ISSN:0975-9646

A Review on Deterministic Finite Automata Compression Techniques for Efficient Pattern Matching Process

Ms. Utkarsha P. Pisolkar[1], Prof. Shivaji R. Lahane[2]

[1]*Student, Computer Engg. Department, GES's R. H. Sapat COE, Nashik*
Pune University, India
[2]*Professor, Computer Engg. Department, GES's R. H. Sapat COE, Nashik*
Pune University, India

Abstract- **Pattern matching is a very significant method in many network security applications, such as intrusion detection, deep packet inspection, IP lookups etc. Deterministic Finite Automata (DFA) and Non deterministic finite automata (NFA) are widely used in pattern matching process to represent the patterns. To match with fast network speed, need of such security applications is a memory efficient and speedy pattern matching process. There are many techniques present which make the pattern matching process fast and memory efficient. Deterministic finite automata compression is one of them. Deterministic finite automata compression can be grouped into state reduction, transition reduction, character set reduction and bit reduction. Prior works on all these categories are discussed in this paper. At the end, the bit reduction technique is discussed. It reduces the number of bits required to represent each state and works on Aho-Corasick DFAs.**

Keywords-**Bit reduction, Deterministic finite automata, Intrusion detection, Network security, Pattern matching.**

I. INTRODUCTION

Today, a computer network has become a very important part of our daily life. Internet has a fast growth from the last decade with increasing dependence of society on it. Internet provides a wide range of advantages to society but it is infected by many security attacks which disrupts the functionality of networking and computing infrastructure. To ensure the protection of network or system from attacks, many security applications uses variety of safety measures. For a system, user can use security software (such as antivirus software). For a network, security mechanisms are made in such way that they directly respond to attacks, Network Intrusion detection System is widely used for this purpose. Basically Intrusion Detection System (IDS) examines all the packets and detects the attack. Like viruses, most attacks have some sort of signatures therefore these signature patterns are compared to detect any attack activity. Pattern matching has become the important module of all network security applications to compare these signature patterns.

As networking speeds doubling every year, the pattern matching algorithm must be able to operate at high speed to compare thousand of string patterns. Therefore, today, the most challenging tasks for security applications are to improve inspecting speed and also reduce their memory requirement. So, it requires well designed algorithms and techniques to speed up pattern matching process.

Pattern matching involves inspection of a given sequence of tokens for the presence of components of some pattern. For a given strings T (text) and P (pattern), the pattern matching problem consist of finding a substring of T equal to P. More complex patterns are described by regular expressions. Regular expressions are grammars that define the regular language. A regular expression forms a search pattern. This search pattern is used in pattern matching process. A finite automaton is a machine which recognizes a regular language. It is further divided into Deterministic Finite Automata (DFA) and Non-deterministic Finite Automata (NFA). Both DFA and NFA are commonly used in pattern matching process. SNORT is a popular and commonly used open source IDS [2]. It has thousands of rules in which the rules refer to the header as well as to the packet payload. SNORT is often used as first step in many pattern matching algorithms to identify regular expressions [2].

Rest of the paper is organized as follows. Section one provides introduction, survey of literature of some of the existing compression techniques and various algorithms used till today are discussed in section two. Section three highlights the DFA compression technique using bit reduction method in detail, and finally section four presents a conclusion.

II. LITERATURE SURVEY

There are standard string matching algorithms are present such as Aho-Corasick [1] and Wu-Manber [10]. Alfred V. Aho and Margaret J. Corasick have proposed efficient string matching algorithm [1]. In that, a finite state pattern matching machine has constructed from the keywords and it is used to process the text string in a single pass. S. Wu, U. Manber have introduced an algorithm to search for multiple patterns at the same time [10]. This multi-pattern matching algorithm could be used instead indexed or sorted data in some applications involving small to medium size datasets. A large body of research literature has concentrated on improving these algorithms to make use of them in networking area.

To make a pattern matching process fast and memory efficient, many DFA compression techniques are carried out. They all are focused on improving algorithm's performance and speed, reducing memory storage requirements. And can be done in different ways such as reduce number of transitions, reduce number of states, reduce character sets and reduce bits encoding the transitions.

A. Transition Reduction

In transition reduction technique, the numbers of transitions are reduced in each state. Under this category, techniques were introduced like D^2FA [9] and CD^2FA [7].

S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner have introduced the Delayed Input DFA (D^2FA), which reduced space requirements by reducing the number of distinct transitions between states [9]. Default transitions were used to reduce memory storage requirement. Since many states can have similar sets of outgoing transitions, redundant transitions were replaced by a single default transition in D^2FA. For two states s_1 and s_2 which have transitions to the similar set of states, {S}, for some set of input characters, {C}. These transitions were removed from one state s_1 and launched a default transition from s_1 to s_2 which followed for all the characters in {C}. So, s_1 maintained distinctive next states for those transitions which are not similar to s_1 and s_2 and used the default transition to s_2 for the similar transitions. A D^2FA is constructed by transforming a DFA via incrementally replacing number of transitions of the automata with a single default transition. D^2FA had provided a trade-off between the memory requirements of the compressed DFA and the number of states visited for each character processed. But the negative aspect of this approach is the traversal of several states when processing a single input character, which needs memory bandwidth increase to evaluate regular expressions.

S. Kumar, J. Turner, and J. Williams have illustrated a transition reduction technique to boost the speed of D^2FA as by storing more information on the transitions edges [7]. Content Addressed Delayed Input DFA (CD^2FA) has introduced and built upon the delayed input DFA (D^2FA), whose state numbers are replaced by content label. These content labels densely contain information that would be stored in the table entry for the state and by using this information consecutive states of a D^2FA are addressed by CD^2FA, instead of "content-less" identifier. Also the content label is used to omit past default transitions. Therefore, selected information is available earlier in the state traversal process, which is enough for the CD^2FA to avoid any default traversal, thus avoiding unnecessary memory accesses and hence achieved higher throughput for pattern matching.

B. State Reduction

In state reduction technique, the numbers of DFA states are reduced. Hybrid DFA-NFA, HFA, XFA are based on state reduction technique.

M. Becchi and P. Crowley [4] have introduced hybrid DFA-NFA state reduction solution based on the study that DFAs are infeasible with large sets of regular expression and NFAs lighten the memory storage problem but lead to a potentially large memory bandwidth requirement. The reason is that multiple NFA states can be active in parallel and each input character can activate multiple transitions. A hybrid DFA-NFA solution is proposed which brought together the strengths of both automata: when constructing the automaton, any nodes that would contribute to state explosion keep an NFA encoding, while the others are transformed into DFA nodes.

S. Kumar, B. Chandrasekaran, J. Turner, and G. Varghese [6] have studied three main limitations of the traditional DFAs based NIDS. First, DFAs do not take advantage of the fact that normal data streams rarely match more than few initial symbols of any signature. Second, the DFAs are extremely inefficient in following multiple partially matching signatures and explode in size. Third, DFAs are incapable of keeping track of the occurrences of certain sub-expressions. And a History-based Finite Automaton (H-FA) is proposed to solve each of these drawbacks by adding some information to keep the transition history and, as a result, reduced the number of states. It is also a state reduction technique. More information is remembered by proposed machine, such as encountering a closure, by storing them in a small and fast cache which signifies a sort of history buffer. So, it has number of conditional transitions per character, which resulted in a large transition table and a slow inspection speed.

Another state reduction technique is extended character set (XFA) [5]. A XFA uses number of automata transformations to eliminate conditional transitions. XFA is limited to one auxiliary state per regular expression, which is inappropriate for difficult regular expressions.

D.Ficara, S.Giordano, G. Procissi, F.Vitucci, G.Antichi, A.DiPietro have proposed a compressed representation for deterministic finite automata, called Delta Finite Automata [3]. The algorithm reduced the number of states and transitions and it is based on the study that most adjacent states share several common transitions, so it stored only the differences between them. Primary feature of the delta finite automata is that it required only a state transition per character, thus allowed a fast string matching. A new state encoding scheme called Char-State compression based on input characters is proposed which exploited the association of many states with a few input characters. This compression scheme has integrated into the delta finite automata algorithm, which resulted in a further memory reduction.

C. Character Set Reduction

In character set reduction technique, character set is getting reduced. S. Kong, R. Smith, and C. Estan have mapped the set of characters in an alphabet to a smaller set of clustered characters that label the same transitions for a substantial amount of states in the automaton [8]. Several alphabet compression tables have used as a lightweight method for reducing the memory requirements of DFAs. This technique has used heuristics approach to partition the states of a DFA and computing a distinct alphabet compression table for each partition [8]. An Alphabet Compression Table can be used to map groups of symbols to a single symbol that is retrieved by a table lookup [8].

III. DFA Compression using bit reduction technique

CompactDFA technique is used to compress the number of bits required to represent each state[2]. It uses DFA compression algorithms that work on a large class of Aho-Corasick-like DFAs. This algorithm reduces the rule set to only one rule per state [2]. Thus, all transitions to a specific state are encoded by a single prefix [2]. An algorithm encodes and stores the DFA in memory. While examining the input, this DFA is access by an algorithm. Encoding scheme stores the set of rules with current state field, symbol field and next state field [2]. A DFA compression algorithm includes three stages: state grouping, common suffix tree construction, state and rule encoding [2].

A. *State Grouping*

In the first stage, making a group of the states is based on common suffix (CS) and longest common suffix (LCS) parameters. These two parameters are calculated for each state in DFA [2]. If a state has more than one incoming transitions then common suffix, denoted as CS(s), is a label of the state without its last symbol. Otherwise, no common suffix is present for a state. Label of a state s, denoted as label(s), is concatenation of the edge symbols of shortest path from initial state to state s [2]. The longest common suffix for state s, denoted as LCS(s), is long length common suffix of a state to which state s has an outgoing edge.

B. *Common Suffix Tree Construction*

In the second stage, common suffix tree is created and suffix rules are map to prefix rules. The nodes in common suffix tree are different LCS values called as set L. For every two values l_1, l_2 in set L, l_1 is an ancestor of l_2 if and only if l_1 is a suffix of l_2 [2]. Connecting nodes are added for every internal node such that the total number of its children is a power of two [2].At the end, states are linked to one of the connecting nodes.

C. *State and Rule Encoding*

In the third stage, there is a procedure of encoding of common suffix tree, states and rules. First code width is calculated which is a number of bits required to encode the common suffix tree. Then edges and nodes are encoded. At the end, using corresponding node in common suffix tree every state is encoded.

Common suffix tree is truncated at predefined depth to minimize total memory requirement. It is a simple variation over common suffix tree such that nodes that appear in the common suffix tree more than predefined depth are connected to their ancestor at predefined depth if they are state nodes [2]. Otherwise nodes are omitted. Speed of the CompactDFA is increase by performing lookup on input sequence of characters of predefine size [2]. A set of compressed rules is the output of CompactDFA technique [2]. When a state matches more than one rule then a rule with longest prefix determines the action [2].

IV. Conclusion

The security of network and computer system is very important. Pattern matching is a basic method used in security applications. It is widely used in Intrusion detection system. It compares signature patterns to detect attacks. It is very important to increase speed and to reduce the memory space requirement of pattern matching method. DFA can be compressed by techniques like transition reduction, state reduction, character set reduction and bit reduction. Transition reduction technique reduces number of transitions of every state. State reduction technique decreases number of states. In character set reduction, alphabet set of patterns are minimize to smaller size. Bit reduction technique decreases number of bits required to represent every state. These techniques can be applied as alone or in mixture of two or more. At the end, Bit reduction technique which works on Aho-Corasick DFA is discussed in this paper. It has three stages: state grouping, common suffix tree construction, state and rule encoding. The compressed rule set of this technique helps in decreasing number of bits required for a state.

Acknowledgment

We are glad to express our sentiments of gratitude to all who rendered their valuable guidance to us. We would like to express our appreciation and thanks to Prof. Dr. P. C. Kulkarni, Principal, G. E. S. R. H. Sapat College of Engg., Nashik. We are also thankful to Prof. N. V. Alone, Head of Department, Computer Engg., G. E. S. R. H. Sapat College of Engg., Nashik. We thank the anonymous reviewers for their comments.

References

[1] A.V. Aho and M.J. Corasick. *"Efficient String Matching: An Aid to Bibliographic Search."* Communications of the ACM, 18(6):333–340, 1975.

[2] AnatBremler-Barr, D.Hay, and Y. Koral, "*CompactDFA:Scalable pattern matching Using Longest Prefix Match Solutions*," in IEEE/ACM Transaction on networking,vol-22,No.2,April 2014.

[3] D.Ficara, S.Giordano, G. Procissi, F.Vitucci, G.Antichi, A.D. Pietro, "*An Improved DFA for Fast Regular Expression Matching*" ACM SIGCOMM Computer Communication Review, Volume 38, Number 5, October 2008.

[4] M. Becchi, P. Crowley, "*A hybrid finite automaton for practical deep packet inspection*", Proc. Of CoNEXT'07, pages 1-12. ACM, 2007.

[5] R. Smith, C. Estan, and S. Jha, "*Xfa: Faster signature matching with extended automata*", in IEEE Symposium on Security and Privacy, May 2008.

[6] S. Kumar, B. Chandrasekaran, J. Turner, G. Varghese, "*Curing regular expressions matching algorithms from insomnia, amnesia, and acalculia",* in Proc. of ANCS '07, pages 155-164. ACM, 2007.

[7] S. Kumar, J. Turner, J. Williams, "*Advanced algorithms for fast and scalable deep packet inspection*", in Proc. of ANCS '06, pages 81-92. ACM, 2006.

[8] S. Kong, R. Smith, and C. Estan, "*Efficient signature matching with multiple alphabet compression tables,*" in Proc. Int. Conf. Security Privacy Commun. Netw. (Securecomm), 2008.

[9] S. Kumar, S. Dharmapurikar, F. Yu, P. Crowley, and J. Turner, "*Algorithms to accelerate multiple regular expressions matching for deep packet inspection*", in Proc. of SIGCOMM '06, pages 339-350. ACM, 2006.

[10] S. Wu , U. Manber, "*A fast algorithm for multi-pattern searching*" Technical Report TR-94-17, Dept.of Computer Science, University of Arizona,1994.