

An independently partial pattern matching for content inspection at multi gigabit networks

Woo-Sug Jung*, Taeck-Geun Kwon **

*: *Electronics and Telecommunications Research Institute*
161, GajeongDong, YuseongGu, Daejeon, Korea

** : *Dept. of Comp. Eng, Chungnam Nat'l University*
220, GungDong, YuseongGu, Daejeon, Korea
wsjung@etri.re.kr, tgkwon@cnu.ac.kr

Abstract — In this paper, we propose a novel algorithm for pattern matching at multi gigabit network. It is called as the IPM. Our proposed algorithm can minimize the system overhead which is generated to extract patterns at out-of-sequence packets since the IPM does not require a buffering, reordering and reassembling to keep sequence of packets. Our proposed algorithm performs independently partial pattern matching to extract the pattern from incoming multiple packets at multi gigabit network. The IPM is implemented by FPGA, TCAM and SRAM. Consequently, we show the IPM can reduce 87% of TCAM entries to extract 17 bytes pattern and $w=4$. Moreover, we observed that IPM can perform the scanning of packets at 10.941Gbps and it spent 5.79LC/Char to implement the IPM. Then, we achieved the 1.89 Gbps/LC of PHC score. This means that IPM provides the optimal performance in contrast with hardware complexity. Therefore, we propose the cost-effective algorithm to extract the pattern at multi gigabit network.

Keywords— Pattern Matching, IPM, TCAM, FPGA, Out-Of-Sequence packet

I. INTRODUCTION

Recently, Internet popularization is caused to increase network traffic for services like e-business, VoIP, VoD and P2P. Data for these services is divided into multiple packets with fixed size. Sometimes, it may not guarantee a stable service by an abnormal situation on network. To guarantee QoS for real time service like VoIP, it must be kept steady packet delivery time between peer to peer points. TTA (Korea Telecommunication Technology Association) had measured the QoS for VoIP service by packet delivery time on August, 2004. MOS (Mean Opinion Score), which is a value to measure QoS for VoIP service, is changed from 3.0 to 3.8 in this experiment. MOS is kept more than 3.8 to supply a stable VoIP service. This means that it is important to guarantee a steady packet delivery time in order to satisfy QoS for the real time service like VoIP.

Internet popularization yields to intrusion type through a network from direct to indirect intrusion as well as increasing network traffic. Indirect intrusion reveals to the information

and destroys the system after worm virus or spyware – Codered, Trojan horse, Nimda - in packets are hidden, and infected to the system. To prevent the indirect intrusion, it detects the abnormal behaviour from incoming packets and then report to administrator. The abnormal behaviour is commonly called the violation of the security policy, and malicious activities which hinder an integrity, confidentiality and availability.

To guarantee QoS for real time service and prevent the indirect intrusion, it is necessary the special function – called DPI (Deep Packet Inspection) to inspect TCP header and payload and extract the pattern. The DPI will identify and classify the traffic based on a signature database that includes information extracted from the data part of a packet, allowing finer control than classification based only on header information [20].

NIDS is the system to perform the DPI. In general, the NIDS (Network Intrusion Detection System) uses the method of pattern matching or a statistical analysis to detect the indirect intrusion activity on network. The method of pattern matching extracts signatures from network traffic by using the pre-defined rule-set. The performance of signature-based NIDS has been shown to dominate by the speed of string matching algorithms. Snort is signature based open source network intrusion prevention and detection system (IDS/IPS) developed by Sourcefire[21]. Measurements on Snort show that 31% of total processing and 80% in the case of Web-intensive traffic are due to string matching[6][15]. Moreover, an implementing of different algorithms achieves up to 500% increasing in performance for Snort 2.0[15][19]. There are two kinds of NIDSes, namely software based NIDS and hardware based NIDS. Software based NIDS is not suitable to perform the pattern matching at multi gigabit network because of a performance issue. For example, Snort equipped with a dual 1-GHz Pentium III system can sustain a throughput of only 50Mbps.[17] Therefore, it is necessary a hardware based NIDS to execute the real time pattern at multi gigabit network.

We propose a novel pattern matching method – called *IPM* (*Independently Pattern Matching*) - for hardware based NIDS. IPM minimizes an additional memory and high performance processor to require for extracting the pattern at out-of-sequence packet. It is commonly called the system overhead in

this paper. We adapt TCAM to extract parts of signatures from incoming multiple packets at wire speed and FPGA to implement our proposed algorithm.

This paper is organized as following: we reviewed the related work and summarized the relevant TCAM background and described the network attribute in Section 2. We define a problem for pattern matching at OOS (Out-Of-Sequence) packet in Section 3. Section 4 and Section 5 explain our proposed scheme and performance evaluation. Finally, we conclude the paper in Section 6

II. RELATED WORK

It has been extensively studied about pattern matching. In this section, we discuss the previous pattern matching scheme and network attributes to explain the system overhead. We first review software based pattern matching scheme and explain to hardware based pattern matching scheme.

A. Software based pattern matching scheme

The most influential software-only algorithms are: KMP (Knuth-Morris-Pratt), BM (Boyer-Moore), Aho-Corasick and Commentz-Walter[13]

These methods are based on the brute force algorithm. The KMP and BM are designed for a single pattern matching. These algorithms build a skip table to avoid back tracking and to help shift forward. The scanning time for m bytes pattern in n bytes packet is $O(n+m)$. If there are k patterns, the scanning time is $O(k(n+m))$ [14].

Aho-Corasick and Commentz-Walter are proposed for multiple pattern matching. These algorithms are pre-process the pattern and build a finite automaton which can process the incoming packet in $O(n)$ time.

The algorithms for contents based packet handling have been proposed from a few years ago. The AC_BM (Aho-Corasick-Boyer-Moore) algorithm, which is proposed by Silicon Defence, combines with the BM and AC algorithms [18]. The SBMH (Setwise Boyer-Moore-Horspool) algorithm increases an average performance better than Aho-Corasick and Boyer-Moore. These algorithms like BM_AC and SBMH can increase a performance of pattern scan. However, a software-only pattern matching is not adapted for a pattern matching at multi gigabit network because of a performance issue.

B. Hardware based pattern matching scheme

It requires hardware based pattern matching scheme to resolve a performance issue of software based pattern matching scheme. Normally, hardware based pattern matching scheme uses a bloom filter, CAM or TCAM with co-processor as a hardware component. We adopt TCAM, SRAM and FPGA to implement our proposed algorithm. TCAM is used for scanning of pattern from the incoming packets at wire speed, and FPGA is used for increasing the

performance of pattern matching due to their reconfigurable capacity and parallel ability.

C. Bloom Filter

Dharmapurikar[11] proposed the parallel bloom filter for a multiple pattern matching. Bloom filter has a space effective data structure to inspect the signature. These signatures have saved to calculate lots of hash functions on rule-set. This approach is difficult to handle with thousands of signatures. Bloom filter has two kinds of disadvantages. First, there is the bottleneck by hash collision which is generated to extract the extra lookup from a hash table. To resolve this disadvantage, Song and Lockwood proposed the EBF (Extended Bloom Filter) which removes an additional hash table and uses the information to increase a lookup speed. Second, the bloom filter has a parallelism restriction since a parallel bloom filter should be built to bloom filters for all kinds of patterns according to a pattern length. That is, the parallel bloom filter and EBF is possible to scan less than hundreds of patterns since a pattern length is possessed diversely from tens to thousands bytes.

D. TCAM (Ternary Content Address Memory)

The general memory - SRAM, DRAM and Flash Memory - searches the data with the address but CAM searches the address with the data. According to the supporting logic state, CAM is classified into the BCAM (Binary Content Address Memory) and TCAM (Ternary Content Address Memory). TCAM provides patterns to search not only '0' and '1' but also "don't care" state. On the other hand, BCAM supports only '0' and '1' logic state. TCAM can contain more information than the same size of BCAM since "don't care" state can minimize the table size to retrieve the information. Entry is a minimum unit to store the information on TCAM. TCAM is a high speed and expensive memory which is possible to search information in one clock. Accordingly, TCAM based pattern matching scheme is important to search more than patterns on same size of TCAM. Figure 1 shows the internal block of TCAM.

F. Yu and R. H. Katz[14] proposed TCAM based pattern matching scheme. This scheme separates the long pattern to several short pattern with TCAM width and is compared with TCAM entries and incoming packets. Gao and Zhang[18] proposed a pattern matching scheme to increase performance of the pattern matching. Gao and Zhang used two kinds of TCAMs which are called the front and the back TCAMs for pattern matching. These methods are considered only single pattern. Sung[16] proposed the *m*-byte jumping window to support the pattern matching at multiple packets and increase the performance of a pattern matching. Sung also performed the pattern matching per m bytes. Here, m is the width of TCAM. The above methods for the pattern matching are based on the hash or the linked list mechanism.

We are commonly called the sequential pattern matching since hardware based pattern matching scheme applies the hash, the linked list, DFA (Deterministic Finite Automata) or NFA (Non-deterministic Finite Automata) for pattern matching.

According to the kinds of packets, TCAM entries are dramatically increased in the sequential pattern matching. We discuss about this problem in Section 3.

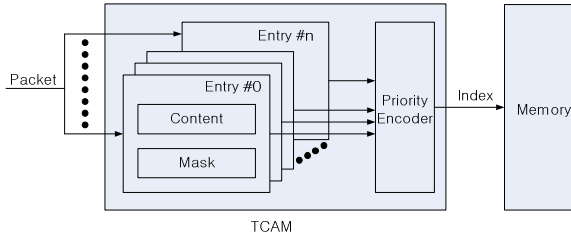


Figure 1. The internal block of TCAM

E. Network Traffic Attribute

Data is transmitted to destination by the single or more than two packets. Sometimes, packets cannot keep the sequence of packets by abnormal situation on network. These packets, which do not keep the sequence of packets, are generated by a changing routing path and a packet loss. In this paper, these packets are commonly called the OOS (Out-Of-Sequence) packet. Receiver performs the fast retransmission algorithm with duplicated ACKs when it received the OOS packets at receiver. The fast retransmission leads to decrease a network bandwidth by reducing CW that TCP transmitter regards to generate network congestion in TCP receiver. TCP receiver delivers packets to higher layer in a regular sequence. Packets are buffered to reassemble and reorder for keeping the sequence of packets when it received the OOS packet. It requires an additional memory for buffering and a higher performance processor for reordering and reassembling to maintain the sequence of packets. The OOS packet is classified into the reordering and retransmission packets. Reordering packets are generated by changing routing path. Retransmission packets are generated by packet loss. According to the study of Bennett [2], Paxson [3] and Jaiswal [9], OOS packet is accounted to 2-5% of the whole packets. Y. Wang [5] analysed the attributes of OOS packet about 200 thousand connections with 3.3 million packets which were generated during 20 days in 2003. In the study of Y. Wang, 5.79% of whole sites received the OOS packet, and the frequency of OOS packet is concentrated on 20% of sites which received the OOS packet. 90% of reordering packets have the time interval less than 5.1ms but, only 3.5% of the retransmission packets have the time interval less than 5.1ms. Some retransmission packets have the time interval more than one second. In case of the packet interval, 95.3% of the reordering packets have a packet interval less than 2 packets but 79.9% of the retransmission packets have the packet interval more than 3 packets. There are the retransmission packets with 55 packet interval.

III. PROBLEM DEFINITION

String $R = r_0, r_1, \dots, r_{m-1}$ is separated to a finite set of substrings $r = b_0, b_1, \dots, b_{n-1}$, $0 < n < m-1$, then transfer to

destination with packet headers through the network. Here, b is a byte. Packet is comprised of a header and payload. In our paper, a payload is called as the substring. Text $T = t_0, t_1, \dots, t_{j-1}$, $j = 1, 2, \dots, n$, which is called as a pattern, is divided to a finite set of substrings t . t is concealed in substrings then transfer to destination. If T is extracted from only one substring, it is called as a SSPM (Single Substring Pattern Matching). If T is extracted more than two substrings, it is called as a MSPM (Multiple Substring Pattern Matching). In this paper, we assume that T has been extracted at two substrings. Two substrings are denoted to r_i, r_{i+1} . The p_i is a packet interval between r_i and r_{i+1} . If p_i is m , it requires an additional memory that size is $m \times l$. Here, l is the length of r_i and r_{i+1} . Generally, hardware based pattern matching schemes saves the incoming packets in order to detect T from OOS substring since it uses the sequential pattern matching.

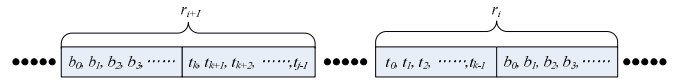


Figure 2. In case of OOS substring, the position of text at r_i and r_{i+1}

IV. PROPOSED PATTERN MATCHING SCHEME

We propose a novel pattern matching scheme – called the IPM (*Independently Partial pattern Matching*) – to minimize the system overhead which is generated by buffering, reordering and reassembling to extract T at the OOS substring. Our proposed algorithm uses TCAM and FPGA for a pattern matching from incoming substrings at multi gigabit network.

A. Proposed pattern matching scheme

IPM is consists of three kinds of stages for pattern matching. IPM constructs the TCAM table in the first stage. To construct TCAM table, T is divided to several number of substrings for MSPM. Here, the length of subtext is equal to TCAM width as denotes w . To minimize the system overhead, our algorithm performs independently partial pattern matching. When IPM performs the partial pattern matching, IPM does not require a relation on the matched sequence between each subtext. If a subtext is extracted, the related information is saved into the lower cost memory like SRAM to decrease hardware cost. We explain the related information at the later section. This information is used to make decision whether T is detected or not in the next stage. The Second stage performs no bearing on the arrived sequence of substring. Finally, the last stage is called as the distance inspection. The distance inspection performs the procedure to calculate each distance of matched subtext. IPM is considered to detect T if the result of calculated distance is zero or meets a matched condition. IPM executes each stage in parallel like pipeline. We explain more detail about each stage at the following section.

B. Stage 1 : constructing searching table on TCAM

Stage 1 performs to construct the searching table on TCAM. As explaining in section 3, our proposed algorithm applies to TCAM for scanning subtexts at real time on multi gigabit network. It cannot be changed the width of TCAM if it is once determined in the initial process. In this paper, entry is mapped to subtext and text is divided into several subtexts for MSPM. The length of a subtext is the w bytes. The w bytes are equal to the width of TCAM. Subtexts are mapped to the entries on TCAM. This process is called as the constructing the searching table on TCAM in this paper. We assume $T = "12345"$ and $w = 4$. It needs only two entities ("--1", "2345") for SSPM. To extract $T = "12345"$ at sequence packet, it needs four pairs of TCAM entries ("--1", "2345"), ("--12", "345--"), ("--123", "45--") and ("1234", "5--"). Unfortunately, in case of the sequential pattern matching, it requires the additional four pairs of TCAM entries ("2345", "--1"), ("345--", "--12"), ("45--", "--123") and ("5--", "1234") as well as TCAM entries to extract $T = "12345"$ at sequence packet. The size of searching table is exponentially increased as length of pattern is linearly increased. If each pair of TCAM entries is extracted, it is a necessary and sufficient condition to detect $T = "12345"$ at our algorithm. This condition is generated at the stage 2. The relationship of TCAM entries can depict to matrix like Figure 3. Each pair of TCAM entries can describe to row elements of matrix.

$$m_i = \begin{bmatrix} ---1 & 2345 \\ --12 & 345- \\ -123 & 45-- \\ 1234 & 5--- \end{bmatrix}$$

Figure 3. Subtext matrix for $T = "12345"$

(1) is the normalized matrix to detect T from the incoming substring. Here, a is $\text{ceil}(l/w)$ and b is $w-l$.

$$m_i = \begin{bmatrix} t_1^1 & t_1^2 & t_1^3 & \dots & t_1^a \\ t_2^1 & t_2^2 & t_2^3 & \dots & t_2^a \\ t_3^1 & t_3^2 & t_3^3 & \dots & t_3^a \\ \dots & \dots & \dots & \dots & \dots \\ t_b^1 & t_b^2 & t_b^3 & \dots & t_b^a \end{bmatrix} \quad (1)$$

C. Stage 2 : IPM (Independently Partial pattern Matching)

This stage is composed of two kinds of processor: IPM_OP and IPM_CON. IPM_OP extracts subtexts from incoming substrings, IPM_CON confirms the necessary and sufficient condition to transit into the next stage.

To extract subtexts, IPM_OP uses a shift register – called *SR* (Scan register) – with w bytes. *SR* fetches the datum from incoming substring and is compared with the datum and TCAM entries. These processes are called the packet scanning. The packet scanning is executed on each clock. If a subtext is extracted at IPM_OP, the related information is saved into SRAM since the possibility for extracting subtexts

is remarkably lower than the frequency of a packet scanning. The related information is used to move into the next stage and to decide whether T is found or not. The related information denotes $i(g, l, m, s, e)$. $i(g)$ is the position of T which is located at strings. This is mapped to a sequential number in case of TCP packet. $i(l)$ indicates the position which is extracted the subtext from substrings. That is, $i(l)$ is equal to the movement count of *SR* for SSPM and MSPM. $i(m)$ shows the matched status whether a subtext is found or not. $i(s)$ and $i(e)$ are the specific position which supports to increase performance of our algorithm. $i(s)$ sets to 1 if $i(l) \leq w$, and $i(e)$ sets to 1 if $i(l) \leq (l-w)$. The related information has one by one correspondence relation with the element of m_i . Following is the matrix for related information.

$$m_i = \begin{bmatrix} i_1^1 & i_1^2 & i_1^3 & \dots & i_1^a \\ i_2^1 & i_2^2 & i_2^3 & \dots & i_2^a \\ i_3^1 & i_3^2 & i_3^3 & \dots & i_3^a \\ \dots & \dots & \dots & \dots & \dots \\ i_b^1 & i_b^2 & i_b^3 & \dots & i_b^a \end{bmatrix} \quad (2)$$

For MSPM as well as SSPM, our algorithm requires only $i(g, l, m, s, e)$ and IPM_CON. IPM_CON is generated in regardless ordering of incoming substring. We denote i_y as all element of y -th row at m_i . IPM condition sets to one when $i_y(m)$ is one. This condition is easily implemented to combination logic with a input and-gates and b input or-gate. Figure 4 shows the combinational logic for IPM condition.

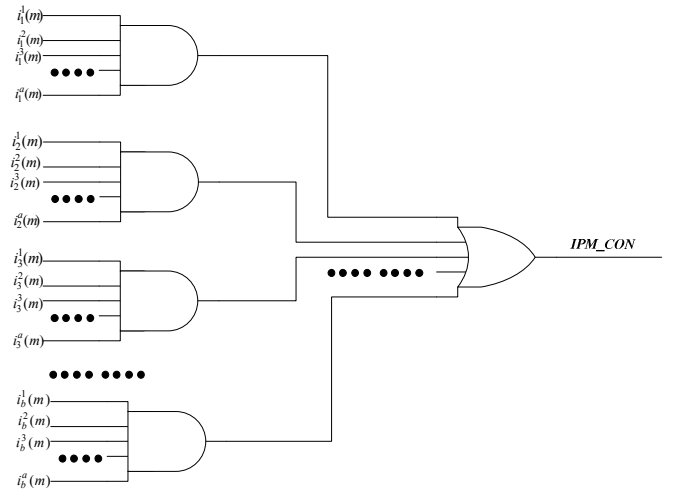


Figure 4. Combinational logic to generate the IPM_CON

D. Stage 3 : Distance Inspection

This stage calculates the distance between each matched subtext and determines whether T is extracted or not. Our proposed algorithm uses $i(g, l, s, e)$ without $i(m)$ to calculate the distance between each matched subtext. We define i_y^x as the related information for an element of x -th row and y -th

column. Here, $x = 1, 2, 3, \dots, a$ and $y = 1, 2, 3, \dots, b$. For SSPM, (3) is the condition to decide whether T is retrieved or not.

$$\begin{aligned} i_y^{x+1}(g) - i_y^x(g) &= 0 \\ i_y^{x+1}(l) - i_y^x(l) &= w \\ i_y^{x+1}(b) - i_y^x(b) &= 0 \\ i_y^{x+1}(b) - i_y^x(b) &= 0 \end{aligned} \quad (3)$$

For MSPM, (4) is the condition to determine whether T is retrieved or not.

$$\begin{aligned} i_y^{x+1}(g) - i_y^x(g) &= 1 \\ i_y^{x+1}(b) - i_y^x(b) &= 1 \end{aligned} \quad (4)$$

V. PERFORMANCE EVALUATION

A. Implementation

We use a few kinds of hardware devices to implement the proposed algorithm. TCAM is used for pattern scanning at multi gigabit network and returns the related information to FPGA and SRAM. FPGA can be used for the fast pattern matching due to their reconfigurable capacity and parallel ability. IPMH (Independently Partial pattern Matching Hardware) is FPGA to implement our algorithm. IMPH has three kinds of internal blocks - MemIF, IPMGen and DisInspect. MemIF generates signals to save/reload the related information on/from SRAM. IPMGen generates PM_CON. IPM_CON is activated to the DisInspect. When DisInspect is activated, this block calculates the distance between each matched subtexts with the related information. The internal block of IPMH is illustrated in Figure 5.

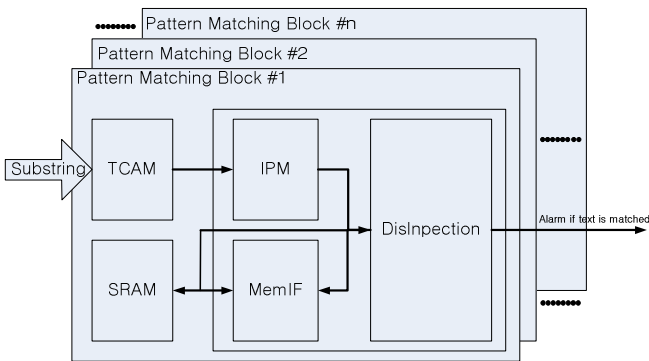


Figure 5. The hardware internal block of our proposed algorithm

IPMH is made by AHDL (Altera Hardware Description Language) which is similar with VHDL or Verilog. We use the Startix II. Startix II is a kind of FPGA family which is manufactured by Altera corp. IPMH has spent 330 LC (Logic Cell) and 14 clocks to extract the text with 57 bytes. Here, width of TCAM is 8 bytes and m_i is matrix with 8×8 . IPMH can execute the pattern matching at 330MHz.

B. TCAM efficiency

TCAM efficiency means how many texts are extracted at the same size of TCAM. Generally, the number of TCAM entries – denotes e_s – for SSPM is $\text{ceil}(l/w)$. It requires $e_s \times w$ of TCAM entries to extract the text at MSPM without OOS packet. That is, e_m is $\text{ceil}(l/w) \times w$. e_m is the number of TCAM entries for MSPM without OOS substring. In case of sequential pattern matching, it is necessary $(l+w-1) \times e_m$ of TCAM entries to extract a text at OOS substring since the sequential pattern matching extracts a text after reordering the OOS substrings. That is, e_o is $\text{ceil}(l/w) \times (l+(w-1))$. e_o is the number of TCAM entries for MSPM including OOS substring. However, our algorithm does not require the ordering and reassembling to extract a text at OOS substrings. Therefore, our algorithm needs only e_m for pattern matching at OOS substring.

Snort version 2.4 has 2,535 unique patterns. The average length of patterns is 17 bytes and maximum length of patterns is 364 bytes. We are depicted to the number of texts in Figure 6. Figure 6 shows the difference of TCAM efficiency between our proposed algorithm and the sequential pattern matching. Here, the size of TCAM is 100Kbyte and the width of TCAM is 4 bytes. IPM is possible to extract 2084 texts on TCAM which size is 100K bytes. However, it is only support 250 texts on same size of TCAM in case of the sequential pattern matching. Consequently, IPM saves 87.99% of TCAM entries which use to extract the pattern at the sequential pattern matching scheme.

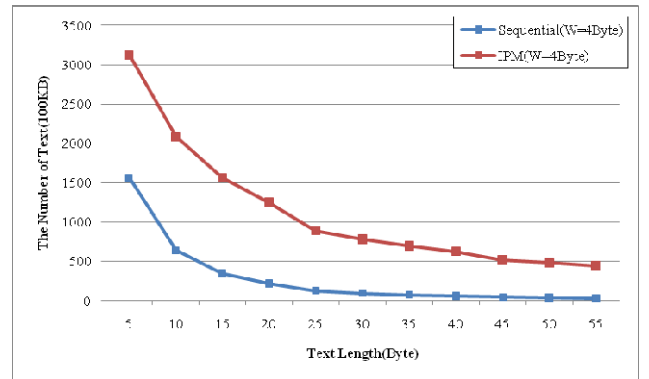


Figure 6. The efficiency of TCAM (TCAM size = 100KByte)

C. Performance in contrast with hardware cost

In this paper, the hardware cost means the amount of resources which is used to implement for pattern matching. The hardware cost of IPMH is 5.79 LC/Char since the proposed algorithm spent 330 LC to extract a text with 57 bytes. Throughput for our algorithm is 10.941 Gbps since IPM spend 14 clocks to extract a text with 57 bytes. IPMH is possible to perform a pattern matching on 330MHz. The PHC (Performance per Hardware Cost) is defined as the throughput per one logic cell. PHC score is 1.89 Gbps/LC on our algorithm. If α algorithm is a higher PHC score than β algorithm, α algorithm is possible to implement the pattern matching with a

lower hardware resource than β algorithm. IPM has higher 87% of PHC score than Sourdis which has the second of PHC score.

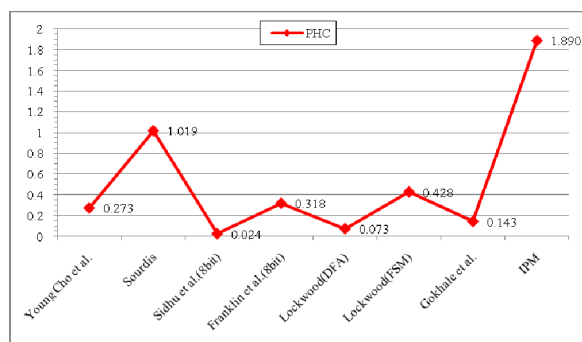


Figure 7. The PHC (Performance per Hardware Cost)

VI. CONCLUSIONS

In this paper, we propose the novel algorithm – called IPM - for real-time pattern matching at multi gigabit network. IPM performs the pattern matching operation to minimize the system overhead which is generated to extract the pattern at OOS packet. Therefore, IPM has higher performance and lower hardware complexity than the sequential pattern matching. IPM can reduce 87.99% of TCAM entries than the sequential pattern matching scheme. Our algorithm spent 330 LC and 14 clocks to extract the pattern with 57 bytes. It gets 1.89Gbps/LC for the PHC. IPM is suitable to a cost-effective system for pattern matching since our algorithm has a higher PHC score than other algorithm. IPM is possible to execute a pattern matching at multi gigabit network. We show IPM can provide better performance in contrast with hardware more complexity than the sequential pattern matching. Therefore, we propose hardware based cost-effective and high performance pattern matching scheme which is possible to adapt at multi gigabit network.

VII. ACKNOWLEDGMENT

This work was supported by the Ministry of Knowledge Economy (MKE) of the Republic of Korea [2006-S-601-01, Development of Distributed Sensor Network System]. This research was supported by the MKE under the ITRC (Information Technology Research Center) support program supervised by NIPA (National IT Industry Promotion Agency) (NIPA-2009-(C1090-0902-0016))

VIII. REFERENCES

- [1] G. A. Stephen, "String Searching Algorithm", Lectures Notes Series on Computing, Vol. 3, 1994
- [2] J. C. Bennet, C. Partidge, N. Shectman, "Packet Reordering is not pathological network behavior", IEEE/ACM Transaction on Networking, Vol. 7, Issue 3, pp 789-798, 1999.
- [3] V. Paxson, "End-to-end Internet Packet Dynamics", IEEE/ACM Transaction on Networking, Vol. 7, Issue 3, pp 277-293, 1999.

- [4] R. Sidhu, V. K. Prasanna, "Fast regular expression matching using", FCCM 2001, 2001
- [5] J. W. Lockwood, "An open platform for development of network processing modules in reconfigurable hardware", IEC DesignCon 2001, 2001.
- [6] M. Fisk, G. Varghese, "An analysis of fast string matching applied to content-based forwarding and intrusion detection", Technical Report CS2001-0670, Univ. of California San Diego, 2002.
- [7] R. Franklin, D. Carver, B. Huchings, "Assisting network intrusion detection with reconfigurable hardware", FCCM 2002, 2002.
- [8] Y. H. Cho, S. Navab, W. H. Magione-smith, "Specialized hardware for deep network packet filtering", FPL 2002, LNCS 2438, pp. 452-461, 2002
- [9] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, "Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone", Sprint ATL Technical Report, 2002.
- [10] I. Sourdis, D. Pnevmatikatos, "Fast, large-scale string match for a 10Gbps FPGA-based network intrusion detection system", FPL 2003, 2003.
- [11] S. Dharmapurikar, et al, "Implementation of a Deep Packet Inspection Circuit using Parallel Bloom Filters in Reconfigurable Hardware", In Hot Interconnects, 2003
- [12] Y. Wang, G. Lu, X. Li, "A Study of Internet Packet Reordering", ICOIN 2004, LNCS 3090, pp 350-359, 2004.
- [13] F. Yu, R. H. Katz, T. V. Laskhman, "Gigabit Rate Packet Pattern Matching with TCAM", UCB technical report, UCB/CSD-04-1341, 2004
- [14] F. Yu, R. H. Katz, T. V. Lakshman, "Gigabit Rate Packet Pattern Matching with TCAM", ICNP 2004, Berlin, Germany, Oct. 2004
- [15] R. Liu, C. Kao, H. Wu, M. Shin, N. Huang, "FTSE: The FNP-Like TCAM searching engine", ISCC 2005, 2005
- [16] S. J. Sung, S. M. Kang, Y.S Lee, T.G. Kwon, B.T. Kim, "A multi-gigabit rate deep packet inspection algorithm using TCAM", Globecom 2005, 2005
- [17] C. C. Wu, S. H. Wen, N. F. Huang, C. N. Kao, "A Pattern Matching Coprocessor for Deep and Large Signature Set in Network Security System", Globecom 2005, 2005
- [18] M. Gao, K. Zhang, J. Lu, "Efficient Packet Matching for Gigabit Network Intrusion Detection using TCAMs", Proc. Of the 20th International Conference on Advanced Information Networking and Application, 2006
- [19] N. Desai, "Increasing performance in high speed NIDS", <http://www.linuxsecurity.com/article>
- [20] http://en.wikipedia.org/wiki/Deep_packet_inspection
- [21] Snort, www.snort.org