# Wired Network Analysis Using Ftp Server in Firewall Load balancing based on Round Robin Method

M.Kowsalya[1], Dr.A.Senthilrajan[2]
1(computer science, Alagappa University, and karaikudi
Email: kowsalyamathiyalagan96@gmail.com)
2 (computer science, Alagappa University, and karaikudi
Email: agni_senthil@yahoo.com)

## Abstract:

In modem technology cluster nodes are build to process of finding network range at load balancing on incoming and outgoing network. It provides a best way to transmit a communication network from one place to other place at anytime and anywhere using network load balancer. It was designed as a routed and transmits between network segments. It has some security permission to block both incoming and outgoing connection based on polices of firewall. When fire wall check redundacncy in networks terms based on clustering methods. In previous work it was implemented on K-Means ++ to get 90% of accuracy on load balancer. Now we proposed with Mean Shift clustering to detect a firewall failure due to network failure detection with accuracy of 94%. In case the network fails firewall load balancer detects security threats and block connection to made a path secured.

***Keywords*** — firewall, load balancing, security, network.

## 1. INTRODUCTION

Traditional firewall is typically implemented as a single device that applies security policies for data packets. In order to avoid this, in this paper, fault-tolerant firewall cluster is proposed as a new solution to achieve the firewall high performance, high reliability, high availability. The load-balancing and state sharing algorithm is designed to achieve load-balancing and fault-tolerant by means of sharing connection states in firewall cluster. When a firewall is overloaded, firewall will transfer part of loads to the light-load partner's nodes; when a firewall is failed, the backup firewall replaces the lapsed firewall smoothly. The experiment shows that the load-balancing and state-sharing algorithm in firewall cluster can effectively improve the performance, and prevent the single-point of failure. In recent years, load balancing method with firewall grows a rapid extension of internet and balancing as a backend server is a primary measure at the security of the network, servers and application. It is a key concern & together with VPN services, SSL loader devices, to the load balancer. Firewall is an important security device that using security rules for arriving packets. Packets are processed by comparing the security rules to determine whether the packets are accepted or denied to go through . With the continuous increasing of network loads and complexity of security policies, the amount of time required to process will also be increasing. A firewall has limited resources in terms of link speed, memory, and processor power. These factors determine the capacity and capability of the device in terms of session scalability and raw packet-switching performance.

In this case, the traditional single firewall probably become a bottleneck and will be down easily so as to lead to low reliability and low availability. Building a faster single firewall is possible; however, it will be high cost and low performance/price ratio. Additionally, single firewall still is a single point of failure, so it is not scalable and unable to provide reliable and differential service. Firewall cluster achieves packet parallel processing by multiple firewalls, which can improve the firewall's processing ability and prevent the firewall's single-point of failure. Firewall cluster extends redundancy beyond the traditional active/standby backup mode provided on a device basis. This cluster greatly increases the availability of firewall protection and improves the stability of the network in general.

It is a bridge between the network and server but load balancer usually need to learn the health status of the server, also don't need to be able to set up the network protocol that packet content information to modify or read. Firewalls are physical devices that exist between network segments, typically in a routed design. They perform stateful inspection of sessions going through them from one segment to the other. Firewalls block or permit sessions based on configured security policies and rules.

## 2. LITERATURE REVIEW

**Section [1]: A Load-Balancing and State-Sharing Algorithm for Fault-Tolerant Firewall Cluster**

**Author:** Peng Zhichao ; Chen Daiwu ; He Wenhua

Firewalls are rule based access control system through process. All incoming and outgoing network traffic based on specific security organization. Compared with a single firewall, firewall cluster achieved high-performance, high reliability, high availability. in this idea we proposed a firewall cluster with load-balancing algorithm. Additionally,

we built a fault-tolerance firewall cluster and finished the comparison experiments. Experiments show that load-balancing and state-sharing algorithm is effective in firewall cluster.

**Section [2]: Cost-Effective N:1 Firewall Array Via Subnet- Levcl Load Balancing by SDN/OpenFlow Switches**

**Author:** Christian I. Quispe ; Cesar A. Santivanez

A novel subnet-level load balancer design based on an SDN / OpenFlow switch is presented. The design guarantees the permanence of the active sessions after the traffic migration (Firewall handover process) in response to traffic bursts. The proposed balancer can be used to implement a N:1 Firewalls array to significantly reduce the Capital Cost (CapEx). We present our prototype implementation in the ExoGeni TestBed. Initial test results for rates in the order of 110 Mbps. Testing at higher rates has been limited by intense traffic fluctuations introduced by the Traffic Generator based on the Python Scapy library, which is unable to deal with high rates or time intervals in the order of microseconds. It is estimated that in real scenarios (transmission rate higher than 1 Gbps) the performance of the balancer will improve, as fluctuations will decrease. Testing of 1Gbps traffic intensities over our SDN testbed composed of Pica 8 switches will be our next task. Future works include: analysis of the impact of choosing the more or less voluminous subnet in the system, the implementation of a "firewall bypass" module that migrate elephant sessions deemed safe such as streaming video (similar to the work done in SciPass[2]), and the testing of the performance of the load balancer in a CyberSecurity Test-Bed with more traffic traces and commercial firewalls.

**Section [3]: Firewall and load balancing as an application of SDN**

**Author:** Nayana Zope ; Sanjay Pawar ; Zia Saquib

To meet the requirements of today's dynamic IT infrastructure such as Server virtualization, user mobility, the need to rapidly respond to changing business conditions, IT-as –a-Service, Software Defined Networking provides an edge over the conventional network architecture. SDN and OpenFlow as a standard are sufficiently explained in this paper. This paper describes how it helps researchers to develop and test their applications in a physical scenario for gradual deployment into production network. The important component of SDN i.e. controller is mainly focused. Floodlight controller itself provides the network visibility GUI and implements a firewall and LoadBalancer module using REST API. Communication between network elements can be monitored and controlled by firewall. To innovate on and form the network through which we can fully interact, SDN is still growing in many areas of the research. We expect to enlarge this work in future for routing component and to observe how more networks can be connected and controlled and discover the communication between host machines of divergent networks.

**Section [4]: Software Defined networking based routing firewall**

**Author:** Karamjeet Kaur ; Sukhveer Kaur ; Vipin Gupta

We were successfully able to create routing firewall application using pox controller which was performing both routing and filtering. We tested this application using mininet emulator. Future work can involve testing it on real hardwareor creating this type of routing firewall application using other SDN controllers such as RYU and OpenDaylight and doingperformance analysis. Other area of future work could beadding deep packet inspection capability into

this routing firewall to make it application level routing firewall

**ALGORITHM USED:**

**FUZZY C MEANS ALGORITHM**

FCM is a powerful algorithm that was invented in the late 19th Century. The same way, build the fuzzy model using FCM then optimize the weighting exponent m by optimizing the least square error between the output of the fuzzy model and the output from the original function by entering the same tested data

**Algorithmic steps for Fuzzy c-means clustering,**

1. Assign an initial random centroid value to each cluster (Group).
2. Compute the distance between each point and the cluster centre using a simple algorithm is used to calculate the fuzzy membership '$\mu_{ij}$' using:

$$\mu_{ij} = 1/\sum_{k=1}^{c}.(d_{ij}/d_{ik})^{(2lm-1)}$$

3. Based on distance between each point and the cluster centre, re-compute the member vj function..

$$v_j = \frac{\left(\sum_{i=1}^{n}.(\mu_{ij})^{m} x_1\right)}{\left(\sum_{i=1}^{n}.(\mu_{ij})^{m}\right)}, \forall j = 1,2,\ldots.c$$

4. Based on the new membership function, re-compute the centroid.
5. It continues till this condition is true.

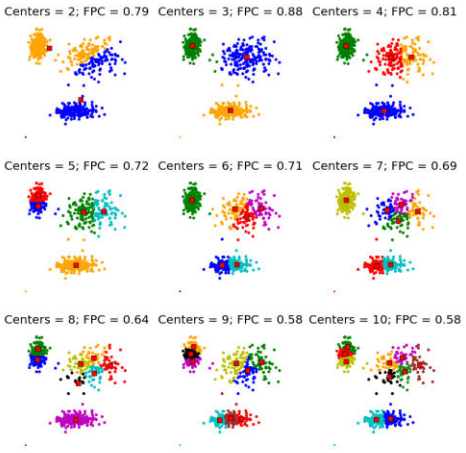



Figure1 fuzzy c means implementation

**K-MEANS ++ ALGORITHM**

K-means++ algorithm is used to clustering. The basic principle is to make the distance between the centers of clusters as large as possible.

For each data from 2 values (vj, yj),are the points to calculate its distance D(j) to the find nearest cluster center

$$D(k) = \sqrt{[v_j - x_i]^2 T(y_j - y_i)}$$

Calculate the distance from each node to the center of the cluster according to formula and join the corresponding cluster.

Then, the center of the cluster is updated by formula Finally, all nodes are evenly divided into K clusters.

$$L(m_2, N_i)\tau = \text{El l E R e ( ) u lec}$$

K means ++ starts from cluster allocation process when it comes randomly then searching the center values of starting point.

1. Take a center point as c1 and make it randomely form the value X.

2. Take a new center point v as

, choosing x ∈ v with probability D(x) 2 P

x∈v D(v)

3. Repeat Step 2 . until we taken a k center with perpective form.

4. This procedure comes in the way of K-Means algoritm to derive the equqtion to k and l .

$$E_L(k,d) = \begin{Bmatrix} k\ x & E_{fm} + k\ x\ \varepsilon_{fs}\ x\ d^2, d < d_0 \\ l\ x & E_{elec} + l\ x\ \varepsilon_{amp}\ x\ d^4, d \geq d_0 \end{Bmatrix}$$

**Pseudocode For K-Means++**

Input: C={$d_1$,$d_2$,…..$d_n$) (Pair of entities for clustered)

K (represents number of clusters)

Max (I) (Limitation of iteration)

Output: D={$c_1$,$c_2$,…$c_n$)(No. Of centroids set)

L={I(d) | d = 1,2,…n}( Set of cluster labels D)

Foreach $c_i$ £ C do

$C_i$,- $e_j$£ E (e.g.random selection)

end

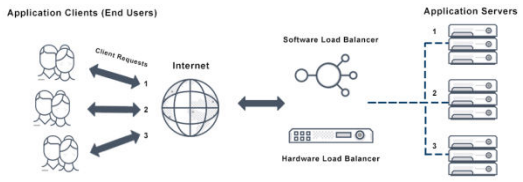
Foreach $e_i$ £ E do

l($e_i$)←argmin Distsance($e_i$, $c_j$)j £{1…..k)

end

**Round Robin Method**

Round-Robin (RR) algorithm is one of the simplest load balancing algorithms being used. It uses a circular list and a pointer to the last selected server to make dispatching decisions. It dispatches user requests to web servers from the first web server to the last one by order.




That means if Si was the last chosen node, the new request is assigned to Si+1, and it is the number of server nodes. The advantage of the algorithm is its simplicity. When using t is algorithm, we don't need to record the state of all the communication. To guarantee fast dispatching of large numbers of requests per second, the web switch cannot use highly sophisticated algorithms. So Round-Robin algorithm is one of the fastest solution to prevent the web switch becoming the primary bottleneck of the cluster.

**Steps**

1- Create an array rem_bt[] to keep track of remaining
burst time of processes. This array is initially a copy of bt[] (burst times array)
2- Create another array wt[] to store waiting times of processes. Initialize this array as 0.
3- Initialize time : t = 0
4- Keep traversing the all processes while all processes
are not done. Do following for i'th process if it is not done yet.
a- If rem_bt[i] > quantum
(i) t = t + quantum
(ii) bt_rem[i] -= quantum;
c- Else // Last cycle for this process
(i) t = t + bt_rem[i];
(ii) wt[i] = t - bt[i]
(ii) bt_rem[i] = 0; // This process is over

## 3. PROPOSED WORK

At present, most research on load balancer utilizes some algorithms that is settled in advance. When a new user request comes, it works and then makes dispatching decisions, which seldom taking load prediction into consideration. So, in our paper, we discuss more about the load balance prediction. According to the result of load balance prediction, we can estimate the current situation of all the servers in the cluster and the dispatcher can adopt suitable load balancing algorithms and make some adjustment to achieve better load balance. By prediction, load balancers can make dispatching decisions while the chosen web server is processing the user request, which is time-saving. And for large-scale network, by keeping the record of load on each server node, we can roughly know the running status, capacity consumption and bearable load amount of web servers, in which case we can even have an overall

assessment on the cost and performance of the web cluster.

**COMPARISON TABLES**

| Algorithm Used | Accuracy | Time taken (in seconds) |
|---|---|---|
| Fuzzy C means | 78% | 900ms |
| K-means ++ | 90% | 720ms |
| Round Robin | 94% | 580ms |

## 4. CONCLUSION

In our project vendor lock is implemented to prevent and build in our firewall application. We also make some simulations on MALTAB. To simplify the simulation, we have N=4, m=3, Į1 1, Į2 0.4, Į3 0.3. And we generate 100 random as different loads to make dispatching for 100 times. To set an example, we show the results of one experiment using Round-Robin scheduling, Firewalls are rule based access control system through process. The results obtained show that the designed system is capable of balancing the load in heterogeneous networks. In our project result show that load-balancing and k-means++ algorithm is effective in firewall cluster.

We also calculate the load range and load variance of four web servers deploying the three different scheduling policies.

## REFERENCES

[1] S. Kaur, J. Singh, K. Kumar, and N. S. Ghumman, “Round-Robin Based Load Balancing in Software Defined Networking,” 2nd International Conference on Computing for Sustainable Global Development, 2015.

[2] Balas, E., & Ragusa, A. (2014). SciPass: a 100Gbps capable secure Science DMZ using OpenFlow and Bro. In Supercomputing 2014 conference (SC14).

[3] K. Govindarajan, V. Kumar, “An Intelligent Load Balancer for Software Defined Networking (SDN) based Cloud Infrastructure” 2nd Intern. Conf. on Electrical, Computer and Comm. Tech. (ICECCT), 2017.

[4] Walber José Adriano Silva, Djamel Fawzi Hadj Sadok, "Control inbound traffic: Evolving the control plane routing system with Software Defined Networking", High Performance Switching and Routing (HPSR) 2017 IEEE 18th International Conference on, pp. 1-6, 2017, ISSN 2325-5609.

[5] K. Kaur, S. Kaur and V. Gupta, "Flow statistics based load balancing in OpenFlow," 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, 2016, pp. 378-381.

[6] N. Kang and M. Ghobadi and J. Reumann and A. Shraer and J. Rexford, “Efficient traffic splitting on commodity switches,” In Proc. 11th ACM Conference on Emerging Networking Experiments and Technologies. CoNEXT 2015.

[6] Kreutz, Diego, Fernando MV Ramos, P. Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. "Softwaredefined networking: A comprehensive survey." proceedings of the IEEE 103, no. 1 (2015): 14-76.

[7] Azodolmolky, Siamak, Reza Nejabati, Eduard Escalona, Ramanujam Jayakumar, Nikolaos Efstathiou, and Dimitra Simeonidou. "Integrated OpenFlow–GMPLS control plane: an overlay model for software defined packet over optical networks." Optics express 19, no. 26 (2011): B421-B428.

[8] Kirkpatrick, Keith. "Software-defined networking." Communications of the ACM 56, no. 9 (2013): 16-19.