An Improved 30 Gbps-Class Large-Capacity Packet Processing Method Using Core Isolation Technology

Youngsun Kwon[1], Moonhee Son[2] and Hoon Chang[1,*]

[1]*Dept. of Computer Science and Engineering, Soongsil University, Republic of Korea*
[2]*Dept. of Technical Research Center, Entobilsoft, Inc., Republic of Korea*
E-mail: dolphini0727@naver.com; muljamjari@entobilsoft.com; hoon@ssu.ac.kr
[*]*Corresponding Author*

Received 22 October 2021; Accepted 20 February 2022;
Publication 30 June 2022

Abstract

With the spread of 5G services and the development of IoT technology, network traffic for information delivery is increasing in capacity. As network traffic increases, cyber threats also increase, resulting in an increasing importance on traffic analysis. The existing packet processing engine generates a signature by analyzing the characteristics of the attack after the occurrence of suspicious traffic, and based on this, it is difficult to properly respond to new and variant attack traffic because a manual response method is performed to detect the same attack. In addition, even during a network operation, only analysis results generated by passive filtering appear, and when abnormal or suspicious traffic is observed, the quality of the report is often affected by the analysis capability of the administrator. The packet processing method proposed in this paper applies the core isolation method to the NUMA structure applied to the existing 20 Gbps packet processing engine to increase the accessibility of the existing NUMA memory structure and lower the packet drop rate to enable high-capacity 30 Gbps traffic processing. Using

Journal of Mobile Multimedia, Vol. 18_6, 1497–1512.
doi: 10.13052/jmm1550-4646.1862

the proposed processing engine, it is possible to determine the degree of possibility of abnormal traffic, preferentially by a quick analysis of suspicious traffic rather than a detailed analysis of traffic.

Keywords: Network, traffic analysis, packet processing, core isolation, huge page.

1 Introduction

Network traffic is rapidly increasing with the development of ICT technology. According to a report released by Cisco, personal IoT devices will increase from 2.4 to 3.6 per person from 2017 to 2022, and the monthly global average IP traffic is expected to more than triple from 122.4 exabytes to 396 exabytes in 2022 [1]. As network traffic increases, cyber threats through networks also increase [2]. Recently, network traffic has been encrypted, making it difficult for network administrators to identify or analyze the payload in a packet [3].

In the case of the existing traffic analysis system, when suspected attack traffic occurs, the characteristics of the attack are analyzed post-mortem to create a signature, and based on this, the manual response to detect the same attack is the main focus. By combining these detection results with the firewall system at the entrance of each local domain, a manual response method is implemented to block the attack traffic flowing into each local domain [4]. In this case, because the detection range and performance are based on the signature of the attack, it is difficult to respond appropriately when using a new attack method that induces new or variant traffic. Collecting and processing traffic information on a network has been performed for a long time from the perspective of network management [5]. However, this is primarily aimed at providing statistical data for traffic processed on the network in terms of network engineering and network management, and therefore, it is inappropriate to analyze traffic characteristics to detect abnormal traffic caused by attacks [6, 7].

In this paper, we propose a method that can handle 30 Gbps-class high-capacity traffic. By improving the NUMA structure used in the existing 20 Gbps-class traffic processing method, some cores are forcibly allocated to only process packets, thereby reducing the load on the system and increasing the packet processing speed. Using the proposed processing engine, it is possible to determine the degree of possibility of abnormal traffic, preferentially by a rapid analysis of suspicious traffic rather than a detailed analysis of traffic.

The remainder of this paper proceeds as follows. Section 2 describes the memory design technique and various traffic analysis methods for processing the existing large-capacity traffic as a related study. Section 3 describes a method for processing high-capacity traffic of a 30 Gbps class. Section 4 reports the experiments and results, and Section 5 provides the conclusion.

2 Related Research

2.1 NUMA Structure for Large Packet Processing

For large-capacity packet processing, non-uniform memory access (NUMA) [8, 9] is used to optimize memory parallelism when large-capacity traffic occurs with a multi-processor model in which nodes are connected to each dedicated memory, as shown in Figure 1.

In the existing Symmetric Multiprocessing (SMP) structure, because all CPUs share the memory and only one processor can access the memory at a time, other processors must wait. In contrast, in the NUMA structure, each processor has an independent local memory, and the memory access speed varies depending on the relative location of the processor. Therefore, when each processor accesses the local memory, other processors are not required to wait and can access the memory at a high speed. In addition, because the local memory and local bus close to each CPU are used, the bottlenecks and delays of the bus are reduced.

However, if multiple processors simultaneously require the same data, remote access is made to the memory of a different processor instead of the local memory. In addition, if each thread is allocated and operated in the NUMA structure using general thread scheduling, there is an unexpected possibility that it will operate in the same core as other threads, and the process of processing packets while utilizing the core increases to 100% instantaneously. There is a limit to the performance improvement of large-capacity packet processing because of the packet drop probability.

2.2 Signature-Based Traffic Analysis Method

The signature-based traffic analysis method [10, 11] is a method for analyzing traffic generated by a specific application to distinguish it from other applications. The classified applications show high accuracy, but there are disadvantages in that it is impossible to immediately respond to a newly created application because it is done manually in the process of determining a feature, and it cannot be classified unless the signature of the application

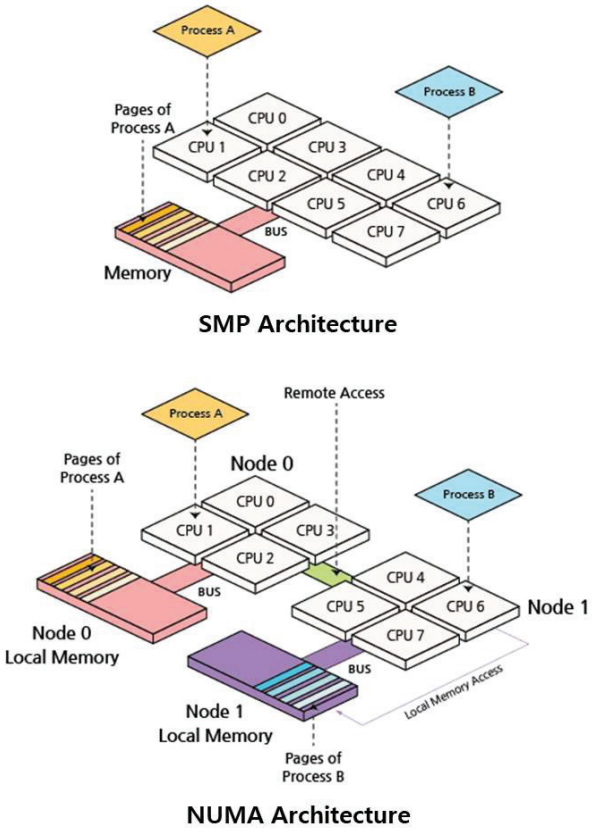

Figure 1 NUMA architecture.

can be confirmed. For example, typical packets such as HTTP and SMTP can be characterized, and examples of HTTP header information that can be extracted are listed in Table 1.

However, encrypted protocols such as HTTPS and SMTPS cannot be classified because their characteristics cannot be determined.

2.3 Machine Learning-Based Traffic Analysis Method

In the machine learning-based traffic analysis method [12], there is a method that can classify traffic by learning various items that can be characteristic of each application using various machine learning methods, as shown in Table 2.

Table 1 HTTP response header

Element	Explanation
Response Line	HTTP/1.1 200 OK
Memory	Server: nginx
	Date: Sun, 10 Apr 2016 05:08:46 GMT
	Content-Type: image/gif
	Connection: close
	Pragma: no-cache
	Expires: Tue, 01 Jan 1980 09:00:00 GMT

Table 2 Typical machine learning-based traffic classification

Element	Features Used	Purpose Protocol
Naïve Bayes	Duration, TCP Port, Interval, Size	FTP, SSH, SMTP, WWW, DNS, etc.
K-Means	Packet length mean, duration	Web, P2P, FTP, etc.
Decision Tree	Protocol, Duration, Byte volume	FTP, Telnet, SMTP, DNS, HTTP

The port number, flow duration, inter-arrival time, and packet size, which are characteristic items, are learned using machine learning models, such as classification and clustering. This method provides a high rate of classification analysis. However, the packets that have data that are not learned or different from what was previously learned cannot be classified into the same application; therefore, it is less accurate when applied to all the Internet.

3 Proposed Scheme

3.1 Architecture

The proposed packet processing engine that can handle more than 30 Gbps traffic is divided into kernel, core processing, and application that are parts that can quickly analyze the detected packet, focusing on the H/W that detects the packet, and can operate as illustrated in Figure 2.

The network interface card (NIC) driver uses dual LAN ports such that the packet detection speed can be detected over 10G. In addition, it supports multi-core processing using a hash map for 30 Gbps-class traffic processing and analysis to make the most of the system resources. In particular, CPU core isolation [13] technology is applied to prevent packet drop during traffic processing owing to the biased use of specific cores. In the NUMA structure of the existing 20 Gbps-class processing process, some cores are forcibly allocated to only process packets. For the packets detected and analyzed in

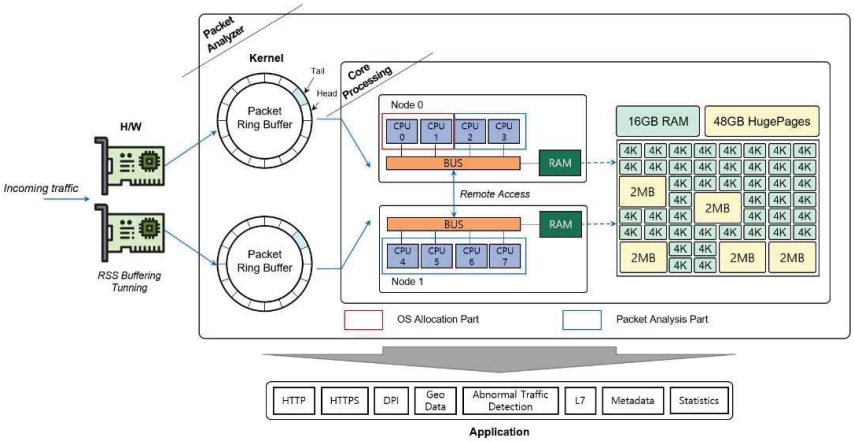

Figure 2 H/W structure for handling large-capacity traffic.

this manner, separate threads such as HTTP, GEO data, and file extraction modules are performed to handle a large number of TCP/UDP sessions. At this time, an abnormal behavior or a suspicious traffic can be checked using metadata (flow log), L7 data (web log), etc.

3.2 Traffic Processing Using Core Isolation

CPU core isolation is a function that specifies the CPU to be used by a process for each process in a server with multiple CPUs and optimizes cache performance by mapping S/W threads to H/W threads, as shown in Figure 3.

It will not be affected by various unpredictable tasks that may occur within the OS, such as when periodic input/output of the OS occurs, when the load of periodic execution registered in the crontab increases, and when CPU/memory usage by other than the packet processing unit increases. In addition, because the functions performed by each core are separated and there is no context switch between threads, the performance of the pure packet processing core is increased, and the packet drop rate owing to the load during traffic processing is reduced.

To apply the CPU core isolation, the core to be controlled by the kernel at boot time is predefined, and settings are added. If two NICs are linked to the hardware structure that connects Core 0, 1, 2, and 3 to NUMA node 0 and Core 4, 5, 6, and 7 to node 1, as shown in Figure 4, two cores on each node can be set to the packet receiver by isolation. The setting value for the tuned profiles real-time solution [14] is shown in Figure 5.

Figure 3 Core isolation architecture.

```
~]# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3
node 0 size: 16159 MB
node 0 free: 6323 MB
node 1 cpus: 4 5 6 7
node 1 size: 16384 MB
node 1 free: 10289 MB
node distances:
node   0   1
  0:  10  21
  1:  21  10
```

Figure 4 Memory NUMA node assignment.

```
isolated_cores=0-1,4-5
```

Figure 5 Applying core isolation.

```
chrt -f taskset -c 0,1,4,5 etapd
```

Figure 6 Running the packet processing daemon.

Subsequently, as shown in Figure 6, the chrt -f command is used to convert the process's thread scheduling to the FIFO format. Cores 0-1 and 4-5 are isolated using the taskset -c 0,1,4,5 command, the main packet processing daemon, etapd, is run, the core is removed from the kernel, and then the assignment is forced to the packet receiver.

In addition, the huge page technique [15] is used to increase accessibility to the memory block size of each core, as shown in Figure 7. By changing the memory block size from 4 Kbytes (OS basic) to 2 Mbytes, the probability of a page failure when using the memory can be reduced, thereby improving the access speed and reducing the access failure probability. When this is used,

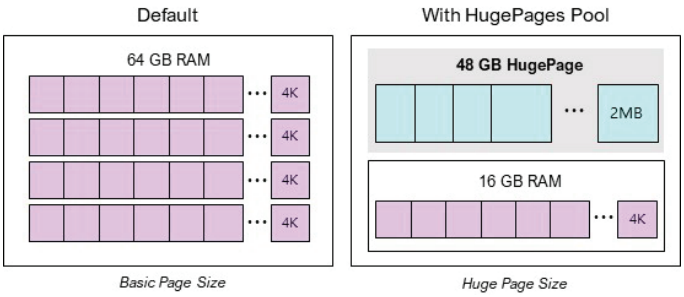

Figure 7 Apply huge page pool.

```
[root@np-234 ~]# grep Huge /proc/meminfo
AnonHugePages:     47104 kB
HugePages_Total:     256
HugePages_Free:      256
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
[root@np-234 ~]# grep Huge /proc/meminfo
AnonHugePages:    575488 kB
HugePages_Total:     256
HugePages_Free:       12
HugePages_Rsvd:       12
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

Figure 8 Before and after applying huge page pool.

the size of the memory processed can be confirmed, as shown in Figure 8. Notably, the processed memory size is improved by approximately 12 times when the huge page is applied.

4 Experimental Results

To implement the proposed traffic processing method that is capable of 30 Gbps-class processing, the experiment was conducted using the following equipment.

The experiment confirmed the packet processing performance with two-way traffic at 30 Gbps by receiving traffic generated by the packet analyzer from the measurement equipment, and the experiment was largely divided into two steps. First, it was confirmed that the packet analyzer handled actual traffic at the 30 Gbps level. Second, the drop rate of packets was checked

Table 3 Experimental environment – packet analyzer

Element	Explanation
CPU	Intel(R) Xeon(R) Sliver 4114 CPU @ 2.20 GHz
Memory	64 GB
OS	RedHat CentOS 7.5.1804
Kernel Version	3.10.0-862.11.6.el7.x86_64
I/O Interface	1000BASE-T × 4 Ports
	10G Fiber × 4 Ports

Table 4 Experimental environment – instrument

Element	Explanation
Use	L4-L7 Information Security Product Performance Measurement
Support interface	Performance Metric based on 1G/10G 8Port
HTTP CPS	1,500,000
HTTP Bandwidth	40 Gbps
HTTP Concurrent Connection	30,000,000

Table 5 Measuring equipment environment setting value

Element	Explanation
Protocol	HTTP
Percentage of Traffic	100%
Number of Clients	400
Number of Servers	40
Sessions per second	2.1K
Content Size	1.1Mbytes

by measuring the number of transmission/reception bytes, the number of transmission/reception packets, and the number of transmission/reception sessions. The type of protocol and traffic ratio generated by the measurement equipment are based on Table 5. The traffic generation time was 10 min.

First, it was confirmed that the packet analyzer handled 30 Gbps class traffic. In the test procedure, the value set presented in Table 4 was applied from the measurement equipment to the packet analyzer, and the processing results were confirmed, as shown in Table 6 and Figure 9.

The measurement equipment sent traffic at an average of 29.1 Gbps after 50 s, and the packet analyzer handled traffic at 29.1 Gbps. When checking the traffic speed processed in seconds, it was confirmed that packets were processed at an average speed of 29.0 Gbps or more.

Table 6 Traffic processing rate per second measurement results

Timestamp	Transmit Rate	Receive Rate
s	*Megabit/s*	
0.483	0.00	0.00
5.488	∼2880	∼2880
6.488	∼3477	∼3477
11.489	∼6357	∼6356
12.488	∼6983	∼6982
13.489	∼7498	∼7497
18.495	∼10430	∼10420
19.517	∼11020	∼11020
25.490	∼14540	∼14530
26.495	∼15100	∼15100
31.489	∼18010	∼18010
32.489	∼18580	∼18570
38.489	∼22140	∼22130
39.490	∼22540	∼22540
44.490	∼25650	∼25650
45.490	∼26090	∼26090
51.527	∼29170	∼29160
52.489	∼28980	∼28980
57.489	∼29080	∼29080
58.490	∼29120	∼29120
59.516	∼28980	∼28980
64.520	∼29100	∼29100
65.491	∼29000	∼29000
70.493	∼29140	∼29140
71.514	∼29090	∼29090
72.491	∼29020	∼29020
77.509	∼29160	∼29150

Second, the drop rate of packets and the CPU and memory utilization rates were checked when processing large-capacity traffic of 30 Gbps. Figures 10, 11, and 12. present graphs showing the number of bytes, packets, and sessions measured for 10 min, respectively, and the amount of transmission/reception should be the same. The final results are presented in Table 7. The graphs confirmed that there was little difference in the number of

```
[2021/05/18-13:55:52] [src/etap.cpp 3771] Resource counts (78)
  sessions  +sessions  -sessions     tuples         mem
    59.74K      6.00K      5.53K    119.47K      8.32G(8318406656/0)

 tuple       total        tcp         udp        icmp         etc       error
   in:       59676       59676           0           0           0           0
  out:       59677       59677           0           0           0           0

     rx          bps         pps       bytes    packets    dropped
enp95s0f0     67.36M     84.44K    266.07M      2.67M       0.00
       v0     29.07G      2.41M    115.45G     76.46M       0.00
       v1     67.33M     84.40K    266.06M      2.67M       0.00
enp95s0f1     29.07G      2.41M    115.46G     76.46M       0.00

     tx          bps         pps       bytes    packets    dropped
enp95s0f0     29.07G      2.41M    115.45G     76.46M       0.00
       v0     67.33M     84.41K    266.06M      2.67M       0.00
       v1     29.07G      2.41M    115.45G     76.46M       0.00
```

Figure 9 Check handling traffic at the 30 Gbps level.

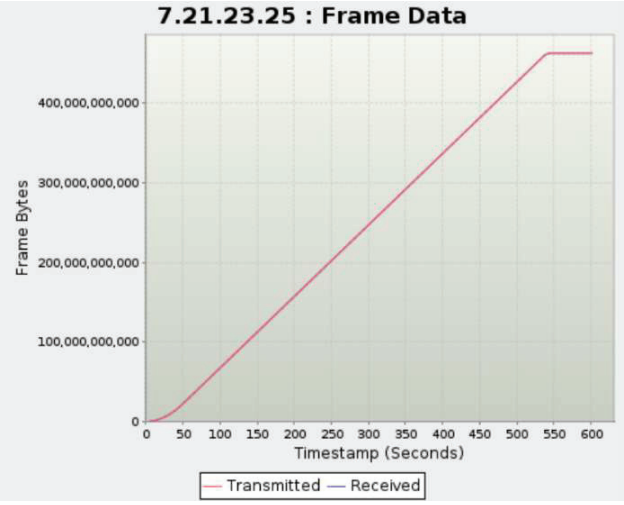

Figure 10 Bytes transmitted/received results.

transmission/reception bytes, the number of transmission/reception packets, and the number of transmission/reception sessions.

When comparing traffic generation and throughput, the drop rate of the number of bytes, packets, and sessions were all normalized to less than 0.01%. In addition, CPU utilization and memory utilization were 4.9% and 6.2%, respectively, confirming that CPU and memory loads did not occur even when processing large amounts of traffic.

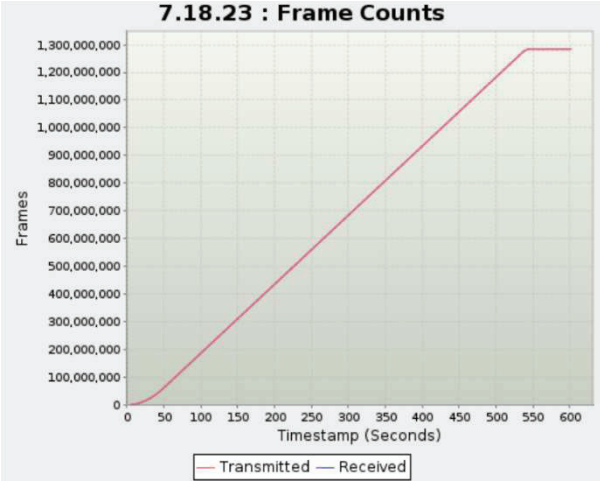

Figure 11 Packet count transmitted/received results.

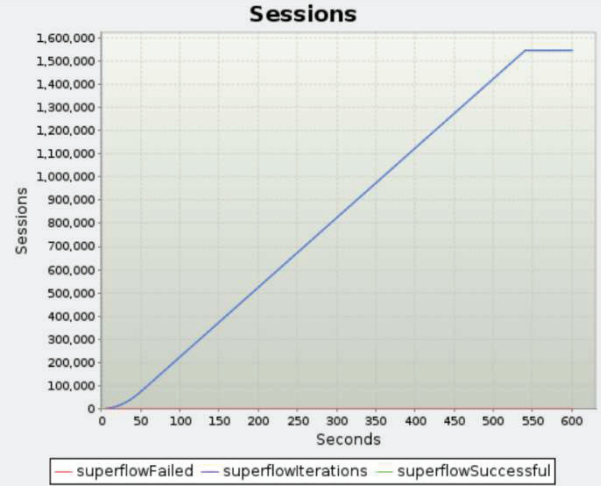

Figure 12 Sessions transmitted/received results.

In addition, the performance was confirmed by comparing and analyzing the existing 20 Gbps-class traffic processing, and the results are shown in Table 8. The experimental results showed that the proposed 30 Gbps-class packet processing method is improved by approximately 65% in terms of processing performance compared to that of the existing 20 Gbps class.

Table 7 Traffic processing test result

	Measuring Equipment	Traffic Analysis System	Drop Rate
Byte	463,084,244,536	463,055,776,875	0.006%
Packet Count	1,284,280,875	1,284,202,508	0.006%
Session	1,546,497	1,546,473	0.002%
CPU Utilization	–	4.9%	–
Memory Utilization	–	6.2%	–

Table 8 Comparison of traffic processing of 30 Gbps-class packet processing method and existing 20 Gbps-class processing engines

Element	20 Gbps	30 Gbps
Byte	308,703,844,583	463,055,776,875
Packet Count	856,135,055	1,284,202,508
Session	1,030,982	1,546,473

5 Conclusion

The proposed 30 Gbps high-capacity packet processing engine enabled 30 Gbps high-capacity traffic processing by increasing memory accessibility and lowering packet drop rates in the NUMA structure of the existing 20 Gbps high-capacity packet processing engine. It improved passive response and the lack of traffic characteristic analysis functions, which are major limitations of existing packet processing engines; thus, the possibility of abnormal traffic could be determined by a rapid analysis of suspicious traffic rather than a detailed analysis of traffic. Using the proposed packet processing engine, the extracted metadata could be used as raw data to analyze suspected attack traffic and detect abnormal traffic, and the protocol and application information could be converted into big data and used as various statistical data.

However, when applying core isolation, different setup methods must be applied to each equipment by CPU, memory, NIC, etc., and the isolated core cannot be controlled and monitored by Kernel; therefore, the current usage rate is unknown. Research is required to address these problems in the future.

Acknowledgement

This research project was supported by the Ministry of SMEs and Startups (MSS) and the Korea Technology and Information Promotion Agency for SMEs (TIPA) in 2021 (S2909329).

References

[1] Cisco, Cisco visual networking index: Forecast and trends, 2017–2022, 2018, Vol. 11.

[2] D. I. Oh, 'In the post-corona era, cyber-attacks will intensify,' Electronic Newspaper, https://www.etnews.com/20200512000181, 2020, Vol. 05.

[3] K. H. Jung, B. H. Lee and D. Yang, "Performance Analysis of Detection Algorithms for the Specific Pattern in Packet Payloads," Journal of the Korea Institute of Information and Communication Engineering, Vol. 22, No. 5, pp. 794–840, 2018. 02. DOI: https://doi.org/10.6109/jkiice.2018.22.4.794

[4] S. H. Lee, J. C. Na and S. W. Son, "Traffic Analysis Technology Trends in Terms of Security, https://www.itfind.or.kr/WZIN/jugidong/1117/111701.html

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville and Yoshua Bengio, 'Generative adversarial nets. In Advances in Neural Information Processing Systems,' NIPS, Proceedings of the 27th international conference on Neural Information processing Systems, '14, Vol. 2. 2672–2680, 2014. 03

[6] Schlegl Thomas, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth and Georg Langs, "Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery," International Conference on Information Processing in Medical Imaging. Springer, Cham, 2017. 03

[7] H. J. Choi, H. S. Kim and D. M. Shin, "Design and Implementation of Tor Traffic Collection System Using Multiple Virtual Machines," Journal of Software Assessment and Valuation, Vol. 15, No. 1, pp. 1–10, 2019. DOI: https://10.29056/jsav.2019.06.01

[8] Lameter Christoph, "NUMA (Non-Uniform Memory Access): An Overview: NUMA Becomes More Common Because Memory Controllers Get Close to Execution Units on Microprocessors," Queue, Vol. 11, No. 7, pp. 40–51, 2013. 07

[9] Christoph Lameter, "An Overview of Non-Uniform Memory Access," Communications of the ACM, Vol. 56, No. 9, 59–54, 2013, https://dl.acm.org/doi/fullHtml/10.1145/2500468.2500477

[10] S. Kim and S. Lee, "Automatic Malware Detection Rule Generation and Verification System," Journal of Internet Computing and Services(JICS),

Vol. 20, No. 2, pp. 9–19, 2019. 09. DOI: https://doi.org/10.7472/jksii.2019.20.2.9

[11] M. Thottan and C. Ji, "Anomaly Detection in IP Networks," IEEE Transactions on Signal Processing, Vol. 51, No. 8, pp. 2191–2204, 2003. 05. DOI: https://doi.org/10.3745/KTCCS.2020.9.5.113

[12] H. H. Lim, D. H. Kim, K. T. Kim and H. Y. Youn, "Traffic Classification Using Machine Learning in SDN," Vol. 26, No. 1 Winter Conference of the Korean Society of Computer and Information Technology, 2018. 01

[13] Vinit Tirnagarwar, "CPU Isolation & CPU Affinity In Linux.", https://www.linkedin.com/pulse/cpu-isolation-affinity-linux-vinit-tirnagarwar

[14] Red Hat Customer Portal, "3.13, Isolating CPUS using tuned-profiles-realtime, https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_for_real_time/7/html/tuning_guide/isolating_cpus_using_tuned-profiles-realtime

[15] S. Ahn, D. Kang and Y. Eom, "Analysis on the Characteristics and Performance Effects of Linux Huge Page," Journal of the Korean Society of Information Sciences, Vol. 2017, No. 06, pp. 73–75, 2017. 06

Biographies

Youngsun Kwon received the B.S. degree in Multimedia from Soongsil University, Korea, in 2020. She is currently a M.S. student in the Department of Computer Science and Engineering, Soongsil University. She is interested in network protocol, internet security and Deep Learning.

Moonhee Son received the B.S. degree in Department of Information and Communication from the Catholic University in 2000. He is currently working for Entobilsoft as a developer. He is interested in network protocol, VR and Multimedia.

Hoon Chang received the B.S., M.S. degree in Electrical Engineering from Seoul National University, Korea, in 1987 and 1989, respectively. He is received PH.D. degree in ECE from University of Texas at Austin, in 1993. He is currently a professor in the Department of Computer Science and Engineering, Soongsil University. He is interested in Computer System (Embedded System), VLSI/SoC, Design Automation.