SSRG International Journal of Electronics and Communication Engineering
ISSN: 2348-8549/ https://doi.org/10.14445/23488549/IJECE-V10I8P111

Volume 10 Issue 8, 108-117, August 2023
© 2023 Seventh Sense Research Group®

Original Article

A SDN-Based Load Balancing Algorithm for IoT Traffic Data and Network Performance Evaluation

V. Tirupathi[1], K. Sagar[2]

[1]*Computer Science and Engineering, UCE, Osmania University, Telangana, India.*
[2]*Computer Science and Engineering, Sreyas Institute of Engineering and Technology, Telangana, India.*

[1]*Corresponding Author : thirulu629@gmail.com*

Received: 18 June 2023 Revised: 19 July 2023 Accepted: 16 August 2023 Published: 31 August 2023

Abstract *- The Internet of Things (IoT) is constantly evolving as more people utilize internet-connected devices in daily life. As this trend continues, network traffic increases, and communication quality requirements, especially IoT devices and application QoS, become difficult to achieve. This research introduces an SDN-based load-balancing technique to address these issues. The approach uses an upgraded Floyd-Warshall algorithm to optimize MQTT client device network connectivity. The suggested sparse graph-specific Floyd-Warshall algorithm is simple but effective. Optimization is essential for successfully managing IoT networks and minimizing the need for more complicated and resource-intensive solutions. The solution automates network configuration with a centralized controller using SDN. This method provides programmability, network visibility, and real-time resource allocation based on dynamic network information. SDN simplifies load-balancing algorithm implementation, improving its effectiveness. This study emphasizes the need for SDN in managing IoT traffic volume. The study focuses on MQTT broker throughput, packet loss, and communication delays to improve network performance. The authors used Mininet, a widespread network emulation tool, to test the method's efficacy. The simulation results show that the suggested strategy accomplishes efficient network load balancing and considerably improves IoT device performance. The research emphasizes the importance of solving the issues IoT device adoption faces. The paper introduces an SDN-based Load Balancing Algorithm (LBA) to increase IoT network communication quality and efficiency. The authors demonstrate that their approach improves network load balancing and device performance through simulations. This research advances IoT technology and its smooth integration into daily life by enabling more resilient and scalable IoT systems.*

Keywords *- Internet of Things, MQTT brokers, SDN, Network performance, Mininet.*

1. Introduction

The fast development of internet networks has increased the number of website service users. This increase in demand can challenge service providers, especially as clients grow. The growing use of IoT devices makes network traffic management and service requirements difficult [1]. IoT devices differ in processing power, storage capacity, energy resources, and functionality, complicating QoS, resource allocation, and load-balancing network architecture. Optimizing resource utilization in IoT networks requires Load Balancing (LB) to allocate resources to user tasks. The system ensures equilibrium among the processing capabilities, storage capacities, energy consumption, and network resources, such as bandwidth, load balancers, and traffic analyzers, of the nodes [2]. The use of an effective Load Balancing (LB) technique has the potential to mitigate overload issues and enhance Quality of Service (QoS) parameters in large-scale Internet of Things (IoT) networks. These QoS parameters include several aspects, such as response time, cost, throughput, performance, and resource utilization. The presence of a network traffic imbalance may lead to increased latency, packet loss, and a decrease in the packet delivery ratio. Message brokers allow devices and servers to communicate using multiple protocols in IoT. MQTT [3] is a popular protocol due to its minimal power consumption and overhead. MQTT exchanges messages between IoT devices using a publish-subscribe message broker. Large-scale IoT systems are challenging to install due to IoT devices' stiffness and limited dependability.

Multiple queries from different customers cause load-balancing problems. Various solutions, including static and dynamic load balancing, have been presented. However, the traditional Internet-distributed administration needs more global knowledge, preventing a holistic solution. Brokers' filtering operations in publish/subscribe systems cause delays due to detours and processing times for matching events against filter rules [4]. Traditional management paradigms must adapt to IoT's heterogeneous, large-scale nature as networks become more complicated. Scalable and effective network management is essential to support many heterogeneous IoT devices and traffic while maintaining QoS [5].

SDN divides the control and data planes, enabling high-level abstraction and virtualized network functionalities that ease IoT management. SDN improves network traffic response by controlling traffic routing and configuration. Network administrators can govern IoT traffic by merging IoT frameworks with SDN [6, 7]. This integration lets the network dynamically configure and manage IoT traffic to meet IoT devices' needs. Figure 1 shows a typical IoT-SDN integration architecture that improves network speed and resource utilization in complicated IoT scenarios.

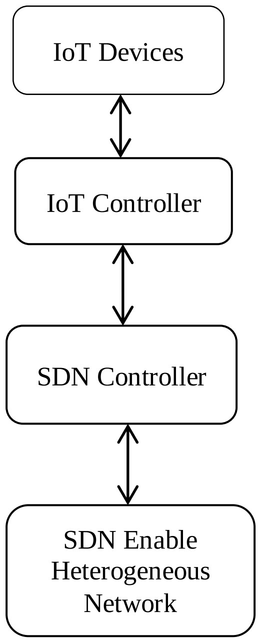

Fig. 1 IoT architecture with SDN: a summary

Software Defined Networking (SDN) revolutionizes networking by separating network switch control and data planes. The SDN controller controls data plane traffic by setting forwarding rules. SDN's flexibility and central control make it perfect for quickly adopting new IoT protocols and regulations. SDN optimizes resource allocation and optimization by managing devices and networks using real-time analytics. Container services can be deployed on an SDN switch as a network computing resource, enabling IoT application integration. IoT environments are unpredictable; unexpected events can lead to a surge in data volume in IoT networks. In order to address these difficulties, Internet of Things (IoT) protocols [8] need the ability to respond in real time, minimize bandwidth use, and optimize energy efficiency. MQTT, also known as Message Queuing Telemetry Transport, is considered a highly suitable option for Internet of Things (IoT) applications. The present study introduces a publish-subscribe messaging system designed specifically for ubiquitous networks to optimize bandwidth utilization and improve device battery longevity. The data is organized into topics, enabling publishers to send messages to a centralized broker, efficiently distributing data to topic subscribers [9, 10]. This approach allows seamless communication among constrained IoT devices while ensuring efficient data delivery. The versatility and practicality of MQTT have been demonstrated in its successful application as a communications protocol for IoT-based smart cities [11, 12].

The article's structure follows: Section 2 provides an overview of SDN-based load balancers, discussing existing techniques and their effectiveness. In Section 3, we present our load-balancing approach, which is based on broker client counts. This technique automatically distributes the load to enhance the overall performance of the IoT network. Section 4 covers the experimental settings, performance metrics, and testing procedures used to evaluate the effectiveness of our proposed load-balancing technique, and outcomes are examined to assess the proposed load-balancing algorithm. Section 5 summarizes the findings and emphasizes the importance of the proposed approach in IoT load balancing.

2. Related work

Current load-balancing research in publish/subscribe systems can be divided into two categories: (1) optimizing routing in topic-based publish/subscribe systems and (2) utilizing SDN-based publish/subscribe systems. Both of these themes are further broken into several subcategories.

In [13], the authors demonstrated a cross-layer QoS-enabled publish/subscribe communication architecture with SDN-like features. The primary purpose was to establish a transparent connection between IoT services and SDN networks, increasing the Quality of Service (QoS) for event delivery. They prioritized subjects, created a cross-layer quality of service management system, and presented SDN-inspired middleware architecture. The architecture handled event routing, subject management, and policy management. It also kept the topology. They used various service models and cross-layer access control case studies to validate their methods. The results of the experiments demonstrated the efficiency of their middleware.

The authors introduced a Software-Defined Networking SDN-enabled Publish/Subscribe system named SDNPS in their work [14]. The objective of this approach was to effectively and succinctly convey information about occurrences. To do this, the system constructed and refined overlays specifically linked to the issue using a comprehensive global topology overview. SDNPS makes it easy to filter and forward events directly on SDN-configurable switches using a hierarchical arrangement of topics called a "topic tree" similar to Huffman encoding and turns topics into binary strings. This hierarchical organization simplified overlays' progressive creation and storage, reducing the routing algorithm's complexity. The method was successful in achieving a balance between optimizing overall load distribution and lowering forwarding costs as much as feasible for each subject overlay.

In [15], the authors presented PSIoT-SDN, a QoS-aware design for publish/subscribe systems. The framework orchestrated Internet of Things traffic and facilitated network resource allocation between pub/sub-consumers and IoT data aggregators. PSIoT combined edge-level QoS control at IoT aggregators with network-level QoS control via SDN features and a bandwidth allocation model to achieve more effective IoT traffic management across the network. The SDN connection enabled dynamic responses to bandwidth sharing, which the SDN controller facilitated. As a result, bandwidth distribution improved, and network utilization for Internet of Things traffic increased.

In [16], the authors presented the "Grey Wolf Optimization Affinity Propagation" (GWOAP) technique to address intelligent load balancing in SDN-IoT-enabled smart city networks. This method entailed using sophisticated cluster-based load balancing across a variety of controllers. GWOAP outperformed other optimization algorithms, such as the Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Affinity Propagation (AP), to lower overall communication costs. The proposed technique efficiently distributed IoT traffic load between controllers in smart city networks [17].

The authors of reference [18] used a different method in their study to create a good slave controller allocation-based load balancing mechanism for a Software-Defined Networking enabled Internet of Things (IoT) network with multiple domains. The primary objective was to expeditiously transfer switches to controllers with untapped resources. The research endeavour used the Multi-Criteria Decision-Making (MCDM) methodology, especially the Analytical Network Process (ANP), to enhance service statistics and communication metrics throughout the process of choosing a target controller from a group of subordinate controllers. To get the most out of the slave controllers, the switch migration was described as a knapsack 0/1 issue. The results in an emulation environment demonstrated the efficacy of the ESCALB technique, and the proposed scheme allowed for flexible decision-making when picking controllers with different resources.

3. Materials and Methods

Some technologies can connect IoT devices, but the physical level is the best, providing high throughput and decreasing energy consumption. Mobility is a major Wi-Fi implementation difficulty. We must always connect most mobile devices to their desired services; switching access points can disrupt them. The RSSI is used to associate Broker Points (BPs) and nodes in typical Wi-Fi networks, which might cause access points to be overloaded. This step significantly affects Wi-Fi network performance.

SDN provides programmability and a versatile platform for protocol developers to implement load-balancing schemes. A significant advantage of SDN is that it eliminates the need for specialized hardware dedicated to load balancing. Instead, leveraging the OpenFlow protocol, an SDN controller takes charge of data path control decisions and incorporates an intrinsic load balancing mechanism [19, 20]. This method enables the SDN controller to optimize the utilization of network resources, dynamically mitigate network overhead, and achieve minimum reaction times by efficiently dividing the workload among IoT devices. One notable advantage of an SDN controller is its capacity to collect data from both higher levels, including networking protocols and lower tiers or sublayers. This dynamic information and real-time network infrastructure conditions allow the SDN controller to make intelligent routing decisions, improving load balancing. Network link status, average packet retransmissions, and router congestion data improve load-balancing decisions.

SDN controllers benefit from a holistic view of network structure and parameters. This thorough understanding lets the controller dynamically evaluate all possible data pathways between each pair of network nodes. The SDN controller monitors routes, traffic, and load by updating global network topological information. We made intelligent routing decisions using this abundance of information to efficiently route traffic through the network, maintain load balance, and optimize network performance.

3.1. Proposed SDN Model

The proposed method involves the SDN controller gathering essential data, such as BP (Broker Point) capacity, current load, distances between BPs, and BP device lists. This data enables the controller to make optimal decisions when establishing new relationships. As a mobile IoT device enters an overlapping region, it initiates the exchange of probe request messages, and the targeted BP forwards information about the linked clients of the SDN controller. Based on this acquired information, the controller decides whether to approve or reject the associated devices' requests to connect to that specific BP. In cases where the targeted BP is overloaded, the controller takes measures to switch the devices to neighboring BPs with lighter loads. OpenFlow switches, which interact with the controller through exchanging OpenFlow messages, manage the mobility and communication of access points [21]. Figure 2 shows that mobile IoT client devices connect with BPs based on RSSI when they reach an overlaying zone. SDN controllers evaluate this BP's load to a threshold.

If the BP is overloaded, the controller considers moving the client to an adjacent BP with a lower load. The methodology used in this study bears resemblance to the one described in reference [22], with the exception that the transmission power of each base station is modified based on its respective load. The controller transmits a beacon frame to the initial Base Station (BP) in order to deactivate the client, subsequently modifying the Received Signal Strength Indicator (RSSI) of the other Base Stations (BPs) to enhance the signal intensity of the BP experiencing the least amount of workload.

The detached device has the capability to establish a connection with the new BP without the need for a handshake message since its MAC address is already stored in the controller's database. The strongest RSSI determines BP association for non-mobile clients in overlying regions [23]. The SDN approves association requests if the BP is lighter. A heavily loaded BP redirects the client to another BP, resulting in improved network load distribution and higher throughput.

Figure 3 shows the client, Broker Points (BPs), and SDN controller exchanging association control messages. BPs protects communication with the SDN controller to maintain confidentiality and integrity.

MQTT client is in an overlying region with 3 MQTT broker points (BP1, BP2, and BP3). The device connects to one of the three BPs (BP1) based on RSSI.

State 1: The MQTT client device receives beacon frames from all three BPs and broadcasts probe request messages to start the association procedure when it enters the overlying region.

State 2: The BPs send Packet-In messages to the controller with probe response messages containing a list of MQTT clients, association event, and their load values.

State 3: When the BP load exceeds a threshold, the SDN controller updates BP information, analyses load and association events, and executes LBA.

State 4: If BP1's load exceeds the threshold, the SDN directs beacon-config signals to dissociate the device. The controller also boosts the strength broker point, which has less-load to make it stronger.

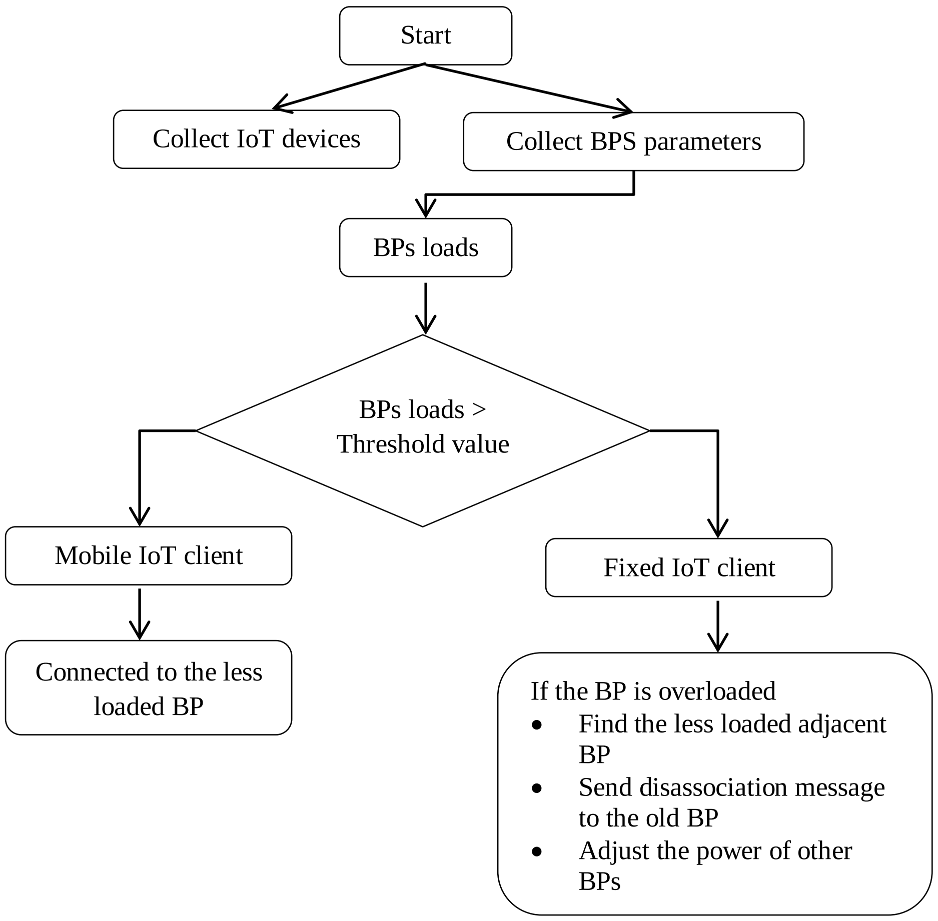
Start
Collect IoT devices
Collect BPS parameters
BPs loads
BPs loads > Threshold value
Mobile IoT client
Fixed IoT client
Connected to the less loaded BP
If the BP is overloaded
Find the less loaded adjacent BP
Send disassociation message to the old BP
Adjust the power of other BPs

Fig. 2 SDN-based load balancing algorithm

State 5: The MQTT client device receives a disassociation notification from BP1.
State 6: Repeat Step 1.
State 7: The client requests association with the highest RSSI broker point and the connection is made when the device receives the BP's association response.

The load-based signal power tuning for each BP (Step 4) is crucial to this approach. This step is critical for MQTT client devices in areas with more than two BPs since it speeds up device-access point association. The method configures the least loaded BPs with the maximum so that the client re-associates without wasting time to join an overloaded BP and then switch to an alternative. This optimization dramatically reduces association time and request exchange, improving MQTT communication efficiency.

3.2. Floyd Warshall Algorithm

The Floyd Warshall algorithm updates the shortest distances between all graph vertices iteratively. The approach initializes the shortest distances as the direct edge weights between vertices. The algorithm then iterates over all graph vertices.

If the shortest path through the current vertex is shorter than the existing shortest path, the method updates the shortest distances between the current vertex and the other vertices in the graph. The algorithm repeats until updates are impossible. This method employs a smart recurrence relation. We use x_{lm} to represent the initial weights on the entries of the graph's adjacency matrix, and x_{lm}^{j} to denote "shortest path weight". Here, x_{lm}^{0} corresponds to the initial x_{lm} values and the "recurrence relation" for this calculation is as follows:

$$x_{lm}^{j} = \begin{cases} p_{lm}\ if\ j = 0 & if\ j = 0 \\ min\left(x_{lm}^{j-1}, x_{lj}^{j-1} + x_{jm}^{j-1}\right) & if\ j > 0 \end{cases} \quad (1)$$

We will introduce the D^{j} responsible for calculating x_{lm}^{j}.

$$if\ \left(\left(p_{lj} + p_{jm}\right) < p_{lm}\right)\ then\ p_{lm} \leftarrow p_{lj} + p_{jm}$$

3.3. Improved Floyd-Warshall Algorithm

Finding meaningful matrix entries is easy by exploring the vertex (j)'s outgoing and incoming edges. The second loop should only inspect the vertices j's entering edges, and the third loop should only investigate its departing entries.

The recurrence relation iteration may build new paths between pairs of vertices that had no path before. Implement extra paths by adding necessary edges to incoming and outgoing edge lists and transforming ∞ matrix items to non-∞ values. Thus, subsequent iterations will examine these matrix elements' advantageous entries, correlating to some vertices.

Adjust the Floyd Warshall recurrence relation to use only the vertex j's incoming and outgoing edges at iteration j to minimize redundant relaxing attempts. In iteration $in_{\alpha}^{\beta}(out_{\alpha}^{\beta})$ represents the list of vertices on the opposite end of the incoming β (i.e., matrix D^{β}). (Outgoing) edges of vertex α.

$$x_{lm}^{j} = \begin{cases} p_{lm}\ if\ j = 0 \\ min\left(x_{lm}^{j-1}, x_{lj}^{j-1} + x_{jm}^{j-1}\right) \\ if\ j > 0 \wedge l \in\ in_{j}^{j-1} \wedge m \in\ out_{j}^{j-1} \end{cases} \quad (2)$$

$$in_{m}^{j} = \begin{cases} in_{m}^{0} \sqcup \{l\} & if\ j = 0 \wedge p_{lm} \neq \propto \wedge\ l \neq m \\ in_{m}^{j-1} \sqcup \{l\} & if\ j > 0 \wedge x_{lm}^{j-1} = \propto \wedge\ x_{lm}^{j} \neq \propto \end{cases} \quad (3)$$

$$out_{l}^{j} = \begin{cases} out_{m}^{0} \sqcup \{l\} & if\ j = 0 \wedge p_{lm} \neq \propto \wedge\ l \neq m \\ out_{m}^{j-1} \sqcup \{l\} & if\ j > 0 \wedge x_{lm}^{j-1} = \propto \wedge\ x_{lm}^{j} \neq \propto \end{cases} \quad (4)$$

3.4. Least Number of Connections-Based Load Balancing

Load balancing is crucial in efficiently managing network traffic by distributing it among multiple brokers. Based on the number of clients connected to brokers, the proposed load-balancing algorithm aims to optimize the performance and utilization of MQTT brokers within the network. Here is a detailed breakdown of the algorithm's steps:

1. Initially, the algorithm retrieves real-time MQTT broker statistics from the SDN controller. This step provides access to up-to-date information about the current load on each broker.
2. After obtaining the broker statistics, the algorithm calculates the total number of MQTT clients connected to each broker. This calculation is essential for assessing the load distribution across the different brokers.
3. Based on the client count, the algorithm identifies the broker with the highest load, which indicates the broker serving the maximum number of connected clients.
4. Simultaneously, the algorithm identifies the broker with the lowest load, signifying the broker with the fewest connected clients.
5. The algorithm then computes the load differential, representing the difference in load between the highest and lowest load brokers.
6. If the load differential surpasses a predetermined threshold, the algorithm initiates load-balancing procedures to achieve a more balanced distribution of clients across the brokers.

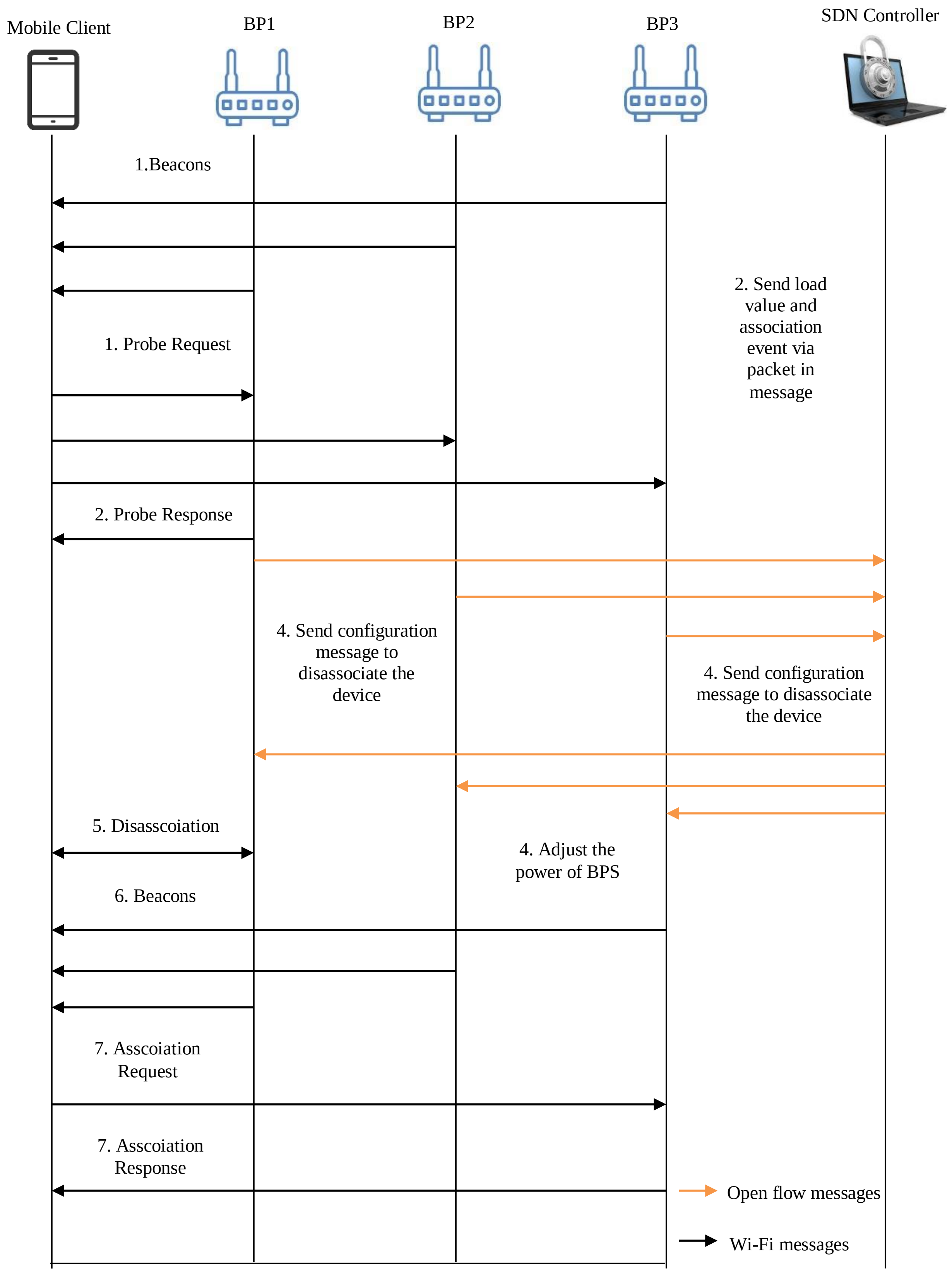
Mobile Client
BP1
BP2
BP3
SDN Controller
1.Beacons
2. Send load value and association event via packet in message
1. Probe Request
2. Probe Response
4. Send configuration message to disassociate the device
4. Send configuration message to disassociate the device
5. Disasscoiation
4. Adjust the power of BPS
6. Beacons
7. Asscoiation Request
7. Asscoiation Response
Open flow messages
Wi-Fi messages

Fig. 3 Association message flow using LBA

- Firstly, the algorithm calculates the number of clients that need to be migrated from the highest load broker to the lowest load broker to achieve a balanced load distribution with the least number of connections.
- The algorithm then retrieves the IP address of the highest load broker from the SDN controller. This IP address is crucial for the subsequent load-balancing steps.
- For each client that requires migration, the algorithm follows these steps:
    i) It calculates the shortest path from the MQTT client to the lowest load broker using the SDN controller. For this task, the algorithm efficiently utilizes the well-known Floyd-Warshall algorithm.
    ii) Once the shortest path is determined, the algorithm updates the forwarding rule in the SDN controller. This rule ensures that traffic from the MQTT client is directed to the lowest load broker along the calculated shortest path. As a result, the client's connection is efficiently redirected, effectively achieving load balancing across the brokers.

Through these steps, the proposed algorithm dynamically adjusts the distribution of clients among MQTT brokers, ensuring optimal utilization of network resources and enhancing overall performance in the IoT architecture.

4. Results and Discussion

We execute the network configuration method in this paper through simulation-based experiments using the Mininet software. Throughput is a crucial metric that indicates the successful transfer of data packets from the source to the destination within a specific time. Achieving fast packet arrival at the destination node is a key indicator of excellent network performance. By analyzing throughout, the study aims to identify the network's underlying causes of packet loss. Load balancing is vital in maximizing throughput by reducing response time intervals and minimizing data traffic congestion. The calculation for throughput can be determined mathematically using Equation 1:

$$Throughput\ T(n) = a\left(1 - \frac{1}{1+\left(\frac{1+\alpha}{1-\alpha}\right)n}\right) \qquad (5)$$

In the initial configuration, our network consists of 20 stations connected to three access points using the conventional RSSI-based method. Among these stations, four clients are associated with BP1, 10 with BP2, and 6 with BP3. These clients are actively exchanging TCP traffic. We visualize the results of the first scenario in Figure 4, clearly indicating an uneven network load distribution. Specifically, the client connected to BP2 experiences a lower throughput, averaging around 76 Mbps, compared to BP1, which achieves an average throughput of 83.5 Mbps, and BP3, with an average throughput of 79.1 Mbps. This disparity in network load suggests that the data traffic is not evenly distributed among the access points, leading to different throughput levels for other clients.

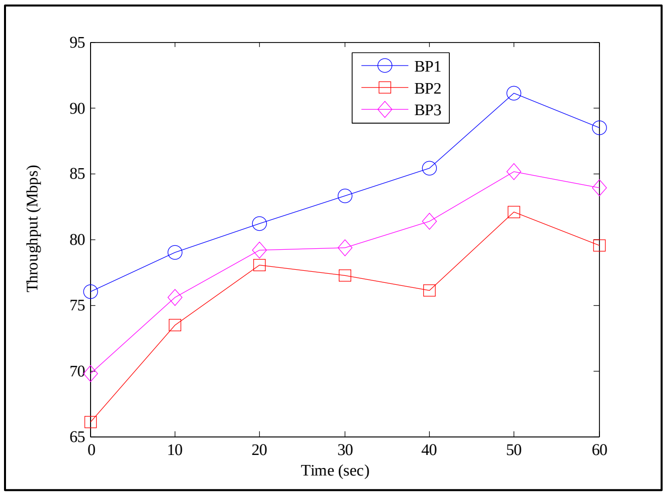

Fig. 4 TCP traffic in classic Wi-Fi network based on SDN (first scenario)

The second scenario maintains the same network topology but uses our SDN-based load-balancing technique. This implementation significantly improves network load balancing. The load-balancing algorithm distributes customers evenly throughout the three BPs. The network load is balanced, and each BP handles an equal share of client traffic. Figure 5 shows how the load-balancing method improves network performance. Each BP's throughput improves with equal client distribution. Because data flow is efficiently controlled, each BP has a more balanced and optimal workload. This balanced network load distribution improves network performance because all BPs share client traffic. The second scenario uses SDN and the load balancing algorithm to provide a more robust and efficient network design, improving throughput and client experience.

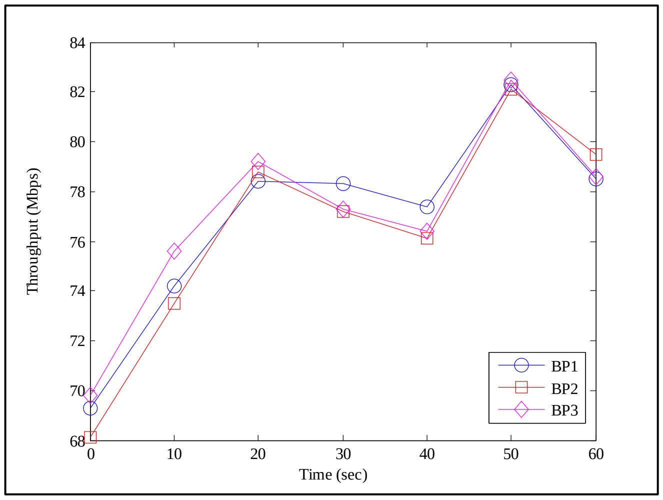

Fig. 5 SDN load balancing method TCP traffic (second scenario)

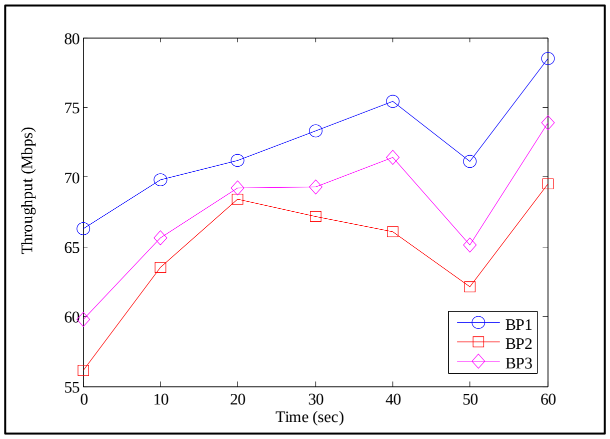

Fig. 6 SDN-based conventional Wi-Fi UDP traffic (third scenario)

Figure 6 shows that BP2 customers have lower throughput than BP1 and BP3. Figure 7 shows the three BPs with balanced loads after installing our load-balancing method.

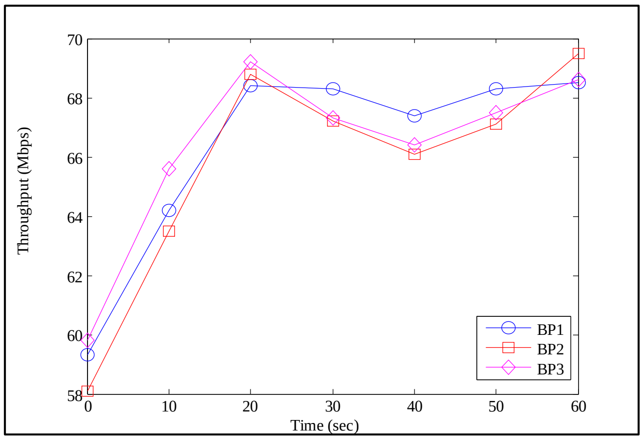

Fig. 7 UDP traffic utilizing SDN load balancing (second case)

4.1. Analysis of Response Time

We ran server load balancing tests during testing to assess web server performance. The server's response time to client queries was crucial in these tests. Lower response times indicate improved performance and client query efficiency. Clients used httperf to simulate real-world events and generate performance metrics by connecting to web servers simultaneously. The response times recorded during these tests they have revealed the load-balancing algorithm's effectiveness.

Table 1 for the round-robin load balancing algorithm and Table 2 for the least number of connection algorithms show the experiment results. Each table offers a complete set of test cases showing load-balancing scenario response times. Table 1 shows round-robin load balancing response times for different settings and client loads. The response times show how the system fared under varied client counts and workload intensities. Table 2 shows the least-connection load balancing algorithm's response time across different test situations. The least-connection approach was used to compare how the system handled client requests in this table. A comparison of response times in both tables can reveal the strengths and limitations of each load-balancing system. The tests evaluate web server performance in different scenarios, helping choose the best load-balancing method to optimize system performance.

Table 1. Round-robin algorithm response time (ms) results

Connection	Response Time (ms) with the Round-Robin Algorithm			
	Case-1	Case-2	Case-3	Case Average Results
1000/500	5.7	5.4	4.5	5.2
2000/1000	6.3	5.3	5.2	5.6
3000/1500	7.8	6.6	6.8	7.06
4000/2000	8.7	7.8	7.8	8.1
5000/2500	12.3	10.2	9.6	10.7

Table 2. Least number of connections algorithm response time (ms) results

Connection	Response Time (ms) with Least-Number Connections Algorithm			
	Case-1	Case-2	Case-3	Case Average Results
1000/500	4.7	4.3	3.8	4.26
2000/1000	5.1	4.9	4.8	4.93
3000/1500	6.3	5.8	5.7	5.93
4000/2000	7.8	6.8	6.5	7.03
5000/2500	9.1	8.7	7.8	8.53

Broker load balancing client-to-SDN controller results are shown in Tables 1 and 2. The SDN controller stages client connection requests. First example: 1000 requests at 500 connections/sec. Maintain 2000 requests at 1000 connections/sec. With 1500 connections/sec, 3000 requests continue. After that, the SDN application receives 4000 requests at 2000 connections/second and 5000 at 2500. Each situation has different outcomes. Because each case scenario is long, this condition develops.

This produces uneven request processing queues. This increases reaction time due to SDN controller service demand. Using case outcomes reaction time, compare the two load balancing approaches to determine performance. Response time comparisons between round-robin and least-connection methods are shown in Table 3 and Figure 8.

Table 3. Round-robin and least-connection average response times (ms)

Connection	Response time (ms)	
	Least-Connection Algorithm	Round-Robin
1000/500	4.26	5.2
2000/1000	4.93	5.6
3000/1500	5.93	7.06
4000/2000	7.03	8.1
5000/2500	8.53	10.7

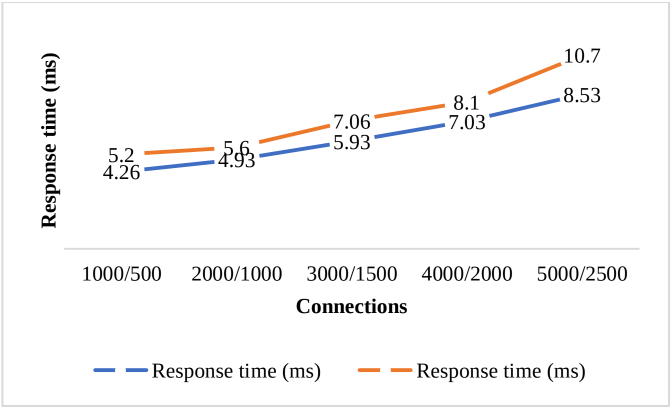

Fig. 8 Response time comparison between round-robin and least connection

5. Conclusion

This paper introduces a novel load-balancing technique utilizing SDN to enhance communication for mobile IoT client devices over Wi-Fi networks. We conducted simulations and analyzed the algorithm's performance using the Mininet to validate this approach. A key aspect of our technique is optimizing the Floyd-Warshall algorithm, eliminating unnecessary relaxation attempts and resulting in faster and more efficient computation of the network's shortest pathways. The simulations confirm the effectiveness of our strategy, as the load balancing algorithm evenly distributes network load among brokers, leading to improved client device throughput. Our architecture outperforms a typical system without load balancing.

Furthermore, we compared response times between our least-connection algorithm and Round-robin for all five connections. The results demonstrate that the least-connection algorithm responds faster than Round-robin. This study represents a significant advancement in load balancing for MQTT clients and brokers. We illustrate how combining SDN with an optimized Floyd-Warshall algorithm can enhance load distribution, network performance, and client experience. Our proposed technique can address IoT devices' dynamic and heterogeneous nature, contributing to developing more resilient and scalable IoT infrastructures.

References

[1] Harshit Gupta, and Kalicharan Sahu, "Honey Bee Behavior Based Load Balancing of Tasks in Cloud Computing," *International Journal of Science and Research*, vol. 3, no. 6, pp. 842–846, 2014. [Google Scholar] [Publisher Link]

[2] G. Ravikumar et al., "Cloud Host Selection using Iterative Particle-Swarm Optimization for Dynamic Container Consolidation," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 10, no. 1s, pp. 247–253, 2022. [CrossRef] [Publisher Link]

[3] Ala Al-Fuqaha et al., "Internet of Things: A Survey on Enabling Technologies, Protocols and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[4] Zhihong Yang et al., "Study and Application on the Architecture and Key Technologies for IoT," *International Conference on Multimedia Technology*, pp. 747-751, 2011. [CrossRef] [Google Scholar] [Publisher Link]

[5] M. Sri Lakshmi et al., "Minimizing the Localization Error in Wireless Sensor Networks using Multi-Objective Optimization Techniques," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 10, no. 2s, pp. 306–312, 2022. [CrossRef] [Publisher Link]

[6] C. Gulzar, and Ameena Yasmeen, "Maximum Network Lifetime with Load Balance and Connectivity by Clustering Process for Wireless Sensor Networks," *International Journal of Computer Engineering in Research Trends*, vol. 3, no. 7, pp. 375–383, 2016. [CrossRef] [Publisher Link]

[7] Ravindra Kumar Chouhan, Mithilesh Atulkar, and Naresh Kumar Nagwani, "A Distributed Attack Detection System for SDN using Stack of Classifiers," *International Journal of Engineering Trends and Technology*, vol. 71, no. 3, pp. 81-90, 2023. [CrossRef] [Publisher Link]

[8] Alex King Yeung Cheung, and Hans-Arno Jacobsen, "Load Balancing Content-Based Publish/Subscribe Systems," *ACM Transactions on Computer Systems*, vol. 28, no. 4, pp. 1-55, 2010. [CrossRef] [Google Scholar] [Publisher Link]

[9] Pasha M. Jahir et al., "Bug2 Algorithm-Based Data Fusion using Mobile Element for IoT-Enabled Wireless Sensor Networks," *Measurement: Sensors*, vol. 24, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[10] Angel Leonardo Valdivieso Caraguay et al., "SDN: Evolution and Opportunities in the Development IoT Applications," *International Journal of Distributed Sensor Networks*, vol. 10, no. 5, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[11] G. Prathyusha, Dunna Nikitha Rao, and Kaipa Chandana Sree, "Enhancing Cloud-Based IoT Security: Integrating AI and Cyber Security Measures," *International Journal of Computer Engineering in Research Trends*, vol. 10, no. 5, pp. 26-32, 2023. [CrossRef] [Publisher Link]

[12] Anatolijs Zabasta et al., "MQTT Service Broker for Enabling the Interoperability of Smart City Systems," *Energy and Sustainability for Small Developing Economies (ES2DE)*, pp. 1–6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[13] Nasi Tantitharanukul et al., "MQTT-Topics Management System for Sharing of Open Data," *International Conference on Digital Arts, Media and Technology (ICDAMT)*, pp. 62–65, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Yulong Shi, Yang Zhang, and Junliang Chen, "Cross-layer QoS Enabled SDN-like Publish/Subscribe Communication Infrastructure for IoT," *China Communications*, vol. 17, no. 3, pp. 149-167, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Wang Yali, Zhang Yang, and Chen Junliang, "SDNPS: A Load-Balanced Topic-Based Publish/Subscribe System in Software-Defined Networking," *Applied Sciences*, vol. 6, no. 4, pp. 1-21, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[16] Moraes F. Pedro, and Martins S. B. Joberto, "A Pub/Sub SDN-Integrated Framework for IoT Traffic Orchestration," *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, pp. 1-9, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[17] T. Arvind, and K. Radhika, "XGBoost Machine Learning Model-Based DDoS Attack Detection and Mitigation in an SDN Environment," *International Journal of Engineering Trends and Technology*, vol. 71, no. 2, pp. 349-361, 2023. [CrossRef] [Publisher Link]

[18] Surendra Kumar Keshari, Vineet Kansal, and Sumit Kumar, "A Cluster-Based Intelligent Method to Manage Load of Controllers in SDN-IoT Networks for Smart Cities," *Scalable Computing: Practice and Experience*, vol. 22, no. 2, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[19] Jehad Ali et al., "ESCALB: An Effective Slave Controller Allocation-Based Load Balancing Scheme for Multi-Domain SDN-Enabled-IoT Networks," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 6, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[20] Komal Purba, and Nitin Bhagat, "A Review on Load Balancing Algorithm in Cloud Computing," *SSRG International Journal of Computer Science and Engineering*, vol. 1, no. 10, pp. 1-5, 2014. [CrossRef] [Google Scholar] [Publisher Link]

[21] Rayikanti Anasurya, "Internet of Things (IoT) in Mining: Security Challenges and Best Practices," *International Journal of Computer Engineering in Research Trends*, vol. 9, no. 5, pp. 93–98, 2022. [CrossRef] [Publisher Link]

[22] Sounni Hind, El Kamoun Najib, and Lakrami Fatima, "Software Defined Network for QoS Enhancement in Mobile Wi-Fi Network," *International Journal of Recent Technology and Engineering*, vol. 8, no. 3, pp. 4863 4868, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[23] Nahida Kiran, Yin Changchuan, and Zaid Akram, "AP Load Balance-Based Handover in Software Defined Wi-Fi Systems," *IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*, pp. 6-11, 2016. [CrossRef] [Google Scholar] [Publisher Link]