these power scaling algorithms in practice.

To address this issue, we propose a *Self-Adaptive Dynamic Frequency Scaling Architecture* (SAFS) to reduce energy consumption. It predicts future network traffic based on historical data, and all the possible operation sequences which can be applied to the main processors are enumerated and sent to sandbox model with the predicted result. Then, the best operation sequence which be evaluated by an objective function will be applied. In this way, our approach is (1) self-adaptive to the network traffic (2) can keep a good trade-off between network performance and energy saving. Experiments indicates that our method can save around 60% power consumption. We make the following contributions in this paper.

- First, we introduce a simplified sandbox model which simulates the procedure of packet receiving and processing in network devices.
- Secoud, we propose a novel scheme to tune the frequency of the main processor automatically based on the sandbox model.

2 Related Work

Many researchers concentrate on the method of reducing the energy consumption of network devices. These methods can be classified as two groups, as *smart standby* and *dynamic power scaling*.

Smart standby lets the complements of devices fall into sleep to reduce the energy consumption. M. Gupta and S. Singh [6] proposed a method by putting idle ports to sleep to save energy, but the method could only be effective when the device utilization rate is under 10% according to [7]. G. Ananthanarayanan *et al.* [8] proposed a novel architecture for buffering ingress packets using shadow ports in low-power state.

Dynamic power scaling tune the frequency of the complements in devices dynamically to achieve the same goal. C. Gunaratne *et al.* [5] firstly raised the approach to tune the link rate of ethernet port. W. Meng *et al.* [9] proposed an approach by tuning the frequency of main processors. This method aimed to reduce the energy consumption by decreasing the ability of processors, which has been implemented on NetFPGA.

3 Self-Adaptive Frequency Scaling

In this section, we propose our approach in detail. The overview of our method are illustrated in Section 3.1 and the complements are proposed in Section 3.2–3.5.

3.1 Overview

Our method is based on the idea of local optimization to adapt to the variation of network traffic. Different from dual-threshold method which set the fixed parameters on the buffer, our approach introduces a sandbox model to predict the behaviour of the system based on the predicted network traffic, then an optimized operation sequence applied in future will be concluded. As the traffic changing, the operation sequence can

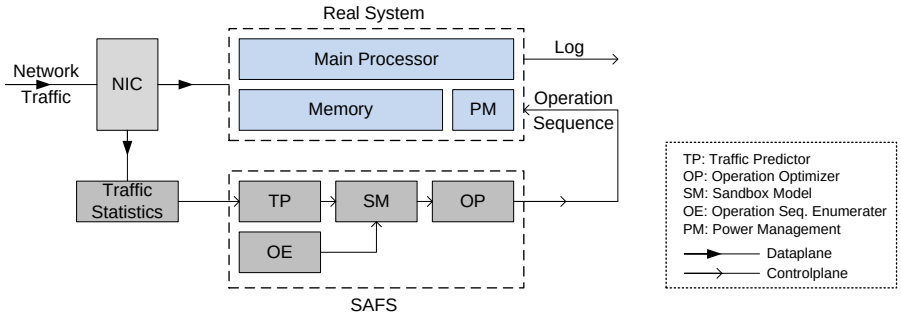


Fig. 1. The Architecture of the SAFS

be recalculated to adapt to the future traffic. Therefore, our method can adapt to the traffic automatically.

Fig.1 illustrates the architecture of SAFS which can be which can be divided into 5 parts: real system, traffic predictor, operation sequence enumerator, sandbox model and operation optimizer. The whole system works as follows:

- As the network traffic being processed by real system, the statistical information of network traffic can be sent to the traffic predictor, and the traffic predictor predicts the trend of future network traffic;
- The operation sequence enumerator enumerates all the possible operation sequence and sends the result to the sandbox model with the predicted traffic trend. Then the sandbox model evaluates these sequences parallelly.
- The operation optimizer chooses the best sequence according to their performance and apply it in the real system.

3.2 Traffic Predictor

To calculate the operation sequence, we need to know the trend of the future traffic. Currently, ARMA (Autoregressive Moving Average) and FARIMA (Fractional Autoregressive Integrated Moving Average) [10] time series are the main model used to fit and predict the network traffic, which is well known and have good performance. Therefore, our work will force on other parts of the architecture.

The output of the traffic predictor N_t is a sequence in time,

$$N_t : \{N_{t^*}, N_{t^*+\Delta s}, N_{t^*+2\Delta s}, \ldots, N_{t^*+(m-1)\Delta s}\} \tag{1}$$

where Δs is the sample interval and have the length of m. t^* is start time of the sequence.

3.3 Operation Sequence Enumerator

The operation sequence enumerator gets all the possible frequency $F_{allow} = \{f_1, f_2, \ldots, f_n\}$ that the main processor can be set, and enumerates the operation sequence in time of

$t_1, t_2, \ldots, t_m$,

$$
\begin{aligned}
F_1 &= f_1, f_1, f_1, \ldots, f_1 \\
F_2 &= f_2, f_1, f_1, \ldots, f_1 \\
&\vdots \\
F_n &= f_n, f_1, f_1, \ldots, f_1 \\
F_{n+1} &= f_1, f_2, f_1, \ldots, f_1 \\
&\vdots \\
F_{m \times n} &= \underbrace{f_n, f_n, f_n, \ldots, f_n}_{m}
\end{aligned}
$$

and we define an function $F_n(i)$ as the ith item of sequence F_n.

3.4 Sandbox Model

The sandbox model is used to evaluate the behaviour of the main processor in the real system with the particular network traffic and frequency sequence by simulation. The simulator in the sandbox model simulates the Producer-Consumer(P-C) system which exists widely in network devices. Packets came from the network are received and push it into a FIFO queue, then the worker pop the packets from FIFO queue and process the packets.

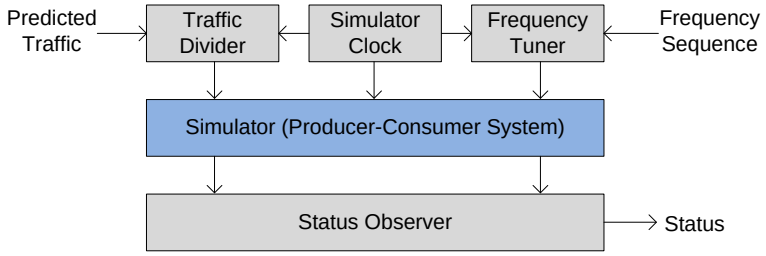


Fig. 2. The structure of sandbox model

The implementation of the sandbox model can be described as follows:

1. Get the simulator time t and Δt. Get the operation sequence F and the traffic sequence like Eq.(1).
2. Let $t^* = t$, get the traffic for the correct simulator time using traffic divider which divide the traffic N_{t^*} by $1/\Delta t$.
3. Get the ith item of the sequence in F as $F(i)$, and the frequency tuner set the processing capacity as $F(i)$.
4. Send $\hat{N}(t)$ into the packet generator. The packet generator generates the packets with length of $L(1), L(2), \ldots, L(n)$ until $\sum_{i=1}^{n} L(i) = \hat{N}(t)$.
5. The packets generated from packet generator is pushed into the FIFO buffer. and pop the $W(t)$ packets from FIFO buffer.

6. Let the simulator clock t equals $t + \Delta t$, and goto the step 3. When the t can be divided with no remainder by Δs, goto the step 2.

The system status observer can get the status of the system at each t. These status can be used to evaluate the performance of operation sequence, which will proposed in Section 3.5.

3.5 Operation Optimizer

Operation Optimizer selects the best operation sequence from the result of sandbox model and apply the sequence in the real system. The performance grade the sandbox model gives are listed below.

- *Energy Saving*(E): We use the frequency to evaluate the energy saving instead of actual power, according to the linear relationship between energy consumption and the frequency. Thus, the energy consumption of devices could be modeled as

$$E = f_1 T_1 + f_2 T_2 + f_3 T_3 + \cdots + f_n T_n \tag{2}$$

  where E is the energy consumption, $f_1, f_2, f_3, \ldots, f_n$ represents the frequency the processor can be tuned, and $T_1, T_2, T_3, \ldots, T_n$ denote the time spent in each frequency.
- *Switch Times*(S): The switch of frequency in processor does not come without cost. The transition between frequencies brings the extra performance lost. So we treat the switch times as one of the objectives to evaluate the scaling method.
- *Packet Loss*(P_{lost}): The count of packet loss in our devices is an important grade to evaluate reliability. The packet loss would be unexpected.

We use an objective function to combine all the performance metrics as

$$f(E, S, P_{lost}, Q_l) = \begin{cases} \alpha E + \beta P_{lost} + \theta S, & Q_l < kQ_{max} \\ \alpha' E + \beta' P_{lost} + \theta' S + \xi' Q_l, & Q_l \geqslant kQ_{max} \end{cases} \tag{3}$$

with $\alpha + \beta + \theta = 1, \alpha' + \beta' + \theta' + \xi' = 1$, where Q_l is the length of the FIFO queue, and Q_{max} is the max length of the queue. $\alpha, \beta, \theta, \alpha', \beta', \theta', \xi'$ and k are the parameters of the objective function given by users.

Operation Optimizer gets all the sequence $F_1, F_2, F_3, \ldots, F_n$ and calculates the performance grade as Eq.(3). The best operation sequence $F|_{\min f(\cdot)}$ will be used in real system.

4 Experiment

4.1 Data Collection and Performance Metrics

We captured two cases of traffic spanning over a period of time of a total day. One of them is from the Internet link of a research institute and another is from the gateway of an office. The characters of traffic cases are listed in Table 1.

There are three performance metrics for the devices that we studies which has been proposed in Section 3.5.

Table 1. Character of Traffic Cases

Name	Traffic Type	Sources	Bandwidth (Mbps)	Avg. Bit Rate(Mbps)	Avg. Pkt Length(B)
Case I	Core router	Institute	10,000	2,770.60	843
Case II	Home router	Office	100	6.68	913

4.2 Experiment Setup

Our simulated experiments are finished on a PC. According to our datasets, we start the simulate at time in minutes of $t = 0$ and end up with $t = 1250$, which obtain the daily dynamic traffic pattern. The simulation clock step Δt are set as 0.1min to reduce the cost of computing for the limited resource.

Table 2. The parameters of the simulator used in experiments

Simulator	CPU Frequency (pkts/Δt)	Queue Length (pkts)	Physical Interpretation
A	32,128	1,024	Simple CPU with small memory
B	32,128	2,048	Simple CPU with medium memory
C	32,128	4,096	Simple CPU with large memory
D	32,128	8,184	Simple CPU with huge memory
E	32,64,96,128	1,024	Complex CPU with small memory
F	32,64,96,128	2,048	Complex CPU with medium memory
G	32,64,96,128	4,096	Complex CPU with large memory
H	32,64,96,128	8,184	Complex CPU with huge memory

Table 2 lists the sets of parameters of the simulator, which represents the devices with different memory and CPUs to test the performance of the methods. The up-threshold of dual-threshold (DT) scaling method is set as Eq.(4) and the down-threshold is set as Eq.(5).

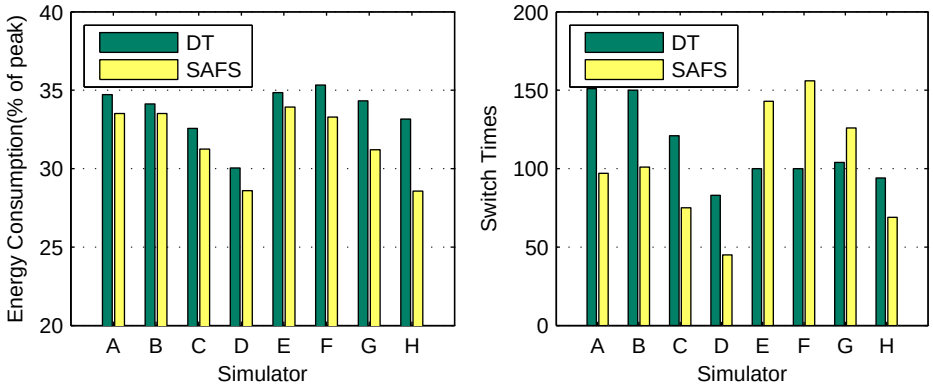
$$0, \frac{Q_l}{n} + \Delta L, 2\frac{Q_l}{n} + \Delta L, \ldots, (n-1)\frac{Q_l}{n} + \Delta L, Q_l \tag{4}$$

$$0, \frac{Q_l}{n} - \Delta L, 2\frac{Q_l}{n} - \Delta L, \ldots, (n-1)\frac{Q_l}{n} - \Delta L, Q_l \tag{5}$$

where Q_l, n are the length of the queue and the count of CPU frequencies for each simulator. In the following experiments, the ΔL is set as 10, and the parameters of Eq.(3) are set as $\alpha = 0.4, \beta = 0.2, \theta = 0.4$, $\alpha' = 0.1, \beta' = 0.3, \theta' = 0.2, \xi' = 0.4$ and $k = 0.8$ based on their importance of metrics.

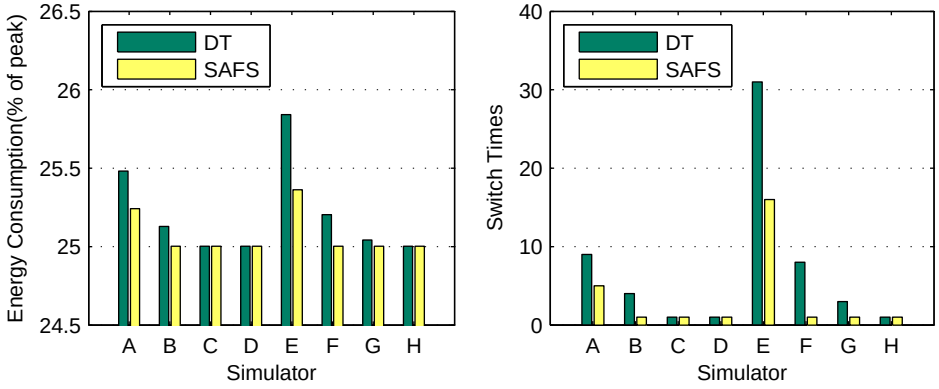
4.3 Result

Fig.3 compares the performance of quad-threshold method (DT)[5] with SAFS in the Traffic Case I (Institute), which illustrates that SAFS has better performance with near 60% energy saving comparing with peak in the huge memory simulator. Fig.3(a) shows the energy consumption of each method in each simulator, note that SAFS has good effect in Simulator C, D, F, G, H which have large memory and complex CPUs. Fig.3(b)



(a) Energy Consumption (b) Switching Times

Fig. 3. The performance of SAFS and DT[5] in Traffic Case I(Institute)



(a) Energy Consumption (b) Switching Times

Fig. 4. The performance of SAFS and DT[5] in Traffic Case II (Office)

shows the times of frequency switch, which SAFS have better performance in Simulator D and H than DT. When it comes to the packet loss, our method lost on average of 633 packets at Simulator A, B, E, H, and both our method and dual-threshold method lose no packets in Simulator C, D, F, G, H.

Fig.4 shows the performance of SAFS in Traffic Case II (Office). SAFS has better performance on the energy consumption and switch times in Simulator A, B, C and E. In all the simulator, both SAFS and DT have no packet loss. SAFS and DT have the same performance in Simulator C, D, G, H because the traffic of Traffic Case II is too small for them which represents the machine with huge memory and have good computing power.

5 Conclusion

This paper is to design power scaling mechanism named by SAFS for energy efficient IDS devices, by adjusting frequency of the main processor in devices dynamically. Specially, SAFS employs a sandbox model firstly to evaluate the performance of the operation sequences that may be applied in real system, thus select the best one and finally use it in the next time period. Experiments indicate that SAFS effectively decreases the energy consumption of the main processor and the switching times. Consequently, our approach, SAFS, is much more effective for energy conservation in IDS devices.

Acknowledgments. This work was supported by The National Science and Technology Support Program (Grant No. 2012BAH46B02); The Strategic Priority Research Program of the Chinese Academy of Sciences under (Grant No. XDA06030200); The National Natural Science Foundation (Grant No. 61402474).

References

1. Di, P.R., Mancini, L.V.: Intrusion detection systems. Volume 38. Springer (2008)
2. Chabarek, J., Sommers, J., Barford, P., Estan, C., Tsiang, D., Wright, S.: Power awareness in network design and routing. In: INFOCOM 2008. The 27th Conference on Computer Communications. IEEE. (2008)
3. Bolla, R., Davoli, F., Bruschi, R., Christensen, K., Cucchietti, F., Singh, S.: The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization? IEEE Communications Magazine **49**(8) (2011) 80-86
4. Song, T., Shi, X., Ma, X.: Fine-grained power scaling algorithms for energy efficient routers. In: Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems, ACM (2014) 197-206
5. Gunaratne, C., Christensen, K., Nordman, B., Suen, S.: Reducing the energy consumption of ethernet with adaptive link rate (alr). Computers, IEEE Transactions on **57**(4) (2008) 448-461
6. Gupta, M., Singh, S.: Greening of the internet. In: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, ACM (2003) 19-26
7. Reviriego, P., Christensen, K., Rabanillo, J., Maestro, J.A.: An initial evaluation of energy efficient ethernet. Communications Letters, IEEE **15**(5) (2011) 578-580 Communications Letters, IEEE.
8. Ananthanarayanan, G., Katz, R.H.: Greening the switch. In: Proceedings of the 2008 conference on Power aware computing and systems, USENIX Association (2008) 7-7
9. Wei, M., Yi, W., Chengchen, H., Keqiang, H., Jun, L., Bin, L.: Greening the internet using multi-frequency scaling scheme. In: Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on, IEEE (2012) 928-935
10. Yantai, S., Zhigang, J., Lianfang, Z., Lei, W., Yang, O.W.W.: Traffic prediction using farima models. In: Communications, 1999. ICC '99. 1999 IEEE International Conference on. Volume 2. (1999) 891-895 vol.2