AntiWorm NPU-based Parallel Bloom filters in Giga-Ethernet LAN[1]

Zhen Chen, Chuang Lin, Jia Ni, Dong-Hua Ruan, Bo Zheng, Zhang-Xi Tan, Yi-Xin Jiang, Xue-Hai Peng,
An-an Luo, Bing Zhu, Yao Yue, Jin-Jun Zhuang, Fu-Jun Feng, Yang Wang, Feng-Yuan Ren

***Abstract*—In this paper, an AntiWorm system, which is based on Intel IXP Network Processor, is implemented using the Parallel Bloom filters technique. The AntiWorm system consists of two components: Bloom filters and Exact Matching engines. The Parallel Bloom filters can identify the suspicious traffic quickly and effectively, and then dispatch them to Exact Matching engines for further investigation. Both the principles and the implementation of the AntiWorm system are introduced in detail. With the consideration of the system performance parameters, two feasible implementation solutions are investigated and the advantages and disadvantages are also compared. The selections of configuration parameters of the AntiWorm system are also discussed. A hash scheme based on MD5's function is proposed for implementing fast hash functions. To test the performance of the AntiWorm system, such as throughput and delay, some experiments are carried out with different simulated traffic condition. The internal statistics of IXP network processor are also collected and analyzed for optimizing the system performance. To demonstrate the working of the AntiWorm system, the assaults of Worm Blaster are used in the test bed and the experiments results prove the working of the AntiWorm system. Software Package WormDetector1.0 is also given as a software release of the research.**

***Index Terms*—Computer Networks, Network Security, Network Processors, IXP Network Processor, Worms, Parallel Bloom filters, Worm Blaster.**

I. INTRODUCTION

Since the outbreak of RTM (Robert T. Morris) worm in 1988, active worm[10-39] is a enduring threat to global Internet, e.g., CodeRedI v2 worm infected approximately 359,000 computers in 24 hours; the SQL Slammer(or Sapphire) worm infected at least 75,000 hosts. Recently, Worm Blaster and Worm Sasser have attacked thousands of Windows-based PC and led many of them to paralysis, and the attack is still ongoing.

A Gigabit Ethernet network scenario is currently the most used network topology in the office and enterprise, which consists of one or two gigabit Ethernet Switches, several workgroups composed of many hosts, and some application server. See Figure 1. In such a high-speed LAN scenario, which have an expected smaller network latencies in communication, hosts are closer in proximity may be more vulnerable. Some worms, such as Nimda, even intend to localize network propagation. Currently, there are few defenses in place to respond to worms, and the human response is very ineffective. Therefore, Giga LAN is more fragile under the worm's attack than expected. It is more urgency to deploy Anti-Worm system in such network scenario.

The remainder of the paper is organized as follows. Section 2 introduces the research backgrounds about the Worms, Deep packet Inspection and Network Processor. Section 3 shows the design of AntiWorm System using Parallel Bloom filters. In section 4, experiments are carried out for demonstrating the working of AntiWorm system and the experiment results are shown. Section 5 introduces the software package of the research. Section 6 presents the further work. Finally, Section 7 gives the conclusion.

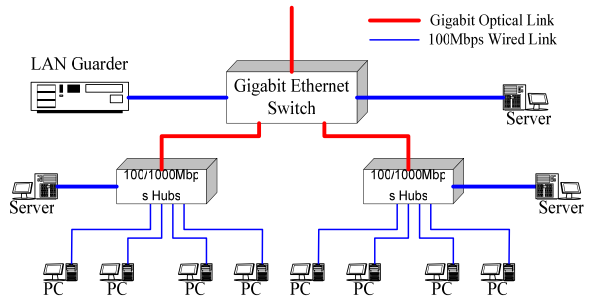


Figure 1. A Gigabit Ethernet Network Scenario

II. BACKGROUND

A. Worms

1) The Definition of Worms

A Worm is malicious code, which can be self-reproduced and propagates over a network. The life cycle of a worm is as follows: (1) Target discovery: the worm in the infected host tries to locate a victim host; (2) Carrier: First, make the new hosts prepare to receive the worm code. Then, transfer the worm code (in plain text (unencrypted) mode or in encryption way) to the new host; (3) Activation: change the new host's registry, and make it to reference the worm code. Then the process repeats.

2) The Detection of Worms

The following symptoms of active worms can be used for the detection of Worms.

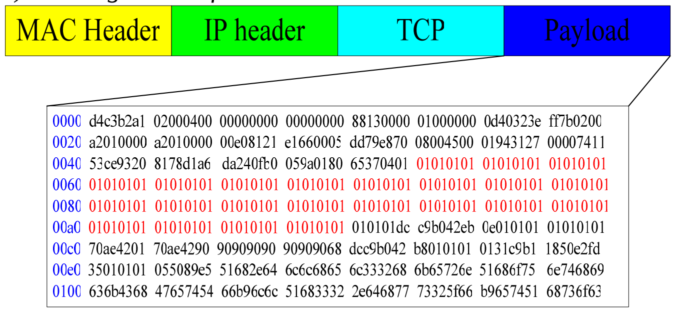
In the victim Target discovery Stage: a simple strategy the worm used is scanning, including sequential and random scanning, e.g. CodeRed. Scanning is a highly anomalous behavior, so devices can effectively detect them.

In the Carrier stage, a worm can either actively transfer itself from host to host, or it can be carried along as part of normal communication, e.g. Worm Blaster exploits the victim's RPC(Remote Procedure Call) flaw and make the victim connect back to the infecting machine using TFTP(Trivial File Transmission Protocol) to download the worm code.

The authors are with the Institute of Network technology, Department of Computer Science and Technology, Tsinghua University, Beijing, China.
Corresponding author: Zhen Chen, Phone: 86-10-62772487, Fax: 86-10-62771138, E-mail: zhenchen@csnet1.cs.tsinghua.edu.cn..

[1]**This work is funded by IXA University Program of Intel Research Council and partially supported by Intel China Research Council.**

3) *The signature of Worms*



```
0000 d4c3b2a1 02000400 00000000 00000000 88130000 01000000 0d40323e ff7b0200
0020 a2010000 a2010000 00e08121 e1660005 dd79e870 08004500 01943127 00007411
0040 53ce9320 8178d1a6 da240fb0 059a0180 65370401 01010101 01010101 01010101
0060 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
0080 01010101 01010101 01010101 01010101 01010101 01010101 01010101 01010101
00a0 01010101 01010101 01010101 01010101 010101dc c9b042eb 0e010101 01010101
00c0 70ae4201 70ae4290 90909090 90909068 dcc9b042 b8010101 0131c9b1 1850e2fd
00e0 35010101 055089e5 51682e64 6c6c6865 6c333268 6b65726e 51686f75 6e746869
0100 636b4368 47657454 66b96c6c 51683332 2e646877 73325f66 b9657451 68736f63
```

Figure 2. SQL Slammer Worm Packet Payload

The SQL Slammer Worm Packet Payload is shown in Fig. 2. The salience is the 01010101 repeating pattern, which is used to result in a buffer overflow exploit in the receipted party, and the actual code executed is followed. To identify the SQL Slammer Worm, the repeating 01010101 patterns and the worm code can be used as the signatures in signature-based systems. Such signatures in many worms can be collected.

B. *Deep Packet Inspection*

Solidum Pax.port, Juniper NetScreen 5000, and ClassPI [58] etc. use the ASIC-based filter in packet content inspection. As we know, new worm occurs instantly and evolves into many variants quickly. The ASIC-based schemes can not be configured quickly, and will fail to keep up the pace of the worm's evolution. Some research groups also use FPGA, another possible implementation platform. But the drawback of such implementation is still the inflexibility: the slow reconfigurable time, appending and deleting even one signature needs to reconfigure the whole FPGA. Thus, the services need to be halted for reconfiguration, which aren't dependable enough.

There are also some software-based packet payload inspection schemes, such as SNORT [47] and Sourcefire[48]. The KMP string matching algorithm, Boyer-Moore algorithm and Karp-Rabin algorithm are the most used algorithms. But with the advent of Gigabit Ethernet, it is becoming increasingly difficult for software-based packet monitors to catch up the high-speed link rate.

III. The Design of AntiWorm System

A. *Parallel Bloom Filters*

To find worms' signatures quickly and correctly, Bloom filter technique is used [1-2][40-41].

Definition 1. (**Bloom Filter**). A Bloom filter is used to represent a set $S=\{s_1,s_2,\cdots,s_n\}$ of n elements from a universe U . It consists of an array of m bits, initially all set to 0. A Bloom Filter uses k independent random hash functions $h_1,\cdots,h_k$ with range $\{0,\cdots,m-1\}$. These hash functions map each element $s\in S$, the bits $h_i(s)$ are set to 1 for $1\le i\le k$. A Bloom filter may yield a false positive, where it suggests that an element x is in S even though it is not, but it will not generate a false negative event.

Given the assumption that hash functions are perfectly random, let $p=e^{-kn/m}$, the false positive probability f is given

$$f=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^{k}\approx\left(1-e^{-kn/m}\right)^{k}=(1-p)^{k} \qquad \text{(Eq. 1)}$$

Definition 2. (**Counting Bloom filter**). A Counting Bloom filter is a bloom filter enhanced with a vector of counters corresponding to each bit in the bit vector (BV) for scalability reason. Whenever a counting Bloom filter adds or deletes a member, it increments or decrements the counters corresponding bit in the bit vector. Only when a counter changes from one to zero, it clears the corresponding bit in BV. Therefore, these counters change only during the addition and deletion of strings in the Bloom filter.

B. *The Introduction of Intel IXP2400*

The Network Processor Unit (NPU)[3-5] enables network researchers and developers to add the latest network services while maintaining high throughput and low latency. The main idea of this paper is to use the flexibility and high performance of Network Processors to detect and scan packet payload to locate worm codes, and cut off their propagation. In the design, Intel IXP Network Processors IXP2400 is used and will be introduced as follows [8-9, 49-53].

IXP2400 consists of 8 Multi-Threaded MicroEngines (2 Clusters), XScale Core, 4K 32-bit word Scratchpad Memory, 2 QDR SRAM Interface controllers, 1 DDR DRAM Interface controller, Hardware Hash Unit, Media and Switch Fabric(MSF) Interface, Half Duplex OC-48/2.5 Gbps Ethernet Interface, and other Peripherals Components.

A ME possesses such features: 640 32-bit Local Memory, 256 GPRs (Bank A and Bank B), 4K × 40-bit instruction control store, 128 SRAM Read (Write) Transfer registers, 128 SRAM Read (Write) Transfer registers, Eight threads per ME (no overhead for context switching), 128 Next Neighbor registers, and other units.

A networking application typically operates on three logical planes: *data plane*, *control plane*, and *management plane*. This programming focuses primarily on writing data plane components utilizing MEv2 microengines.

Logical networking functions named microblocks are used in the fast path processing. Each microblock is a macro (Assembly Language) or microengine C function written using underlying low-level libraries.

C. *The Architecture of AntiWorm System*

Signatures are extracted from the worm's sample binary codes, and compared with the payload in every packet appeared in the network. Such signatures are hashed and stored in the BV of Parallel Bloom filters, which are distributed across the Microengines among Network Processor, to match and identify those signatures. Bloom filters boost the string matching by separating most of the strings from the network data and only sending those most probable candidate strings of matching to the supervisor. A string of interest never passes without notification since the Bloom filters never give false negatives. Consequently, such scheme can quickly detect and locate the trace of worms. Figure 3 shows the architecture of the design.

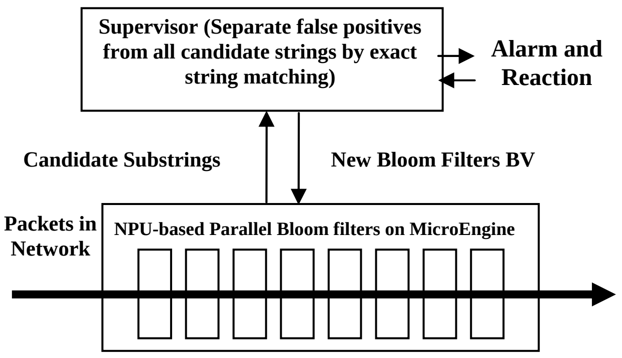

Supervisor (Separate false positives from all candidate strings by exact string matching)
Alarm and Reaction
Candidate Substrings
New Bloom Filters BV
Packets in Network
NPU-based Parallel Bloom filters on MicroEngine

Figure 3. An AntiWorm system based on Network Processor

D. *Implementation Details*

1) *The System Configuration*

The system is developed based on Radisys ENP2611 platform [54-56]. WindRiver System Vxworks5.0 is running on the XScale core of IXP 2400. The Tornado 2.2 is used as programming environment on the side of XScale. Intel IXP Developer Workbench 3.61 is used as programming environment on the side of Microengines.

In current design, the following system parameters are chosen:

(1) The size of the Worm Signature Library is about 128;

(2) The false-positive probability of the Bloom filter should be less than 2^{-8};

(3) The size of signature is fixed and 16 bytes long.

2) *Software framework Implementation*

In current design, three microblocks namely GERX, Bloom filters, GETX are implemented in the fast path of data path. Functions to initialize and maintain Bloom filters are realized in the control plane. Functions to maintain and update Worm Signature Library are realized in the manage plane. MEs are allocated as follows: Packet receiving function is implemented in one ME, Packet Transmitting function is implemented in another ME, and the other MEs are used as Bloom Filters (One Bloom Filter per ME).

3) *Two Organization Mode of MEs*

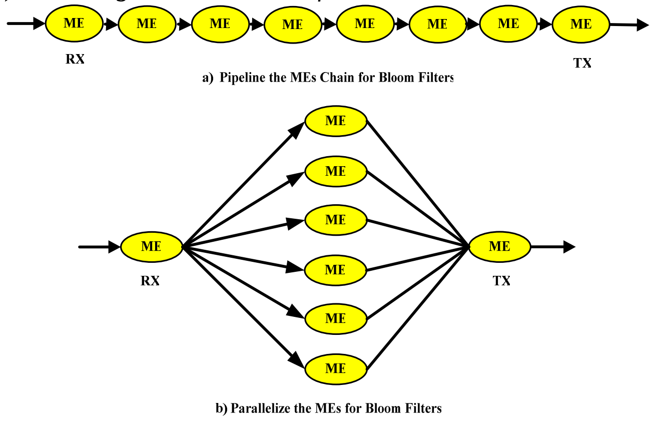

ME
RX
TX
a) Pipeline the MEs Chain for Bloom Filters
b) Parallelize the MEs for Bloom Filters

Figure 4. Two Practice Implementation of Parallel Bloom Filters

There are two modes to organize MEs: Pipeline Mode of MEs (ME Chain) and Parallel Mode of MEs (ME Para).

a) *Pipeline Mode of MEs (ME Chain)*

Figure 4(a) shows the pipeline mode which can be further classified into two cases:

(1) The same copy of the signature Library is distributed in the Bloom filter ME;

In this case, the Bloom filters are concatenated with parameters $m = 4096$, $k = 2$. According to the Equation (1), one Bloom filter per ME can support $n \le 1419$ signatures. The false-positive probability is about $(1/2)^2 = 0.25$. By employing six such MEs, one construct a Bloom filter with a maximum of 1,419 signatures and false-positive probability of $f = (1/2)^{10}$.

Packet, if suspicious, will bypass the remaining MEs, and be sent directly to the XScale for further investigation. Every ME uses different Hash functions.

(2) The signature Library is distributed over a ME chain.

In this case, the Bloom filters are concatenated with parameter $m = 4096$, $k = 12$. According to the Equation (1), one Bloom filter per ME can support $n \le 236$ signatures. The false-positive probability is about $(1/2)^{12}$. By employing six such MEs, one can construct a Bloom filter with a maximum of 1,419 signatures and false-positive probability of $f = (1/2)^{12}$.

All packets must be transferred from one ME to another, and can not be abort the checking if the previous check is miss.

Both cases have their strength and weakness. With the system requirements in mind, in case (a) the load of MEs are not distributed equally, and in case (b) introduce hard work for every ME.

To minimize the False Positive Probability, $e^{-kn/m} = 0.5$, i.e. $kn = m \ln 2$. The Local Memory size is about 2^{14} bit ($640 \times 4 \times 8$). Some headroom is used for other purpose. If the number of hash functions is $k = 8$, and we use all Local Memory to store BV of Bloom filters, then the maximum number of signatures in one ME is about 1774.

b) *Parallel mode of MEs(ME Para)*

Figure 4(b) shows parallel mode of MEs(ME Para). ME Para has the highly parallelizability and achieves higher throughput, but the cost is the packet's disorder. Hence the *Asynchronous Insert and Synchronous Remove Ring* (AISR Ring) can be used to maintain the End-To-End ordering of Packets.

ME Para is the choice for AntiWorm System. We implement the Bloom Filter in Intel IXP 2400. In the prototype, the 8 MEs in IXP 2400 are configured in the following way: one for the receiving module, one for the transmitting module; the other 6 for Bloom Filter module.

4) *ME Programming Model*

MEs can be programmed in two model, namely **ordered thread model** and **unordered thread model**.

In the ordered thread model, namely HyperTask Chaining **(HTC)**, threads execute in some application-defined order in critical sections. A critical task is divided into different phases and each thread takes a turn executing a phase and then passes control to the next thread.

In the unordered thread model, namely Pool of Threads **(POTs)**, threads can run in any arbitrary order. Threads work until their current task is done, get more work and continue. **POTs** is the alternative for programming in Microengines in the design.

5) Parallel Bloom Filters in MicroEngine

The implementation of the Bloom Filter can be divided into three parts: the initialization operations; the filtering operations for each packet, and managing operations.

a) The Parameters of Bloom Filters

System parameters is transformed into Bloom Filters Parameters, the following parameters are acquired in current implementation:

The size of BV (Bit Vector): $m = 2048$;

The size of Signatures Library: $n = 128$;

The number of hash functions: $k = 8$.

b) The Hash Functions Choice

(1) *F*, *G*, *H* and *I* functions used in MD5

Let & denote AND operation, ~ denote INVERSE operation, | denote OR operation, ^ denote EXLUSIVE OR(XOR) operations, *F*, *G*, *H* and *I* functions in Rivest's MD5 (RFC1321)[57] are given as follows:

$F(X,Y,Z)=(X\&Y)|(\sim X)\&Z)$; $G(X,Y,Z)=(X\&Z)|(Y\&(\sim Z))$;

$H(X,Y,Z)=X \wedge Y \wedge Z$; $I(X,Y,Z)=Y \wedge (X|(\sim Z)$.

(2) Ramakrishna-Fu-Bahcekapili Hash functions

Given a bit string $X=(x_1, x_2, x_3, \cdots, x_b)_2$, the hash function is like this: $h_r(X)=r_1 \cdot x_1 \oplus r_2 \cdot x_2 \oplus \cdots r_b \cdot x_b$, where "$\cdot$" represents the AND operator and "$\oplus$" represents the bitwise XOR operator. The random number string $(r_1, r_2, \cdots, r_b)$ is previously generated in $[0, m-1]$, hence every hash function maintains b different random numbers.

(3) CRC unit and Hash unit in IXP2400

The CRC Unit operates in parallel with the Execution Datapath. It takes two operands, performs a CRC operation, and writes back a result. There are two polynomials in CRC unit, i.e., CRC-CCITT and CRC-32. By setting the *CRC_Remainder Local CSR*(Control and Status Registers) initially, distinct hash functions can be constructed in use of the same polynomial. The resultant 32bits data of the CRC operation is written back into *CRC_Remainder Local CSR*.

The Hash Unit that can take 48-bit, 64-bit or 128-bit data and produces a 48-bit, a 64-bit or a 128-bit hash index, respectively. Issue a command to the Hash Unit requesting that the data in the transfer registers be moved to the hash unit, a hash operation performed on the data, and the result returned back to the transfer registers.

(4) Hash Function Implementation Discussion

In current design of the AntiWorm system, three methods above have been tried. To implement 8 hash functions, the MD5's function method costs 34 ME's cycles, while using CRC unit takes 152 ME's cycles.

Ramakrishna-Fu-Bahcekapili Hash functions are also tested. It is not very convenience to implement in NP, which prefers the byte operation. This method is more suited for hardware parallel bit operation. So we do not adopt it in the system.

c) The Initialization of the BV of the Bloom Filters

The BV of Bloom filters represents the characteristics of all signatures. Every signature is hashed into eight values in the range $[1, m-1]$ by using 8 distinct hash functions. After hashing, all 8 corresponding bits in BV are set with 1. The XScale core is responsible for the construction of BV and writes the BV into the local Memory.

d) Packet Payload Filtering

The filtering operations are the backbone of the AntiWorm system. The filtering operations run in 8 threads of each ME in order to hide I/O latency.

Each time, 64 bytes of the packets payload are read from DRAM and the reading offset is only added by 48. 48 groups of 16 bytes strings are extracted in the way that byte 0 to 15, 1 to 16, …, 47 to 63. For each string, 8 hash values are calculated, and the corresponding bits of BV are checked. If a hash result is missed, then the following hash functions are aborted, and this string is considered as a ***non-signature***. Otherwise further more exact matching is needed. The packet payload is shifted one byte per step, and the above operation continues. The exact matching operation is to compare the string with all 128 original signatures. Once the string matches one signature, the whole packet is *hit* and will be handled as an exception. If the string does not match any signature, then it is ***false-positive packet***, and the above operation continues. A packet can pass through the Bloom filter, only if every string in the packet payload from head to tail can pass the filter.

e) The LiveUpdate of Signature of Bloom Filters

Worms evolve rapidly and always have several variants; hence the Signature Library often needs update. To add or remove new signatures into Bloom filters, recomputing the BV is required. To reduce the computing cost, countable Bloom filters technique is used.

Functions named AddSignature() and RemoveSiganture() are provided in the software package on the XScale side. To liveupdate the BV in ME, there is a negotiation scheme needed to graceful stop a ME after the ME has finished processing the current packet. The BV in MEs is live updated one by one to keep the system in operation with least performance loss.

f) The Report of Hit in Bloom Filters

The hit of Bloom Filters will cause a ME thread to generate an interrupt of the XScale core. The interrupt invokes ISR (Interrupt Service Routine) to handle the hit packet.

6) String Match engine in XScale

String Match Engine under implemented is used for the stateful flow inspection for higher layer application, and will be delivered in the next version of software package.

IV. Experiments

A. Simulation Experiments

1) The Simulation Experiments Environment Setup

Firstly, simulation experiments are carried out on Intel SDK Developer WorkBench 3.61. All the results are collected under the Intel SDK Developer WorkBench 3.61. In this experiment, *Foreign Model* and *Network Traffic Simulation Plug-in* are used on WorkBench, see Figure 5.

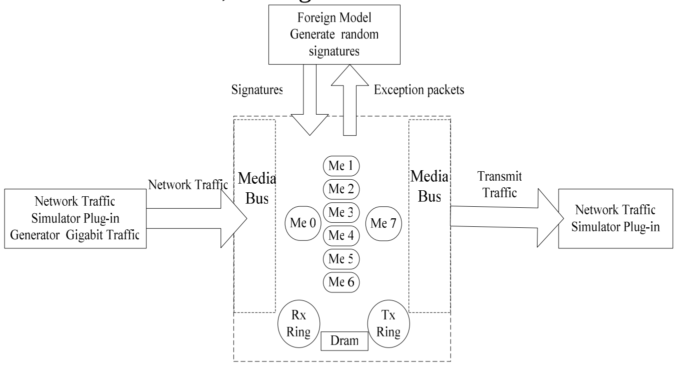

Foreign Model Generate random signatures
Signatures
Exception packets
Network Traffic
Media Bus
Media Bus
Transmit Traffic
Network Traffic Simulator Plug-in Generator Gigabit Traffic
Me 0
Me 1
Me 2
Me 3
Me 4
Me 5
Me 6
Me 7
Network Traffic Simulator Plug-in
Rx Ring
Dram
Tx Ring

Figure 5. The simulation environments.

When the simulation starts, the Network Traffic Simulation Plug-in module begins to send packets to the media bus of IXP 2400 continually. Each byte of these packets' payload is randomly generated and the length of the payload can also be changed. The effect of choice of the hash functions for Bloom Filter are evaluated by checking the false-positive probability. In order to compare the performance of packet forwarding rate, signatures are purposely inserted into the payload of packets. With the variance of the length of the payload and the hit rate, the packet forwarding rate are evaluate in different environment.

2) Results

a) The False Positive Ratio of Packets

Let packet payload length be L, a packet contains $L-15$ strings needed to be investigated, Given the false positive probability $p_f = \left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k$ and the assumption of the independence of the strings, the false positive ratio of packets is about $1-(1-p_f)^{L-15} \approx (L-15)\cdot p_f$, where m, n, k denote the length of BV, the cardinality of the set of signatures, and the number of hash functions respectively.

In the completed implementation, $m = 2048$, $n = 128$, $k = 8$ are chosen, so the possibility is about $p_f = (1-(1-\frac{1}{2048})^{8\times128})^8 = 5.75\times10^{-4}$. Given a packet with payload 64Bytes, 49 strings needed to be investigated, the false positive ratio of packets is about $1-(1-p_f)^{49} \approx 49\cdot p_f \approx 2.8175\%$.

In the proceeding implementation, parameters $m = 16384$, $n = 256$, and $k = 8$ are chosen. In this case, the false positive probability is about 3.63×10^{-8}. The false positive ratio of a packet is about 10^{-5}, when the packet length is maximum (1514 for IP packet in Ethernet).

b) Validation the Bloom filters and MD5's hash function

100,000 packets with payload length 64Bytes are simulated and signatures are intentionally inserted into packets, which accounts for 20%. The experiments adopt both hash function of CRC-unit and MD5, the false positive ratio are compared as performance index.

All packets inserted signatures have been captured, the hash functions bring few effects on the false positive ratio; hence the MD5 method is feasible. **Table 1** shows the results.

Table 1 False positive ratio by using Hash function based on MD5 and CRC-unit.

Hash	Pass Packets	Hit Packets	False positive Packets	False Positive Ratio (Empirical)	False Positive ratio (Theoretical)
MD5	78,409	21,591	1,080	1.08%	2.8175%
CRC	80,008	19,992	733	0.733%	2.8175%

Table 1 shows that the chosen of the hash function take little effect on the false positive probability. Hence, the hash functions can be further simplified to be faster. The actual results are smaller than the theoretical results because of the assumptions are based on the independence of the string, which is not hold on the practice.

c) Throughput

The *Media and Switch Fabric*(MSF) interface is configured in x32MPHY4 mode. There are two media bus devices connected in Media bus, i.e., Device 0 and Device 1. Each Media Bus Device consists of 4 ports, which use POS3 (POS-PHY Level3) protocol. Each port parameters are shown in **Table 2**.

Table 2. Port parameters configured for Media Bus Device

	Receive	Transmit
Data Rate (Mbps)	1000	1000
Buffer size	65536	16384
Interpacket gap (ns)	96	96

The Bloom filters' throughput (Packet forwarding rate) is measured when the packet is input with full rate. Such throughput can be represented with transmitting rate on media bus collected on the WorkBench 3.61. Figure 6 shows the maximum packet forwarding rate of the system with the payload size and hit ratio varying. In the experiments, the packets are simulated with length of 1024, 512, 256, 128 and 64 Bytes, and the probability carrying the signatures is 0%, 20%, 40%, 60%, 80%, 100%.

As the system needs to scan the entire payload by shifting byte one by one, the maximum packet forwarding rate is related to the size of the payload and the hit ratio. When the size of payload increased, the forwarding rate comes down slowly.

The experiments with different packet hit ratio prove that the working of AntiWorm system when worms spread out. In the simulation experiments, the exact matching is also executed on ME. Once a string is hit and proved true by using exact matching, the filter just stops and alarms without any more filter operations. Therefore, it still takes less cycles when handling packets with signatures especially with long packets. It means the performance of the AntiWorm system will not be impaired when a worm breaks out and the network is jammed with worm's packets.

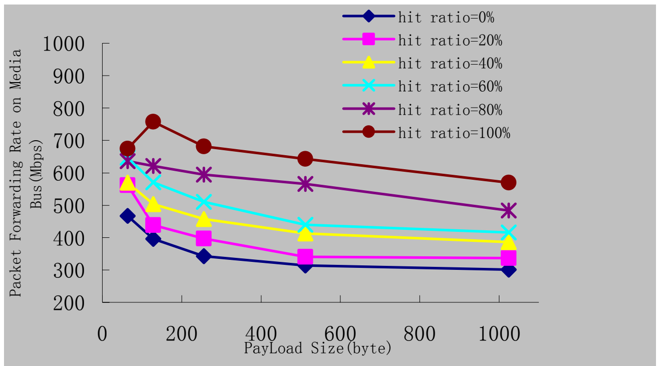


Figure 6. Packet Forwarding Rate on Media bus varies with the hit ratio.

d) Delay

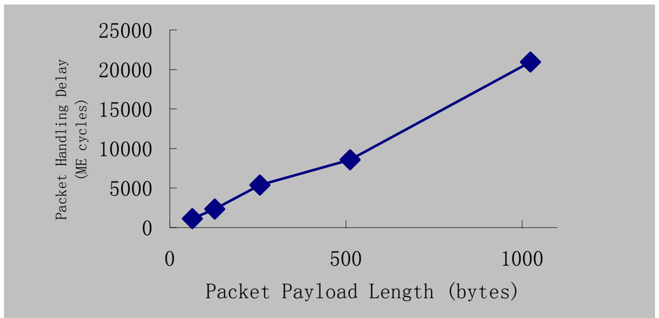


Figure 7. Packet handling delay and packet payload length.

Packet's handling delay is accurately measured in the unit of cycles. The handling delay of each packet is recorded with payload varying from 64 to 1024. The packets are generated without signatures (i.e., hit ratio is 0%), so the Bloom Filter must scan the whole payload, which is the worst case in packet delay. The relation between packet handling delay and packet payload length is shown in Figure 7.

e) Network Processor Internal Statistics

The internal statistics of IXP 2400[6] are collected, which consists of TX ME utilization rate, Bloom filters ME utilization rate, RX ME utilization rate, two SRAM Channel Utilization Rates, and DRAM Utilization Rate. Packet payload with 64bytes, 256bytes and 1024bytes are investigated respectively. Figures 8-12 show the internal statistics. From these figures, it can be concluded that the performance bottleneck of the system is the Bloom filters operation. The SRAM and DRAM are under utilized because each packet only needs one write and one read operations. There are still many spaces remaining further improvement.

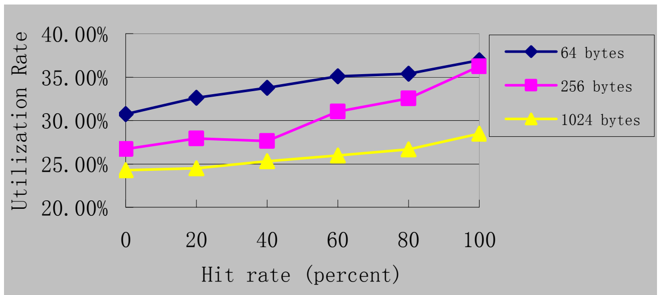


Figure 8. ME Utilization Rate (RX)

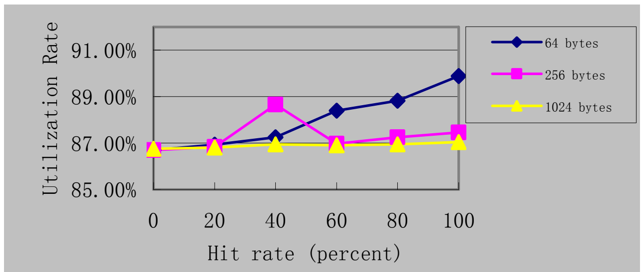


Figure 9. Average ME Utilization Rate (Bloom filters)

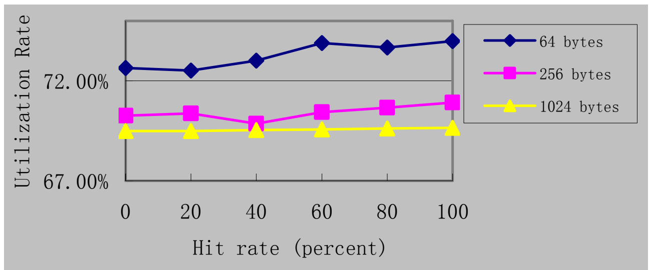


Figure 10. ME Utilization Rate (TX)

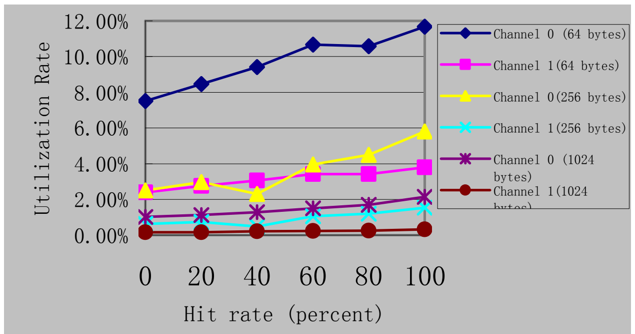


Figure 11. SRAM Channel Utilization Rate

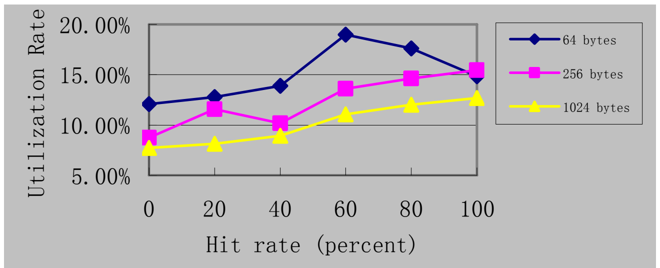


Figure 12. DRAM Utilization Rate

B. Real Experiments

1) The Real Experiments Environments Setup

The test Environment consists of the following Devices: A Gigabit Ethernet Switch (TP-LINK TL-SL2226P+, 24+2G), IXIA 1600 Traffic Generator, ENP 2611 Platform (Containing one IXP2400 Network Processor), and several PC and Servers. The configuration is shown in Figure 13.

The AntiWorm System can work in listen mode or probe mode. The former means the NPU passively intercept all the traffic in transfer in such network scenario, while the later means the system store-handle-forwarded all traffic. In this experiment, listen mode is tested, and all packets carrying the worm's code are captured.

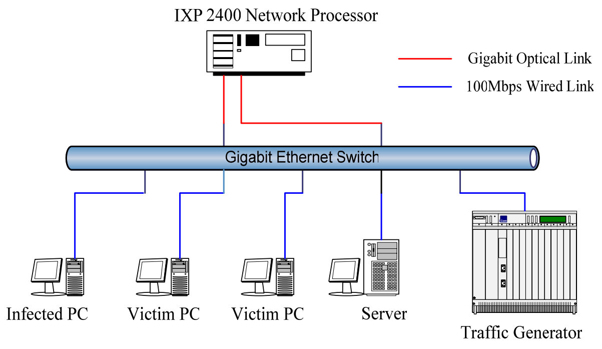


Figure 13. Gigabit Ethernet Test Environments

V. SOFTWARE PACKAGE WORMDETECTOR1.0

For performance issues, Bloom filter is implemented by using microcode [49]. The initialization and management procedures on XScale are written in C language.

We have packed all the codes into one software package, named WormDetector1.0, which integrates about one hundred signatures of Worms. We will distribute software package as shared resource for research purpose in Network Processor field.

VI. FURTHER WORK

The further work includes the following content:

A. Enlarge Worm Signatures Library in Bloom Filters

Worm Signature Library needs to be enlarged to enable AntiWorm System to detect almost all existing worms. Technically speaking, the Bit Vector of Bloom filters must be enlarged, and the number of the hash functions must be increased. One scheme using Local memory as Bit Map to fast index the BV is devised and under development.

B. Migrate Current Software Package to IXP2800

IXP2800 integrates security functionality which is more convince for high speed security issues and provides more remarkably performance issues. In current effort, the software package will be migrated to IXP2800 platform for further study.

C. Function Integration and Augment

More intelligent alarm functions will be augmented for early worm alarming, especially for new worms and their new variants. Technically speaking, a traffic measurement function block will be implemented to identify the instantaneous jumbo suspicious traffic.

At the same time, protocol parsing function block will be added into the current system to analyze the payload with stateful inspection.

D. Performance evaluation and optimization

Performance is the enduring issues, we will continue to optimize the data plane and control plane to improve the performance issues. The statistics collected in Figure 8-12 also gives us some guides for further performance improve.

VII. CONCLUSIONS

The worm immigrates from the uplink of a Giga-Ethernet switch and propagates among hosts in LAN. To protect the hosts in a LAN environment from worms, this research project targets on the implementation of an Anti-Worm system in Giga-Ethernet, i.e., a device to search and locate the track of worms, and deter them from propagating. The main theme is how to discover the worms by using the available information. With the fast packet-handling and highly programmable capability provided by NPU, we implement a searching engine based on Parallel Bloom filters technique. The main idea is to find and locate the worm by detecting the signatures of worms in every packet en route. NPU-Based implementation can scan every packet with high performance and can live update the worm's signature flexibly to keep pace with fast evolution of worms. A prototype of AntiWorm System is implemented, the design principles and details are also introduced, and the simulated network scenarios are constructed. The simulation results are collected and the performance issues of the AntiWorm system are discussed and analyzed, and the experiments carried in test-bed also demonstrate the working of the AntiWorm system.

ACKNOWLEDGMENTS

We thanks for Intel Corporation for donating Intel IXP 2400 developer platform and provides technique supports. We are grateful to Jeffery Cao in Intel China University Program for his support and also thank Brad Burres in Intel Communications Group for his insight suggestions.

REFERENCE

[1] B. Bloom, "Space/Time Trade-Offs in Hash Coding with Allowable Errors," Comm. ACM, vol. 13, no. 7, May 1970, pp. 422-426.
[2] Michael Mitzenmacher, "Compressed Bloom Filters," IEEE/ACM Transaction on networking, vol. 10, no. 5, october 2002.
[3] Douglas E. Comer, "Network System Design: Using Network Processors," Pearson Education Inc., 2004.
[4] Wajdi Feghali, Brad Burres, Gilbert Wolrich, Douglas Carrigan, "Security: Adding Protection to the Network via the Network Processor," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.
[5] Ram Bhamidipati, Ahmad Zaidi, Siva Makineni, Kah K. Low, Robert Chen, Kin-Yip Liu, Jack Dahlgren, "Challenges and Methodologies for Implementing High-Performance Network Processors," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.
[6] Sridhar Lakshmanamurthy, Kin-Yip Liu, Yim Pun, Larry Huston, Uday Naik, "Network Processor Performance Analysis Methodology," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.
[7] Uday Naik, Alex Shoykhet, Larry Huston, Donald Hooper, Raj Yavatkar, Duke Tallam, Travis Schluessler, Prashant Chandra, Adrian Georgescu,

"IXA Portability Framework: Preserving Software Investment in Network Processor Applications," Intel Technology Journal, Volume 06, Issue 03, Published August 15, 2002.
[8] Bill Carlson, Intel Internet Exchange Architecture and Applications: A Practical Guide to Intel Network Processors, Intel Press, 2003.
[9] Erik J. Johnson and Aaron R. Kunze, IXP2400/2800 Programming: The Complete Microengine Coding Guide, Intel Press, 2003.
[10] Peder Jungck and Simon S.Y. Shim, "Issues in High-Speed Internet Security," IEEE computer magazine, Vol. 37, No. 7, pp. 36-42, July 2004.
[11] Darrell M. Kienzle, Matthew C. Elder, "Recent worms: a survey and trends," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 1–10, October 2003.
[12] Nicholas Weaver, Vern Paxson, Stuart Staniford, Robert Cunningham, "A taxonomy of computer worms," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 11- 18, October 2003.
[13] Stuart E. Schechter, Michael D. Smith, "Access for sale: a new class of worm," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 19–23, October 2003.
[14] Michael Liljenstam, David M. Nicol, Vincent H. Berk, Robert S. Gray, "Simulating realistic network worm traffic for worm warning system design and testing," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 24–33, October 2003.
[15] Arno Wagner, Thomas Dübendorfer, Bernhard Plattner, Roman Hiestand, "Experiences with worm propagation simulations," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 34–41, October 2003.
[16] Dan Ellis, "Worm anatomy and model," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 42-50, October 2003.
[17] Cliff Changchun Zou, Weibo Gong, Don Towsley, "Worm propagation modeling and analysis under dynamic quarantine defense," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 51-60, October 2003.
[18] Yang Wang, Chenxi Wang, "Modeling the effects of timing parameters on virus propagation," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 61 – 66, October 2003.
[19] Linda Briesemeister, Patrick Lincoln, Phillip Porras, "Epidemic profiles and defense of scale-free networks," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 67 – 75, October 2003.
[20] Jesse C. Rabek, Roger I. Khazan, Scott M. Lewandowski, Robert K. Cunningham, "Detection of injected, dynamically generated, and obfuscated malicious code," Proceedings of the 2003 ACM workshop on Rapid Malcode, Pages: 76 – 82, October 2003.
[21] Daniel R. Ellis, John G. Aiken, Kira S. Attwood, Scott D.Tenaglia, "A Behavioral Approach to Worm Detection," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[22] Phillip Porras, Linda Briesemeister, Keith Skinner, Karl Levitt, Jeff Rowe, Yu-Cheng Allen Ting, "A Hybrid Quarantine Defense," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[23] Cynthia Wong, Stan Bielski, Jonathan M. McCune, Chenxi Wang, "A Study of Mass-mailing Worms," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[24] Jintao Xiong, "ACT: Attachment Chain Tracing Scheme for Email Virus Detection and Control," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004),, October 29, 2004, Washington, DC, USA.
[25] Nicholas Weaver, Ihab Hamadeh, George Kesidis, Vern Paxson, "Preliminary Results Using ScaleDown to Explore Worm Dynamics," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[26] James E. Just and Mark Cornwell, "Review and Analysis of Synthetic Diversity for Breaking Monocultures," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[27] Stuart Staniford, David Moore, Vern Paxson, Nicholas Weaver, "The Top Speed of Flash Worms," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[28] Evan Cooke, Michael Bailey, Z. Morley Mao, Danny McPherson, "Toward Understanding Distributed Blackhole Placement," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[29] Frank Castaneda, Emre Can Sezery, Jun Xu, "WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism," Proceedings of the 2004 ACM workshop on Rapid Malcode (WORM'2004), October 29, 2004, Washington, DC, USA.
[30] I. Arce and E. Levy, "An analysis of the Slapper worm," IEEE Security & Privacy, Vol. 1 No. 1, Pages: 82-87, Jan.-Feb. 2003.
[31] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. "Slammer Worm Dissection: Inside the Slammer Worm," IEEE Security & Privacy, Vol. 1 No. 4, Pages: 33-39, July-August 2003.
[32] S. J. Stolfo, "Worm and attack early warning: piercing stealthy reconnaissance," IEEE Security & Privacy, Volume: 02, Issue: 3, Pages: 73–75, May-June 2004.
[33] J. Pincus and R. Baker, "Beyond stack smashing: recent advances in exploiting buffer overruns," IEEE Security & Privacy Magazine, Volume 2, Issue: 4, page(s): 20- 27: July-Aug. 2004.
[34] C. Shannon and D. Moore, "The spread of the Witty worm," IEEE Security & Privacy Magazine, Volume 2, Issue: 4, page(s): 46- 50, July-Aug. 2004.
[35] E. Levy, Approaching zero [attack trends], IEEE Security & Privacy Magazine, Volume 2, Issue: 4, page(s): 65- 66, July-Aug. 2004.
[36] J. Newsome, B. Karp and D. Song, "Polygraph: Automatically Generating Signatures for. Polymorphic Worms," to appear in Proceedings of the IEEE Symposium on Security and. Privacy (Oakland 2005), May 2005.
[37] Tadayoshi Kohno, Andre Broido and KC Claffy, "Remote Physical Device Fingerprinting," to appear in Proceedings of the IEEE Symposium on Security and. Privacy (Oakland 2005), May 2005.
[38] Y. Xie, V. Sekar, DA Maltz, MK Reiter, H. Zhang, "Worm Origin Identification Using Random Walks," To appear in Proceedings of the IEEE Symposium on Security and Privacy (Oakland 2005), Oakland, CA, May 2005.
[39] D. Moore, C. Shannon, and J. Brown. "Code-Red: a case study on the spread and victims of an internet worm," Proceedings of the Internet Measurement Workshop 2002, Marseille France, November 2002.
[40] Sarang Dharmapurikar, Praveen Krishnamurthy, T.S. Sproull and J. W. Lockwood, "Deep packet inspection using parallel bloom filters," IEEE Micro, Volume 24, Issue 1, Pages:52 – 61, Jan.-Feb. 2004.
[41] D. V. Schuehler, J. Moscola and J. W. Lockwood, "Architecture for a hardware-based, TCP/IP content-processing system," IEEE Micro, Volume 24, Issue 1, Pages: 62 – 69, Jan.-Feb. 2004.
[42] Zhangxi Tan, Chuang Lin, Hao Yin, Bo Li, "Optimization and Benchmark of Cryptographic Algorithms on Network Processors", IEEE Micro., Volume 24, Issue 5,Sep-Oct issues, 2004.
[43] Signature of worm, see http://www.clamav.net
[44] Symantec Inc., see http://www.symantec.com/
[45] System Detection Inc., see http://www.sysd.com/index.html
[46] Silicon Defense, see http://www.silicondefense.com/
[47] Snort, see http://www.snort.org/
[48] Sourcefire, see http://www.sourcefire.com/
[49] Intel Corporation, IXP2400/IXP2800 Network Processor Programmer's Reference Manuals, May 2004.
[50] Intel Corporation, IXP2400/IXP2800 Network Processor Datasheet.
[51] Intel Corporation, IXP2400/IXP2800 Network Processor Hardware Reference Manual, November 2003.
[52] Intel Corporation, IXP2400/IXP2800 Network Processor Development Tools User's Guide, July 2004.
[53] Intel Corporation, IXP2400 and IXP2800 Network Processors Packet Generator Reference Manual, July 2004.
[54] Radisys Corporations, ENP-2611 Hardware Reference, 007-01419-0000, August 2003.
[55] RadiSys Corporation, ENP Software Development Kit Programmer's Guide, 007-01420-0003, Version 3.5 Release 4, April 2004.
[56] RadiSys Corporation, ENP Software Development Kit: 007-01476-0000, April 2004.
[57] R. Rivest, The MD5 Message-Digest Algorithm, www.ietf.org/rfc/rfc1321.txt.
[58] S. Iyer, R. Rao Kompella, and A. Shelat, "ClassPI:an architecture for fast and flexible packet classification," IEEE Networks, Volume: 15, Issue: 2, Mar/Apr 2001.