IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 5, May 2015.
www.ijiset.com

ISSN 2348 – 7968

Evaluation of different software based approaches for Deep Packet Inspection

Dr. Manish Shrivastava[1] Saurabh Singh[2]

[1] Department of Computer Sc. & Engg, Institute of Technology,
Guru Ghasidas University, Bilaspur, CG., India

[2]R&D Engineer Tally Solutions Pvt Ltd.

***Abstract*:**
Deep Packet Inspection (DPI) is a relatively new technology that inspects the content portion of packets flowing through the network in real time. It plays a vital role for providing safe and congestion free network. It finds if the incoming traffic's payload portion matches any signature present in the database or not. More specifically, DPI technology isn't just about the inspection of individual packet's payload because it often takes more than one packet to form meaningful content. In fact, it might require analyzing hundreds or thousands of packets before malicious code or activity passing through a DPI device can be detected. This ability to analyze a deep string of packets in real time and perform high level content analysis is where the "deep" in DPI comes from. As DPI consider s the whole packet for inspection it proves to be better than the other surveillance techniques. The most usual problem faced by DPI system is the low packet throughput, low accuracy and high memory requirement at high speed lines. This paper gives the thorough review of various software based methodologies currently being followed. It then attempts to present a fresh solution to the existing research gaps.

Keywords: *Deep packet inspection system, network intrusion detection, pattern matching algorithms, multilayer perceptrons neural network*

1.Introduction

The mainstream adoption of the Internet has brought along with it all the good and ills of the physical world. Because everything is globally connected over a common network, the Internet is a massive local community of a billion computers. That means every criminal on the planet has the potential to directly access any computer connected to the Internet. Malicious software such as worms and viruses, which were once the handiwork of curious mischief, are now tools for professional cyber criminals. Any computer connected to today's Internet has to be best prepared to defend itself against worms, viruses, spam, Denial of Service (DoS) attacks. But current practice of safety proves to be almost impossible against the most sophisticated attacks. Thus traffic monitoring and its accurate filtering prove to be the key task for network protection. It demands lot more than what Intrusion Detection Systems (IDS) provide. DPI In the current paper we review the different approaches for DPI which includes the various string matching algorithms, regular expression and then we attempt to provide a soft computing based approach which will provide high accuracy along with high throughput.

2.Approach To Packet Analysis

A. Shallow Packet Inspection

Shallow Packet Inspection (SPI) refers to the inspection of packet headers for optimization of packet routing, detection of network abuse, and statistical analysis. Such inspection does not disclose the contents of data packets. Unlike DPI, SPI makes broad generalities about traffic based solely on evaluating the packet header. Shallow packet inspection cannot provide the same refined/detailed traffic assessments as DPI. It has access to layer 3 and 4 of OSI stack. Generally 5 tuples [1] are observed in this technique viz. source and destination transport layer address, source and destination IP address and service being used. Based on this analysis decision over the packet is taken.

B. Medium Packet Inspection

Medium Packet Inspection (MPI) is typically used to refer to 'application proxies', or devices that sta nd between end-users' computers and ISP/Internet gateways. These proxies can examine packet header information against their loaded parse-list. Application proxies are typically placed in line with network routing equipment – all traffic that passes through the network must pass through the proxy device – to ensure that network administrators' rul e sets are uniformly applied to all data streaming through the network.

C. Deep Packet Inspection

Deep Packet Inspection techniques customarily focus on the content portion of the packet analyzing if the packet is safe enough to be allowed to flow through the network. Unlike the IDS which observe the content only below the layer 4 of IP packet, the DPI system analyzes the data up to the 7 layers of OSI model.

ISSN 2348 – 7968

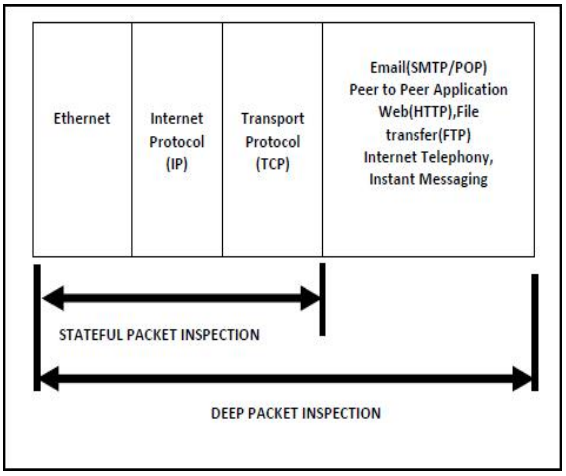

Figure 1: Comparison of SPI and DPI

3. Challenges Faced By DPI

There are few challenges faced by every DPI systems. As stated by AbuHmed et al.[2] the most frequent among them are:

1) *High complexity of search algorithms:* The complexity of the string matching algorithm is the very important aspect of DPI systems. As 40% to 70% [3] of the total execution time accounts to string matching so it has to be very efficient.
2) *Rapid increment in the intruder signature:* There has to be a specific signature corresponding to the different types of vulnerabilities present over the internet. Hence with the advancements in the types of attacks the number of signatures is rapidly growing. Snort [4] contains more than 17209 such rules as in 2010.
3) *False Alarms:* When deployed DPI systems may generate thousands of false each day. This needs extensive revision of the system. Snort [4] generates as much as 69% false alarm.
4) *Complete analysis of the payload:* Unlike IDS the signatures are not localized in any particular location but can be present anywhere in the whole packet. Hence it needs extensive analysis of the complete packet.
5) *Problem with encrypted data:* For DPI to inspect the encrypted data, it needs to be placed behind some decryption units.
6) *Bottleneck issue:* The present communication systems are functional at 10GbE/OC192 speed and its getting better. Hence DPI systems need to work at those speeds or else it would end up as a bottleneck in the system.

4. Existing Software Based Approaches For DPI

In order to implement DPI certain techniques such as exact string matching algorithms, regular expressions presenting the pattern of characters and symbols and a Deterministic Finite Automata (DFA) of included signatures. The DFA and Nondeterministic Finite automata (NFA) represent the signatures corresponding to the application it has been implemented for. Lastly the exact string matching algorithms are brought in use to match the signatures. There are numerous such algorithms but significant changes are required in them to be implemented in DPI. The relation between the components is shown in the figure 2.

A. Exact String Matching Algorithms

DPI uses exact string matching algorithms to avoid false alarms. Exact string searching technique means finding all the occurrence of a particular pattern in the text. Aho-Corasick algorithm[5] is one of the oldest such algorithm. It uses the pattern of length m may be denoted by [0..m-1]. The text is of length n is denoted by [0..n-1]. Both strings consist of finite set of alphabets represented by Σ with size σ.
The approach of this algorithm is very basic as it begins from the starting state of any Finite State Machine and reaches its final state if pattern if found. Else in case of failure the state is transferred back to the starting state for further matching.

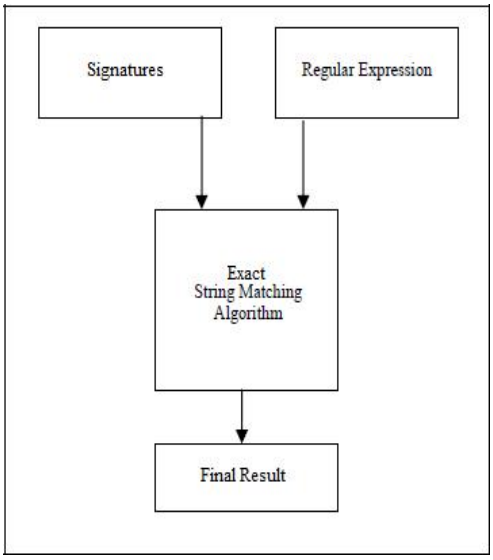

Figure 2: DPI Implementation Diagram

There are several other pattern matching algorithms too proposed for IDS systems. However implementation in DPI systems is more challenging as compared to normal IDS systems. Performance evaluation of such algorithms referenced from[6] have been presented in the table 1[6,7].

ISSN 2348 – 7968

B. Regular Expression

One of the limitations faced by the exact string matching algorithms is that sometimes the intruder may intentionally perform character stuffing in the actual packet content. Thus the new packet may contain dot-star notation or some wildcard characters. Also there may exist some counting constraints as bounded repetition of sub patterns or character set. Hence it is must for signatures to be represented in form of regular expressions rather than actual strings. The main software based approaches that have been presented of regular expression matching is the String Tree. For the passing data stream the String Tree makes parallel signature matching in each clock cycle. It is a brute-force method and many character comparators are required. Hence because of its large size it becomes infeasible to implement the String Tree. Also its computational time is much more.

5.Issues With The Current Systems

The major problems faced by these implementation techniques are that none of them is solely efficient enough in terms of space, time and the accuracy. Some have high cache memory requirement, some have execution time beyond permissible limits while others produce false alarms and show poor accuracy.

6.Proposed System Using Neural Network

Due to some of its intrinsic properties such as adaptability and self learning capacity soft computing is gaining popularity day by day. It has strength of processing the data containing huge amount of noise. The Artificial Neural Network (ANN) is very useful for pattern matching applications. Pattern matching consists of the ability to identify the class of input signals or patterns. Pattern matching ANN are typically trained using supervised learning techniques. One application where artificial neural nets have been applied extensively is optical character recognition (OCR). OCR has been a very successful area of research involving artificial neural networks.

As implemented by Deng et al.[8] the multilayer perceptrons neural network find a very vital application in intrusion detection model. Here the data stored in a special form of data structure is used by the pattern matching module to filter out some of the known intrusions.

Evolving from the IDS the concept multilayer perceptrons neural network can be implemented in DPI systems with some novel changes. To process a high speed data the neural network needs to perform a position independent pattern matching. The ability of such a network to perform a complete position independent pattern matching have already been suggested by Hirai et al.[9] Each signature must be present in form of multiple stored templates in parallel. The captured packet is matched with the multiple candidates of the stored signature. The best matching template having its features arranged in the same order as those of the input data is found. Here it avoids considering the positional difference between the corresponding features. The proposed multilayer network contains a pattern-matching layer, a minimum distance recognition layer, and an identification layer.

Initially in the foremost pattern-matching layer pattern matching is done between the captured packet data and the candidate templates of signatures. The candidate templates are chosen in the minimum distance recognition layer, and are sent back to the pattern-matching layer in parallel. The candidate template having the features in the same positional order as that of present in the input data is identified in the pattern-matching layer. The Identification layer presents the matching signature.

This position independent pattern matching technique of multilayer neural network may even be implemented with the current DPI systems. Its working has been shown in the flowchart in figure 3.

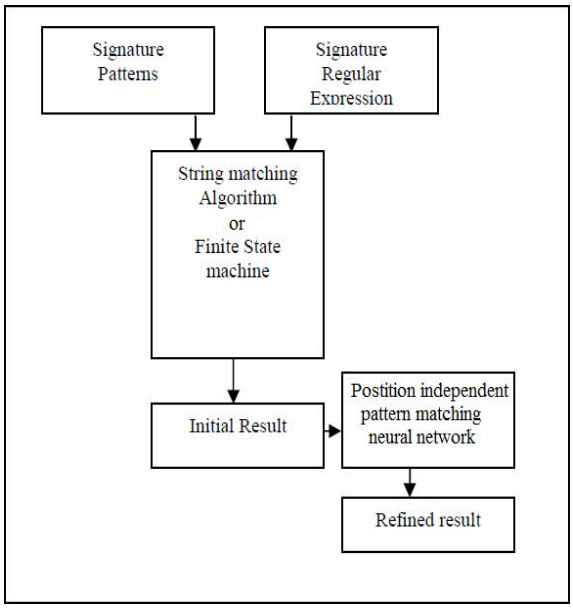

Figure 3: Flowchart of current DPI systems implemented with multilayer perceptrons neural network

7. Conclusions

Through this paper various software based approaches for Deep Packet Inspection systems have been evaluated. We have shown that it requires an efficient string matching algorithms. Through the comparative study of various proposals we showed that we still need a very fast algorithm to operate on 10GbE/OC192 line which is even approaching a better speed. We presented that how the exact string matching algorithms are costly in terms of time complexity. The FSMs are poor in terms of space complexity and require high cache storage. These systems need to curve down the false alarm rates which are treated well by the artificial neural networks. It has high capacity of processing the data having noise. Hence, the multilayer perceptrons neural network can perform position independent matching to solve the purpose. The three layer segment model stated above will be very powerful and will provide high throughput along with accuracy at very high data transfer speeds.

ISSN 2348 – 7968

Table 1: Performance Evaluation of Various Exact String Matching Algorithms

Algorithm Name	Implementation Method	Complexity		
		Processing		Searching
		Space	Time	Time
Aho-Corasick Algorithm	This algorithm locates elements of a finite set of string within an input text hence also termed as dictionary-matching algorithm. The complexity of the algorithm is combination of length of patterns plus the length of the searched text plus the number of output matches.[2]	A:O(m+z)	A:O(m+z)	A:O(n)
Wu-Manber Algorithm	Utilize the concept of skipping characters and also find the candidate for matching. It uses Hash Based Searching. A variant of Boyer-Moore Algorithm.	A:O(m+z)	A:O(m+z)	A:O(Bn/m)
Boyer-Moore algorithm	This algorithm process the pattern from right to left and if a mismatch occurred it uses two precomputed functions to shift the window to the right.[10]	A:O(m+σ)	A:O(m+σ)	A:O(mn)
Turbo-Boyer Moore algorithm	This is confined version of Boyer-Moore algorithm. It remembers text patterns for matching. Only constant extra space is needed as compare to the original one without any extra preprocessing.[11]	A:O(m+σ)	A:O(m+σ)	A:O(n)
Apostolico-Giancarlo algorithm	This is an extension of algorithm boyer_moore. SKIP table is used for storing the information about the longest suffix of pattern ending at the right position of the window for that matching attempts.[12]	A:O($\overline{\quad}$ m^2)	A:O(mσ)	A:O(n)
Horspool algorithm	It uses the bad-character shifts of the rightmost character of the window. It is less complicated version of well known Boyer-Moore.[13]	A:O(m+σ)	A:O(σ)	A:O(mn)
Quick Search algorithm	Very efficient algorithm for short pattern and large text to be searched. It's the Easiest implementation of BM algorithm and works only on bad-character shift table.[14]	A:O(m+σ)	A:O(σ)	A:O(mn)
Optimal Mismatch algorithm	Efficient to implement when patterns are searched from least frequent to most frequent one. This is an alternative to Quick Search algorithm.[14]	A:O($\overline{\quad}$ +σ) m^2	A:O(m+σ)	A:O(mn)
Maximal Shift algorithm	Characters are scanned from the larger shift first before the shorter shift which help to maximize the length of each shift. This is another implementation of optimal mismatch.[14]	A:O($\overline{m^2}$+σ)	A:O(m+σ)	A:O(mn)
Zhu-Takaoka algorithm	Scans and computes two consecutive text characters in order for bad-character shift. Version of Boyer-Moore algorithm.[15]	A:O(m+σ^2)	A:O(m+σ^2)	A:O(mn)

ISSN 2348 – 7968

Berry-Ravindran algorithm	It shifts two consecutive text characters in the given pattern from the right side of pattern window.[16]	A:O(m+) σ^2	A:O(m+) σ^2	A:O(mn)
Smith algorithm	Main theme of this algorithm is based on the concept which uses maximum of the Horspool bad-character shift function and the Quick Search bad-character shift function.[17]	A:O(σ)	A:O(m+σ)	A:O(mn)
Raita algorithm	Initially it checks for the equality of character patterns in order of last, then the first and finally the middle then move on the other pattern. It's similar to that of Horspool algorithm[18]	A:O(σ)	A:O(m+σ)	A:O(mn)
Brute force algorithm	Checks for the pattern at every position in text and shifts pattern by one position to the right after each attempt till an exact match is found or it reaches end of pattern. Expected number of text character comparison is 2n.	Not required	Not required	A:O(mn)(W,B,A)
Automaton based algorithm	It require to build a minimal deterministic automaton which recognizes the language Σ*x;[19]	A:O(mσ)	A:O(mσ)	A:O(n) W:O(nlog(σ))
Karp-Rabin algorithm	Hashing technique is used for preprocessing in order to reduce quadratic time complexity.[20]	Constant space	A:O(m)	A:O(mn)
Shift Or algorithm	Used efficiently when pattern size is less than the memory-word size of the machine. Based on bitwise technique and good for approx string matching.[21]	A:O(m+σ)	A:O(m+σ)	A:O(n)
Morris-Pratt algorithm	Uses technique like Brute force algorithm from left to right, and keep track of information gathered during the scan of the text.[22]	A:O(m)	A:O(m)	A:O(n+m)
Knuth-Morris-Pratt algorithm	Uses tight analysis of Morris-Pratt for comparing from left to right. It's the improvised version of Morris-Pratt algorithm.[23]	A:O(m)	A:O(m)	A:O(n+m)
Simon algorithm	Assist Economical implementation of A(x) the minimal DFA recognizing Σ*x.[34] by reducing the size of Automata.[24]	A:O(m)	A:O(m)	A:O(m+n)
Colussi algorithm	Set is partitioned in two disjoint subsets; first is scanned from left to right and it checks for a match. If a match is found, second subset is scanned from right to left.[25]	A:O(m)	A:O(m)	A:O(m)
Galil-Giancarlo algorithm	This Algorithm is simplified and improved version of Colussi algorithm[26]	A:O(m)	A:O(m)	A:O(m)
Apostolico-Crochemore algorithm	Kmp Next shift table is used to compute the shifts and for remembering the most recent shift activities.[27]	A:O(m)	A:O(m)	A:O(n)
Not So Naïve algorithm	The character comparisons are made with the pattern in the following order 1,2,……m-1,0.[28]	A:O(m)	A:O(m)	A:O(nm)

ISSN 2348 – 7968

Turbo RF algorithm	Efficient and refined version of Reverse Factor algorithm.[29]	A:O(m)	A:O(m)	A:O(n)
Forward Dawg Matching algorithm	The smallest suffix automaton is used to compute the longest factor of the pattern ending at each position in the text.[30]	NA	NA	W:O(n)
Backward ND Matching algorithm	Efficient to implement if length of pattern is less than the memory size of the machine. This is just another variant of the Reverse Factor algorithm.[31]	W:O(nm) A:O(n/m)	NA	W:O(n)
Backward Oracle Matching	Basically used in the Reverse factor algorithm. Efficient in long patterns and small alphabets sets.[32]	A:O(m)	A:O(m)	A:O(nm)
Galil-Seiferas	Decomposition technique is used for preprocessing. Implements the concept of constant extra space complexity.[33]	Constant space	A:O(m)	A:O(n)
Skip Search algorithm	There exist a bucket of positions for each character of the alphabet. for each character of alphabet, bucket collects all positions of that character in pattern x.[34]	A:O(m+σ)	A:O(m+σ)	A:O(nm)
KMP Skip Search	It's a linear skip search algorithm which uses two shift tables of Morris-Pratt and KMP. This is refined version of Skip Search algorithm.[34]	A:O(m+σ)	A:O(m+σ)	A:O(n)
Alpha Skip Search algorithm	This algorithm uses buckets of positions for each factor of length logσ(m) of the each pattern.	A:O(m)	A:O(m)	A:O(nm)

W: Worst case *A: Average Case* *B: Best case*

m: Length of pattern *n:length of text* *σ:Size of alphabet(a finite set of character)*

z: z is the number of pattern occurrences *NA: No description available in literature*

ISSN 2348 – 7968

References

[1] Esoft, "Modern Network Security: The Migration to Deep Packet Inspection," 2005.

[2] T. AbuHmed, A. Mohaisen, and D. H. Nyang, "Deep Packet Inspection for Intrusion Detection Systems: A Survey," Technical Report,Security Research Laboratory, Inha University, Information Incheon 402-751, Korea 2007.

[3] A. Spyros, G. A. Kostas, and P. M. Evangelos, "Generating realistic workloads for network intrusion detection systems," in Proceedings of the 4th international workshop on Software and performance Redwood Shores, California: ACM, 2004, pp. 207-251.

[4]- SNORT, "Network intrusion detection system," Available: www.snort.org.

[5] A. V. Aho and M. J. Corasick, "Efficient string matching: an aid to bibliographic search," Communications of the ACM, vol. 18, p. 340, 1975.

[6] C. Charras and T. Lecroq, "EXACT STRING MATCHING ALGORITHMS." vol. sept.,29,2010: Laboratoire d'Informatique de Rouen,Université de Rouen,Faculté des Sciences et des Techniques, pp. [Online]. Available: http://www-igm.univ-mlv.fr/~lecroq/string/

[7] A. N. M. Rafiq, M. W. El-Kharashi, and F. Gebali, "A fast string search algorithm for deep packet classification," Computer Communications, vol. 27, pp. 1524-1538, 2004.

[8] Deng, Quan-cai, "A New Intrusion Detection System Based on Multilayer Perceptrons Neural Network" E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference.

[9] Hirai, Yuzo, "Position independent pattern matching by neural network" Systems, Man and Cybernetics, IEEE Transactions on Jul/Aug 1990.

[10]- R. S. Boyer and J. S. Moore, "A fast string searching algorithm," Commun. ACM, vol. 20, pp. 762-772, 1977.

[11]- M. Crochemore, A. Czumaj, L. Gasieniec, S. Jarominek, T. Lecroq, W. Plandowski, and W. Rytter, "Deux méthodes pour accélérer l'algorithme de Boyer-Moore," Théorie des Automates et Applications, Actes des 2e Journées Franco-Belges, D. Krob ed., Rouen, France, pp. 45-63, 1992.

[12]- A. Apostolico and R. Giancarlo, "The Boyer-Moore-Galil string searching strategies revisited," SIAM J. Comput., vol. 15, pp. 98-105, 1986.

[13]- R. N. Horspool, "Practical fast searching in strings," Software: Practice and Experience, vol. 10, pp. 501-506, 1980.

[14] D. M. Sunday, "A very fast substring search algorithm," Communications of the ACM, vol. 33, pp. 132-142, 1990.

[15] R. F. Zhu and T. Takaoka, "On improving the average case of the Boyer-Moore string matching algorithm," Journal of Information Processing, vol. 10, pp. 173-177, 1987.

[16] T. Berry and S. Ravindran, "A fast string matching

[17] algorithm and experimental results," in Proceedings of the Prague Stringology Club Workshop`99, J. Holub and M. Simánek ed., Collaborative Report DC-99-05, Czech Technical University, Prague, Czech Republic, 1999, pp. 16-28.

[18] P. D. Smith, "Experiments with a very fast substring search algorithm," Software: Practice and Experience, vol. 21, pp. 1065-1074, 1991.

[19] T. Raita, "Tuning the Boyer-Moore-Horspool string searching algorithm," Software: Practice and Experience, vol. 22, pp. 879-884, 1992.

[20] G. Rozenberg and A. Salomaa, Handbook of Formal Languages: Beyond Words: Springer Verlag, 1997.

[21] R. M. Karp and M. O. Rabin, "Efficient randomized patternmatching algorithms," IBM Journal of Research and Development, vol. 31, pp. 249-260, 1987.

[22]- R. Baeza-Yates and G. H. Gonnet, "A new approach to text searching," Communications of the ACM, vol. 35, p. 82, 1992.

[23] J. H. Morris and V. R. Pratt, "A linear pattern-matching algorithm," Technical Report 40, University of California, Berkeley, 1970

[24] D. E. Knuth, J. H. Morris Jr, and V. R. Pratt, "Fast pattern matching in strings," SIAM Journal on Computing, vol. 6, p. 323, 1977.

[24] I. Simon, "String matching algorithms and automata," in R. Baeza Yates, N. Ziviani (Eds.),First American Workshop on String Processing, Universidale Federal de Minas Gerais, Brazil, 1993, pp. 151–157.

[25] L. Colussi, "Correctness and efficiency of pattern matching algorithms," Information and Computation, vol. 95, pp. 225- 251, 1991.

[26] Z. Galil and R. Giancarlo, "On the exact complexity of stringvmatching: upper bounds," SIAM Journal on Computing, vol.21, p. 407, 1992.

[27] A. Apostolico and M. Crochemore, "Optimal canonization of all substrings of a string," Information and Computation, vol. 95, pp. 76-95, 1991.

[28] C. Hancart, "Une analyse en moyenne de l'algorithme de Morris et Pratt et de ses raffinements," Actes des 2e Journées franco-belges: Théories des Automates et Applications, vol. 176, pp. 99–110.

[29] T. Lecroq, "Experimental results on string matching algorithms," Software: Practice and Experience, vol. 25, pp. 727-765, 1995.

[30] M. Crochemore and W. Rytter, Text algorithms: Oxford University Press, Inc. New York, NY, USA, 1994.

[31] G. Navarro and M. Raffinot, "A bit-parallel approach to suffix automata: Fast extended string matching," in Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching, Lecture Notes in Computer Science 1448, Springer-Verlag, Berlin,, 1998, pp. 14-33.

[32] C. Allauzen, M. Crochemore, and M. Raffinot, "Factor oracle: a new structure for pattern matching," in

ISSN 2348 – 7968

Proceedings of SOFSEM'99, Theory and Practice of Informatics, Lecture Notes in Computer Science 1725,, J. Pavelka, G. Tel and M. Bartosek ed., Milovy,Czech Republic,Springer-Verlag, Berlin1999, pp. 291-306.

[33] Z. Galil and J. Seiferas, "Time-space-optimal string matching* 1," Journal of Computer and System Sciences, vol. 26, pp. 280-294, 1983.

[34] C. Charras, T. Lecrog, and J. Pehoushek, "A very fast string matching algorithm for small alphabets and long patterns," in Proceedings of the 9th Annual Symposium on Combinatorial Pattern Matching , M. Farach-Colton ed.,Lecture Notes in Computer Science 1448, Piscataway, New Jersey, 1998, pp. 55-64.