2009 International Conference on Machine Learning and Computing
IPCSIT vol.3 (2011) © (2011) IACSIT Press, Singapore

Storing Scheme for State Machine Based Rule Base of Genetic Feedback Algorithm Based Network Security Policy Framework Depending on Memory Consumption

Atish Mishra, Prakash Kumar

Jaypee Institute of Information Technology University, Noida

Abstract. The work done hitherto suggests that the rule base of genetic feedback algorithm based network security framework can be represented using Finite State Machines. In this paper it is discussed how just by a simple modification in the storing scheme can reduce the worst case time complexity to be as low as 4 comparisons i.e. constant, for a FSM based rule base model. In this paper three types of storing schemes are discussed namely are, "Time Efficient Storing" which gives constant search time, second is "Trade Off Storing" in this scheme how a trade off can be done between space and time complexities to get a system according to availability of resources is discussed, time complexity for this scheme will be in the range of 2n , where n=2,3,…,10, and value of n is indirectly proportional to the space used and the third scheme is "Space Efficient Storing", this scheme will give worst case complexity of 1024, but the space utilization is maximum for this scheme. In this paper a brief overview of arrays, linked list and arrays of linked list is given, and the various storing schemes are discussed in detail, with comparisons between them.

Keywords: Finite state machine, genetic feedback algorithm based network security policy framework, rule base, array, linked list, array of linked list.

1. Introduction

The work done on rule base of genetic feedback algorithm based network security policy framework [3] only handled its basic structure which was based on Finite State Machine [1]. It was only discussed how by only using simple structure the time complexities can be reduced to very good extent. This paper discusses the storing scheme of rules in rule base. By simply changing the storing scheme the time complexity can be further reduced to the extent of a constant time and a trade off between the space and time can be done to get a system according to resources available. In this paper three types of storing schemes are discussed which can be classified as "Time efficient Storing Scheme", second "Trade off Storing Scheme" and third "Space efficient Storing Scheme". Rest of the paper is divided as follows: Section 2 gives a brief overview of array, linked list and array of linked list, Sections3 discusses in detail various storing schemes and there structure, in section4 comparison between the storing scheme is done, and section5 gives conclusion and future work.

2. Arrays, Linked List and Array of Linked List

Arrays and linked list are most basic data structures they can also be called as basic building block of any application dependent of data structures. The major difference between an array and linked list is the time complexity, searching, storing, retrieving and memory required. Array provides good time complexity because an element can be accessed directly if its index number is known, by using a simple formula (base address + index*size of single element) but it lacks in space complexity as it require continuous memory allocations, where as linked list is more space efficient as they can be stored in noncontiguous memory location, but lacks a bit in time complexity as they need to be searched almost linearly so these two can be combined to get desired results. They can be combined in quite a few ways i.e. when arrays are stored in the form of linked list, they can be called as linked list of array and when linked list are stored in array, this combination is called array of linked list.

3. Storing Scheme

In this section various storing scheme is discussed. How by just modifying the scheme a little bit a relevant change occurs in time and space complexity. Three storing schemes are discussed namely:

- Time efficient storing
- Trade off storing
- Space efficient storing

Before discussing these schemes in detail a simple rule base structure based on FSM is considered for reference (figure1). Then all the three storing scheme are applied to see the effect on time and space complexity of the rule base. Rule base in figure1, contains only two rules or IP addresses, they are 1.*10.*, 1.*121.0, (all the addresses are in IPV4). Here "*" means don't care condition i.e. all the value from 0 to 255 are acceptable.

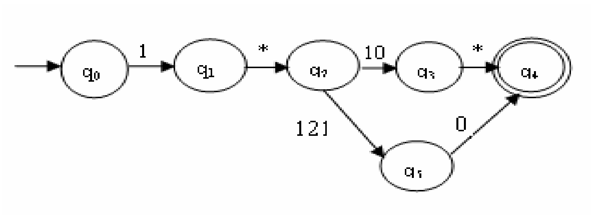

Figure1: Rule Base represented in the form of FSM

3.1. Time efficient storing

This storing technique will give constant time complexity of 4 comparisons plus some overhead of simple calculations for an IPV4 addresses for finding an entry. It can be done by using arrays of links (pointers) of size 256 where each pointer will point out the table of address equivalent to the index number of the array. If the address is not in the rule base then the pointer at that index value is a null pointer i.e. it doesn't point to any table. Whenever a "*" is encountered in the rule base it is stored in a simple link list having two fields i.e. data field and link field. In data field "*" is stored and the link field will point to the next table or next link if another "*" is encountered, depending on the rule base composition. Figure2 shows how rule base in figure1 will be stored when using this scheme.

For example whenever the system has to check whether the rule is present or not, it will see the start link and go to the index location 'a' (i.e. considering IPV4 addresses in the format a.b.c.d) and if the location at index 'a' points to a table then it will go to that table. Suppose we call this table as table 'a' so in table 'a' it will go to 'b' index and if the pointer is present in the table at "b" location, then it will go to table 'b' and check index 'c' and if the pointer is present, system goes to table "c" and checks the index "d" and if index "d" points to the "END' link, then the system terminates with successful search. And the index location can be calculated by simple formula, which is as follows:

Index location= base address + (index value * size of single link).

3.2. Trade off method

In this method we store the entries in the array of link list and simple links called spares with each link consisting of three fields one is Boolean data field and two pointers fields. Size of each array will be 2n , where n=0 to 8. Here for example n=6 is taken. So the size of array is 64, wherever system wants to store or check an entry it will divide each part of address by 64 i.e., (a/64) - 1, (b/64) - 1, (c/64) - 1, (d/64) – 1and take the floor function of each value, then search for the entries i.e. if a= 192 then, (a/64) - 1 = 2 then, if the table pointer is present it is present at the second spare of a mod 64 position.

The rule base in figure1 is stored as shown in figure3 when trade off storing scheme is used. The worst case time complexity when the array size is 64 will be 16, as the system will have to do at most 4 comparisons before going to next part of the address, and the address has 4 parts so the total number of comparisons will be 4*4=16. Worst case time complexity in case of trade off storing can be calculated by using simple formula i.e.:

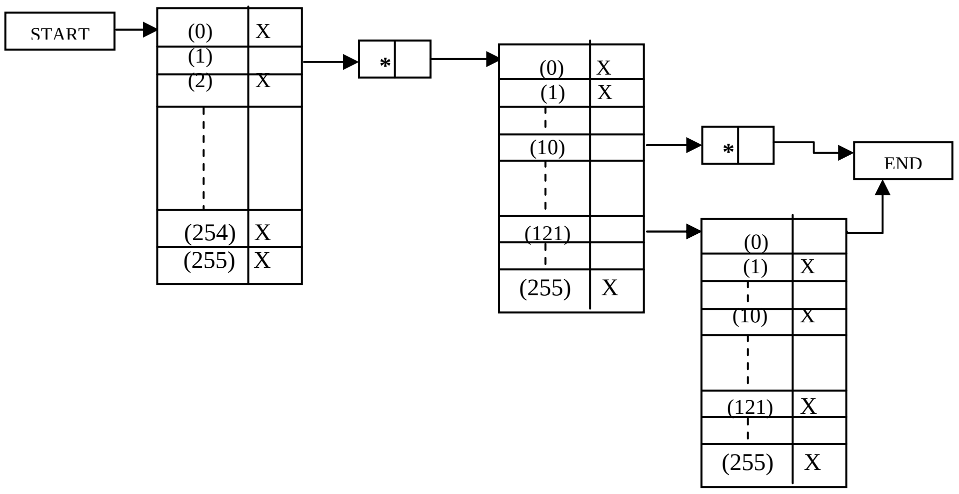

Figure2: Time Efficient Storing Scheme

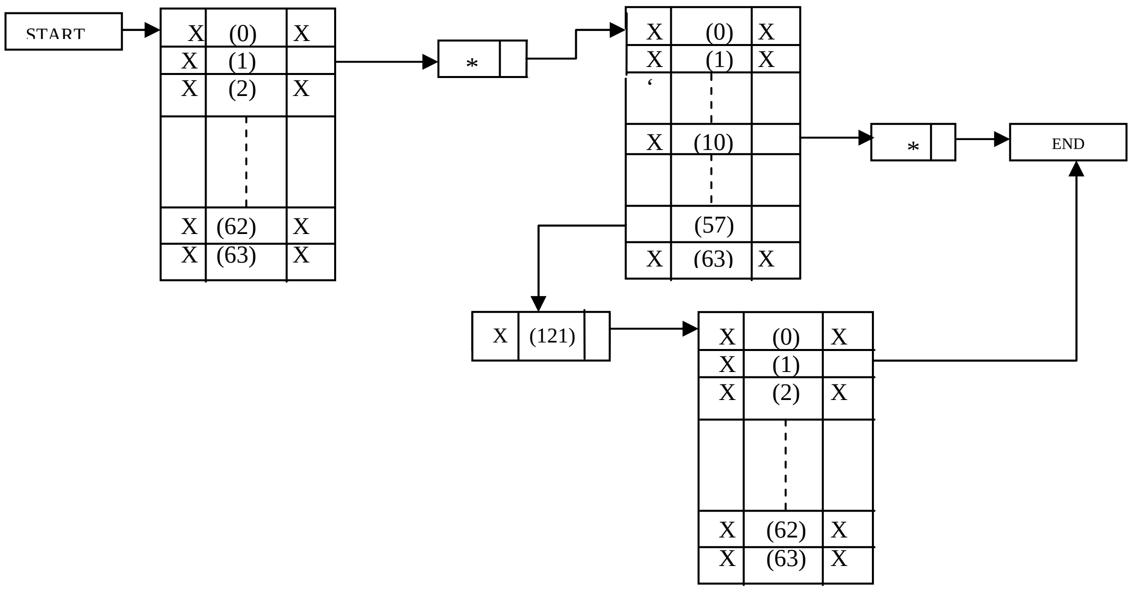

Figure3: Trade off Storing Scheme

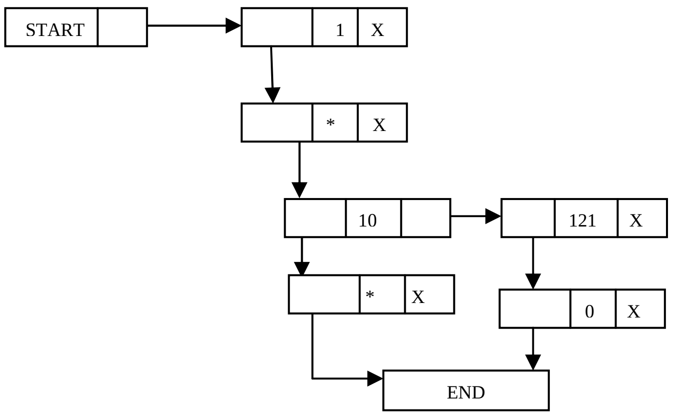

Figure4: Space Efficient Storing Scheme

Worst case time complexity= (256/2n)*P

Where, n= 0 to 8, depending on the size of array, and P is number of parts in IP address i.e. P= 4 if IP version 4 is used and P=16 is IP version 6 is used.

3.3. Space efficient Storing

This method is special case of trade off method where n=0 i.e., size of a table is 1. In this method the link will have three parts i.e. value part which will store value between 0-255 and two pointer fields one pointer pointing the next value at the same level and the second pointer pointing the values at the next level. Here same level means the same octet of the address. And next level means the next octet of the address i.e. if the system is in ‘a’ position of the IP address then it will go to the ‘b’ octet of the address.

Rule base in figure1 will be stored as shown in figure4 when space efficient storing is used. The worst case time complexity in this case will be 1024, because the system will have to do at most 256 comparisons in each level and there are 4 levels in case of IPv4.

4. Comparison

In this section comparison between the three storing scheme is done in terms of worst case time complexity and space requirement. Table1 shows the comparison between the three storing schemes. Comparison is done for IPv4 addresses and after assumption that there are approximately 232 rule base entries

Table1: Comparison between storing schemes

Type of Storing Scheme	Worst case Time Complexity	Space required
Time Efficient Storing	4(approx)	Maximum
Trade off Storing	4*256/Size of table(approx)	Depends on size of table
Space Efficient Storing	1024(approx)	Minimum

Selection of storing scheme can be done on the basis of number of entries in the rule base i.e. if the number of entry is more than half of the total possible IP addresses then time efficient storing should be given preference. In case of moderate number of entries trade off storing scheme can be preferred and value of “n” can be decided depending on the number of entries. And when the number of entries is very small then space efficient storing can be given preference over other two.

5. Conclusion and Future Work

This paper introduces how FSM based rule base of a genetic feedback algorithm based network security policy framework can be designed using simple data structures and this paper also discusses 3 different storing schemes based on the resources available and how the worst case time complexity can be reduced to constant value; in this paper the comparison between all the three storing scheme based on time and space complexity is also discussed. Future work includes designing the algorithm for implementing these three schemes

6. References

[1] Atish Mishra, Arun Kumar Jhapate, Prakash Kumar “Designing Rule Base for Genetic Feedback Algorithm Based Network Security Policy Framework using State Machine”, *ICCD 2009: 2009 International Conference on Computer Design and Applications,* May 2009

[2] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo, “A Goal-based Approach to Policy Refinement”, Proceedings 5th IEEE Workshop on Policies for Distributed Systems and Networks (Policy 2004), *IBM TJ Watson Research*

Centre, New York, USA, June 2004, PP22-229.

[3] CHEN Xiao-su, WU Jin-hua, NI jun “Genetic-Feedback Algorithm Based Network Security Policy Framework” Wireless Communications, Networking and Mobile Computing,, *WiCom 2007*, pp2278-2281.

[4] Hoi Chan and Thomas Kwok “A Policy Based management System with Automatic Policy Selection and Creation Capabilities by using Singular Value Decomposition Technique”, *Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks,*2006, pp96-99.

[5] J.O. Kephart and W.E. Walsh, “An AI Perspective on Autonomic Computing Policies”, *Fifth IEEE International Workshop on Policies for Distributed Systems, Networks,* 2004, pp3-12.

[6] RFC 2573, “A Framework for Policy-based Admission Control”, http://www.faqs.org/rfcs/ rfc2753. Html.