Using Software-Defined Networking for Data Traffic Control in Smart Cities with WiFi Coverage †

Basima Kurungadan and Atef Abdrabou *

Abstract: The growth of smart cities is fueled by the vast rise in wireless smart gadgets and uninterrupted connectivity. WiFi is the dominant wireless technology, enabling Internet-of-Things (IoT) connectivity in smart cities due to its ubiquitous access points and low deployment cost. However, smart city applications offer a wide range of services with different quality-of-service (QoS) demands. This paper addresses packet delivery latency as one of the QoS metrics affecting many time-sensitive smart city services. Thus, the paper proposes employing software-defined networking (SDN) to control the traffic load of WiFi access points (APs), preserving its symmetry in a city-wide coverage of WiFi-connected IoT gateways or fog nodes. These gateways receive data packets from smart city/IoT devices via wireless links and forward them over a city-deployed WiFi network to their management entities or servers. Three SDN-based algorithms are devised to reduce the gateways’ packet-forwarding delay and keep a symmetric traffic load at the WiFi network APs. The algorithms are developed and tested using a real hardware setup constituting WiFi devices without additional requirements on the IoT gateways (WiFi clients) or the APs, such as support for a specific roaming protocol or bandwidth-consuming signaling such as sending probe packets. Extensive hardware experimentation shows that the SDN controller, via the proposed algorithms, can effectively reduce the packet forwarding latency of IoT gateways by carefully selecting the IoT gateway with the highest packet latency and seamlessly handing it over to the least-loaded covering AP.

Keywords: software-defined networking; WiFi; fog; traffic; load; control; delay; IoT; smart city; e-Health

1. Introduction

With WiFi seeping its roots deep into human life day by day, it has even influenced the author of Ref. [1] to add it to Maslow’s Hierarchy of Needs [2] on a basic level before human physiological needs. A new study expects more than two-thirds of the IP traffic volume to be from portable and WiFi-associated devices, outnumbering the wired traffic by more than double the amount [3]. With the leap in usage of wearable smart gadgets, which are mostly connected through smartphones instead of having embedded cellular connectivity [4], WiFi connection outdoors is becoming inevitable. Moreover, WiFi offloading is a current topic of interest due to its efficiency in providing uninterrupted video streaming [5].

In addition, the presence of the COVID-19 pandemic broadly expanded the interest in involving the Internet through WiFi associations to achieve a large throughput in both communication directions (transfer and download), as in web-based video conferences/meetings and online teaching. WiFi is commonly used as a candidate technology for the communication network design of smart cities to support the functional requirements of many IoT applications [6]. However, the capabilities of a smart city and its importance...
in the future are still unappealing to ordinary people due to numerous concerns over the latency, privacy, reliability, and availability [7] it can offer. Information delivery latency is one issue that needs to be addressed seriously. Besides numerous tasks, transmitting a large real-time data volume from IoT nodes and processing this data represent important tasks of various smart city services. Among those, some are time-critical, for which any communication lag above a particular threshold means total worthlessness of such services. For instance, information transfer latency is vital for e-health services to realize their objectives of continuously monitoring patients and reporting emergencies [8].

Indeed, the number of wireless sensor nodes required to realize a full-fledged smart city is massive. These nodes are of different types and should be installed in different locations. Some of them are not easy to replace frequently, like in the case of structure monitoring, and hence they need to maintain low-power consumption. Others require long-range communication due to obstacles or being installed in remote city areas. This mandates the usage of different wireless technologies such as Zigbee and LoRa, which cannot support a high data rate. Thus, using WiFi-connected IoT gateways becomes inevitable [9]. These gateways can receive sensed data over different wireless technologies and forward it to its management entities or servers using WiFi. Moreover, they can function as edge or fog computing nodes [10]. However, these gateways need to maintain acceptable packet latency requirements of the forwarded data, given that these gateways often lie in spots with overlapped WiFi coverage. Admittedly, this is triggered in a situation where some APs receive periodic routine data while others are loaded with delay-sensitive traffic, such as alarm notifications or critical health status reporting. Consequently, this demands a dynamic network configuration to connect an IoT gateway to the AP with sufficient resources preserving symmetry in each AP traffic load, which cannot be achieved with classic methods.

While there are very few studies regarding the uplifting of smart cities using SDN in its functioning, the potential of such a system is boundless. With its agility and programmability [11], SDN can dynamically configure the wireless network resources efficiently [12] since it can untie the complexity of the network infrastructure. This includes WiFi networks or even future cellular networks beyond the fifth-generation (6G) [13,14]. In SDN, there are two planes; a control plane in which a central controller monitors and controls the whole network and a data plane that constitutes all the network devices.

In this work, the power of SDN is harnessed by programming a central SDN controller to decide to which AP an IoT gateway (WiFi client) will be connected. In a traditional network, such a decision is made by the WiFi client using the measured received signal strength (RSSI), causing unbalanced traffic load assignment to different APs [15,16]. In this research, the SDN controller is programmed to select which IoT gateway should be handed over to another AP and which AP can accommodate the gateway traffic with a lower end-to-end (E2E) packet delivery latency. This results in redistributing the traffic load at the WiFi network APs, leading to a considerable decrease in overall network delay.

The contributions of this paper are three-fold.

- A detailed description of an algorithm is introduced to find the IoT gateway with the highest end-to-end packet delivery delay transparently without altering the IoT gateway or the receiving node by adding any software agent or measuring probes (The algorithm is presented in part in the proceedings of CITS 2021 [17]). The algorithm is non-invasive as it solely depends on packet interarrival time measurements at the SDN controller;
- Another algorithm is devised to find the IoT gateway with the highest end-to-end latency by estimating this delay for each gateway connected to a certain AP. This is done by performing the necessary measurements at the SDN controller to carry out an M/G/1 analysis using the arrival rate information received from the IoT gateway. For both algorithms, the SDN controller reassociates the tagged IoT gateway seamlessly
with another AP without changing the gateway configuration or exchanging handover-related messages;

- An algorithm is developed on the SDN controller to continuously find the least loaded AP to handover the IoT gateway with the highest end-to-end packet delivery delay.

The outline of the remaining sections of the paper is as follows. The most related research papers are surveyed in Section 2. Section 3 highlights the system model. In Section 5, the devised algorithms are presented. Section 4 introduces the experimental setups used to test the proposed algorithms. The experimental results are provided and discussed in Section 6. Section 7 outlines the conclusions of the paper.

2. Related Works

Addressing WiFi AP selection is different from load balancing for latency reduction. Even though various studies address AP selection mechanisms, few papers discuss WiFi AP load balancing. A considerable number of these studies demand the availability of specific functionality in WiFi clients, such as transmitting test packets [18,19] and running certain protocols [20].

For instance, the authors of Ref. [18] present an online algorithm in which a WiFi node chooses the AP that minimizes the norm of the traffic loads of other APs in its transmission range. Only throughput is addressed in Ref. [18], which requires a modification of the client WiFi adapter to send custom test packets. The research work in Ref. [21] proposes a system named Virgil. This system chooses an AP as a result of associating the client with the rest of the nearby APs to evaluate the connection quality. Range extender selection for home APs is addressed in Ref. [20]. The paper considers not only the traditional method of RSSI values but also the channel load, but it demands the existence of WiFi hardware supporting IEEE 802.11 k/v [20]. Industrial applications employ many IoT services. These applications also constitute an integral part of a smart city. Thus, the authors of Ref. [22] propose a deterministic load balancing algorithm (Det-LB) based on a game-theoretic auction model. The research work in Ref. [23] presents a video load balancing solution (ViLBaS) to increase the performance of video applications in a multihop wireless mesh network to enhance the user quality of experience. However, this method is more like a routing mechanism to avoid congested nodes in the route.

In SDN, monitoring and decision-making can be done at a centralized controller. Thus, in the literature, a multitude of research works investigate the usage of SDN in AP selection. For instance, in Ref. [24], a centralized SDN controller is used for AP selection according to a date rate-based fittingness factor without taking packet latency into account. The authors of Ref. [25] present a software defined-WiFi network that addresses packet delay and loss for e-Health applications. However, both Refs. [24,25] use computer simulations to verify the proposed schemes. The authors of Ref. [26] propose a new learning-based algorithm to monitor load imbalances in APs and perform handover accordingly to balance the load in all APs in a software-defined wireless network (SDWN). Their work mainly focuses on the throughput of TCP connections. Using SDN to optimize the client association process with the help of a centralized controller while ensuring QoS for software-defined wireless local area network (SDWLAN) is studied in Ref. [27]. However, the authors in Ref. [27] assume that all the APs and clients are tuned to the same radio channel and conducted only computer simulations to measure the performance of the proposed scheme. The authors of Ref. [17] present an SDN-based algorithm for selecting WiFi nodes based on packet latency to be handed over to other another AP.

To the best of the authors’ knowledge, no other research work in the literature presents an SDN-based laboratory-validated scheme for reducing the delivery latency of data packets forwarded by WiFi-connected IoT gateways covered by multiple smart city APs working on different radio channels. The packets forwarded by IoT gateways are continuously monitored by centralized SDN controller(s) for latency. In any region of the smart city network, the SDN controller handling the region selects the IoT gateway with the highest
packet latency by running the proposed algorithms and seamlessly hands it over to the least loaded AP covering this gateway.

3. System Model

The paper addresses a smart city outdoor setting where a city-wide WiFi network consisting of many APs with overlapped coverage is deployed to support different smart city services/applications via IoT communications, as shown in Figure 1. The data traffic carried by the city-wide WiFi network involves two data traffic types. The first is the smart city traffic (i.e., machine or device-type traffic), whereas the second is the background (human-based) traffic. The human-based traffic is originated from user devices, such as smartphones, laptops, and tablets. On the other hand, the smart city traffic is assumed to be generated by different IoT devices serving numerous applications supported by various wireless technologies such as Zigbee, Bluetooth, LoRa, and others. These devices are assumed to be connected to low-cost IoT gateways equipped with WiFi transceivers supporting the legacy IEEE 802.11n standard without any other amendments, such as IEEE 802.11 v/k, which support roaming and handover. Thus, the IoT gateways or fog nodes are used to aggregate the data traffic of IoT devices. They also take care of the different wireless technologies used by such devices, as a gateway is assumed to have two wireless interfaces. One matches the wireless technology for the group of IoT devices supported by the gateway, whereas the other uses WiFi to connect to the city-wide WiFi network through its APs. Some smart city applications are assumed to be delay-sensitive, whereas others are delay-tolerant. The gateways are considered fixed in their locations (or have limited mobility) and not suffering from energy limitations as they are deployed outdoors across the city with access to the electric grid. The city-wide WiFi network consists of APs with overlapped coverage. The adjacent APs are tuned to non-overlapping radio channels. An SDN controller or several controllers are assumed to communicate with all APs in the network via the OpenFlow [28,29] protocol and obtain information about the packets transmitted over any WiFi radio channel. The OpenFlow protocol is adopted as it is the current industry standard for SDN, although some other researchers propose the usage of other protocols [30] or architectures [31] for particular applications such as e-Health. An IoT gateway/fog node can also communicate with the controller via a software agent that reports the packet arrival rate at the gateway. The internal architecture of the IoT gateway/fog node is assumed to be similar to the one proposed in Ref. [32], where the fog node can classify the traffic and, hence, prioritize it to be forwarded based on latency requirements as an aggregated traffic flow. Thus, it does not need to use the OpenFlow protocol at the IoT device level to reduce the number of flows handled by the SDN controller, alleviating some of the scalability issues mentioned in [33]. In addition, it is assumed that the controller can seamlessly handover an already connected gateway to another AP without reconfiguration or disconnection.
4. Experimental Setup and Procedure

Two experimental setups are used to mimic the system model and validate the proposed algorithms, as revealed in Figure 2a,b. Two experiments were done with the first experimental setup shown in Figure 2, whereas the third experiment used the second experimental setup. Both setups use real hardware, not computer simulators. Different hardware equipment and software packages are employed in both setups, as described in the sequel.

4.1. Hardware Equipment

APs are emulated using TP-LINK wireless dual-band routers in both experimental setups. For the first setup, two APs are used (marked as AP1 and AP2), whereas, for the second setup, a third AP is used (AP3). Three PCs emulate three IoT gateways...
in both setups. The WiFi adapters used in the three PCs (WiFi clients) support IEEE 802.11 n. The data generated by the three gateways are sent to the destination PC, emulating a management entity (server) in the real world. Apart from the three gateway nodes, two other background WiFi clients (PCs) are used to emulate background data traffic that shares the city WiFi network but is not a part of the smart city sensed data traffic. All the used computers run Ubuntu 19.10 or 20.04 operating system. All Ethernet connections are made via Gigabit Ethernet ports.

4.2. Software Packages

In order to realize the SDN environment in the experiments, various software packages are used, namely, Ryu, Open vSwitch (OVS), and Empower-5G framework. Ryu is a component-based software-defined networking framework [34] used to create the SDN controller functionality.

The SDN controller runs on a separate computer (PC) and supports the OpenFlow protocol. The controller uses the Open Flow protocol to communicate with the OVS software, which runs on a multi-Ethernet port computer to emulate a soft SDN switch (referred to hereafter as the OVS switch) and also on the APs’ firmware, as shown in Figure 2a,b. The SDN controller, the APs, and the destination PC are connected through the OVS switch.

4.3. Procedure

The seamless handover of WiFi gateways from one AP to another is performed using the Empower-5G framework [35], which is an open SDN platform for radio access networks. This framework is installed and run on the SDN controller PC. When APs are registered with Empower-5G, it establishes a WiFi network of a certain service set identifier (SSID) for each of the registered APs. When a WiFi gateway connects to any of these APs, an agent corresponding to this gateway is installed at the AP by Empower-5G, known as a light virtual access point (LVAP) agent. Thus, handover is performed by transferring these virtual agents between two APs (i.e., removing the LVAP agent from the current AP and creating a new LVAP agent in another AP) without changing the WiFi configuration of the gateway [36].

The data traffic received by the gateway from IoT devices is emulated using the RUDE/CRUDE software tool, a UDP traffic generator running in a client/server fashion. The RUDE/CRUDE tool is also used to generate the background traffic with different data rates. The synchronization of senders and receivers is crucial to measure the end-to-end delay precisely. Hence, the precision time protocol (PTP) runs on all the senders and destination computers.

5. Analysis and Proposed Algorithms

This section introduces the three proposed algorithms. Two algorithms aim at selecting the gateway with the highest end-to-end packet delivery latency to undergo a handover to another AP. The first algorithm is mainly based on packet service time estimation, whereas the second relies on queuing analysis. The third algorithm selects the least loaded AP to which the handover of the gateway, selected by any of the first two algorithms, will be performed.

5.1. Based on Service Time Estimation

The end-to-end delay, $W_j$, for a packet of a gateway node $j$, is given by the service time, $V_{t_j}$, and the queuing delay, $U_{t_j}$, as

$$W_j = V_{t_j} + U_{t_j}.$$  \hspace{1cm} (1)

Generally, $U_{t_j}$ varies with the arrival rate and the service time. It is assumed that either the arrival processes of all gateways are similar and have almost the same arrival rate $\lambda$ or the arrival processes have different arrival rates, but the queuing delay is negligible compared
with the service time. This happens when the network works in a region sufficiently far from the saturation point [37]. Hence, the end-to-end delay variation is captured by the service time variation (i.e., the longer the service time, the longer the end-to-end packet delay). A generalization of the average service time $E[V_{ij}]$ of IEEE 802.11 mentioned in Ref. [37] can be obtained as

$$E[V_{ij}] = \sum_{i=1}^{N-1} \gamma_i \left[ D_{ij} + \frac{D_{ij}}{2} \frac{p_c}{1 - p_c} \right] + B(p_c) + D_{ij} + \frac{D_{ij}}{2} \frac{p_c}{1 - p_c}$$  \hspace{1cm} (2)

where $\gamma_i = V_i \lambda$ is the node $i$ queue utilization, $p_c$ is the probability of collision (constant for background nodes and gateways), $\lambda$ is the rate of packet arrivals, and $B(p_c)$ is the average backoff interval given by Ref. [37]

$$B(p_c) = \frac{1 - p_c - p_c(2p_c)^{bs} \frac{CW_{\min}}{2}}{1 - p_c}$$  \hspace{1cm} (3)

where $CW_{\min}$ is the minimum contention window size. The transmission and collision times of a packet of node $i$ are represented by $D_{ij}$ and $D_{ij}$, respectively. They can be expressed as $D_{ij} = \frac{S}{R_i} + D_{ij}$ and $D_{ij} = \frac{S}{R_i} + D_{ij}$, respectively, where $S$ is the packet size, $R_i$ is the channel transmission rate for node $i$, and $D_{ij}$ and $D_{ij}$ are the constant times related to the IEEE 802.11 protocol operation [37]. The proposed algorithm can be outlined as follows.

Step 1: Once a packet arrives at the OVS switch, a Packet-In event is triggered and sent to the SDN controller. The controller obtains the average and standard deviation of the interarrival times of Packet-In events using the recorded arrival time of these events. The average and standard deviation of the interarrival time of Packet-In events correspond to their packet service time counterparts.

Step 2: The gateway that has the maximum average interarrival time will be the candidate for handover to the lowest loaded AP. In case the same maximum value is recorded for more than one gateway, the gateway that records the maximum standard deviation among them will be selected for handover.

Step 3: The interarrival time of Packet-In events will be continuously monitored for the gateway that underwent the handover. Step 2 shall be redone with another AP if the interarrival time is not decreased.

Step 4: The interarrival time of Packet-In events is rechecked by the SDN controller whenever a new gateway is attached to an AP or the traffic load of one of the APs increases because this influences the service time of all the attached gateways to this AP based on (2). Step 2 shall be repeated when the average values of the packet interarrival time of the connected gateways increase.

Figure 3 outlines the flowchart of the algorithm.

5.2. Using M/G/1 Analysis

Since the traffic received by a large number of IoT sources (mainly characterized as on-off in nature) is aggregated at the IoT gateway, modeling this aggregated traffic can be approximated to a Poisson arrival process [38]. Thus, M/G/1 queuing analysis [39] is used to estimate the end-to-end delay for each IoT gateway using the Pollaczek–Khinchine (PK) formula. The mean waiting time for a packet in the queue at a sender gateway $j$ according to the PK formula is

$$T_{ij} = \lambda * \frac{(\sigma_{ij}^2 + E(V_{ij})^2)}{2 * (1 - \lambda * E(V_{ij}))}$$  \hspace{1cm} (4)

where $\lambda$ is the packet arrival rate, $\sigma_{ij}$ is the standard deviation of the service time at sender $j$, and $E(V_{ij})$ is the mean service time at sender $j$. From (1) the average end-to-end packet delay, $E[W_{ij}]$, for a packet sent by node $i$ can be formulated as

$$E[W_{ij}] = T_{ij} + E[V_{ij}]$$  \hspace{1cm} (5)
Figure 3. Service time estimation algorithm flow chart.

The steps of the algorithm are provided as follows.
Step 1: By recording the arrival time of packet-In events with the aid of the OVS, the SDN controller calculates the average and standard deviation of these events’ interarrival time. This translates directly to the average and standard deviation of each gateway’s packet service time.

Step 2: Each gateway informs the SDN controller about its packet arrival rate through a software agent as a client/server communication. The controller applies the PK Formula (4) to estimate the end-to-end delay for each gateway (WiFi client).

Step 3: The client that has the maximum estimated end-to-end delay is selected for handover to an AP loaded with less traffic.

Step 4: The end-to-end packet delay of all the gateways, which recently performed a handover, is continued to be estimated by the SDN controller to check if the delay is lowered. Otherwise, Step 2 shall be redone, but with another AP.

Step 5: The end-to-end packet delay for all IoT gateways (clients) is estimated by the SDN controller when a new gateway joins an AP or an AP traffic load increases. Once the estimated end-to-end delay becomes higher, Step 2 shall be redone.

Figure 4 shows the flowchart of the algorithm.

5.3. Finding the Least-Loaded AP
This algorithm complements the aforementioned algorithms since it finds the least loaded AP to which the SDN controller shall carry out a handover. This is performed after finding the gateway with the maximum average interarrival time (as in Section 5.1) for Packet-In events or the maximum estimated end-to-end packet delay (as in Section 5.2). The integration between the algorithm mentioned in Section 5.1 and this algorithm is depicted as a flowchart in Figure 5 for illustration purposes.

Step 1: The SDN controller receives Packet-In events from all the network APs after the packets are delivered to the OVS switch.

Step 2: Using the available information from the packet-In events, the SDN controller extracts the number of packets received from each AP separately over a certain time interval.
Step 3: The AP with the least packet count is identified by the SDN controller as the least loaded AP.

Step 4: The Packet-In events’ data is continuously monitored by the SDN controller, which applies Step 2 to update the AP with the least load status.

Figure 4. The flowchart for the M/G/1 analysis-based algorithm.

Figure 5. The scheme for choosing least loaded AP and highest E2E client.

6. Performance Results and Discussion

This section presents and discusses the performance results of the proposed algorithms.
6.1. Results

This section introduces the performance results of the three algorithms described in Section 5. Extensive laboratory experimentation is used to validate the performance of the algorithms in Section 5. The results of around 30 samples are recorded for each experiment.

6.1.1. Handover Based on Service Time Estimation

Three performance indicators are used to validate the algorithm presented in Section 5.1. In the first indicator, the average end-to-end packet delay for the emulated gateways before handover is compared with its value after handover by varying the gateway traffic rate and a constant background traffic rate of 120 packets/s (each packet is 1450 Bytes). Figure 6a shows a considerable decrease in the end-to-end packet delay after handover, particularly when the gateway traffic rate is high.

In the second indicator, the relationship between varying the interarrival time of Packet-In events at the SDN controller and the end-to-end delay is revealed in Figure 6b for an extensive range of gateway traffic rates.

The third indicator addresses the applicability and correctness of performing the handover based on the service time estimation algorithm by comparing it with the decision of an observer who can record the end-to-end delay at the destination computer (ideal decision). As depicted in Figure 6c, the number of times the algorithm decision was different from the ideal one decreases when the gateway traffic rate increases.

Furthermore, more experiments are conducted to validate the algorithm’s performance against the previously mentioned performance metrics when the gateway traffic rate is kept constant at 475 packets per second while the background traffic load of the network is varied.

Figure 6. Cont.
Figure 6. Service time estimation-based algorithm performance with different client rates. (a) End-to-end delay before and after handover. (b) Variation of end-to-end delay with interarrival time at the SDN controller. (c) Ideal handover and algorithm-based handover percentage difference.

As shown in Figure 7a, a notable end-to-end packet delay difference between the cases before and after handover is observed. Figure 7b reveals that the variation of the end-to-end delay is affected by changing the packet interarrival time at the controller in general. However, with a large background traffic volume, a small variation in the packet interarrival times leads to a considerable change in the end-to-end delay. Figure 7 shows that the algorithm handover decision approaches the ideal handover decision when raising the load of the network background traffic.

Figure 7. Cont.
6.1.2. Handover Based on M/G/1 Analysis

This section presents the results of performing the handover using the algorithm introduced in Section 5.2. Figure 8a shows the variation of the end-to-end delay as the gateway (WiFi client) packet rate changes. As Figure 8a reveals, the algorithm is successful in significantly decreasing the end-to-end packet delay after handover. However, it is depicted in Figure 8b that the change of the packet interarrival time at the SDN controller with Poisson-distributed traffic arrivals does not change significantly with the arrival rate. Thus, it does not reflect the change of the end-to-delay, as shown previously with a fixed arrival rate (Figure 6b). Similar to the first algorithm, the performance of this algorithm in terms of end-to-end packet delay is investigated when the background traffic load is varied, as depicted in Figure 9. Figure 9a shows a significant reduction in end-to-end packet delay after handover is performed. However, the difference between the end-to-end delay before and after handover becomes smaller as the background traffic volume increases. The results revealed in Figure 9b affirm that the variation of the end-to-end packet delay is not reflected by the packet interarrival time at the SDN controller, as noticed in Figure 8b. In addition, a comparison between the measured end-to-end delay before handover and the estimated end-to-end delay calculated by the SDN controller is depicted in Figure 8c. The difference in the delay values is attributed to the measurement error due to the time taken by the operating system kernel to set a timestamp of the incoming and outgoing packets [40].

Figure 7. Service time estimation-based algorithm performance with different traffic loads. (a) End-to-end delay vs. background traffic load before and after handover. (b) End-to-end delay and interarrival time at the SDN controller with background traffic load. (c) deal handover and algorithm-based handover percentage difference.

Figure 8. Cont.
Figure 8. Handover based on M/G/1 analysis with different client rates. (a) End-to-end delay before and after handover. (b) Variation of end-to-end delay with interarrival time at the SDN controller. (c) Measured and estimated end-to-end delay.

Figure 9. Cont.
Figure 9. Handover based on M/G/1 analysis with different traffic loads. (a) End-to-end delay vs. background traffic load before and after handover. (b) End-to-end delay and interarrival time at the SDN controller with background traffic load.

6.1.3. Handover to AP with Least Load

Figure 10 depicts the performance of the AP selection algorithm. The performance is evaluated by comparing the end-to-end packet delay of the gateways (clients) before and after handover for an algorithm randomly choosing the AP, as in Figure 10a, and the algorithm proposed in Section 5.3, as in Figure 10b. The percentage difference in the traffic load between the available APs is varied in both figures, and the corresponding end-to-end packet delay is recorded. It is evident from Figure 10a,b that the reduction in the end-to-end delay packet delay after handover is significantly larger when the AP is chosen using the proposed algorithm. An exception to this is when the percentage difference in traffic load between the available APs is very low.

6.2. Discussion

The presented results indicate that the three proposed algorithms can efficiently perform their intended functions.

As evident from Figures 6a and 7a, the end-to-end packet delay of the IoT gateways (WiFi clients) can be significantly reduced by performing handover based on an algorithm running on the SDN controller that estimates the service time by measuring the packet interarrival time of Packet-In events. For a constant rate of gateway traffic, which often happens with synchronized periodic IoT sources [41], Figures 6b and 7b reveal that the interarrival time of Packet-In events is indicative of the end-to-end packet delay at the receiving node. Moreover, Figures 6c and 7c show that the algorithm handover decision is less erroneous as the network traffic load increases either by increasing the gateways’ traffic rate or the background traffic (increasing $\gamma_i$ in (2)). This pushes the network towards saturation (a higher violation probability and backoff time as in (3)), making the end-to-end delay more sensitive to service time variation.

While using M/G/1 analysis for handover decisions, a similar trend to the one with the service time estimation can be observed. Gateways’ traffic exhibited a lower end-to-end packet delay after handover based on M/G/1 analysis, as shown in Figures 8a and 9a. However, the packet interarrival time at the SDN controller does not significantly vary with increasing neither the gateway traffic rate (Figure 8b) nor the background data rate (Figure 9b). This is due to the randomness of Poisson traffic, which makes the queuing delay impact the end-to-end delay more than the service time of the shared IEEE 802.11 channel.

The AP selection algorithm described in Section 6.1.3 is driven by the awareness of the SDN controller of the traffic load in each AP, which affects the available channel capacity, and hence, the end-to-end packet delay.
Figure 10. Comparison of algorithm-based and random AP selection. (a) End-to-end delay before and after handover of with random AP selection. (b) End-to-end delay before and after handover of with AP selection algorithm.

It is worth noting that the mobility of IoT devices does not affect the performance of the proposed algorithms. If an IoT device moves, it may connect to a different IoT gateway. The IoT gateways are assumed stationary and deployed to cover the connectivity of the IoT devices over the whole city. On the other hand, if any IoT gateway changes its location or is deployed elsewhere, this is considered limited mobility as it will still be under overlapped coverage of the other city APs. This also does not affect the operation of the proposed algorithms as they run mainly on the SDN controller based on the data received from the APs.

7. Conclusions

This research harnesses the SDN abilities to control the AP traffic load, keeping its symmetry, and reduce the end-to-end packet latency in a smart city setting. The paper tackles a scenario where WiFi-connected IoT gateways or fog nodes are covered by a city-wide WiFi network, where multiple APs typically cover each gateway/fog node. An SDN controller or several controllers observe all the APs in the city WiFi network, which supports smart city applications of different delay requirements. Three algorithms are introduced to control and preserve the symmetry of the traffic load of each AP and reduce the end-to-end delay of connected gateways by performing a seamless handover to the least-loaded AP. Two algorithms address the selection of the IoT gateway with the highest end-to-end packet delay. The first algorithm selects the gateway by estimating the packet service time at the SDN...
controller in a non-invasive fashion that does not require any extra configuration, protocols, or software agent to run on the gateways. With constant-rate gateway forwarded traffic, the algorithm efficiently reduces the end-to-end delay of the gateways and makes the right handover decision according to hardware experimentation. The second algorithm addresses the gateway traffic of Poisson arrivals, which typically happens when many independent on-off IoT sources are aggregated at the gateway. This algorithm uses M/G/1 analysis to estimate the end-to-end packet delay by estimating the service time statistics at the SDN controller. Extensive hardware experiments using different packet rates of gateways and background traffic show that the algorithm efficiently decreases the end-to-end packet delay of IoT gateways. Finally, a third algorithm that selects the least-loaded AP via the SDN controller is shown to effectively minimize the end-to-end packet delay of the traffic of IoT gateways compared with selecting any AP randomly for handover. The future work of this research includes using machine learning techniques for SDN-based traffic load control in smart cities with heterogeneous wireless (WiFi and cellular) coverage.

Author Contributions: B.K. built the testbed and contributed to devising the proposed algorithms. She conducted the experiments using the testbed. She also contributed to paper writing. A.A. designed the network architecture of the testbed. He contributed to the analytical models and proposed algorithms. He also participated in reviewing and writing the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the UAE University UPAR grant number 12N008/31N456.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References


