Load Balancing Algorithms in Software Defined Network

Mustafa Hasan Al Bowarab, Nurul Azma Zakaria, Z.Zainal Abidin

Abstract— Compared with the traditional networks, the SDN networks have shown great advantages in many aspects, but also exist the problem of the load imbalance. If the load distribution uneven in the SDN networks, it will greatly affect the performance of network. Many SDN-based load balancing strategies have been proposed to improve the performance of the SDN networks. Therefore, in this paper a finding form comprehensive review help to improve further understanding of lead b balancing algorithms in SDN.

Index terms—Software Defined Networking; SDN; Load Balancing.

I. INTRODUCTION

The expand of the global networks which has demand new specification of requirements to integrate the internetworking systems and networking for current and future networks[1]. Nowadays, the huge information and big data bang crucial challenge of implementing the networks, which leads to find an intelligent [2], efficient and reliable network.

The current networks have different hardware equipment such as switches, routers and load balancer which quiet difficult to deal with them by traditional architecture networks [3]. To overcome these challenges, the term of Software Defined Network (SDN) is involved in the general network systems [4].

The SDN system architecture primarily comprises 3 layers (illustrated in Fig.1). These include the application layer, control layer and infrastructure layer (may also be termed the data layer). The application and control layers interact with one another via API, which is the northward interface. The control and infrastructure layers interact via the control data surface interface, which is located in the southward interface[5].

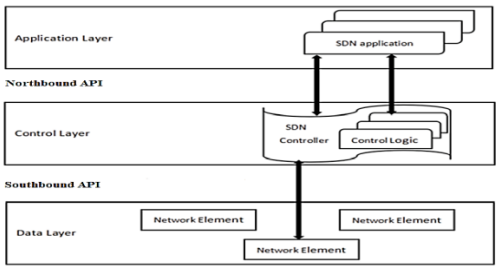

Fig.1SDN architecture

Revised Manuscript Received on December 22, 2018.

Mustafa Hasan ,Center for Advanced Computing Technology, Faculty of Information Communication Technology UniversitiTeknikal Malaysia Melaka UTeM Melaka, Malaysia

However, SDN controllers have a global view of the network and can produce more optimized load balances[6].

In this paper we introduce and review the load balance algorithms. Load balancing is a significant component of current network infrastructure and computer systems where resources are distributed over vast ranges of systems and require sharing from a populous end user-base. Load balancing seeks to utilize resources, obtain minimum response time and reduce overloads as optimally as possible by distributing the workload. Load balancing is also a basic problem in many practical systems in daily life. The supermarket model is a popular example, whereby a central balancer or dispatcher is in charge of assigning queued customers to a particular server in order to reduce the response time [7]. These Load Balancing techniques are widely observed in the data-center and enterprise network settings in order to bolster scaled-up services. Early works [8] are based on Round Robin Domain Name System (RR-DNS) to allocate inbound connections towards a group of servers. Other well-known load-balancing approaches are based on Internet Protocol (IP) level according to flow tuple [9], or according to the relative load on the different network instances [10].

Layers 4 and 7 employ further load-balancers. These load balancers are employed for numerous other network services, including acting as network proxy servers, whereby the load balancer is builds on the proxy-server's cache content. The main aim of this utilization is to raise the cache hit ratio instead of achieving an equally spread out server load balance [11].

The construction of this paper contains three sections, where the first section explained the evolution of load balancing, while the second section discuss the types of the load balancing, finally the review the algorithms of load balancing has been discussed in the third part.

II. EVOLUTION OF LOADBALANCING

In this section, we discuss the background of load balancing in the networks that is illustrated in Fig. 2.

Domain Name System (DNS) is employed to achieve the first load balancing technology. In this step, a name is configured for numerous IP addresses, so as to enable clients who query the name to receive one of the addresses. This allows various clients to have access to different servers and

thus fulfill load balancing. DNS load balancing is a fundamental tool to this aim, but may suffer from not being able to differentiate between servers and is not able to reflect the servers' current running status [12].

Due to the restriction of the DNS load balance approach, Hardware load balancer (HLD) was introduced by several manufacturers in the mid-1990s [13]. Decoupling load balance function from application enables the DNS to use network layer techniques such as Network Address Translation (NAT) [14] or Direct Server Return (DSR) [15] to send inbound and outbound traffic to the servers. Such techniques are used to process the requests and replies to the client.

Atypically, Software Load Balancing (SLB) can be implemented into Server Operating System (Server OS) such as Windows Server 2016 [16] or Red Hat's High Availability Linux Server [17]. Most of the existing solutions are focused on distributed network traffic betweenclustered servers or server farms. SLB is flexible for cloud visualization environment in which the servers have individual OSs, or share an operating system. SLB in a cluster environment that allows scaling of the networkservices where additional servers can be added dynamically to the cluster. SLB distributes the load between servers, while a server cluster provides fault tolerance in the system.

The proliferation of dynamic content led to the delivery of dynamic services, content-rich applications that need to understand the application-specific traffic. The traditional load balancer could not cope with these growing requirements. Therefore, HLB has evolved into Application Delivery Controllers (ADCs) [18] over the past ten years. Typically, in the data center, ADC is a device that sits between the firewall and a web farm to provide several tasks [11] One of these tasks is loading the traffic between web servers. ADC can inspect packet headers and distribute the traffic to a selected server based on this information.

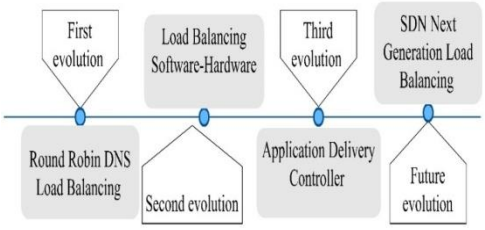

Fig.2The evolution stages of load balance

Various load balancing algorithms and methods currently exist. These include the round robin, fasted node selection, weighted round robin, IP-based hashing and multi-tier round robin methods. SDN is very flexible, enabling the installation of company-defined software based on white-box switch. It further allows the programming of current equipment to fit the network requirements and decrease costs related to deployment and management [19]. In the context of data centers, the decision of customers to apply a wide variety of load balancing algorithms depend on the server size, resource availability, flexibility and peak traffic hours. [20].

III. LOAD BALANCING ALGORITHMS

The core SDN controller mechanisms lays the foundation for the autonomous network operations. These are utilized by respective network applications to enable advanced features for network operators, network providers and users. Network applications and their elements may be deployed onto SDN controller (SC) directly. Although in practical settings, network owners restrict deployment due to security and reliability concerns. Thus, only applications which support network management and network service provisions are enabled.

Load balancing occupies an important position to solve over-load traffic problem in the network. It has been one of the first appealing applications in SDN networks. These are some commonly used load balancing approaches [21]:

- Random: This approach randomly distributes the traffic to the available paths. Generally, hash function is used to map requests to available paths.
- Round Robin: This approach distributes the request to the paths in sequence, starting from the first path to the last one in rotation continuously.
- Weighted Round Robin: This approach assigns weight for each path, then distributes requests sequentially with respect to the assigned weights.
- Least Connections: This approach forward the request to the path that has the least number of current connections.

IV. TYPES LOAD BALANCING TECHNIQUES

Load balancing methods may be classified as either static, dynamic or both [1] as shown in Fig. 3. Prior knowledge of the system is crucial to static methods, where the rule is directly programmed within the load balancer. The caveat is that user behavior cannot be forecasted and is thus not optimal for networks. On the other hand, dynamic methods avoids this problem and is more efficient as load is spread out in a dynamic manner following a pre-programmed load balancing pattern [2].

A. Static Load Balancing

In a static algorithm, equal division of traffic is done within the servers. Static algorithm is suitable for systems with low load variation. This algorithm needs to have prior information of the system resources in order to be able to make sure that decision of load shifting does not depend on current system state.

The master processor delegates the initial tasks to be executed to individual processors. The same processor is always responsible for the tasks delegated. Thus, the work load performance is determined from the onset via the master processor [23]. The slave processors compute the designated work and the results are fed to the master processor. Tasks are always carried out on the processorwhich receives the designated task. Static load balancing methods are not pre-emptive, and is used to decrease overall execution time for concurrent programs while reducing the possible delays in communicating among processors [24].

The features of Static Load Balancing methods are:

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

- It needs less communication in order to minimize the communication delays, where this reduces the execution time.
- It needs less communication in order to minimize the communication delays, where this reduces the execution time.
- Weighted algorithms achieve a better response time and processing time.
- Load balancing methods load the distribution depending on the load at the time of selecting the node before the execution starts.
- Static methods are mostly suitable for the constant work application, and for homogeneous and stable environments that can produce better results in

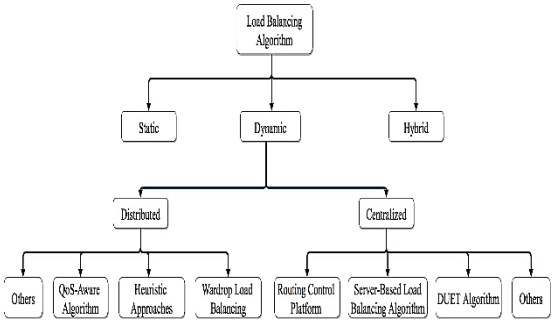

Figure 3: The Classification of Load Balancing in SDN

- within these environments.
- It is easy to be implemented.
- It is relatively simple to forecast the static methods' behavior.
- It is very difficult to forecast the loads' arrival and processing times prerequisite for future loads [25].

B. Dynamic Load Balancing

Dynamic load balancing techniques are more effective than the static counterparts due to the dynamic distribution of pre-programmed load balancer patterns [22], [26]. The proper load balancing is crucial to optimizing minimum response time, maximal throughput, minimal resource consumption, scalability and not running into any resource overloads. The Dynamic Load Balancing method can be achieved based on three ways: non-distributed, distributed, or semi-distribute methods. In the non-distributed method, there is one node (centralized) that receives all requests and distributes them to the servers. In the distributed method, all nodes are shared with the distribution of the requests. As for the semi-distributed method, the nodes are divided up into a group of clusters, where each cluster works as a central node in order to distribute the requests, and all clusters are responsible for the load balancing distribution[25].

The features of Dynamic Load Balancing methods are as follow:

- It selects the suitable node that requires real time communication with the networks, which will lead to an extra traffic to be added to a system. Dynamic methods provide better performance.
- It is difficult to be implemented.
- Dynamic methods are suitable for adaptive applications where the workload is unpredictable, or keeps changing during an execution [27].
- Dynamic methods are also mostly suitable for heterogeneous and distributed systems.
- These methods require that each node must know the states of other nodes.
- Even within nodes during execution, processes may freely migrate from one another in order to guarantee equal loads.

1) Distributed Algorithms: Distributed dynamic scheduling methods distribute load balancing over all slave nodes rather than only on the master node. Knowledge of the work load is retrieved depending in the level of demand present. These methods possess average scalability over a particular centralized scheme, but suffer from the disadvantage of being expensive to retrieve and maintain due to the dynamic nature of system information [28]. There are a number of distributed algorithms as described in the following subsections.

QoS-Aware Algorithm: SDN systems are unique in that they are programmable due to control and data plane decoupling. In particular, they offer basic and user-friendly programmable network devices instead of complicated network devices, as those found in active network protocols. SDN also attempts to separate control and data planes within the architectural design of the network, with which the network could be controlled on a separate control plane and not impact data flows. Therefore, network intelligence could be removed from switching devices and implemented within controllers. Subsequently, switching devices could be manipulated by software without built-in intelligence. Control plane decoupling within the data plane seeks to both simplify the programmable environment as well as enable more flexibility in defining the network behavior [29],[30].

Much of the modern network applications, from media streaming, cloud services and so on, necessitate the use of predictable and steady network resources possessing intensive Quality of Service (QoS) criteria. On the other hand, OpenFlow is a software-defined network which enables flow level programmability which supports the network programming as per QoS criteria as well as network traffic conditions, all in a dynamic manner. Consequently, QoS-aware network reprogramming is crucial to traffic steering within the multi service SDN-based networks. For the particular resource reallocation, network traffic flows comprise links as per QoS criteria and traffic engineering requirements.

Thus, various problems arise hindering the role of flow level resource allocation, such as big flows and network resource partitions, burst and dynamic traffics, as well as many traffic classes operating under different criteria. Resource partitioning remains an essential side effect of the level of dynamicity of network traffic and big flows. If it is assumed that two 10 Gb/s path from node A and B exist,

each with 500 Mb/s free bandwidth, then there is 800 Mb/s rate from the first to the second node. Because of resource partitioning, flow cannot be routed appropriately, thus network bears 1 Gb/s free bandwidth capacity. To combat these negative effects, flow routing must be executed with a holistic view of network while taking into consideration its effect on all other flows. To elaborate, big flow routing requires the rerouting of a number of other flows because of resource partitioning. Therefore, it is apparent that static network configurations are not sufficient. Furthermore, big flows and dynamic network traffic necessitates the reprogramming of the network on a timely basis, which may lead to instability and undesirable impact on QoS of flows. Within SDN networks, network features comprise the main problem of network reconfiguration with minimum side-effect and overhead. It is important to note that network reconfiguration overhead depends on the number of rerouted flows. Raising this number may lead to network instability as well as heightened packet loss and end-to-end delay [3].

a) *Heuristic Approaches:* are utilized to achieve optimal hybrid routing configurations. Studies have found that hybrid routing achieves optimal load balancing in comparison to pure explicit routing. On the other hand, latency, overhead and method utilization may not be taken into account [6] [32] [33]. These studies have compared among various heuristic approaches in order to heighten the resilience of software-defined networks in order to oppose connection failure among nodes and controllers. Attempts have been made to maximize controller placement reliability [34]. A minimum number of controllers are heuristically searched and assigned to individual nodes in conjunction with proper node placement, with the ultimate aim of achieving a particular threshold reliability. Many studies have focused on resilience opposing network failures but do not take into account the further metrics such as πimbalance or πmax latency [34], [35]. More specifically, trade-offs among metrics and objectives, such as πmax latency is also not discussed. As opposed to evaluating entire solution spaces, no guarantee is made for optimizing the results obtained in the study [36].

Heuristic-based mechanisms have illustrated novel optimization methods. Studies have attempted to implement such optimization methods for load balancing problems. These methods range from honey bee swarm algorithm [37], lion optimization algorithm [38], whale optimization algorithm [39], gray wolf optimization algorithm [40], bat optimization algorithm [41].

b) *Wardrop Load Balancing:* The Wardop Load-Balancing algorithm is employed to converge arbitrarily small neighborhoods of particular equilibrium for loads within providers. Due to its features, the algorithm is feasible for various SDN scenarios, where service requests originate from network nodes and controlled by SDN controllers.

Various load-balancing methods have been studied previously by researchers [42]. It has been recommended to categorize load-balancing algorithms as either global-based, cooperative-based or non-cooperative based approaches. Global algorithms entail that individual nodes transmit current status to centralized load balancers via an extensive and cohesive system network. Thus, jobs are delegated to each resource which also optimizes a particular objective, such as the response time of the whole system over all jobs. This method has been popularly used in conjunction with methods such as nonlinear optimization, until being outclassed by the other two aforementioned methods. Cooperative algorithms utilize various decision makers which agree on decisions cooperatively such that each operates optimally individually. On the other hand, non-cooperative algorithms utilize multiple decision makers optimizing individual response times regardless of the status of others, and is thus not cooperating with others. In these cases, Nash equilibrium condition is achieved when no decision makers are able to obtain further benefits by altering its own decisions unanimously. To specify, the network stability in these cases are studied in terms of achieving load distribution so that individual jobs are able to switch between nodes with lower number of jobs.

Load-balancing algorithms may be classified as either static or dynamic. The former relies on the currently possessed knowledge for application load, whereas the latter relies on settings where information pertaining to load distribution is unknown.

Many studies have analyzed load balancing, with a notable studies by [43] studying centralized static cooperative load balancing; [44] and [45] studying centralized static non-cooperative load balancing; [46] and [47] studying centralized dynamic load balancing; and [48] addressing the issue of distributed dynamic load balancing relying upon local cooperation among neighboring network nodes.

One study analyzed the non-cooperative dynamic load-balancing method, which is popularly implemented in game theory frameworks with problems consisting of dynamic load-balancing game where users distribute loads in non-cooperatively and selfishly [49]. The study also considered the renowned game theory traffic model proposed by Wardrop [50], which sought to describe road traffic with infinitely numbered agents, each in charge of a very small amount of traffic. The framework considers flow demand to be routed from a given source to a destination by utilizing various paths. The agents are able to distribute its own flow within admissible paths. The network bears the trait of non-decreasing latency functions which are based on edge flows. Multiple flows ensure that latencies of employed paths are as low as possible, which is called the Wardrop equilibrium. Therefore, the Nash equilibrium is dubbed a Wardrop equilibrium when there exists an infinite number of decision makers [51].

2) Centralized Algorithms: In this approach, the concern of the load-balancing decision remains with the master node and the data used for the load balancing is gathered from the remaining (slave's) nodes on either on demand basis or after a certain predefined time interval. The obvious point is since the data is not sent arbitrarily; the unnecessary traffic over the network is minimized. But, the scalability remains limited with this technique [28] [29]. There are a number of centralized algorithms as described in the following subsections.

a) *Routing Control Platform (RCP):* Roughly a decade earlier, numerous studies have focused on the Routing Control Platforms (RCP) [52-56]. These studies were the first to refactor IP routing architecture and made a

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

logically centralized control plane distinct from forwarding elements and more towards the BGP decision processes and route control requirements given by a large operator.

At its core, RCPs comprise three architectural principles: path computations based on a consistent network state, expressively specified routing policies and controlled communication between routing protocol layers [52]. Compared to BGP configurations distributed complexly over various routers, these methods allow individual AS to rapidly utilize novel, customer-facing services [54].

In the RCP architecture, legacy network routing comprises iBGP Route Reflector (RR), which obtains network information and switches it with RCP. Network information is retrieved consistently by retrieving eBGP learned routes. Therefore, the retrieved information is directly distributed over border routers possessing full route access. Consequently, the optimal routes are calculated depending on the route learned from the eBGP. The entire routing configuration and states are contained within the RCP control plane. For proper handling, RCP maintains local registries with global views and information exchanges with external RCP for routing between domains. Utilizing the RCP appropriately within the SDN is referred to as 'Intelligent Route Service'. Control Point (IRSCP) is an architecture mainly employed prior to the advent of SDN [57].

The current RouteFlow Control Platform (RFCP) architecture has advanced from prior prototypes [58] to more sophisticatedly-layered, distributed system design and possessing ample flexibility to be employed within diverse virtualization cases. This method is based on a modular architecture comprising three major components, as show in Fig. 4. These include the RF-Slave, RF-Server and RF-Controller. The RF-Slave collects routing and forwards information from the Linux host. Alternatively, in order to extract complementary routing information, it utilizes peers such as iBGPs to hook using a routing engine. Next, the RF-Server is a standalone application in charge of the core system logic, including event processing, VM-to-switch mapping, and resource management. RFCP Services are also utilized as operator-tailored modules utilizing a knowledgeable information base to carry out arbitrary high-level routing logic. Lastly, the RF-Controller is an application above the OpenFlow Controller serving RFCP via switch interaction and the topology sate collection of the network [59].

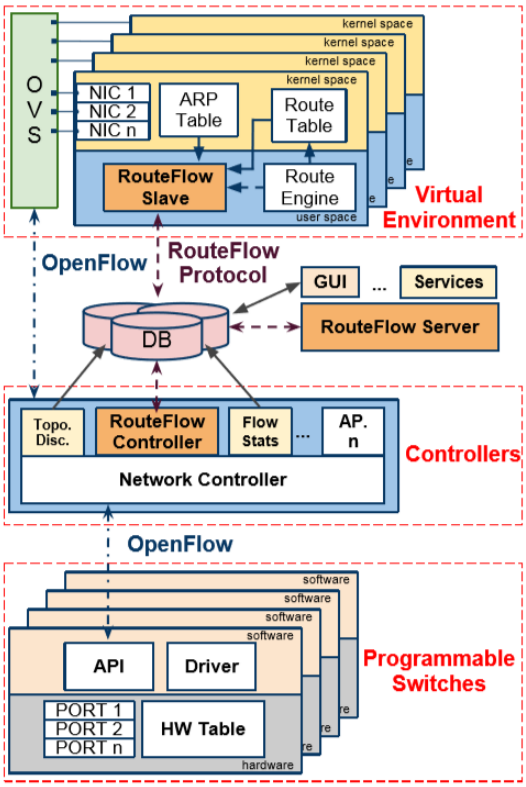

Fig. 4 Route Flow Control Platform (Architecture design) [57].

b) *Server-Based Load Balancing Algorithm (SBLA):*The SBLA load balancing algorithm was proposed in [60] and it was suitable for server - cluster in virtual environment Firstly, the controller used the SNMP protocol to collect the state information of the servers, then calculated the load of the serve is according to the SBLA algorithm, and finally selected the lightest load server to respond to the users. The algorithm minimized the server's response time, but was unsuitable for unstructured networks and data centers.

c) *DUTE Algorithm:* Gandhi et al. proposed a DUTE scheme which combined the hardware with the software [61]. The scheme used the existing switches to build a hardware load balancer to effectively increase the capacity, reduce the cost and delay, but the flexibility was poor. Especially, when the switch failed, the disadvantage was especially obvious. The hardware load balancer dealt with a large number of traffic, while the software load balancer served as a backup to ensure high availability and Gandhi et al proposed a DUTE scheme which combined flexibility, but it was difficult to implement.

C. Hybrid Load Balancing

These methods are achieved and employed to get rid of the disadvantages associated with Dynamic and Static Load Balancing methods, and they are being used to aggregate the benefits and merits of static and dynamic algorithms in order to design a new one [22]. In fact, this implies that combinations the benefits of two or more existed algorithms either dynamic or static algorithms are able to present a new one.

The features of the Hybrid Load Balancing methods are:

- The word “data” is plural, not singular.
- The main disadvantage lies in its incapacity to enable noncomplex methods.
- Hybrid methods take over the attributes of both static and dynamic LB techniques, seeking to overcome the drawbacks of both methods. They are more scalable.

V. EVALUATION OF LOAD BALANCING ALGORITHMS

This paper classifies the load balancing researches in SDN networks, as show in Fig. 3. The SDN architectures can be divided into centralized single controller architectures and distributed multiple controller’s architectures according to the number and organization of the controllers in SDN networks, in the centralized architectures load balancing researches are divided into the data plane and the control plane. The data plane mainly includes link load balancing and server load balancing. The distributed architectures are divided into the flat architecture and the hierarchical architecture. This paper introduces, analyzes and summarizes types of load balancing researches, so that researchers can quickly understand the relevant knowledge in this field.

VI. CONCLUSION

Software Defined Network has been developed to manage large networks like data center big data. Due to huge expanding of internet, enormous number of request is arriving at server per second. In this paper a detailed survey of load balancing algorithms has been done. The classifications of load balancing have been divided to three types of static, dynamic and hybrid load balancing. There is requirement of efficient algorithm to balance the load of server to avoid network degradation. The centralized controller of SDN has the global view of network which makes load balancing in SDN easy. The load balancing algorithm must consider the current load to reflect the real time change. Using single centralized controller can lead to single point of failure. So load balancing algorithm should be mainly based on distributed decision. Researchers should do more detailed study of distributed architecture to develop better load balancing algorithms taking advantage of SDN architecture. The algorithm should be designed in such a way that it minimizes the latency and response time and maximize the throughput.

ACKNOWLEDGMENT

The researchers would like to thank the university technical Malaysia Melaka (UTeM) for sponsoring this work.

REFERENCES

1. P. Martinez-julia and A. F. Skarmeta, “Empowering the Internet of Things with Software Defined Networking,” White Pap. IoT6-FP7 Eur. Res. Proj., 2014.
2. M. Nasser et al., “Cyber-Security Incidents: A Review Cases in Cyber-Physical Systems,” Int. J. Adv. Comput. Sci. Appl., vol. 9, no. 1, 2018.
3. P. Kumari and D. Thakur, “Load Balancing in Software Defined Network,” Int. J. Comput. Sci. Eng., vol. 5, no. 12, pp. 227–232, 2017.
4. J. Li, E. Altman, and C. Touati, “A General SDN-based IoT Framework with NVF Implementation,” ZTE Commun., vol. 13, no. 3, pp. 42–45, 2015.
5. X. You and Y. Wu, “Software Defined Network Architecture based Research on Load Balancing Strategy,” in AIP Conference Proceedings, 2018, vol. 1967.
6. A. A. Neghabi, N. J. Navimipour, M. Hosseinzadeh, and A. Rezaee, “Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature,” IEEE Access, vol. 6, pp. 14159–14178, 2018.
7. S. Islam, “Network Load Balancing Methods : Experimental Comparisons and Improvement,” arXiv Prepr. arXiv1710.06957., 2017.
8. E. D. Katz, M. Butler, and R. McGrath, “A scalable HTTP server: The NCSA prototype,” Comput. Networks ISDN Syst., vol. 27, no. 2, pp. 155–164, 1994.
9. A. Bestavros, M. Crovella, J. Liu, and D. Martin, “Distributed Packet Rewriting and its Application to Scalable Server Architectures,” in Proceedings Sixth International Conference on Network Protocols (Cat. No.98TB100256), 1998, pp. 290–297.
10. L. Aversa and A. Bestavros, “Load Balancing a Cluster of Web Servers Using Distributed Packet Rewriting Luis,” in Conference Proceedings of the 2000 IEEE International Performance, Computing, and Communications Conference (Cat. No.00CH37086), 2000, pp. 24–29.
11. E. Doron and M. Sekiguchi, “Techniques for providing scalable application delivery controller services,” US9386085B2, Apr-2016.
12. W. Chen, H. Li, Q. Ma, and Z. Shang, “Design and implementation of server cluster dynamic load balancing in virtualization environment based on OpenFlow,” in International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013), 2014, pp. 691–697.
13. J. Ju, G. Xu, and K. Yang, “An Intelligent Dynamic Load Balancer for Workstation Clusters,” ACM SIGOPS Oper. Syst. Rev., vol. 29, no. 1, pp. 7–16, 1995.
14. P. Srisuresh and K. Egevang, “Traditional IP Network Address Translator (Traditional NAT),” RFC 1631, pp. 1–16, 2000.
15. C. Kopparapu, Load Balancing Servers, Firewalls, and Caches. 2002.
16. J. H. Huh and K. Seo, “Design and test bed experiments of server operation system using virtualization technology,” Human-centric Comput. Inf. Sci., pp. 1–21, 2016.
17. S. Cash, A. Karve, T. Mathews, S. Mullen, and M. Mulsow, “Managed infrastructure with IBM Cloud OpenStack Services,” vol. 60, no. 2, pp. 1–12, 2016.
18. K. Salchow Jr, “Load Balancing 101 : The Evolution to Application Delivery Controllers,” 2007.
19. S. Khan, M. Ali, N. Sher, Y. Asim, W. Naeem, and M. Kamran, “Software-Defined Networks (SDNs) and Internet of Things (IoTs): A Qualitative Prediction for 2020,” Int. J. Adv. Comput. Sci. Appl., vol. 7, no. 11, pp. 385–404, 2016.
20. D. Mithbavkar, H. Joshi, H. Kotak, D. Gajjar, and L. Perigo, “Round Robin Load Balancer using Software Defined Networking (SDN),” Capstone Team Research Project. pp. 1–9, 2016.
21. M. Alanyali and B. Hajek, “Analysis of simple algorithms for dynamic load balancing,” 1997.
22. A. S. Milani and N. Jafari, “Load balancing mechanisms and techniques in the cloud environments : Systematic literature review and future trends,” J. Netw. Comput. Appl., vol. 71, pp. 86–98, 2016.
23. H. Kameda, E. S. Fathyt, I. Ryut, and J. Lis, “A Performance Comparison of Dynamic vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network,” in Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187), 2000, pp. 1415–1420.
24. I. Journal and F. Technological, “Comparison of Static and Dynamic Load Balancing in Grid Computing,” Int. J. Technol. Res. Eng., vol. 2, no. 7, pp. 1337–1340, 2015.
25. S. Hamadah, “A Survey : A Comprehensive Study of Static , Dynamic and Hybrid Load Balancing Algorithms,” Int. J. Comput. Sci. Inf. Technol. Secur. (IJCSITS), vol. 7, no. 2, pp. 27–32, 2017.
26. S. Chen, Y. Chen, and S. Kuo, “CLB: A novel load balancing architecture and algorithm for cloud services,” Comput. Electr. Eng., vol. 58, pp. 154–160, 2017.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

27. I. Publication, “A comparative study of static and dynamic Load Balancing Algorithms,” IJARCSMS, vol. 2, no. 12, pp. 386–392, 2014.
28. B. Kang and H. Choo, “An SDN-enhanced load-balancing technique in the cloud system,” J. Supercomput., pp. 1–24, 2016.
29. A. A. Hassan, W. Shah, M. F. Iskandar, M. S. Talib, and A. Abdul-jabbar, “K Nearest Neighbor Joins and MapReduce process enforcement for the cluster of data sets in BigData " Jour of Adv Research in Dynamical & Control Systems,vol. 10, pp. 690-695, 2018
30. W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, “Survey on software-defined networking,” IEEE Commun. Surv. Tutorials, vol. 17, no. 1, pp. 27–51, 2015.
31. M. M. Tajiki, B. Akbari, and N. Mokari, “Optimal Qos-aware network reconfiguration in software defined cloud data centers,” Comput. Networks, vol. 120, pp. 71–86, 2017.
32. Y. N. Hu, W. D. Wang, X. Y. Gong, X. R. Que, and S. D. Cheng, “On the placement of controllers in software-defined networks,” J. China Univ. Posts Telecommun., vol. 19, pp. 92–97, 2012.
33. H. Yannan, W. Wendong, G. Xiangyang, Q. Xirong, and C. Shiduan, “Reliability-aware Controller Placement for Software-Defined Networks,” EEE Int. Symp. Integr. Netw. Manag., no. February, pp. 672–675, 2013.
34. F. J. Ros and P. M. Ruiz, “Five nines of southbound reliability in software-defined networks,” Proc. third Work. Hot Top. Softw. Defin. Netw. - HotSDN ’14, no. May, pp. 31–36, 2014.
35. Y. Zhang, N. Beheshti, and M. Tatipamula, “On resilience of split-architecture networks,” in IEEE Global Telecommunications Conference - GLOBECOM 2011, 2011.
36. S. Lange et al., “Heuristic Approaches to the Controller Placement Problem in Large Scale SDN Networks,” IEEE Trans. Netw. Serv. Manag., vol. 12, no. 1, pp. 4–17, 2015.
37. D. Karaboga, “An idea based on honey bee swarm for numerical optimization,” 2005.
38. M. Yazdani and F. Jolai, “Lion Optimization Algorithm (LOA): A nature-inspired metaheuristic algorithm,” J. Comput. Des. Eng., vol. 3, no. 1, pp. 24–36, 2016.
39. S. Mirjalili and A. Lewis, “The Whale Optimization Algorithm,” Adv. Eng. Softw., vol. 95, pp. 51–67, 2016.
40. S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey Wolf Optimizer,” Adv. Eng. Softw., vol. 69, pp. 46–61, 2014.
41. X. Yang and A. H. Gandomi, “Bat Algorithm : A Novel Approach for Global Engineering Optimization,” Eng. Comput., vol. 29, no. 5, pp. 464--483, 2012.
42. D. Grosu, A. . Chronopoulos, and M. . Leung, “Cooperative load balancing in distributed systems,” Concurr. Comput. Pract. Exp., vol. 20, no. 16, pp. 1953–1976, 2008.
43. S. U. Khan and I. Ahmad, “A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids,” IEEE Trans. Parallel Distrib. Syst., vol. 20, no. 3, pp. 346–360, 2009.
44. D. Grosu and A. T. Chronopoulos, “Noncooperative load balancing in distributed systems,” J. Parallel Distrib. Comput., vol. 65, no. 9, pp. 1022–1034, 2005.
45. R. Subrata, A. Y. Zomaya, and B. Landfeldt, “Game-theoretic approach for load balancing in computational grids,” IEEE Trans. Parallel Distrib. Syst., vol. 19, no. 1, pp. 66–76, 2008.
46. E. Even-Dar, A. Kesselman, and Y. Mansour, “Convergence Time to Nash Equilibria,” Int. Colloq. Autom. Lang. Program., vol. 2719, pp. 502–513,2003.
47. E. Even-Dar and Y. Mansour, “Fast Convergence of Selfish Rerouting,” Proc. Sixt. Annu. ACM-SIAM Symp. Discret. Algorithms, pp. 772–781, 2005.
48. S. Shah and R. Kothari, “Convergence of the dynamic load balancing problem to Nash equilibrium using distributed local interactions,” Inf. Sci. (Ny)., vol. 221, pp. 297–305, 2013.
49. H. Ackermann, S. Fischer, M. Hoefer, and M. Schöngens, “Distributed algorithms for QoS load balancing,” Distrib. Comput., vol. 23, no. 5–6, pp. 321–330, 2011.
50. J. Wardrop, “Some theoretical aspects of road traffic research,” in trid.trb.org, 1952, vol. 1, no. 3, pp. 325–362.
51. H. Kameda, J. Li, C. Kim, and Y. Zhang, Optimal load balancing in distributed computer systems. 1997.
52. R. Ramjee et al., “Separating Control Software from Routers,” 2006 1st Int. Conf. Commun. Syst. Softw. Middlew., pp. 1–10, 2006.
53. N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, “The case for separating routing from routers,” Proc. ACM SIGCOMM Work. Futur. Dir. Netw. Archit. - FDNA ’04, pp. 5–12, 2004.
54. J. Van der Merwe et al., “Dynamic connectivity management with an intelligent route service control point,” Proc. 2006 SIGCOMM Work. Internet Netw. Manag. - INM ’06, pp. 29–34, 2006.
55. Y. Wang, I. Avramopoulos, and J. Rexford, “Design for configurability: Rethinking interdomain routing policies from the ground up,” IEEE J. Sel. Areas Commun., vol. 27, no. 3, pp. 336–348, 2009.
56. T. D. Mohammed AA, Burhanuddin MA, Basiron H, “Key enablers of IoT strategies in the context of smart city innovation.,” *Adv. Res. Dyn. Control Syst*, vol. 10, no. 4, 2018.
57. P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. E. van der Merwe, “Wresting control from BGP: scalable fine-grained route control,” 2007 USENIX Annu. Tech. Conf. Proc. USENIX Annu. Tech. Conf., pp. 295–308, 2007.
58. M. R. Nascimento, C. E. Rothenberg, M. R. Salvador, C. N. A. Corrêa, S. C. de Lucena, and M. F. Magalhães, “Virtual Routers as a Service: The RouteFlow Approach Leveraging Software-Defined Networks,” Proc. 6th Int. Conf. Futur. Internet Technol. - CFI ’11, vol. 1, pp. 34–37, 2011.
59. C. Rothenberg et al., “Revisiting IP Routing Control Platforms with OpenFlow-based Software-Defined Networks,” Futur. Internet Exp. Res. Work., 2012.
60. W. Chen, Z. Shang, X. Tian, and H. Li, “Dynamic Server Cluster Load Balancing in Virtualization Environment with OpenFlow,” Int. J. Distrib. Sens. Networks, vol. 11, no. 7, pp. 1–9, 2015.
61. R. Gandhi et al., “Duet : cloud scale load balancing with hardware and software,” ACM SIGCOMM Comput. Commun. Rev., vol. 44, no. 4, pp. 27–38, 2014.

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication