Centralized Parallel form of Pattern Matching Algorithm in Packet Inspection by Efficient Utilization of Secondary Memory in Network Processor

N.Kanniya Raja
M.E.,(PhD).,A.P/CSE Dept.
Arulmigu Meenakshi Amman College of Engg,
Thiruvannamalai Dt, near Kanchipuram.

K. Arulanandam
Prof & Head,
CSE Dept
Ganadipathy Tulsi's Jain Engineering College,
Vellore.

B. Raja Rajeswari
M.E/CSE.,
Arulmigu Meenakshi Amman College of Engg,
Thiruvannamalai Dt, near Kanchipuram.

ABSTRACT

The network equipment has capable of inspecting packets in order to discover the worms and virus over the network. Many network users are hacked by attackers through malicious functions are mapped on network applications. Such unauthorized activities are required to delete by deep packet inspection in application layer. The high level network equipment provides in-depth packet inspection through pattern matching in network detection system. By presenting centralized parallel pattern matching algorithm for efficient packet inspection with network processor and coprocessor in order to retrieve the pattern with less time.

General Terms

Packet Inspection , Pattern Recognition, Network Security.

Keywords

Intrusion detection, Pattern matching , Packet inspection, Packet payload.

1. INTRODUCTION

Now – a – days network services are tremendously increased by the fast emerging technologies. Many of the companies shares their information by providing the network service over the internet. Hence the Network Intrusion Detection System (NIDS) has introduced to control and analyses the packets. Network Intrusion Detection System (NIDS) that contains the detection engine that provide in-depth packet inspection. There are several algorithms are developed to design the detection engine. But most of the algorithm provide low – layer detection which inspect only the packet header that contains only specified fields. High layer network equipment have able to inspect the packet payload that contains the entire up –to – date information of the packet . The number of patterns is typically a few thousands, and the lengths of the patterns are varied. The patterns may appear anywhere in any packet payload. Consequently, the emerging high-layer network equipment needs a pattern detection engine capable of in-depth packet inspection, which searches the entire packet headers and payloads for pattern matching. Network equipment then employs the detection results to manage network systems intelligently. For instance, Snort is an open- source network-based intrusion detection system (NIDS) and is adopted for detecting anomalous intruder behavior with a set of patterns and generating logs and alerts from predefined actions [1]. The patterns of Nimda worm is prescribed in the detection engine of Snort which finds this pattern existing in a packet, the corresponding alert is generated to warn net-work administrators. The pattern matching is considered as the most resource-intensive task in the Snort detection engine [2]. Hence, this study focuses on the nascent issues of the payload inspection. This study proposes a Centralized Multi Parallel Pattern matching algorithm (CNMPPMA) for fast packet inspection, which simultaneously searches the packet payload for a set of patterns for the multiple nodes. This study contributes modifications to the Hierarchical matching algorithm (HMA) [9] and introduces the idea of a sampling window and a Safety Shift Strategy in addition. CNMPPMA is a multi-tier and cluster-wise matching algorithm and can perform fast skippable payload scan. Based on the occurrence frequency of grams, this study discovers a small set of signatures from the patterns themselves to narrow the searching domain. A Min-Max strategy is used in the CNMPPMA. The hit rate of the initial table in the CNMPPMA is minimized, while the spread of patterns in the next table is maximized. Accordingly, CNMPPMA significantly reduces the number of memory accesses and pattern comparisons. CNMPPMA can skip unnecessary payload scans by applying the proposed Safety Shift Strategy, which is based on a frequent – based bad gram heuristic. The frequency- based bad gram heuristic is a modification of the bad grouped character heuristic of Wu-Manber (WM) algorithm [10]. Therefore, CNMPPMA has the advantages of both HMA and WM. The memory space and the number of external memory accesses required by the proposed CNMPPMA are much smaller than those required by state-of-the-art multipattern matching algorithms. Simulation results reveal that CNMPPMA outperforms the state-of-the-art algorithms. Even under real-life intense attack,CNMPPMA still outperforms others. Because it employs only basic instructions and two small index tables, CNMPPMA is very simple for hardware and software implementations. Consequently, the proposed CNMPPMA is a very cost-effective and efficient mechanism for real-life network detection systems. The rest of this paper is organized as follows: Section 2 presents previously proposed pattern matching algorithms and the fundamental definitions. Section 3 then describes the proposed CNMPPMA in detail. Next, Section 4 presents the results and implementations of CNMPPMA. Section 5 draws up the Conclusion .

2. RELATED WORK

Multipattern matching is one of the well-studied classical problems in computer science. The most notable algorithm Aho-Corasick and Commentz - Walter algorithms which can be considered as the extension of well-known KMP and Boyer-Moore single pattern matching algorithms, respectively[3],[13]. Both these algorithms are suitable only for software implementation and suffer from throughput limitations. The current version of Snort uses an optimized Aho-Corasick algorithm. In the past few years, several interesting algorithms and techniques have been proposed for multipattern matching in the context of network intrusion detection. The hardware-based techniques make use of commodity search technologies such as ternary content addressable memory (TCAM) or reconfigurable logic/FPGAs]. Some of the FPGA-based techniques make use of the on-chip logic resources to compile patterns into parallel state-machines or combinatorial logic. Therefore, scalability with pattern set size is the primary concern with purely FPGA-based approaches. The technique proposed in [4] seeks to accelerate the Aho-Corasick automation by considering multiple characters at a time.

3. THE CENTRALIZED MUTI PARALLEL PATTERN MATCHING ALGORTITHM (CNMPPMA)

In the Centralized Multi Parallel Pattern Matching Algorithm(CNMPPMA) is a centralized system based on a multi-tier and cluster-wise architecture as shown in Fig.1. In the Fig.1.,the Centralized Network Intrusion Detection System(NIDS) is constructed to monitor the packets that arrives from 'n' number of nodes. The Internet traffic is made of streams of fragmented packets consisting header and payload. Since attack can span more than one packet of a stream, every stream needs to be reassembled before applying the deep packet inspection.

There are some class of attacks that use unconventional protocol features and packet fragmentation to elude the intrusion detection system. One such attack uses overlapping fragmented IP packets. Such attacks can be eliminated by normalizing the packets. Packet normalization produces consistently clean network traffic without abnormalities . Fig 1 shows an effective deep packet inspection steps which begins with packet normalization followed by static and dynamic inspections. Static inspection step classifies the incoming packets using the header information, while dynamic inspection searches through the payload of the packet to find the patterns defined in the attack signature database . CNMPPMA comprises two small index tables, namely the first table (H^1) and the second table (H^2).These two tables act as filters to avoid unnecessary external memory accesses and pattern comparisons and, thereby, pass the innocuous packets quickly in the online matching process. The second-table procedure activates only after the first-table procedure gets a match.

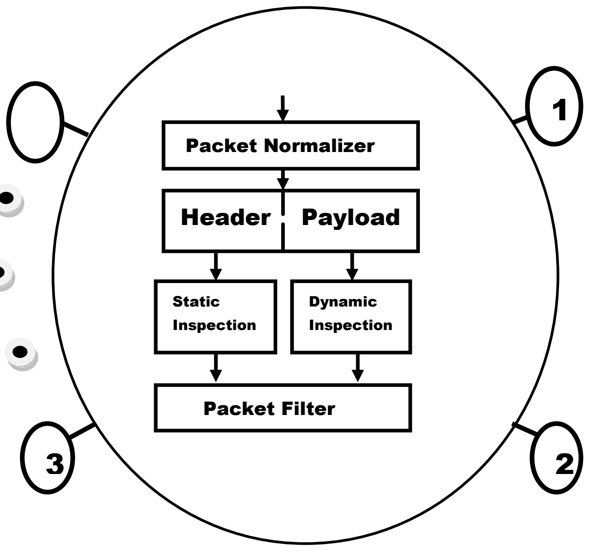


Fig.1. Block Diagram of CNMPPMA

Using H_2 , which indicates a small subset of patterns that are similar to the input packet, P with the CNMPPMA compares only a few selected patterns of suspected substrings of the packet, rather than comparing all patterns with all substrings of the packet. Furthermore, a frequency-based bad gram heuristic is proposed in the CNMPPMA to determine the safety shifts on the input strings during the matching process. However, frequently accessing the external memory (to read patterns or tables) significantly decreases the matching efficiency due to the external memory access latency being very long and indeterminable. The memory latency strongly affects the throughput of pattern matching. Therefore, reducing the number of required external memory accesses is more important.

3.1 Common – Gram Searching (CGS)

The efficient Intrusion Detection System(IDS) provided with high speed detection engine which can be able to handle large pattern set .The pattern set contains thousands of pattern. For the String matching algorithm it is difficult to search the pattern. This will take large amount of time to fetch the pattern. By implementing the CGS algorithm can able to reduce the searching time. CGS aims to detect the gram that common to the pattern set. 50% of the searching process is completed with the CGS algorithm.

It follows the fact : that if packet payload contains the common gram then it came to be conclude that pattern may exist for packet partially. In the high-layer intrusion detection, patterns may appear anywhere in the packet payload, making the attacking packets difficult to recognize.

CNMPPMA assumes that a small set of signatures can be found from the patterns themselves, then the suspicious substrings of T may be easier to distinguish from the innocent parts, and the pattern matching is therefore faster. A set of significant grams is defined as representatives of a pattern set P.

Fig.2 illustrates the sampling window, where T_1 is the size of a frequent-common gram, $T_1 < W$, and T_2 is the size of the second pivot in the H^2 table, which is explained later . The Common Gram Searching (CGS) algorithm is presented in Fig. 3. A bitmap vector V and a matrix M are temporary memories, where $0 \leq i, j < |P|$. Vector V records the occurrence of each T_1-gram in a pattern; M is used for recording frequency, where $i \neq j$, indicates the number of concurrent occurrences of two T_1-grams g_i and g_j in P; and r_{ii} records the frequency of the T_1-gram g_i occurring in distinct patterns.

Matrix M is then derived from V (as shown in line 4 of Fig. 3). Second, the largest occurrence frequency is found, and its corresponding gram g_f is selected as one of F. The elements of M relating to g_f are subtracted accordingly to renew M. CGS is repeated until all elements on the diagonal of M become zero. Hence, CGS can easily discover the most frequent grams from patterns and obtain a small F as the signatures of pattern set. This will achieve the fast matching process with less time consumption as well as to enhance the multi tier organization.

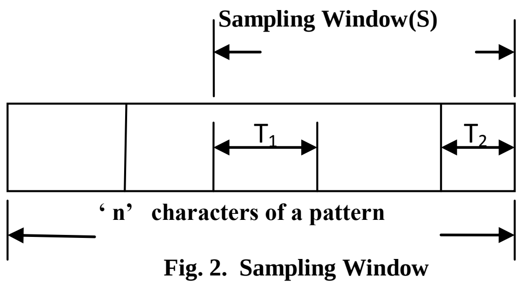


Fig. 2. Sampling Window

```
Input: Given a set of patterns P, the parameters: S, $T_1$, $T_2$ and n.

Output: A set of common gram C.

1   Initialize: C,V,M are set to zero

2   For each pattern $P_i$, of P, $0 \leq i \leq |P|$ do /* build a matrix M */

3   Shift the first S-$T_2$ bytes of the sampling window of the pattern $p_i$ in to $T_1$- grams and set the element of a vector V.

4   Get the flag value V. For each $v_j = 1$, set the elements of matrix R: $r_{jk} = r_j + v_k$ .

5   While ( $r_{ii}$ != 0 ) do

6   Find a common gram $c_k$, where $r_{ff} = \max \{ r_{ii} \}$

7   Add this gram into I: I=I + { c.I }

8   For $0 \leq i \leq |P|$ do/* act on the diagonal of M */

9   $r_{ii} = r_{ii} - r_{ji}$ if $r_{ii} > r_{ji}$ otherwise $r_{ii} = 0$ ;

10  Return ;
```

Fig. 3. The CGS Algorithm

3.2 Cluster Balancing Strategy (CBS)

Most packets are innocent in general situations. Even a harmful packet may contain only few patterns. Therefore, comparing all of the patterns in the large P with each input packet is time consuming. If the patterns in P can be distributed into different small clusters based on their similarity, then only the pattern in each cluster that is most similar to the suspected packet needs to be compared, thus improving the efficiency of the matching process. This section presents strategies to attain this goal. First, the method of clustering a set P based on the similarity of patterns is described. Then, a CBS is adopted to balance the cluster size. The clustering pivots are the keys used to distribute patterns, where each clustering pivot is a common gram of patterns defined previously. Two common grams are employed as a pair of clustering pivots, called a pivot pair, say (a,b) where the first pivot is a frequent-common gram, and the second pivot is the substring following the frequent-common gram. Notably, a pattern is assigned to only one cluster in the clustering strategy, although a pattern may have more than one pivot pair. Since a pattern may have several opportunities to select a cluster, a better assignment can lower the maximum cluster size and ,thereby, improve the worst-case performance of CNMPPMA. The pattern grouping is based on I. To lower the worst matching time, CBS is adopted to balance the size of all clusters. The CBS is given as follows :

1. First, read one pattern at a time from P and scan the pattern.
2. According to CGS, for any given p_i , there exists a T_1- gram $g \in I$ where T_1 is the length of a frequent common gram. To balance the cluster size, CBS finds the smallest $n_{a,b}$, given by $n_{x,y}$, among all available pivot pairs (a,b)'s of p_i for all $a \in I$ and 'a,b' c p_i .
3. After grouping p_i into the smallest cluster $p_{x,y}$, the corresponding $n_{x,y}$, is also incremented.

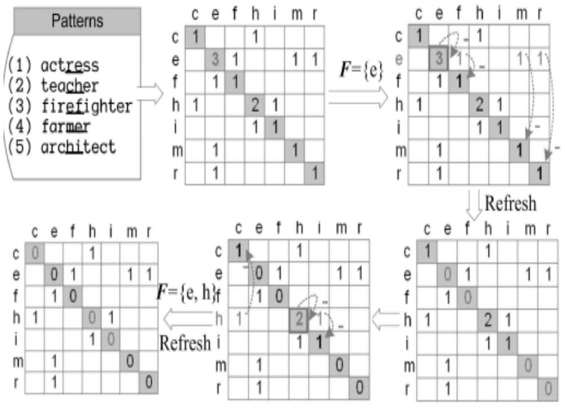
3.3 Safety Shift Strategy

This section presents a safety shift strategy to derive the values of the shift fields of H^1 and H^2. H^1 and H^2 can use the same strategy to derive their safety shifts, respectively. The proposed strategy helps CNMPPMA to speed up the matching process, since certain characters can be skipped unhesitatingly. since x is all the possible of the pivot pairs (a,b) , $x \in I$. The basic concept of the safety shift strategy is that: if x is not a gram of any pattern, and any suffix of x is not any prefix of any pattern in P, then it is safe to shift m when x is scanned. Two parameters are needed to derive the safety shifts, namely W and m, as shown in Fig. 2. Assume that $B \leq W \leq m$, and define the safety shifts in which the values associated with the table that carries the address for the gram. The safety shifts are related with the link pointer for the frequent gram obtained from CGS are given as input to the table of multi-tier of various hierarchy table of each entry , (H (x). shift) as follows:

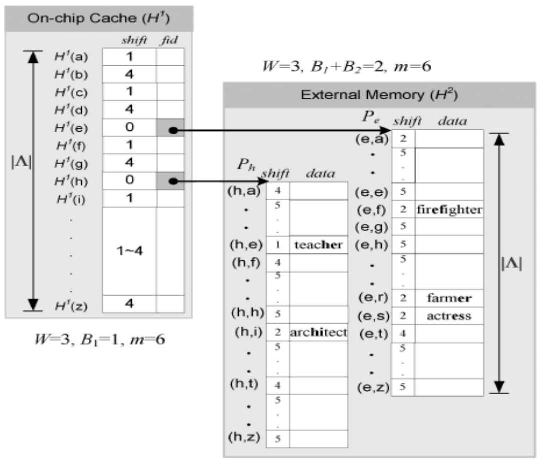
```
1.  Initially , all shift fields of the table H are set a
    If $m>W$, then  $H(x).shift=m-W+q$,
         Else              $H(x).shift= r$;

2.  Scanning every pattern p, for each ith B- gram of each pattern $p^B[i]$, where
         $1 \leq i \leq m-W$,set $x[i] \leftarrow p^B$ if the entry H(x) exists:
    If the current  $H(x).shift > m - W - i + 1$,
         then, update, as $H(x).shift = m - W - i + 1$;

3.  For each ith B-gram of each $p^B[i]$, where $m-W<i/, B+ 1$, set $x \leftarrow p^B[i]$ if
         the entry H(x) exists:
    If $x \in F$, then
              $H(x).shift=0$;
    Else If the current $H(x).shift>r$, then update the entry:
              $H(x).shift=r$;
```

3.4 Table Construction

The result of CGS, I, is used to construct the small table H^1,which is stored in the on-chip memory. A direct index table of entries is used for H^1 to achieve fast lookup. B^1 is usually very small entries is used for H^1 to achieve fast lookup An entry of H^1 is denoted as $H^1(a)$, where a is a T_1-gram, and each entry has three fields: the frequent-common gram ID, $H^1(a)$.fid; the pattern ID when a itself is a pattern, H^1.(a).pid, and the safety shift number in the first table, $H^1(a)$.shift. Fig . 4 shows an example of CNMPPMA, which has five patterns: "actress," teacher," "firefighter," "farmer," and "architect," where the alphabet set comprises the 26 English letters. The parameters for CNMPPMA are assumed T_1 = 1, T_2 = 1, n = 6, and W = 3. Fig. 4a demonstrates the CGS. According to the CGS (lines 2-4 of Fig. 3), after scanning the first S - T_2 characters of the sampling window of every pattern (the underlined characters of the patterns in Fig. 4a), the matrix M is obtained and shown in the figure. In the first run, the maximum value on the diagonal of M is three, and thus the corresponding gram "e" is added into F.



(a)



(b)

Fig. 4. An Example of CNMPPMA, where B1=1, B2=1,m=6,W=3, and F={e,h}. (a)An example of GFGS. (b)The architecture of the hierarchical hash tables.

Fig. 4b displays the logical architecture of the tables of CNMPPMA . The fid fields of H^1 point to the corresponding offsets of H^2. As the pattern "actress" has 'e'∊ I and the pivot pair "es" according to CBS it is grouped to the cluster. The shift fields of H^1 and H^2 are obtained from the proposed safety shift strategy. Initially, since T_1 = 1, H^1.shift = 4. H^2.shift is set to 5 for those entries whose second pivot is not the prefix of any pattern . otherwise, H^2.shift is set to 4. When scanning the pattern "actress," the shift fields of H^1('a'), H^1('c'), and H^1('t') are updated to 3, 2, and 1, respectively (the second safety shift strategy); the shift fields of H^1('r') and H^1('s') are both updated to 1, while the H^1('e').shift is updated to 0, because 'e'∊ I (the third strategy). As for the table H^2, only the existing entry H^2 ('e','s') has to be updated to 2.The remainders of the patterns follow the same clustering and safety shift strategy . When scanning the pattern "actress," H^1 ('a').shift = 3 (as $p^1[i]$='a',i=1,m-W-i+1=3); while scanning the pattern "teacher," H^1 ('a').shift is updated to 1 (as "a" is the third character of "teacher" i = 3, then m-W-i+1=1), because the new value is smaller than the previous one (the second strategy).

Assume that the input T is "kangaroo" as given in Fig 5. The scan runs from left to right. Because the pointer goes beyond [T]-B_1 after the shift, CNMPPMA completes scanning the input T. This example only requires one on-cache table lookup. Considering another example where T = 'iamanactress' as shown in Fig. 6, the first scanned B_1-gram is "a". Thus, the matching process stays in the Multi – Tier Matching, and the next T_1-gram "n" is read after shifting one character. Similarly, staying in the Multi - Tier Matching, and the next B1-gram "n" is read after shifting one character.

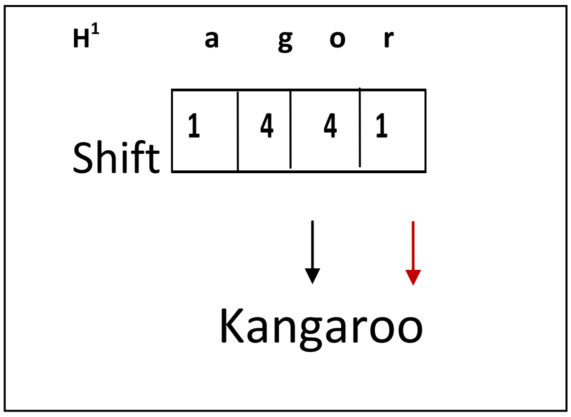


Fig. 5. An example of matching process with input "kangaroo."

Similarly, staying in the Multi - Tier Matching, in order after shifting. After checking the field and finding that it is not NULL, CNMPPMA knows a suspected pattern may exist.

The Multi - Tier Matching then compares input T with the pattern in the cluster 'actress', and gets a match. Because this cluster contains no other patterns, the matching process returns to Multi - Tier Matching.Since the pointer goes beyond |T| - B_1 after shifting two characters, the matching process for the input T is finished. In this case, H^1 is checked four times, and H^2 is fetched only once for the string T of 12 characters. CNMPPMA thus significantly reduces the latency caused by memory accesses.

4. RESULTS AND IMPLEMENTATION

In Figs. 7, 8 the results labeling CNMPPMA in the following simulations use the sampling window with parameters W = m = |pi| which means that each pattern is sampled in its entirety.

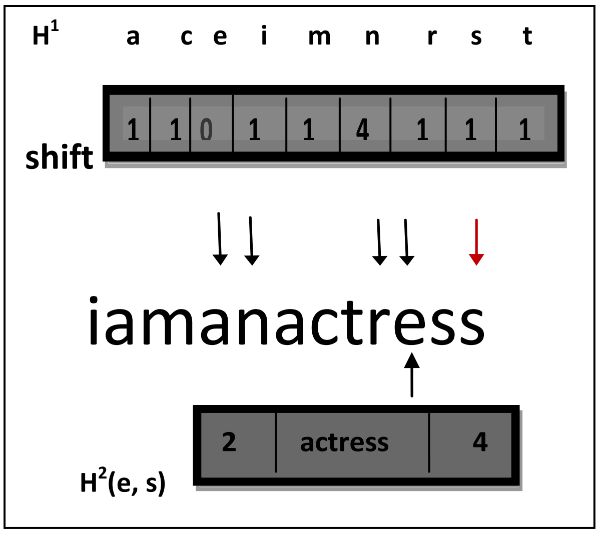


Fig. 6. An example of matching process with input "iamanactress."

The results show that the CNMPPMA is close to that of HMA and WM-PH, but CNMPPMA is much less than others. The BMH seems smaller than others, because the whole skip table of a pattern is idealistically assumed to be loaded within one external memory access and kept in the cache during the matching process for each pattern. Because AC-C compresses the data structure of the state machine, it requires more time to derive the next state pointer. Therefore, AC-C does not have the smallest value . Simulation results show that how significantly memory rise with in any of the experiments, because each algorithm has already tried to reduce the computation load . However, it dominates the overall matching cost. This reveals that the number of external memory accesses is the bottleneck of almost all algorithms. This result also reflects our opinion mentioned previously that the essential issue in designing a high-speed detection engine is to reduce the number of required external memory accesses.

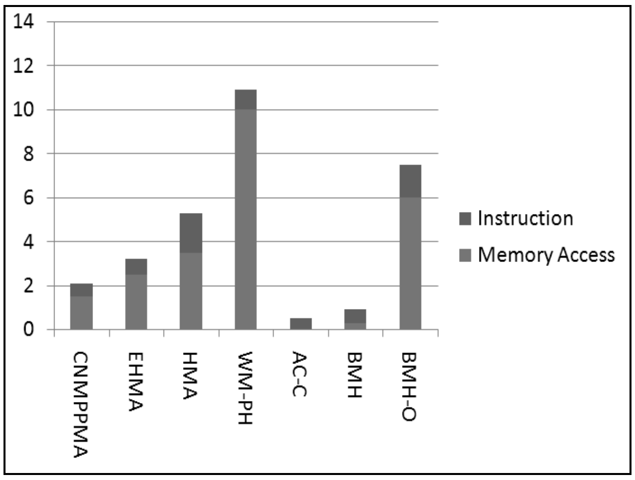


Fig. 7. The Comparison of Memory Access for each instruction with various algorithms

As shown in Fig. 8, CNMPPMA is highly effective in reducing the required external memory, providing efficient performance even in the virus-detection-like model. The results in Fig.8, indicate that CNMPPMA significantly outperforms others in both cases of small and large pattern set sizes even in the intense attack. CNMPPMA still performs better than others even when the penalty on the external memory access is reduced . Comparing CNMPPMA with HMA in Figs. 7, 8, reveals that the proposed safety shift strategy significantly reduces the number of external memory accesses and thus improves the matching performance.

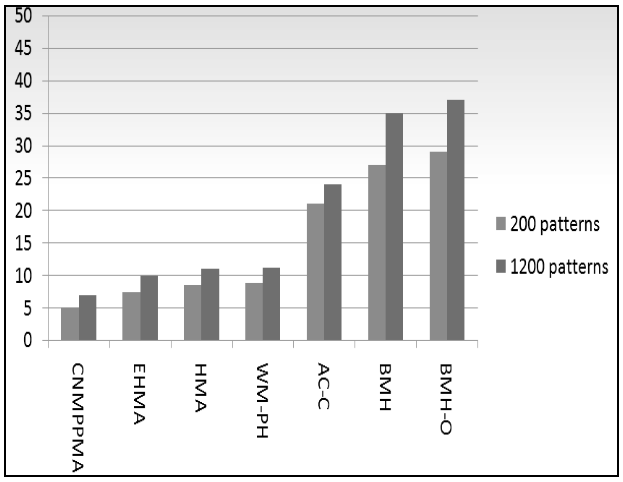


Fig. 8. The comparison of average number of external memory accesses (E)

5. CONCLUSION

The most important component of in-depth packet inspection is an efficient multipattern matching algorithm. This study proposes a CNMPPMA for packet inspection. CNMPPMA applies the common grams obtained by the proposed CGS to narrow the searching scope and to quickly filter out the innocent packets. The hierarchical matching significantly reduces the average number of external memory accesses to only 6 percent to 19 percent, thus improving the matching performance. Particularly, CNMPPMA is very simple and can be easily implemented in both software-based and hardware-based platforms. This study also discusses and evaluates current multipattern matching algorithms for NIDSs. CNMPPMA also works well for the systems with larger minimum pattern size, such as virus detection systems. In conclusion, CNMPPMA facilitates the creation of efficient and cost-effective pattern detection engines for packet inspection.

6. REFERENCES

[1] Snort, http://www.snort.org, 2008.

[2] Antonatos, K.G. Anagnostakis, and E.P. Markatos,"Generating Realistic Workloads for Network Intrusion Detection Systems,"Proc. Fourth Int'l ACM Workshop Software and Performance (WOSP),2004.

[3] R.N. Horspool, "Practical Fast Searching in Strings," Software Practice and Experience, vol. 10, no. 6, pp. 501-506, 1980.

[4] A.V. Aho and M.J. Corasick, “Efficient String Matching: An Aid to Bibliographic Search,” Comm. ACM, vol. 18, no. 6, pp. 330-340,June 1975.

[5] M. Fisk and G. Varghese, “Fast Content-Based Packet Handling for Intrusion Detection,” UCSD Technical Report CS2001-0670,May 2001.

[6] S. Lakshmanamurthy, K.-Y. Liu, Y. Pun, L. Huston, and U. Naik,“Network Processor Performance Analysis Methodology,” Intel Technology J., vol. 6, Aug. 2002.

[7] S. Wu and U. Manber, “A Fast Algorithm for Multi-Pattern Searching,” Technical Report TR94-17, Dept. Computer Science,Univ. of Arizona, May 1994.

[8] E. Markatos, S. Antonatos, M. Polychronakis, and K. Anagnostakis, “Exclusion-Based Signature Matching for Intrusion Detection,” Proc. IASTED Int’l Conf. Comm. and Computer Networks (CCN ’02), Oct. 2002.

[9] R.-T. Liu, N.-F. Huang, C.-H. Chen, and C.-N. Kao, “A Fast String Matching Algorithm for Network Processor-Based Intrusion Detection System,” ACM Trans. Embedded Computing Systems,vol. 3, no. 3, Aug. 2004.

[10] R.S. Boyer and J.S. Moor, “A Fast String Searching Algorithm,”Comm. ACM, vol. 20, no. 10, pp. 762-772, Oct. 1977.

[11] C.J. Coit, S. Staniford, and J. McAlerney, “Towards Faster String Matching for Intrusion Detection or Exceeding the Speed of Snort,” Proc. Second DARPA Information Survivability Conf. and Exposition (DISCEX), 2001.

[12] S. Antonatos, M. Polychronakis, P. Akritidis, K.G. Anagnostakis, and E.P. Markatos, “Piranha: Fast and Memory-Efficient Pattern Matching for Intrusion Detection,” Proc. 20th IFIP Int’l Information Security Conf. (SEC ’05), May 2005.

[13] S. Kim and Y. Kim, “A Fast Multiple String-Pattern Matching Algorithm,” Proc. 17th AoM/IAoM Int’l Conf. Computer Science,Aug. 1999.

[14] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J.Lockwood, “Deep Packet Inspection Using Parallel Bloom Filters,” Proc. 11th Symp. High Performance Interconnects, Aug.2003.

[15] Vitesse Network Processors, http://www.vitesse.com, 2008.

[16] Intel Network Processors, http://www.intel.com/design/network/products/npfamily/index.htm, 2008.

[17] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, “Stateful Intrusion Detection for High-Speed Networks,” Proc. IEEE Symp.Security and Privacy (SP ’02), May 2002.

[18] M. Handley, V. Paxson, and C. Kreibich, “Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics,” Proc. Ninth USENIX Security Symp., 2000.

7. AUTHORS PROFILE

^{1}**N. Kannaiya Raja** received degree MCA from Alagappa University and ME from Anna University Chennai in 2007 joined assistant professor in various engineering colleges in Tamil Nadu affiliated to Anna University and has eight year teaching experience. his research work in deep packet inspection. He has been session chair in major conference and workshops incomputer vision on algorithm, network, mobile communication, image processing papers and pattern recognition. His current primary areas of research are packet inspection and network. He is interested to conduct guest lecturer in various engineering in Tamil Nadu.

^{2}**Dr.K.Arulanandam** received PhD doctorate in 2010 from Vinayaka Missions University Salem. He has twelve years teaching experience in various engineering colleges in Tamil Nadu which are affiliated to Anna University and his research experience are network, mobile communication networks, image processing papers and algorithm papers. Currently working in Ganadipathy Tulasi’s Jain Engineering College Vellore.

^{3}**B.Raja Rajeswari** received degree B.E Computer Science and Engineering from Anna University Chennai in 2010. Now pursuing second year ME Computer Science and Engineering in Arulmigu Meenakshi Amman College of Engineering Kanchipuram affiliated to Anna University Chennai.