Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 12, (2023)
https://www.jneonatalsurg.com

OPEN ACCESS

A Comparative Analysis Of Ipv4 And Ipv6 Networks Using Finite State Machines Based On Rules

Manoj Kumar[1*], Ajeet Kumar[2] , Ajay Giri[3]

[1]Assistant Professor, Department of Computer Science & Engineering, Government Engineering College, Arwal, Bihar, India

Email ID : manojngp13@gmail.com

[2]Assistant Professor, Department of Computer Science & Engineering, Bakhtiyarpur College of Engineering, Bakhtiyarpur, Bihar, India

Email ID : azit.bce@gmail.com

[3]Assistant Professor, Department of Mechanical Engineering, B.P. Mandal College of Engineering, Madhepura, Bihar, India

Email ID : ajaygiri1973@gmail.com

.Cite this paper as: Manoj Kumar, Ajeet Kumar ,Ajay Giri, (2023) A Comparative Analysis Of Ipv4 And Ipv6 Networks Using Finite State Machines Based On Rules. *Journal of Neonatal Surgery*, 12 79-88.

ABSTRACT

Reaching network addresses based on the guidelines established in the rule base is one of the most effective ways to achieve quick results for IP addresses. These guidelines consist of IP addresses, combinations of these addresses, or combinations that include port numbers. This paper evaluates various designs and advantages of rule bases when it comes to accessing network addresses utilizing finite state machines. In a network, whenever new events occur, it is imperative that their detection or evaluation is efficient in terms of time. This can be accomplished by creating effective rules for monitoring network events. One method to achieve this is through the implementation of finite state machines. In this paper, we analyze the search times for IPV4 and IPV6 network addresses. For IPV4 addresses, which involve approximately 232 entries in a table, the mathematically calculated search time for a system based on FSM are around 210, while the time complexity for a linear search system with the same number of entries is approximately 232. For IPV6 addresses, with about 2128 entries in a table, the search time using FSMs is roughly 212, whereas a linear search system would take about 2128. This paper demonstrates that the search time for any IPV6 address using FSMs shows a significant improvement at approximately 29.

Keywords: *IPV4 address, IPV6 address, finite state machine, rule base*

1. INTRODUCTION

Currently, a significant number of individuals are interconnected through the internet on a global scale. The popularity of the internet continues to grow each day. Governments and various organizations are also linked through the internet. The usage of IP addresses is rapidly increasing. Therefore, when a new IP address emerges, the process of evaluating it should be more efficient. This can be achieved by establishing a rule base for the searching of rules. One effective approach to accomplish this is by utilizing a Finite State Machine.

The Internet Protocol version 4 (IPv4) utilizes a 32-bit address. In the worst-case scenario, the time complexity for locating an IP address with a linear search approach will be 232 (approximately). Conversely, employing a Finite State Machine based system will require about 210. For IPv6 addresses, which have a 128-bit address space, the worst-case time complexity for searching an IP address through a linear search system will be 2128 (approximately), but when employing a rule-based finite state machine, it will only take 29 (approximately).

This paper illustrates how finite state machines can be utilized to represent rule bases and how they can be optimized for the system's efficient operation, as well as their advantages compared to linear systems. It also compares the two network addresses and analyses them for speedup among them.

This document consists of eight sections and is arranged as follows: section 1 covers the Introduction, section 2 discusses Rule-Based Systems, section 3 addresses Finite State Machines, section 4 focuses on IPv4, section 5 covers IPv6, section 6 presents a Literature Review, section 7 identifies the Problem, and section 8 proposes a Solution, followed by a conclusion and directions for future work.

2. Rule based system

Rule-based systems, also referred to as production systems or expert systems, are the most basic type of artificial intelligence. The concept of a rule-based system is largely derived from expert systems, which emulate the decision-making processes of human specialists when tackling knowledge-intensive issues. Rather than encoding knowledge in a straightforward, fixed manner that states what is true; a rule-based system expresses knowledge through a collection of rules that dictate actions or conclusions in various circumstances.

A rule-based system is a method of encoding the knowledge of a human expert in a specific domain into an automated framework. A rule-based system can be constructed by employing a collection of assertions along with a set of rules that dictate how to respond to the assertions. These rules are typically articulated in the form of if-then statements (known as IF-THEN rules or production rules).

IF P THEN Q

This can also be represented as

P=>Q.

A rule-based system is made up of a collection of IF-THEN rules, a set of facts, and an interpreter that manages the application of these rules based on the available facts. The concept behind an expert system is to capture an expert's knowledge and translate it into a series of rules. When presented with the same information, it is anticipated that the expert system will operate in a manner comparable to that of the expert. Rule-based systems are straightforward models that can be modified and used for a wide variety of issues. The key requirement is that the knowledge regarding the problem domain must be articulated in the form of IF-THEN rules. Additionally, the scope should not be too extensive, as an excessive number of rules can lead to inefficiencies in the problem-solving capability of the expert system.

3. Finite State Machine

A Finite State Machine (FSM) represents a behavioural model that utilizes states and transitions between those states. A transition refers to a change of state that is initiated by an input event, meaning that transitions link specific state-event combinations to different states. As the name suggests, the collection of states must be limited. Additionally, it is expected that there exists a limited number of unique input events or their categories (types, classes). Consequently, the collection of transitions is also finite.

Finite State Machines (FSMs) are among the most commonly utilized models in computer programming overall. They are especially prevalent in the programming of embedded systems and in the depiction of digital circuits. The success of modeling with FSMs has led to their inclusion as a component of the Unified Modeling Language standard by the Object Management Group.

4. Internet Protocol Version 4 (IPv4)

The Internet Protocol is a key protocol within the TCP/IP protocol suite. This protocol operates at the Network layer of the OSI model and at the Internet layer of the TCP/IP model. Therefore, it is tasked with identifying hosts through their logical addresses and routing data among them across the underlying network.

IP offers a way to uniquely identify hosts using an IP addressing scheme. It employs a best-effort delivery mechanism, meaning it does not assure that packets will be delivered to the intended host, but it will make every effort to reach the destination. Internet Protocol version 4 utilizes a 32-bit logical address.

5. Internet Protocol Version 6 (IPv6)

Internet Protocol version 6 (IPv6) is the most recent iteration of the Internet Protocol (IP), which is the communication protocol responsible for identifying and locating devices on networks as well as directing traffic on the Internet. IPv6 was created by the Internet Engineering Task Force (IETF) to address the long-expected issue of the depletion of IPv4 addresses.

IPv6 is designed to succeed IPv4, which continues to handle the majority of Internet traffic as of 2013. As of February 2014, the proportion of users accessing Google services via IPv6 exceeded 3% for the first time. [16]

Every device connected to the Internet is assigned an IP address for identification and location purposes. As more devices continue to connect to the Internet, the limitation of available addresses in the IPv4 address space has become apparent. IPv6 utilizes a 128-bit address format, which allows for 2^{128}, or around 3.4×10^{38} addresses, equating to more than 7.9×10^{28} times the number available under IPv4, which relies on 32-bit addresses. IPv4 offers roughly 4.3 billion possible addresses. The two protocols were not originally built to work together, making the transition to IPv6 more challenging.

IPv6 addresses consist of eight segments of four hexadecimal digits, divided by colons, as seen in 2001:0db8:85a3:0042:1000:8a2e:0370:7334, though there are ways to shorten this full representation. On June 6, 2012, several major Internet companies decided to enable IPv6 on their servers. Currently, they are utilizing a Dual Stack approach to run IPv6 alongside IPv4. [10]

6. Literature Review

The current state of research on IPv4 vs IPv6 comparative analysis reveals a significant gap in finite state machine (FSM) and rule-based approaches. While extensive literature exists on performance comparisons, transition mechanisms, and deployment strategies, there is a notable absence of comprehensive FSM-based theoretical frameworks for analyzing protocol state complexity and behavior modeling. Most studies focus on empirical performance metrics such as throughput, latency, and packet loss [13], [14], [15], with limited attention to formal verification approaches and mathematical models for protocol state representation. The research landscape shows strong emphasis on practical implementation challenges and transition mechanisms [16], [17], [18] but lacks systematic FSM-based analysis frameworks that could provide deeper insights into protocol behavior and security state management differences.

This manuscript [1] primarily examines the design and advantages of a rule base that is predicated on Finite State Machines. These rules consist of either individual IP addresses or a combination of an IP address along with additional parameters, such as port numbers, among others. For an IPv4 address, which comprises 32 bits, a table encompasses 2^{32} entries. Employing linear search for locating an address necessitates approximately 2^{32} operations in the worst-case scenario. Conversely, utilizing a Finite State Machine facilitates this process in approximately 2^{10} operations, which is significantly more efficient than linear search. In this context, the Finite State Machine serves as an effective means for delineating the rule base. In this scenario, the rule base is fundamentally based on IP addresses.

The paper [12] surveys IPv6 transition solutions, emphasizing the critical role of state management in networking performance and addressing. It discusses various state management types and their impacts on network solutions. The authors summarize the applicability of state management and suggest future research directions in IPv6 transition.

7. Comparison of IPv4 and IPv6

7. 1 Rule Base Design of IPv4

For designing a rule base following steps to be followed-

Step 1. A collection or aggregation of Internet Protocol (IP) addresses that share similar characteristics or attributes is meticulously constructed..

Step 2.- To enhance optimization, a Finite State Machine (FSM) of Internet Protocol (IP) addresses is constructed, incorporating the highest possible quantity of "don't care" conditions, denoted by the symbol "*", which signifies all values ranging from 0 to 255.

Step 3.- The remaining address is verified by providing an input to the FSM.

Step 4.- If the FSM meets the conditions of the given address (rule), it may be removed from the rule base.

For example- if the following rules

149.*.54.*

149.30.54.*

149.*.50.252

The initial rule encompasses the other two rules, so we will eliminate the last two.

Step 5. If the machine does not accept the address, then the machine is altered by integrating the rule (address) into the FSM prior to examining the other addresses in the cluster.

Step 6. After optimizing all the clusters, the FSMs from each cluster are combined to create a single FSM that represents the Rule Base.

For example the rule base may consists following IP addresses

142.20.10.0

142.*.0.*

149.12.10.30

152.21.23.44

142.*.10.45

149.*.200.67

Step 1- Clusters of IP addresses are made as follows:-

Cluster 1- 142.20.10.0

142.*.0.*

142.*.0.45

Cluster 2-

149.*.200.67

149.12.10.30

Cluster 3-

152.21.23.44

Step 2- Clusters are optimized one by one

Optimization of cluster 1 – The address 142.*.0.* presents the highest number of don't care conditions; therefore, a finite state machine (FSM) for this IP address has been created, as illustrated in figure 7.1.1.

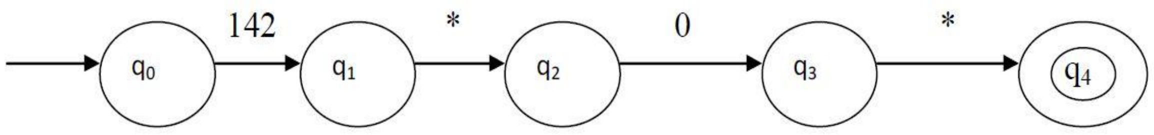

Fig. 7.1.1 FSM of address 142.*.0.*

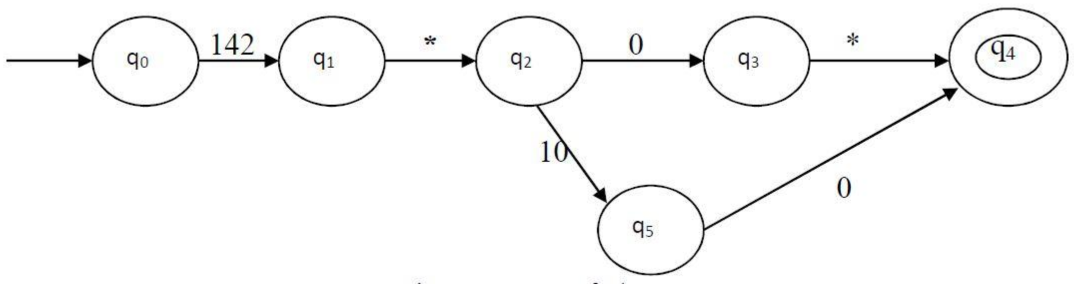

Fig. 7.1.2 FSM of cluster 1

Optimization of Cluster 2

Address 149.*.200.67 has maximum don't care.

So FSM of this IP address is constructed and merging them FSM of 149.12.10.30

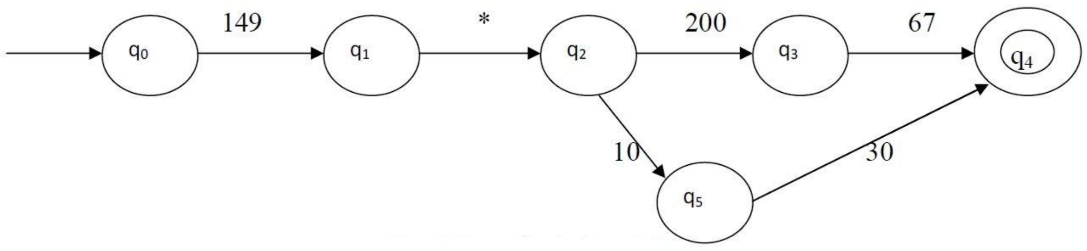

Fig. 7.1.3 FSM of cluster 2

Optimization of Cluster 3

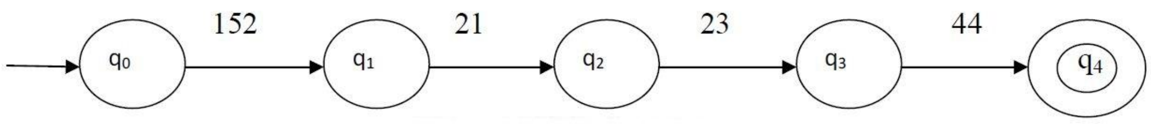

Fig.7.1.4 FSM of cluster 3

Once optimization of all the clusters is done; all the FSM are merged to get FSM for the complete rule base as in fig 7.1.5.

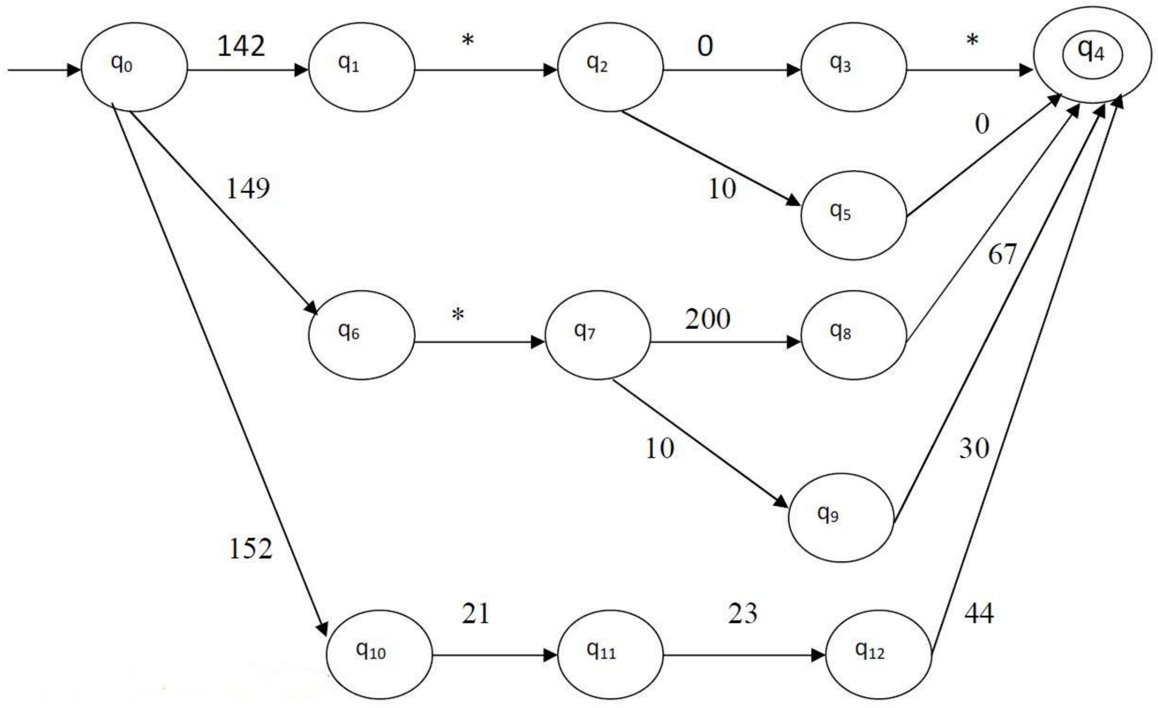

Fig-7.1.5 FSM of rule base

After completing the FSM drawing in fig-7.1.1, the subsequent rule of the cluster, 142.*.0.45, is evaluated and found to be satisfied by the machine, prompting its deletion. Next, the rule 142.20.10.0 is assessed, and since it does not meet the machine's criteria, the machine is adjusted accordingly. The updated machine will be depicted in fig-7.1.2.

The optimization of cluster 1 is now complete. Likewise, clusters 2 and 3 have also been optimized, with their optimized finite state machines illustrated in figures 7.1.3 and 7.1.4, respectively. Once all the clusters were optimized, the finite state machines from each cluster were combined to produce the finite state machine for the entire rule base, as depicted in figure 7.1.5.

Hence, the rule base can be defined by (Σ, *S*, *s*0, δ, *F*), where

Σ = {0-255.*}

S= {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12}

s0= {q0}

δ =S x Σ→ S

F= {q4}

For explaining Transition function only transition table for fig-7.1.1 is drawn

Table 7.1: Transition table for fig. 7.1.1

State	142	0	*
q_0	q_1	-	-
q_1	-	-	q_2
q_2	-	q_3	-
q_3	-	-	q_4
q_4	-	-	-

Similarly when we make table for rule base there will be 257 column and number of rows indicates number of States.

2. ADVANTAGE

It reduces the time complexity for searching a rule as compared to linear system.

For example:

Maximum number of entries a rule base can have=$(255)^4=2^{32}$(approx.)

Worst case linear searching time "X1"=2^{32}(approx.)

Now when FSM based system is used

Worst case searching time"X2"=255+255+255+255=2^{10}(approx)

Since there will be at most 255 comparisons in each state of machine and one have to pass only four states (excluding initial state). So total comparisons will be 255*4=1024 which is very small in comparison to 2^{32}. If any field is increased in the rule base i.e. combination of IP addresses and port numbers together are used to define a rule and with the consideration that there are only 1000 ports which are active. Then increase in complexity will be as follows:

Thus increase in time complexity of linear system=X1* 10^3

Increase in time complexity of FSM based system=X2+ 10^3

Thus time complexity in case of linear system increases multiplicatively, where as in case of FSM based system it will increase additively.

7.2 Rule base design of IPv6

Rule base contains a set of rules that tells what to do or what to conclude in different situations. Here it is IP address based.

For example- in the following IPv6 address

2001:0DB8:*5A3:0042:1000:8A2E:00*2:7334

2001:0DB8:85A3:0042:1000:8A2E:00*2:7334

2001:0DB8:*5A3:0042:1000:8A2E:0042:7334

Here * means any hexadecimal number between 0-9 or A-E

Here the first rule has included the other two rules, so we delete the last two rules from rule base.

IPv6 address is 128 bits long and it is represented as eight groups of four hexadecimal digits separated by colons. For example

2001: 0DB8: 85A3: 0042: 1000: 8A2E: 0370: 7334

For optimizing a rule base

Step 1. Form a cluster of similar type of IPv6 address group.

Step 2. Create a FSM with maximum number of don't care condition which is represented by "*" meaning all the values of hexadecimal number from 0-15(0-9, A, B, C, D, E) are accepted.

Step 3. The remaining group of IPv6 address is checked by passing them through the FSM as input string.

Step 4. If the address group(rule) passed is satisfied by the machine then it can be deleted from the rule base, if the address (group) is not satisfied by the machine, then the machine is modified by merging that rule (address group) in the FSM before checking the remaining address group of the cluster.

Step 5. Once all the clusters are optimized, FSM's of all clusters are merged forming the FSM representing the rule base.

For example the rule base may consists following IPv6 address-

2001:0DB8:85A3:0042:1000:8A2E:0042:7334

2531:0CB8:*5A3:0042:1000:8A2E:00*2: FFFA

2*0*:0DB8:*5A3:0042:1000:8A2E:00*2:8329

3*51:0EB9:45A3:0542:1000:8A2E:0082:7384

3A69:0EB8:*5A3:0042:1000:8A2E:00*2:F334

FD02:0DB8:*5A3:0042:1000:8A2E:00*2:0004

Now we consider only one group i.e. first group of every IPv6 address

Step1. Cluster the similar type of group

Cluster1. - 2001

2531

2*0*

Cluster2. - 3*51

3A69

Cluster3. - FD02

Step2. Every cluster is optimized one by one

Optimization of cluster 1

Since the group (address) 2*0* has maximum don't care so we draw FSM for this address as in fig-8.1

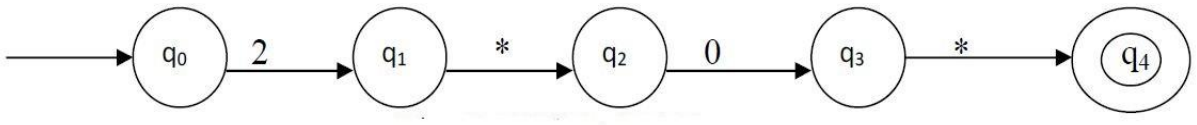

Fig- 7.2.1 FSM of 2*0*

Now the next rule of cluster is checked i.e. 2 0 0 1. As it is satisfied by the machine so it is deleted and then the next rule is checked which is 2 5 3 1. As it is not satisfied by the machine so machine is modified and the modified machine will be as in fig-8.2

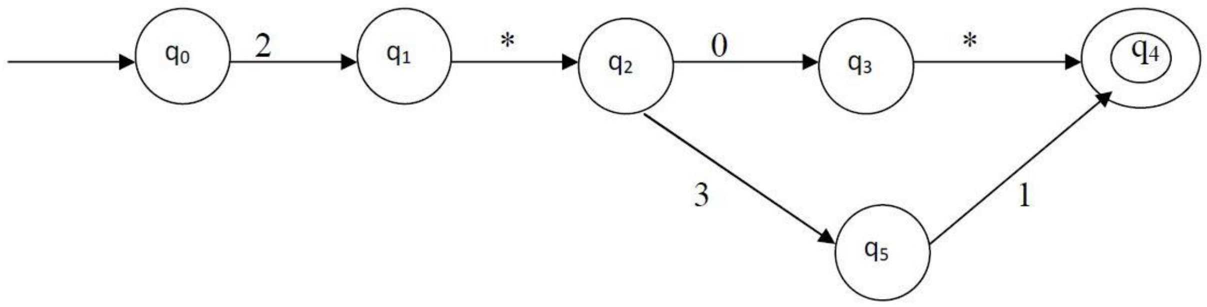

Fig-7.2.2 FSM of cluster 1

Finite State Machine of Cluster 2

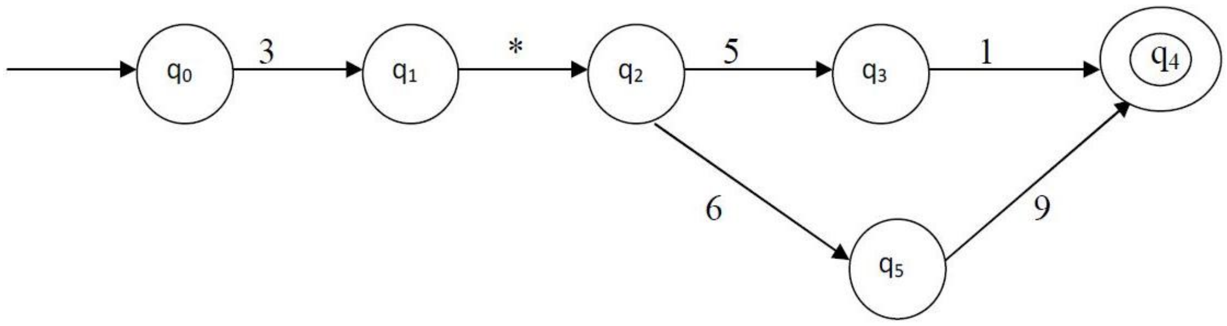

Fig-7.2.3 FSM of cluster 2

Finite State Machine of Cluster 3

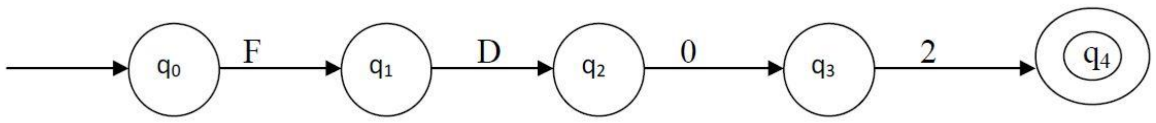

Fig-7.2.4 FSM of cluster 3

After optimization of all the clusters is done. All FSM are merged to get the FSM for the complete rule base as in fig-8.5

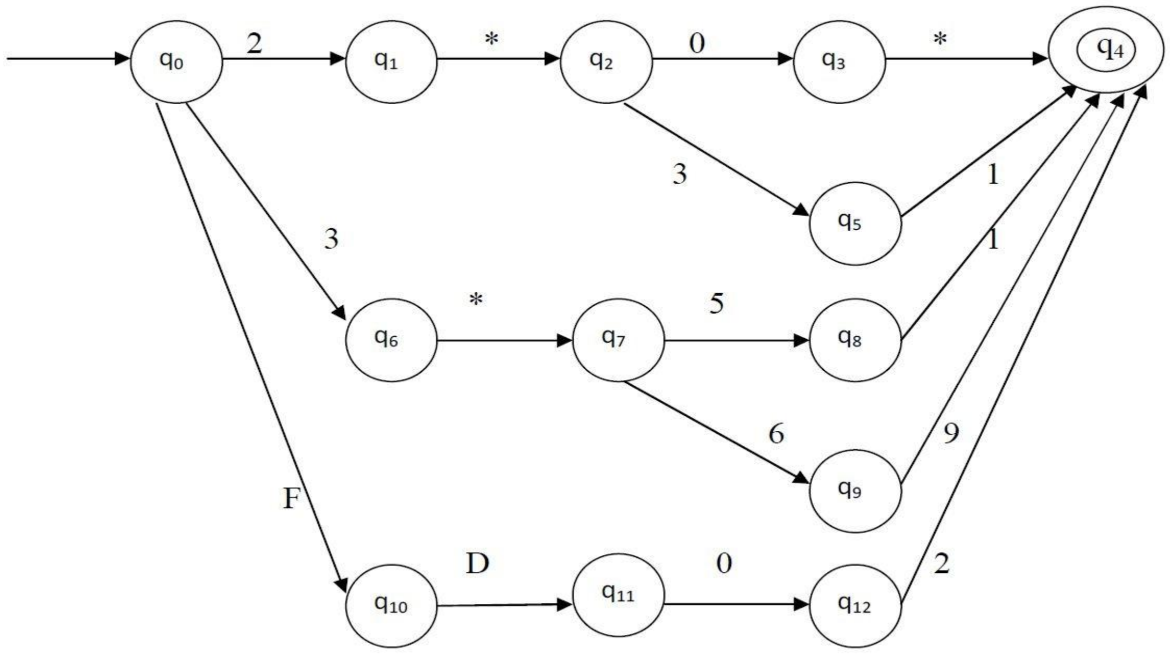

Fig-7.2.5 FSM of rule base

Hence the rule base can be defined by (Σ, *S*, *s*0, δ, *F*), where

Σ = {0-15}

S= {q0, q1, q2, q3, q4, q5, q6, q7, q8, q9, q10, q11, q12}

s0= {q0}

δ =S x Σ→ S

F= {q4}

For explaining Transition function only transition table for fig-8.1 is drawn

Table 7.2: Transition table for fig 8.1

State	2	*	0
q_0	q_1	-	-
q_1	-	q_2	-
q_2	-	-	q_3
q_3	-	q_4	-
q_4	-	-	-

Similarly when we make table for rule base there will be 16 columns and number of rows indicates number of States. This is only for one group out of eight group of IPv6 address.

Similarly this process is applied on every group of IPv6 address.

It reduces the time complexity for searching a rule as compared to linear system.

For example:

Maximum number of entries a rule base when applying on all groups can have

= $\{(16)^{4\}8}$= 2^{128}(approx.)

Worst case linear searching time =2^{128}(approx.)

Now when FSM based system is used

Worst case searching time of one group out of eight group of IPv6 address

$=16+16+16+16=2^6$(approx.)

Worst case searching time $=2^6 + 2^6 + \ldots\ldots\ldots\ldots 8$ times$=2^6*8=2^9$(approx.)

Since there will be at most 16 comparisons in each state of machine and one have to pass only four states (excluding initial state). So total comparisons will be 16*4=64 which is for only one group out of eight group of IPv6 address. Now overall total comparison will be $64*8=2^9$ which is very small in comparison to 2^{128}.

Thus we have seen that time complexity in case of linear system increases multiplicatively; where as in case of FSM based system it will increase additively.

3. CONCLUSION AND FUTURE WORK

As a result of our comprehensive analysis, it can be decisively concluded that the implementation of a Rule-based Finite State Machine for the purpose of conducting IP address searches represents a significantly more efficient and effective approach in comparison to traditional methodologies. Through meticulous observation and empirical evaluation, it has become evident that in linear search systems, the time complexity escalates in a multiplicative fashion, leading to substantial inefficiencies, whereas in systems that leverage Finite State Machines, the time complexity tends to rise in an additive manner, thereby contributing to enhanced performance and optimization in the search process. This distinction underscores the advantages of adopting FSM-based systems, as they not only streamline the searching procedure but also mitigate the computational burdens associated with more conventional linear search algorithms, ultimately resulting in superior operational efficacy.

4. FUTURE WORK

Future works includes developing a robust framework for the implementation of a finite state machine (FSM) that will serve as a foundational rule base, it is imperative to ensure that this theoretical construct is effectively applied within the context of a real-world system, thereby facilitating a seamless integration of computational logic with practical applications in various operational environments.

REFERENCES

[1] Atish Mishra, Arun Kumar Jhapate, Prakash Kumar, “Designing Rule Base for Genetic Feedback Algorithm Based Network Security Policy Framework using State Machine”, 2009 International Conference on Signal Processing Systems.

[2] A.K. Bandara, E.C. Lupu, J. Moffett, and A. Russo, “A Goal-based Approach to Policy Refinement”, Proceedings 5th IEEE Workshopon 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, 2007. 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] L.P. Kaelbling, M. Littman, and A. Moore, “Reinforcement Learning: A Survey”, Journal of Artificial Intelligence Research, May 1996,pp237-285.

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

[8] Xin Yue, Wei Chen, Yantao Wang, “The Research of Firewall Technology in Computer Network Security”, 2009 Second Asia-Pacific Conference on Computational Intelligence and Industrial Applications.

[9] Young-Ho Kim, Jeong-Nyeo Kim, “Design of Firewall in Router using Network Processor”.

[10] Crina Grosan, Ajith Abraham, “Intelligent Systems A Modern Approach”, Springer

[11] Charles M.Kozierok ,”The TCP/IP Guide version 3.0”

[12] Cui, Yong and Chen, Yuchi and Liu, Jiangchuan and Lee, Yiu-leung and Wu, Jianping and Wang, Xingwei, IEEE Network journal, 2015, “State management in IPv4 to IPv6 transition”.

[13] Li, K. H., & Wong, K. Y. (2021). Empirical Analysis of IPv4 and IPv6 Networks through Dual-Stack Sites. Information, 12(6), 246.

[14] Bachilo, V. V., Dravitsa, V. I., & Listopad, N. I. (2023). Comparative Analysis of the Capabilities of IPv6 and IPv4 Protocols to Provide the Designated Quality of Service. Doklady BGUIR, 20(8), 75-83.

[15] Dobrinov, N., Parra, L., Garcia, L., & Romero, O. (2016). Comparative Study of a Router Performance with IPv4 and IPv6 Traffic. Network Protocols and Algorithms, 8(3).

[16] Hanumanthappa, J., Manjaiah, D. H., Aravinda, C. V., & Joshi, V. B. (2010). A Comparison of Performance evaluation metrics and Simulation parameters of a Novel IPv4/IPv6 Transition Mechanism: BD-SIIT vs. DSTM.

[17] Basit, A., & Hussain, R. (2017). Performance evaluation of simultaneous network configuration using dual stack and tunnel transition techniques: An enterprise level analysis. International Journal of Advanced and Applied Sciences, 4(1), 102-109.

[18] Almutlaq, W. M., & Elfadil, N. (2022). A Comparative Performance Evaluation of IPv4/IPv6 Using Network Simulation and Virtualization Tools. International Journal of Computer Science and Mobile Computing, 11(10)