네트워크 프로세서 기반의 침입탐지 시스템 구현

김 형 주[†]·김 익 균[††]·박 대 철[†††]

요 약

많은 공격과 네트워크 데이터 처리량이 증가하는 오늘날의 네트워크 수요를 NIDS가 유지시키기 위해 하드웨어 및 소프트웨어 시스템 구조에서 급진적 새로운 접근이 필요하다. 본 논문에서는 패킷 필터링과 트래픽 측정 뿐아니라 고의행위를 검출하는 패킷 페이로드 검열을 지원하는 네트워크 프로세서 기반의 인라인 모드 NIDS를 제안하고, 특히 2단계 검색구조를 사용하여 심층 패킷 검열기능으로부터 필터링과 측정기능을 분리한다. 그래서 심층 패킷 검열기능의 복잡하고 시간소비 동작이 인라인 모드 시스템의 기본 기능을 멈추게 하거나 방해하지 않게 했다. 프로토타입 NP 기반의 NIDS는 PC 플랫폼에서 구현하였으므로 실험결과는 제안한 구조가 첫 단계에서 두개의 기가비트 포트의 전체 트래픽을 측정과 필터가 신뢰할 수 있음을 보였다. 일반목적 프로세스 기반의 검열 성능과 비교 가능한 두 번째 단계에서 실시간으로 320Mbps까지 패킷 페이로드를 주사할 수 있었다. 그러나 시뮬레이션에서 10Gbps APP 해법을 선택할 때 선로상 속도가 2Gbps까지 가능한 심층 패킷 검색 결과를 얻었다.

Experiments on An Network Processor-based Intrusion Detection

Hyeong-Ju Kim[†] · Ik-Kyun Kim[††] · Dae-Chul Park[†††]

ABSTRACT

To help network intrusion detection systems(NIDSs) keep up with the demands of today's networks, that are the increasing network throughput and amount of attacks, a radical new approach in hardware and software system architecture is required. In this paper, we propose a Network Processor(NP) based In-Line mode NIDS that supports the packet payload inspection detecting the malicious behaviors, as well as the packet filtering and the traffic metering. In particular, we separate the filtering and metering functions from the deep packet inspection function using two-level searching scheme, thus the complicated and time-consuming operation of the deep packet inspection function does not hinder or stop the basic operations of the In-line mode system. From a proto-type NP-based NIDS implemented at a PC platform with an x86 processor running Linux, two Gigabit Ethernet ports, and 2.5Gbps Agere PayloadPlus(APP) NP solution, the experiment results show that our proposed scheme can reliably filter and meter the full traffic of two gigabit ports at the first level even though it can inspect the packet payload up to 320 Mbps in real-time at the second level, which can be compared to the performance of general-purpose processor based inspection. However, the simulation results show that the deep packet searching is also possible up to 2Gbps in wire speed when we adopt 10Gbps APP solution.

키워드 : 네트워크 프로세서(Network Processor), 네트워크 침입탐지시스템(Network Intrusion Detection System), PDU(Protocol Data Unit)

1. Introduction

Network intrusion detection systems(NIDS) are an important part of any network security architecture. As networks become faster, there is an emerging need for security analysis techniques that can keep up with the increased network throughput. That is, technological enhancements of new general-purpose processors and supplemental technologies such as memory and interconnect subsystems are not advancing with the same pace as network interface rates on the Internet. As demand grew for even greater bandwidth, fixed-hardware accelerators in form of Application-Specific Integrated Circuits(ASIC) were introduced to try to provide more processing power in their systems. Although they solved the performance issue, in a very short period of time, the closed architecture of such systems could not keep up with another dimension of the problem - frequent standard changes and new protocol and application requirements. Also, hardware development cycles tended to be too slow to accommodate them. The need for adaptability, differentiation, and short time-to-market brought about the idea of using

† 정 회 원 : 정보통신연구진흥원 선임연구원
†† 정 회 원 : 한국전자통신연구원 선임연구원
††† 정 회 원 : 한남대학교 정보통신공학과 교수
논문접수 : 2004년 2월 18일, 심사완료 : 2004년 2월 26일

Application Specific Instruction-set Processors(ASIP) called Network Processors(NP). NPs can be programmed easily and quickly according to specific products and deployments needs.

In this paper, we propose an in-line mode NIDS using APP (Agere PayloadPlus) Network processors(NP) that supports high-speed links up to Gigabits per second. In particular, we implement an x86 processor based NIDS running Linux with APP. In this prototype system, we adapt a two-level searching scheme for guaranteeing fully the wire-speed intrusion detection operation. We also illustrate the rule matching algorithm for effective use of APP, and show the performance.

2. Motivation

When most people think of network security, they think "Firewall". Firewalls are widely deployed as a first level of protection in a multi-layer security architecture, primarily acting as an access control device by permitting specific protocols to pass between a set of source and destination address. Integral to access policy enforcement, firewalls usually inspect data packet headers to make flow decisions. However, they do not inspect the entire content of the packet and can't detect or thwart malicious code embedded within normal traffic. This deep packet processing which is the original purpose of IDS relies on signature inspection and/or behavior-based systems. So we believe that the IPS(Intrusion Prevention System), which is combined with firewall and IDS systems, will be promise in near future. In this paper, we note that IDS detects and filters the attacks, which is the same function as IPS.

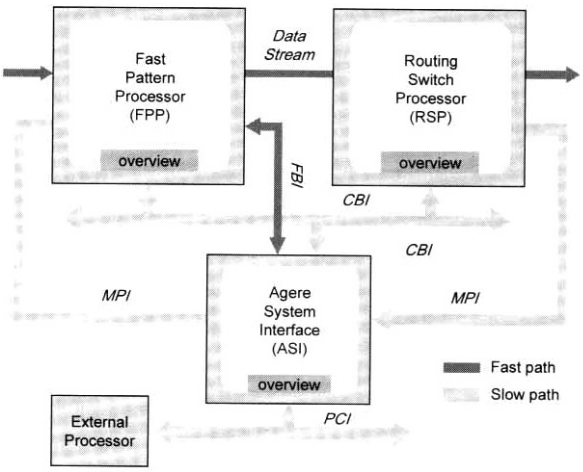
We can classify Firewalls into two categories - PC server based and hardware accelerated firewalls - according to the implementation techniques. Today, however, people focus on the hardware accelerated firewalls due to the packet handling throughput and the real-time analysis and response. For instance, the Sapphire (also called Slammer) worm was the fastest computer worm in history [1]. It is noted that the worm infected more than 90 percent of vulnerable hosts within 10 minutes. This is why we are interested in the real-time analysis of input data stream.

There are several rule based NIDSs available today. Most of these systems use one or more general purpose processors running rule-based packet filtering software. Due to exhaustive pattern detection algorithm, software system running on single general-purpose processor may not be able to inspect all network traffic [3]. In order to identify the effectiveness of rule based NIDS, we first review the background of Snort [2] and its rules-based traffic collection engine, and evaluate the performance of Snort from a network throughout point of view. Then, in order to improve the performance of Snort, we present the numerical results of alert detections from Snort running on kernel not as an application level in terms of offered attack loads and number of rules.

In order to overcome the limitations of rule based NIDS running on general purpose processor, [3] have designed a rule based pattern matching based on a parallel architecture which is capable of processing over 2.88 gigabits per second of network stream on an Altera EP20K series FGPA without manual optimization. However, the design time for new reconfiguration according to the updates to the rule set may be unacceptable in service time. On the other hand, there are some network processors, such as Intel IXP, IBM Power NP, and Agere PayloadPlus NP, are available that support faster network. In order to build an in-line mode NIDS for high-speed networks, we use Agere PayloadPlus NPs, and show the possibilities of NP based NIDS.

3. A Brief Review of APP

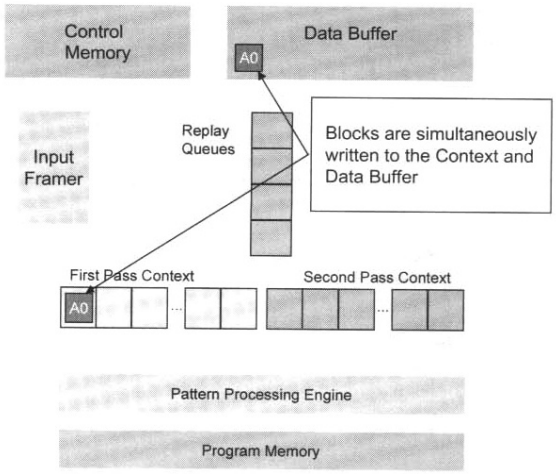
APP consists of three NPs(Network Processors), such as FPP(Fast Patter Processor), RSP(Routing Switching Processor), and ASI(Agere System Interface). The FPP accepts a data stream through industry-standard POS-PHY/UTOPIA Level 3 interface. The FPP analyzes and classifies these frames and cells, reassembles them into PDU(Protocol Data Unit), then transmits the PDUs and their classification conclusions to the RSP. The information provided by the FPP is used by the RSP to assign the PDU to a queue that has been programmed with QoS, CoS, and PDU modification instructions. Internally, the RSP uses custom logic and three programmable Very Large Instruction World(VLIW) computer engines to process PDUs while maintaining a high throughput. The processed traffic of the RSP is output on a configurable 32-bit POS-PHY Level 3/UTOPIA Level 3 interface. The ASI provides a Peripheral Component Interconnect(PCI) bridge to a host processor for the FPP and the RSP. It also provides programmable function processing and state maintenance for the FPP. For more detail information, refer to [14-17].



(Figure 1) Block Diagram of 2.5G PayloadPlus System

In PayloadPlus systems, processing tasks occur in one of two domains, called the fast(wire-speed) path and the slow path. The fast includes all tasks necessary for normal data stream processing. The slow path includes tasks such as exception handling, management, configuration, and updates.

The FPP is a pipelined, multi-threaded processor that simultaneously analyzes and classifies 64 PDUs. Each PDU is processed by a separate processing thread, called a context, which keeps track of the blocks that compose the PDU. As shown in (Figure 2), the operation of FPP is divided into two passes - first and second passes. The first pass stores data as 64-byte blocks and computes data offsets for each block, creating a linked-list data structure that defines the reassembled PDU. On the contrary, the second pass processes the whole PDU, simultaneously performing pattern matching and transmitting the conclusions of the pattern-matching processes for that PDU.



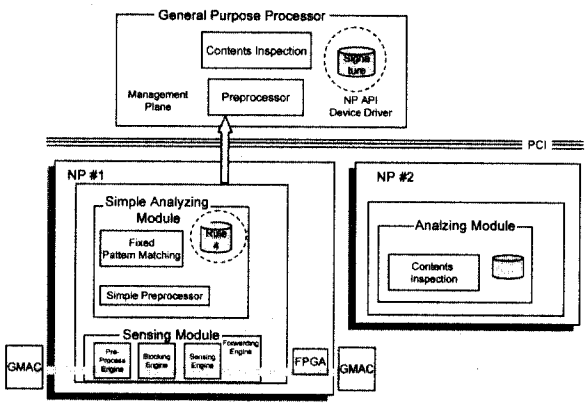
(Figure 2) First Pass and Second Pass Processing of FPP

4. Implementation

4.1 Two Level Architecture for the In-Line Mode IDS

The IntruShield architecture proposed in [5] enables sensor systems to capture network attacks in a number of ways : Switched Port Analyzer and Hub Monitoring, Tap Mode, In-Line Mode, and Port Clustering. The In-Line mode IDSs sit in the data path with active traffic passing through them, and prevent network attacks by dropping malicious traffic in real-time. As noted in Section 2, we are interested in this In-Line mode IDS using network processors.

(Figure 3) is a block diagram of our architecture. There are three main blocks, such as general-purpose processor, NP-1 Card, and NP-2 Card blocks. Particular, we adapt two-level NP scheme to guarantee the gigabit 2-port wire-speed operation. The NP-1 Card provides the filtering, flow statistics, and sensing for the NP-2 Card with fully wire-speed. It shows the logical view of three main functions of NP-1 Card. The NP-2 provides the deep packet processing by a signature inspection. The general purpose processor also provides the deep packet processing by same signature and heuristic analysis.



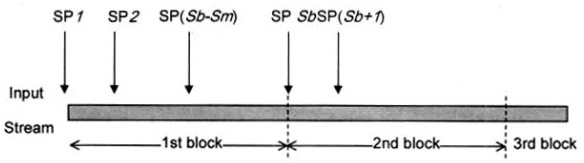
(Figure 3) Two-Level Architecture

The incoming packet data from gigabit MAC is passed to the FPP through a 32-bit bus. Pre-Processing Engine records the various flow statistics based on statistics conditions, which is the full combinations of the IP head five-tuple information (source and destination IP addresses, source and destination port addresses, and Protocol Type). Blocking Engine filters the incoming packets based on the blocking rules, which are also the five-tuple combinations. Sensing Engine copies the corresponding packet data based on the five-tuple and TCP flags information, and sends them to the NP-2 Card and general purpose processor using PCI bus.

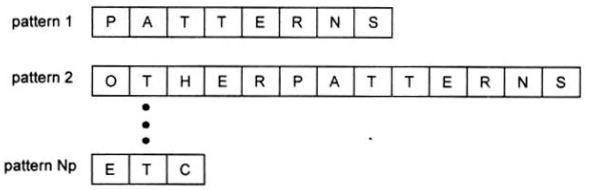
4.2 Rule Matching Algorithm

First of all, we note that the packet content matching is performed at the first pass in FPP at NP-2. The first pass processes data at the block level, which is 64-byte. Hence, the entire input packet stream is matched block-wised with a given matching patterns, as shown in (Figure 3). For the simple explanation, we assume that the maximum size of a matching pattern S_m is less than block size S_b, where S_b is 64-byte. (Figure 4) shows the relationship between searching point (SP) and input data stream and block boundary. Until the SP is $S_b - S_m$, the content pattern match is performed every SP as shown in (Figure 5), where N_p is the number of matching patterns. When the SP reaches $S_b - S_m + 1$, the matching patterns are divided into two parts, partial and truncated patterns as shown in (Figure 6). Theses two kinds of patterns are generated at every SP until SP reaches the end of boundary.

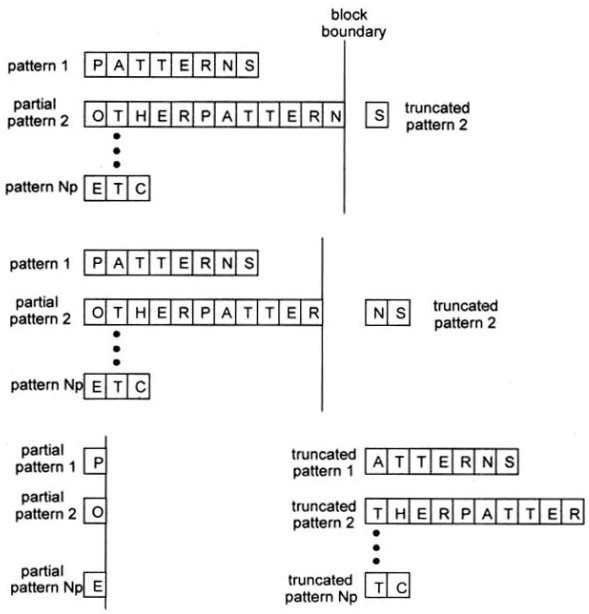
When the SP is between $S_b - S_m + 1$ and $S_b - 1$, the matched results for partial patterns are stored as a bit-wise information of registers. As we can expect, the maximum number of partial or truncated patterns is $N_p \times (S_m - 1)$. When the SP reaches at S_b, which is the starting point of the next block, the pattern matching is performed with the matching patterns (shown in (Figure 5)) as well as the truncated patterns (shown in (Figure 6)). If the truncated patterns are matched, then the corresponding bits of the registers should be compared to check whether the corresponding partial patterns were matched or not.



(Figure 4) The Relationship between Searching Points, Input Data Stream, and Block Boundary



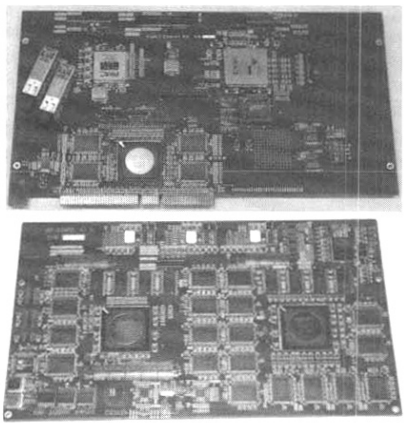
(Figure 5) Example of Matching Patterns



(Figure 6) Example of Partial and Truncated Matching Patterns

5. Evaluation

The set of experiments were primarily aimed at evaluating the effective of the two-level architecture based on network processors. For these experiments, we used an industrial PC platform running Linux and developed two kinds of circuit board using 2.5Gbps APP solutions such as the FPP, the RSP and the ASI. Through the experiments, we especially focused on the performance of deep packet inspection of the second level NPM as well as general purpose processor and the possibility of wire-speed operation of packet filtering and traffic metering at the first level of NPM. The next section presents the prototype architecture and test environments, and the section after that gives the corresponding results. We also present some simulation results for the foundation of further works.



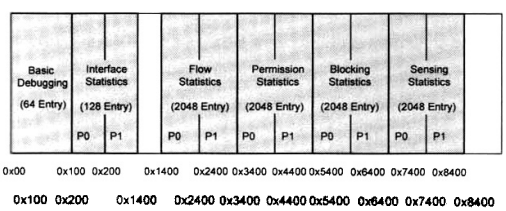
(Figure 7) Board Design(Upper card : Line and main processor interface, Lower card : Packet classification, filtering and rate limiting)

In order to implement the proposed architecture in (Figure 3), we have developed two kinds of circuit board as shown in (Figure 7). The LIM board is designed for interfacing two gigabit Ethernet ports and providing an industry-standard PCI interface by the ASI between a host processor at PC and network processors. It also has a gigabit MAC chip for Ethernet control and some management functions. These functions include hardware configuration and exception handling. In addition, the LIM board maintains the metering information as discussed in <Table 2> and <Table 3>, which must be accessed by some applications running at the host processor. The ASI has a 64-bit, 66MHz PCI interface that is a full master/slave implementation with DMA and interrupt support.

The NPM board mainly consists of the FPP and the RSP which classify and switch the PDUs to accomplish the three main functions of IDS, such as packet filtering, traffic metering, and signature detection. The NPM board should be directly mounted on the LIM board by connectors so that it communicates with host processor via the LIM board. The FPP and the RSP use SSRAM arrays for program and control (we can find them around the network processors in (Figure 7)). Hence, the control or program will be also downloaded from the host processor via the LIM board.

In this prototype system we use 2.5G APP solution to implement the NPM design so that the board cannot handle the full traffic stream if we assume every PDU is sensed as discussed in Section IV. However, we believe this prototype is enough to evaluate the effective of our proposed architecture in this paper. In addition to the experiments results, we present the simulation results by SPA [20] to evaluate the full traffic on 2.5G and 5G APP solutions,

The ASI has external synchronous static RAM (SSRAM) arrays that may be used for keeping state and statistics related to traffic seen by the FPP. (Figure 8) shows the memory mapped diagram of the ASI external SSRAM, which stores the diverse counter values. In this prototype system, we configured the memory up to 2048 entries for flow, permission, blocking, and sensing statistics, respectively.



(Figure 8) Memory Map of ASI external SSRAM

Our test environments have two pairs of the LIM and the NPM boards are mounted on the industrial PC platform via 64-bit PCI slots; that is a Pentium 4 host processor running Linux. We interconnect the two NPM boards with a 5-inch blue ribbon cable which supports the parallel transmission at 104Mbps to transmit the sensed PDUs.

The Gigabit links are directly connected between the LIM and Shomiti's THGs (Ten Hundred Gigabit system) [18] which provides two ports full duplex 1Gbps Ethernet. THGs generates Ethernet frames with full line rate whilst the actual traffic rate is less than full line rate due to the 12-byte gap time between Ethernet frames, and supports the real-time monitoring and measurement.

In our setup, Shomiti's THGs can generate the attack traffic as defined in (Figure 8), and the LIM and the NPM-1 boards filter and meter the traffic according to the blocking and metering rules, respectively. Whenever a PDU is sensed at NPM-1 board based on the sensing rule, two copies of the PDU are forwarded to the NPM-2 board and general purpose processor using PCI bus. Finally, the NPM-2 board looks up the attack signatures in the payload and the general purpose CPU inspect and analysis the packets by heuristic mechanism.

5. Experiment Result

We have developed our prototype system based on the proposed architecture. The prototype is programmed in a combination of C and FPL programming language. Most of all, our prototype system is implemented in programming languages that is best suited for the task it has to perform. Basically, Application Task block on the CPU is implemented in C programming language, but Kernel Module block is implemented to the kernel module programming that is best suited for high-speed pattern matching operation. NPM-2 Logic block is implemented in FPL that is best suited for high-speed packet processing in hardware. Most of all, the prototype system focus on kernel logic and NPM-2 logic for real-time traffic analysis and intrusion detection on high-speed links. Also, we employed inline mode capable of effective response by using two Gigabit Ethernet links. That is, our prototype system has developed in the side of improvement in performance for packet processing.

The goal of the set of experiments described in this section is to get a preliminary evaluation of the effectiveness of our approach. The first experiment was to test for the packet

filtering and traffic metering. The Shomiti's THGs generated two TCP traffic streams at 1 Gbps line rate for each gigabit Ethernet port, which are 64-byte IP packet flows. This traffic was injected on two gigabit Ethernet ports simultaneously. As we explained, the actual full traffic of 64-byte IP packets corresponds to 76% of line rate due to the gap time. From the experiment, we confirmed that the prototype system blocked the full traffic in wire-speed, and exactly counted the number of discard packets. When we permitted the traffic by updating the ACL list, the full incoming traffic from port 0 (or port 1) was forwarded to port 1 (or port 0) while counted the number of passed packets. We note that the number of filtering or metering rule sets does not affect the performance.

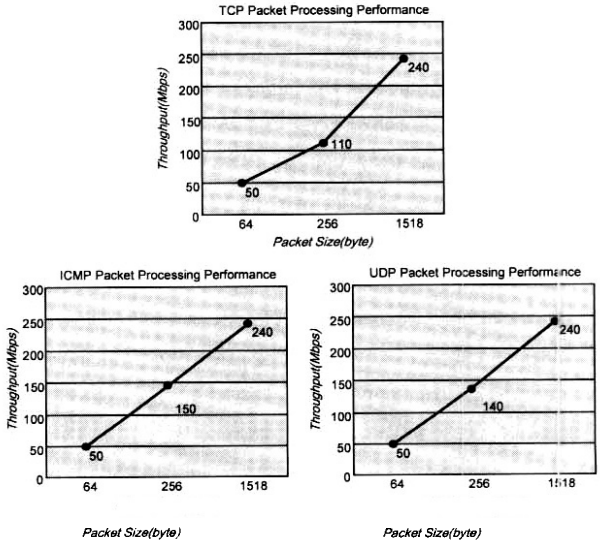
For performance evaluation of our prototype system, our tests use traffic generated by the traffic generator, Shomitti. The test data is configured as attack traffic according to the protocol and packet size. Therefore, the whole traffic is matched by FPP logic, and is sent to kernel logic and Second APP. That is, we show performance of the whole detection mechanism from FPP logic to kernel logic. <Table 1> and (Figure 9) shows the packet processing performance as the experimental results. As shown in the results, our prototype system is capable of processing until a maximum throughput of 240Mbps about packets matched in NPM-1. And, basic packet sensing and header pattern matching is capable of processing until 2Gbps. Therefore, rule configuration of our system is very important.

〈Table 1〉 Experimental Results

Protocol \ Packet size	64byte	256byte	1518byte
TCP Traffic	50Mbps	110Mbps	240Mbps
UDP Traffic	50Mbps	140Mbps	240Mbps
ICMP Traffic	50Mbps	150Mbps	240Mbps

Currently, we are in the process of improving the implementation as well as developing new ones. That is, our prototype system leaves much to be desired. Furthermore, we analyzed the functions of various intrusion detection systems in our testbed network. And now, we are defining more effective analysis functionality in order to improve the performance of detection mechanism on high-speed links.

The second experiment was to test for the deep packet inspection in NP-2. The Shomiti's THGs generated the attack traffic, which were example 1, 2 and 4 shown in (Figure 8). As we explained before, the number of rule set doest not affect the performance of the FPP and the RSP. That means the performance of the proposed architecture dose not degraded as an increasing number of signatures. We also evaluated the performance with different packet length, such as 64, 128, 256, 512, and 1024 bytes. The attack signatures are inserted at the end of payload of the generated PDU, hence the time consumption for the pattern matching process at the NPM-2 must be under the worst condition. But the attack traffic was injected on only one gigabit Ethernet port at 1Gbps full line rate, because the all attack traffic was sensed at NPM-1 board, and then forwarded to NPM-2 board at 2Gbps which might be the limitation of our prototype system. To find the maximum throughput of deep packet inspection of the NPM-2, we disabled intrusion prevention module running on the host processor, which receives the alert from the NPM-2, and then updates the ACL list of the NPM-1 to prevent the attack. So the attack traffic continuously forwarded to the NPM-2 even if the attack was detected.

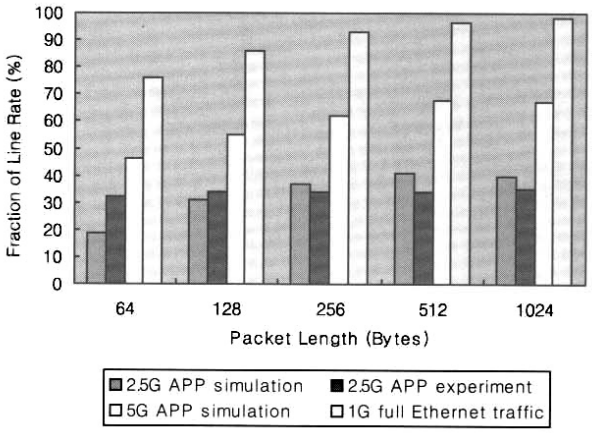


(Figure 9) Packet Processing Performance in Kernel Logic on General Purpose Processor

(Figure 10) shows the results of this second experiment. For each packet length displayed on the x-axis, we show the maximum throughput (represented by the fraction of 1 Gbps line rate) of deep packet inspection at the NPM-2 for each experiment and simulation results.

Note that the actual 1 Gbps Ethernet full traffic increases as the packet length increases due to the gap time between PDUs. As shown in the figure, the experiment result of 2.5G APP solution shows roughly the same percentage at 34% regardless of the different packet length. However, the simulation result of 2.5G APP solution slightly varies with the

packet length. Comparing the experiment and simulation results of 2.5G APP solution, we can predict that the performance of 5G APP solution may be over 60%. Moreover, we believe the 10G APP solution will support the deep packet inspection with the wire speed.



(Figure 10) Experiment and Simulation Results on NPM-2

6. Conclusion

In this paper we have proposed an NP based In-Line mode NIDS using Agere System's APP solutions, which might be the first approach to apply network processor to IDS. Comparing to ASIC approach, the NP-based implementation can dynamically update each rule set for filtering, metering, and signature; that is the system can keep up with another dimension of the problem - frequent standard changes and new protocol and application requirements. Most of all, the prototype system focuses on reducing a lowing of performance caused by high-speed traffic analysis to the minimum. Therefore, it is run by the NPM logic and kernel logic proposed for improvement in performance. Also, it has the advantage that is capable of supporting the effective response by using inline mode monitoring technique on two Gigabit links.

The evaluation of the first prototype based on 2.5G APP solution showed that the proposed architecture is very promising as an in-line mode system because it can filter and meter network traffic with two 1 Gbps Ethernet ports at wire speed, even if it can not inspect the payload contents at wire speed. This is why we separate the deep packet inspection from these basic functions, and it greatly promotes the availability of the in-line mode system.

From our experiment and simulation results, we expect that our implementation can search the signatures on a data bandwidth of several gigabits per second by using more high performance NP solutions, even though the presented prototype can handle only several hundred megabits per second.

Reference

[1] David Moore et al., The Spread of the Sapphire/Slammer Worm, available at http://www.caida.org/outreach/papers/2003/sapphire/sapphire.html.

[2] Martin Roesch, "Snort - Lightweight Intrusion Detection for Networks," Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, Jan., 2000.

[3] Young H. Cho et al., "Specialized Hardware for Deep Network Packet Filtering".

[4] G. Memik and W. H. Maggion-Smith, "NEPAL : A Framework for Efficiently Structuring Applications for Network Processor," Proc. Second Workshop on Network Processors, 2003.

[5] Fengmin Gong, Next Generation Intrusion Detection System (IDS), IntruVert Networks Report, March, 2002.

[6] Y. H. Cho, S. Navab and W. H. Maggione-Smith, "Specialized Hardware for Deep Network Packet Filtering," Proc. Field Programmable Logic and Applications (FPL), 2002.

[7] M. J. Ranum, "Thinking about Firewalls," Proc. SANS-II, 1994.

[9] Martin Roesch, Snort Users Manual Snort Release : 1.8, Snort, 2001.

[10] P. Ferguson and D. Senie, "Network Ingress Filtering : Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," RFC 2267, http://www.ietf.org, 1998.

[11] H. Wang, D. Zhang, K. G. Shin, "Detecting SYN Flooding Attacks," Proc IEEE INFOCOM 2002, 2002.

[12] D. Moore, G. Voelker and S. Savage, "Inferring Internet Denial of Service Activity," Proc. USENIX Security Symposium '2001, 2000.

[13] Riccardo Russo, et al., "Scalable and Adaptive Load Balancing on IBM Power PC," IBM Research Report.

[14] Agere Systems Inc., PayloadPlus Fast Pattern Processor, http://www.agere.com, 2001.

[15] Agere Systems Inc., PayloadPlus Routing Switch Processor, http://www.agere.com, 2001.

[16] Agere System Inc., PayloadPlus Agere System Interface, http://www.agere.com, 2001.

[17] Agere Systems Inc., Functional Programming Language User's and Reference Guides for SDE Version 3.0, 2001.

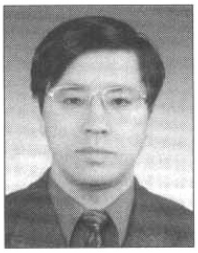
[18] A Finisar Company, Shomiti THGs ; Distributed 10/100/1Gb Network QoS System, http://www.shomiti.net/shomiti/thgs.html.

김 형 주

e-mail : hjookim@iita.re.kr
1982년 경북대학교 전자공학과(학사)
1987년 동아대학교 전자공학과(석사)
1987년~1997년 한국전자통신연구원 선임연구원
1998년~현재 정보통신연구진흥원 선임연구원
관심분야 : 고속통신망 성능평가, 네트워크 보안 등

김 익 균

e-mail : ikkim21@etri.re.kr
1994년 경북대학교 컴퓨터공학과(학사)
1996년 경북대학교 컴퓨터공학과(석사)
1996년~현재 한국전자통신연구원 선임연구원
관심분야 : 초고속 인터넷, 네트워크 트래픽, 네트워크보안 등



박 대 철

e-mail : daechul@hannam.ac.kr
1977년 서강대학교 전자공학과(학사)
1985년 미국 Univ. of New Mexico 전기공학과(석사)
1989년 미국 Univ. of New Mexico 전기공학과(박사)
1977년~1982년 국방과학연구소 연구원
1989년~1993년 한국전자통신연구원 선임연구원
1993년~현재 한남대학교 정보통신공학과 교수
관심분야 : 통신신호처리, 디지털 워터마킹 등