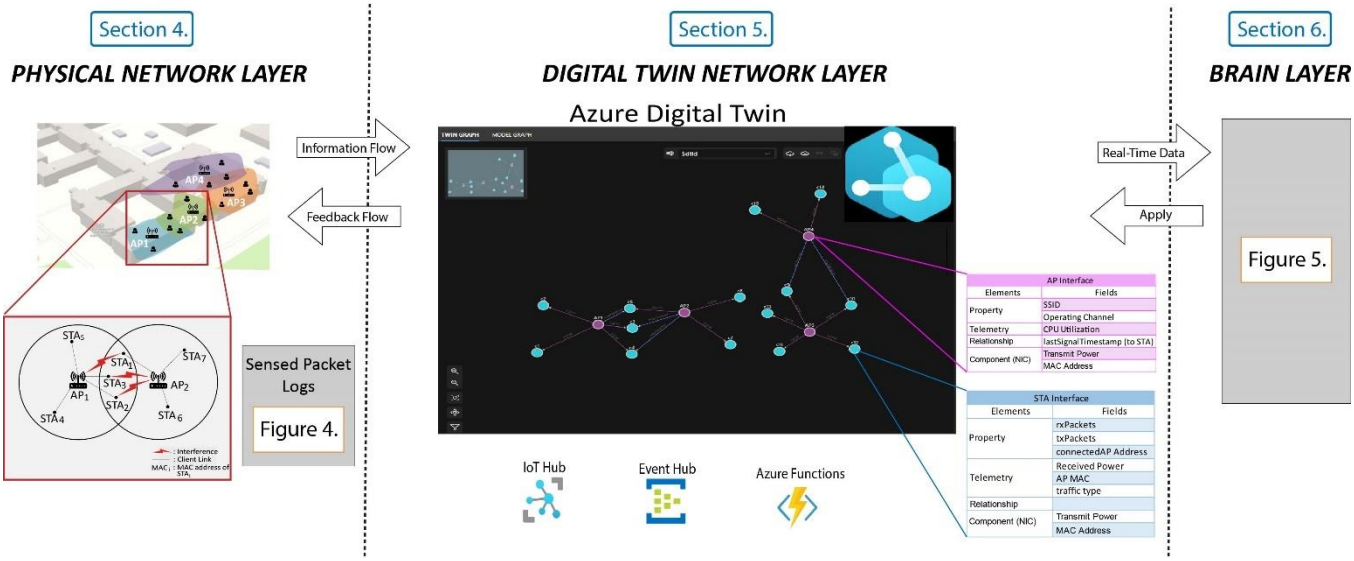

Figure 2. The proposed Digital Twin WiFi Network (DTWN) architecture

Service Sets (BSSs). [8] proposes a two-scale control in which CST is adaptively adjusted at STAs and parameters for CST are adjusted using Artificial Intelligence at APs. While such advancements on 11ax SR mechanisms might increase resource utilization, it is limited because they work without communication among neighbouring APs.

To address the issues, the IEEE802.11be standard introduced Multi-AP coordination concept which includes coordinated spatial reuse (CSR) and several others [5]. In [6], two different CSR options are compared with 11ax SR and simulation results shown that CSR achieves better throughput. However, the CSR protocols' [7] use of the wireless medium may result in overhead which should be taken into consideration. Moreover, the transmission powers of all APs are calculated with a rule-based approach by an AP. Thus, it is challenged by scarce resources on APs such as computational and memory resources.

The proposed power control mechanism in [9], allocations are done with a rule-based approach using characteristics of users. Moreover, in [10] a coordinated power control scheme is proposed which is also a rule-based method. While such algorithms might perform well under predefined conditions, they might be unable to adapt to real word examples.

Additionally, in [12] performances of TPC, CST tuning, and TPC with CST tuning is investigated. The results show that out of these three methods, setting Transmit Power to a value just over the required value for a successful transmission gives the most performance improvement in terms of throughput. The paper concludes that performance gain is dependent on the network topology meaning adapting to the change of the network is needed. This might limit the effectiveness of rule-based TPC schemes.

The study in [11] proposes a centralized controller that adjusts the transmit power and channel of APs based on Q-Learning. They define the state of the network using two-dimensional STA locations that are presumably collected from the devices. However, this collection requires additional communication over the wireless medium, which may result in an overhead. Moreover, deducting interference from locations might be inaccurate [19]. Furthermore, while the proposed solution bares desirable results, the use of offline learning strategy might prevent achieving lower interference.

Digital twins (DTs) give promising results in data analysis with machine learning [2]. Also, DTs have been used to iteratively optimize latency [3]. Moreover, digital twin network (DTN) is an emerging concept in its drafting stage [13]. Although the applications of DTN technology are prominent in smart manufacturing and several other fields [14], to the best of our knowledge, DTN technology has not been applied to WiFi networks for the purpose of solving interference issues by adjusting transmit power.

3. Problem Formulation

We define the WiFi network as an undirected weighted graph $G = (V, E, w)$, where V is a set of vertices consisting of clients and APs that are denoted as V_c and V_{AP}. E is a set of edges that corresponds to reached signal from an AP to a client. We separate the edges formed between v_c and v_{AP} into two groups as signal (E_s) and interference (E_i). The signal type edge is formed when v_{AP} is serving to v_c and interference type is formed when there is interference between them. Moreover, w is the weight function, and the weights are the received signal strength of APs at clients.

Wireless communication quality is measured with a signal-to-interference-plus-noise ratio (SINR). Therefore, we assume SINR can represent users' quality of service,

consequently, performance. However, we cannot do measurements on station side, we define a signal-to-interference indicator using G.

3.1. Signal-to-Interference Indicator (ϕ)

It is calculated for client vertices V_c. A client vertex $client \in V_c$ was shown in Figure 3. It forms edges with m different APs where $AP_i \in V_{AP}$. One of these edges must be of signal type, and in this case, it is the one with AP_m.

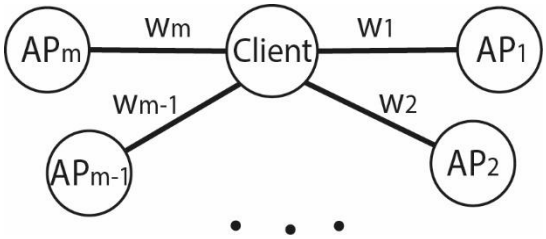

Figure 3. Example Client Vertex

The ϕ for the vertex $client$ in the Figure 3 is calculated as

$$\phi = w_m - 10\log_{10}\sum_{i=1}^{m-1} 10^{w_i/10} \qquad (1)$$

where w_i is the weight of the edge $e = (AP_i, client)$. The weights are in decibels, so to get the ratio, total interference is subtracted from the weight of the signal type edge (w_m). As for the total interference, weights are summed after converting units from dBm to mW. Then, the total value is converted back into dBm. Moreover, if there is no interference type edge, interference is taken equal to the thermal noise power which is $-100\ dBm$.

3.1. Requirement Classes

The ϕ values also give an insight into the performance of the vertex. Depending on the clients' traffic characteristics, their perceptions will change. Therefore, we need to identify how low is too low for a client. For this purpose, we define requirement classes in Table 2. Requirement Classes and assign clients to these based on their analyzed traffic patterns that were acquired through the analytics from the DTN.

Table 2. Requirement Classes

Requirement Class	Threshold
A	$\phi > 35dB$
B	$35dB > \phi > 25dB$
C	$\phi < 25dB$

The level of performance degradation on the WiFi network caused by AP to STA interference is mainly a result of transmit powers of APs. We denote the transmit power of AP_i as θ_{AP_i}. We further define the configuration of the APs in a given time as

$$\boldsymbol{\Theta}^{(t)} = [\theta_{AP_0}^{(t)}, \theta_{AP_1}^{(t)}, \dots, \theta_{AP_m}^{(t)}] \qquad (2)$$

where m is the number of APs in the network.

In general, the goal is to find the optimal vector $\Theta^{(t)}$ so that clients can have adequate levels of ϕ.

4. Physical Network Layer

The physical network layer consists of a WiFi network as shown in **Figure 2**.

Here, APs are deployed close to each other, so the interference issues threaten the performance. In such networks, interference may be in 3 types: AP to AP, AP to

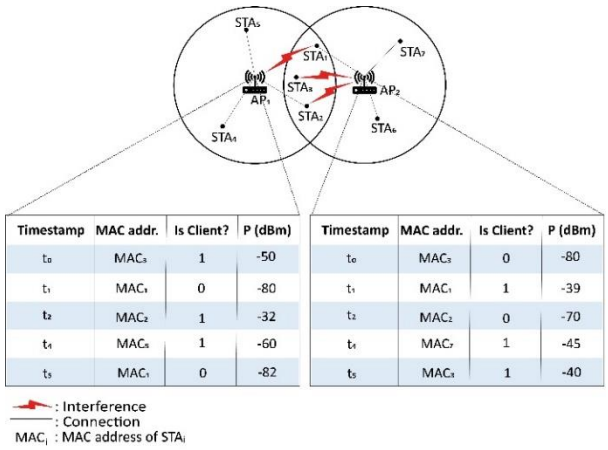

Figure 4. Sample network and example sensed packet logs

STA, and STA to STA. In this paper, we focus on mitigating AP to STA interference. To do so, we need to gather information continuously. Then, using this, decide on the transmit powers APs and later apply the decided actions. This process depends on APs providing an information flow to the next layer and them receiving feedback flow from it. To enable these, we deploy agent programs to APs.

The information flow contains the configuration of the AP, details about clients and their traffic, and the sensed

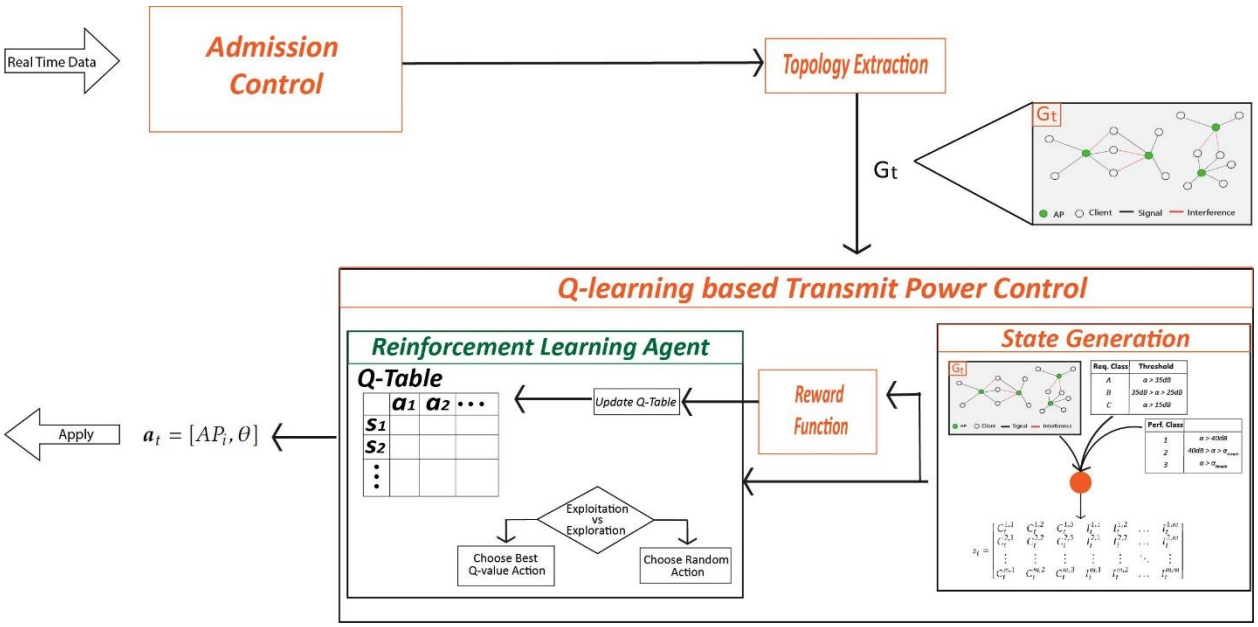

Figure 5. Brain Layer

packet log that is kept by the agent program. We explain the logging procedure by a use case that was shown in Figure 4. It contains two APs (AP_1 and AP_2) and seven client stations. For example, at time t_0, STA_3 communicates with AP_1, in the meantime, the sent packet is also received by AP_2. Both APs log the timestamp, the source address, whether the packet was from its client, and the received signal power strength in dBm.

In addition, the periodicity of the information flow will affect the accuracy of the digital twin. In other words, the digital twins' performance depends on how often it receives information from the real world. This nature brings the agent to send information periodically. Moreover, agents should send simultaneously to obtain the DTN as a whole twin of the physical network. Based on this, we introduce the twining frequency f as a parameter of DTWN. f should be tuned according to the topology at hand because of it can affect the timeliness of our proposed approach and the resource consumption on the physical network by agent programs.

5. Digital Twin Network Layer

In this layer of the architecture, we construct the twin of the physical WiFi network using the information flow from the previous layer. Then, we use this for monitoring and management purposes in the next layer, called as Brain Layer in Section 6.

5.1. Microsoft Azure

As for the implementation, we utilize Microsoft Azure IoT Hub as a gateway to the Physical Network Layer. We have installed agent programs to the APs, and these agents send information to their IoT Hub instances. Next, we used Azure Digital Twins (ADT) to form the DTN. Then, we use Event Hubs to capture a stream of data and Azure Functions to link all services together.

In detail, ADT is used to represent the physical objects using models coded with Digital Twins Definition Language (DTDL) [17]. Moreover, we defined two interfaces, AP and STA, that was shown in the **Figure 2**. They contain the following fields:

- **Property** fields represent physical object's status. The SSID and Channel information is stored for AP interfaces. In STAs, received and transmitted packet count is stored.
- **Telemetry** represents measurements that are not stored in the digital twin. Furthermore, telemetries compose the output stream of data from the DTN layer. For AP, CPU utilization and for STA, all the received power and AP Mac is streamed.
- **Relationship** is formed between AP and client models. It is directly mapped to the sensed packet log. The relationship contains the last timestamp of the signal sensed by the AP.
- **Component** is a part of the model that does not require separate identification. Since APs and clients are both equipped with network interface controller (NIC) cards for receiving and transmitting signals, we model a NIC interface and include it in the models as a component. The component contains transmit power and MAC address as a property.

6. Brain Layer

In this layer, we utilize the DTN to form transmission power adjustments of APs for managing interference. This layer consists of *Admission Control*, Topology *Extraction*, and *Q-learning-based Transmit Power Control* as it was shown in **Figure 5**.

6.1. Admission Control

Whenever, a new client enters to the network, it is detected at the brain layer with a delay based on the twining frequency. After detection, an optimal adjustment-seeking process begins. In this process, the G_t is converted to s_t and is given to the reinforcement learning agent. Then, the agent decides on an action which is then applied. This process is repeated until the decided action is to do nothing.

6.2. Topology Extraction

As mentioned earlier, we represented the network using graph G. We retrieve transmission power configurations and the most recent sensed packet log telemetries from the DTN. We use this information to construct the graph G.

The transmission power configurations of the devices within the network are not all known. In other words, APs are sending their configuration through the information flow, but configurations of the clients are not accessible in this setting. Therefore, we assume that all clients are transmitting at the same level. We denote the transmission power as θ. In detail, for the AP_i it is θ_{AP_i} and for the clients, it is θ_c.

The sensed log telemetries are used alongside θ values to form the edges. For example, a log was taken by the AP_i regarding the client $c_j \in V_c$. We represent the "P" column of the log as $P_{c_j \to AP_i}$. We map this information either to a signal type edge or an interference type edge, depending on the "Is Client?" column. As previously mentioned, the weights of the edges will be the received signal strength at the client, meaning it is denoted as $P_{AP_i \to c_j}$. Therefore, we calculate $P_{AP_i \to c_j}$ using the (3.

$$P_{AP_i \to c_j} = \theta_{AP_i} - \theta_{c_j} + P_{c_j \to AP_i} \qquad (3)$$

As a result, an edge $e = (AP_i, c_j)$ with the weight $P_{AP_i \to c_j}$ is put to the graph with the following condition: In case the edge is type interference, the $P_{AP_i \leftarrow v_j}$ needs to be higher than a significance level.

6.3. Q-learning based Transmit Power Control

RL problems are described as Markov Decision Process (MDP) and expressed in (S, A, p, r) tuple where S is the state space, A is the action space, p is the probability of transition from a state to the another after an action is applied, and r is the immediate reward. The goal of the RL agent is to find the optimal policy that maximizes the reward. The policy is a map of state to action, $\pi: A \times S \to [0,1]$.

In our solution, we use *Q-learning* algorithm. *Q* refers to the quality function that calculates the expected reward for action in a given state, also referred to as *Q-Table*. The algorithm updates *Q-Table* based on interactions with the environment.

State Space

The $S \in R^{M \times (M+3)}$, where M is the number of APs, represents the state space, and the value 3 equals to number of performance classes. The s_t, the observed state at time t, constitutes of 2 parts, C_t and I_t. C_t is the performance breakdown matrix and I_t is the interference matrix.

State Generation

We construct the s_t, using G_t and ϕ. Initially, we find ϕ values for client vertices and categorize clients into performance classes that were shown in Table 3. We use the number of clients in these classes while denoting it as $C^{i,k}$ where k is the performance class. Then, we determine how many of the clients that are connected to a given AP_i experience interference from AP_j; we denote this value as $I^{i,j}$.

Table 3. Performance Classes

Performance Class	Limits
1	$\phi > 40dB$
2	$40dB > \phi > \phi_{thresh}$
3	$\phi < \phi_{thresh}$

Consequently, we define the state s_t as

$$s_t = \begin{bmatrix} C_t^{1,1} & C_t^{1,2} & C_t^{1,3} & I_t^{1,1} & I_t^{1,2} & \cdots & I_t^{1,M} \\ C_t^{2,1} & C_t^{2,2} & C_t^{2,3} & I_t^{2,1} & I_t^{2,2} & \cdots & I_t^{2,M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ C_t^{M,1} & C_t^{M,2} & C_t^{M,3} & I_t^{M,1} & I_t^{M,2} & \cdots & I_t^{M,M} \end{bmatrix} \qquad (4)$$

Action Space

The action space is denoted as A and for a given s_t the RL agent decides on the action $a_t \in A$. We define the action vector a_t as in Equation 5 where $\theta \in [0dBm, 30dbM]$is the transmit power.

$$\boldsymbol{a_t} = [AP_i, \theta] \qquad (5)$$

Reward Function

After each action applied, we calculate the reward for the state and action pair $(s_t, \boldsymbol{a_t})$. In this calculation, we use the change between states. We subtract s_{t+1} from s_t.

$$s_d = s_{t+1} - s_t = [C_{t+1}|I_{t+1}] - [C_t|I_t] = [C_d|I_d] \quad (6)$$

Reward calculation is made using C_d and I_d matrices alongside of the reward factor λ. The reward factor is the mapping of desirability of change in performance classes. Consequently, we define the reward $r(s_t, a_t)$ as

$$r(s_t, \boldsymbol{a_t}) = C_d \lambda U_C - U^T I_d U_I \quad (7)$$

where U is all-ones matrices, U_C is the size 3×1 and U_I is the size $M \times 1$.

Reward Factor Calculation

We assume that while the actions are applied, the network topology remains unchanged so the change in the number of clients in the performance classes corresponds to transitioning between clients. Therefore, the sum of the C_d entries will always be equal to 0.

Since the goal is to have adequate levels of ϕ over the network, a trade-off dynamic appears; therefore, minimizing the performance class 3 is more desirable than increasing the performance class 1. Based on these, the reward factor $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3]^T$ should be selected with the constraints: $\lambda_3 < 0 < \lambda_1, |\lambda_1| > |\lambda_3|$. Moreover, we take the λ_2 as 0 to avoid repeating the calculation for the same transition between classes.

Update Formula

The Q-learning algorithm has a function Q that map state-action pairs to respective rewards.

$$Q: S \times A \rightarrow R \quad (8)$$

Whenever the agent selects an action $\boldsymbol{a_t}$ for a given state s_t, it receives the next state s_{t+1} and a reward r_t. Then, Q is updated based on these values with the following iterative update formula.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (9)$$

where α is the learning rate and γ is the discount factor.

Exploration vs. Exploitation

Exploration enables the agent to learn more about the environment and creates accurate estimates. On the other hand, exploitation selects the action that yields to most reward and may get more reward than the exploration. In order to leverage the benefits of the two methods, we use the ϵ-greedy action selection mechanism that balances those two by randomly choosing between them.

In a given time, the agent takes a random action with the probability of ϵ. Or it takes the $\max_a Q(s_t, a)$ action with the probability of $1 - \epsilon$.

7. Performance Evaluation

For simulations we used ns-3 network simulator [18]. Moreover, we simulated on 20Mhz channels while the APs channel assignments done beforehand. Other simulation parameters are shown in the Table 4. We first tune twining frequency and greedy rate parameters for the proposed DTWN architecture. Then we compare our results to Power Control [9], Coordinated Power Control [10], Joint Power Control Reinforcement Learning [11] mechanism to validate the performance of our proposed Q-learning-based TPC for DTWN.

Table 4. Parameters

Parameters	Value
Number of APs, M	5
Transmit Power of AP_i, θ_{AP_i}	$[0dBm, 30dBm]$
Transmit power of clients, θ_{C_i}	$12dBm$
Carrier Frequency	5Ghz
Bandwidth of the channel	20Mhz
Learning Rate, α	0.001
Discount Factor, γ	0.7
Reward Factor	$[1,0,-2]^T$

7.1. Tuning of Twining Frequency and Greedy Rate

We first conduct the set of experiments to optimally find twining frequency and greedy rate parameters for the given WiFi topology. These two parameters are then utilized in the proposed DTWN model.

Figure 6 shows detection delays of new clients entering the topology in different twinning frequencies, f. The detection delay of new clients entering originate from the admission control in Section 6.1. If the digital twin is updated less frequently, changes in the physical network may be mirrored with a delay, consequently affecting the response of our mechanism. However, higher f settings cause greater consumption of APs CPU resources as seen in Figure 7. Therefore, we set twining frequency f as 0.2 through the remaining simulations for the proposed DTWN approach.

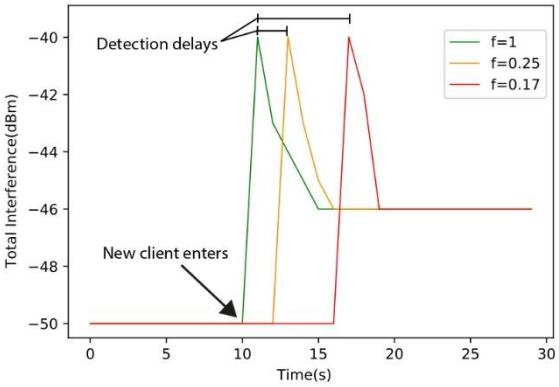

Figure 6. Detection delay comparison with different twinning frequencies

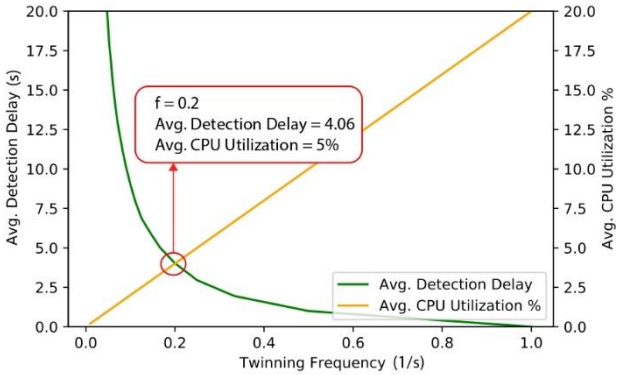

Figure 7. Average detection delay and CPU utilization

Moreover, in Figure 8, it is seen that higher greedy rates converge quicker but lower average rewards. In other words, learning speed increases with greedy rate while quality of strategies decreases. This is because when greedy rate is higher, the algorithm is able to explore more under the same number of iterations leading to faster learning. However, a random action's return might be negative meanwhile in exploitation always the action with the highest reward is selected. Due to this, average rewards converge to lower values when the exploration is higher. As a result, we set ϵ as 0.4 for Q-learning-based TPC for DTWN brain layer.

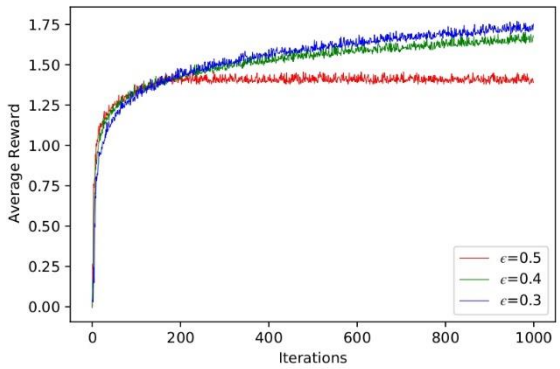

Figure 8. The average reward with the different exploration rates ϵ

7.2 Performance Comparison

We perform comparisons based on average throughput of users and total interference metrics. We compare our proposed DTWN approach to PC, Coordinated PC and JPCRL mechanisms.

As seen in Figure 9, our proposed scheme performs substantially better than PC and Coordinated PC. This is mainly due to the enabled learning and adaptation capabilities of Q-learning approach. However, the JPCRL method also leverages Q-learning, but the proposed approach outperforms it by achieving better state to action pairing which is a result of the interference focused state representation in Equation (4). Additionally, the other methods throughput decreases might be a result of the introduced overhead by the information-gathering.

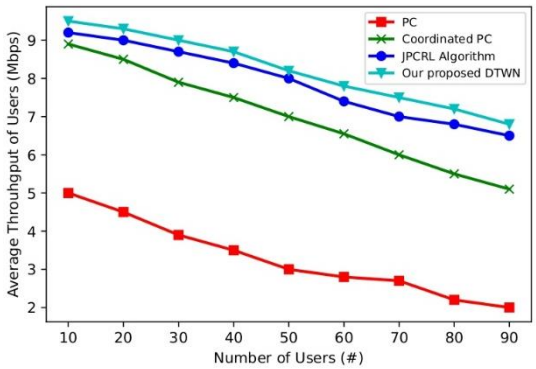

Figure 9. Average throughput versus number of users

We observe the average throughput considerably decreases as the number of users rises. This is clearly due to the fact that interference may not be mitigated fully even the transmit powers adjusted optimally. Moreover, the AP to AP and STA to STA interference is still in existence which are included in the total interference. The measurement of total interference over growing number of

users in also prove the previous observation. In other words, Figure 10 shows that the least raise is obtained in the proposed approach. Main reasons of this are that the proposed approach is able to monitor and manage the network in real-time and the Q-learning TPC can continue to learn thanks to digital twin. To put it another way, the proposed interference indicator ϕ, requirement classes and performance classes used in the state representation of Q-learning-based TPC to represent the interference in the WiFi topology.

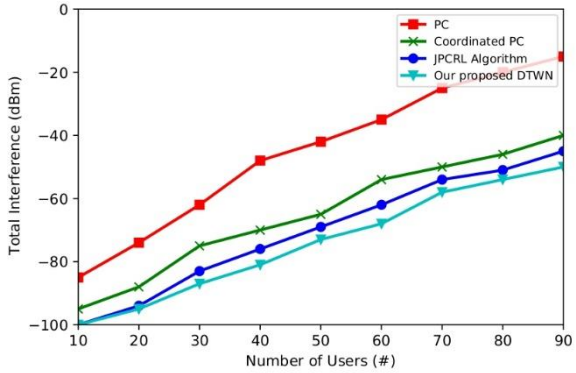

Figure 10. Total interference(dBm) versus number of users

8. Conclusion

In this paper, we proposed a Q-learning-based TPC to mitigate AP to STA interference in WiFi networks. With the reinforcement learning-based approach, we adjusted the transmit powers based on the current state while continued the learning of the TPC. We avoided overhead to the wireless medium preventing performance decrease. We achieved live analysis and management of the WiFi network, thanks to Digital Twin technology. Finally, we showed the performance of our proposed approach under different conditions, and it resulted in substantial advancements in total interference and throughput. As a future direction, deep learning can be leveraged in the TPC mechanism to handle the complicated state spaces formed in larger networks. For this purpose, a Deep Q-learning-based TPC mechanism can be developed and compared to the Q-learning-based TPC.

References

[1] Zhong Z, Kulkarni P, Cao F, Fan Z, Armour S. Issues and challenges in dense WiFi networks. In: 2015 International Wireless Communications and Mobile Computing Conference (IWCMC) [Internet]. IEEE; doi: 10.1109/IWCMC.2015.7289210

[2] Fahim M, Sharma V, Cao T-V, Canberk B, Duong TQ. Machine Learning-Based Digital Twin for Predictive Modeling in Wind Turbines. IEEE Access [Internet]. 2022;10:14184–94. doi: 10.1109/access.2022.3147602

[3] Do-Duy T, Van Huynh D, Dobre OA, Canberk B, Duong TQ. Digital Twin-Aided Intelligent Offloading with Edge Selection in Mobile Edge Computing. IEEE Wireless Communication Letters [Internet]. 2022;11:806–10. doi: 10.1109/lwc.2022.3146207

[4] Khorov E, Kiryanov A, Lyakhov A, Bianchi G. A Tutorial on IEEE 802.11ax High Efficiency WLANs. IEEE Communications Surveys & Tutorials [Internet]. 2018;21(1):197–216. doi: 10.1109/COMST.2018.2871099

[5] Deng C, Fang X, Han X, Wang X, Yan L, He R, et al. IEEE 802.11be Wi-Fi 7: New Challenges and Opportunities. IEEE Communications Surveys & Tutorials [Internet]. 2020; 22(4):2136–66. doi: 10.1109/COMST.2020.3012715

[6] Aio K. Coordinated Spatial Reuse Performance Analysis [Internet]. 2019. Available from: https://mentor.ieee.org/802.11/dcn/19/11-19-1534-01-00be-coordinated-spatial-reuse-performance-analysis.pptx

[7] Wang JJ-M, Ku C-T, Bajko G, Anwyl GA, Feng S, Liu J, et al. MULTI-ACCESS POINT COORDINATED SPATIAL REUSE PROTOCOL AND ALGORITHM [Internet]. European Patent. 3 809 735 A1, 2021. Available from: https://data.epo.org/publication-server/document?iDocId=6519834

[8] Ak E, Canberk B. FSC: Two-Scale AI-Driven Fair Sensitivity Control for 802.11ax Networks. In: GLOBECOM 2020 - 2020 IEEE Global Communications Conference [Internet]. 2020. doi: 10.1109/GLOBECOM42002.2020.9322153

[9] He C, Hu Y, Chen Y, Fan X, Li H, Zeng B. MUcast: Linear Uncoded Multiuser Video Streaming With Channel Assignment and Power Allocation Optimization. IEEE Transactions on Circuits and Systems for Video Technology [Internet]. 2019;30(4):1136–46. doi: 10.1109/TCSVT.2019.2897649

[10] Zhang Y, Jiang C, Han Z, Yu S, Yuan J. Interference-Aware Coordinated Power Allocation in Autonomous Wi-Fi Environment. IEEE Access [Internet]. 2016;4:3489–500. doi: 10.1109/ACCESS.2016.2585581

[11] Zhao G, Li Y, Xu C, Han Z, Xing Y, Yu S. Joint Power Control and Channel Allocation for Interference Mitigation Based on Reinforcement Learning. IEEE Access [Internet]. 2019;7:177254–65. doi: 10.1109/ACCESS.2019.2937438

[12] Jones W, Eddie Wilson R, Doufexi A, Sooriyabandara M. A Pragmatic Approach to Clear Channel Assessment Threshold Adaptation and Transmission Power Control for Performance Gain in CSMA/CA WLANs. IEEE Transactions on Mobile Computing [Internet]. 2019;19(2):262–75. doi: 10.1109/TMC.2019.2892713

[13] Zhou C, Yang H, Duan X, Lopez D, Pastor A, Wu Q, et al. Digital Twin Network: Concepts and Reference Architecture [Internet]. IETF Datatracker. 2022 [cited 2022 Apr 24]. Available from: https://datatracker.ietf.org/doc/draft-irtf-nmrg-network-digital-twin-arch/00/

[14] Wu Y, Zhang K, Zhang Y. Digital Twin Networks: A Survey. IEEE Internet of Things Journal [Internet]. 2021;8(18):13789–804. doi: 10.1109/JIOT.2021.3079510

[15] Barricelli BR, Casiraghi E, Fogli D. A Survey on Digital Twin: Definitions, Characteristics, Applications, and Design Implications. IEEE Access [Internet]. 167653 - 167671;7. doi: 10.1109/ACCESS.2019.2953499

[16] Ak E, Canberk B. BCDN: A proof of concept model for blockchain-aided CDN orchestration and routing. Computer

Networks [Internet]. 2019;161:162-171. doi: 10.1016/j.comnet.2019.06.018

[17] Microsoft. DTDL models - Azure Digital Twins [Internet]. Microsoft Docs. 2022 [cited 2022 Apr 24]. Available from: https://docs.microsoft.com/en-us/azure/digital-twins/concepts-models

[18] ns-3 | a discrete-event network simulator for internet systems [Internet]. [cited 2022 May 6]. Available from: https://www.nsnam.org/

[19] Ak E, Canberk B. Forecasting Quality of Service for Next-Generation Data-Driven WiFi6 Campus Networks. IEEE Transactions on Network and Service Management [Internet]. 2021;18(4):4744-4755. doi: 10.1109/TNSM.2021.3108766