International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 8, Number 2, pages 254–281, July 2018

Dynamic Access-Point Configuration Approach for Elastic Wireless Local-Area Network System and Its Implementation Using Raspberry Pi

Md. Manowarul Islam, Nobuo Funabiki, Minoru Kuribayashi, Sumon Kumar Debnath, Kwenga Ismael Munene, Kyaw Soe Lwin, Rahardhita Widyatra Sudibyo
Department of Electrical and Communication Engineering, Okayama University
3-1-1 Tsushimanaka, Okayama 700-8530, Japan

Md. Selim Al Mamun
Department of Computer Science and Engineering, Jatiya Kabi Kazi Nazrul Islam University
Mymensingh, Bangladesh

Received: January 25, 2018
Revised: April 25, 2018
Accepted: June 2, 2018
Communicated by Yoshiaki Kakuda

Abstract

Recently, *Wireless Local Area Networks (WLANs)* have increased popularity around the world, where users can easily access to the Internet through associations with *access points (APs)* using mobile devices like smart phones, tablets, and laptops. In a WLAN, it is common that the number of users is always changing by time and users are not evenly distributed in the field. To optimize the number of active APs and the host associations in the network depending on traffic demands from users, previously, we proposed the *AP configuration algorithm* for the *elastic WLAN system*. Unfortunately, this algorithm can find the solution for the fixed user state in the network, although users often repeat joining and leaving the network. In this paper, we propose the extension of the *AP configuration algorithm* to deal with this dynamic nature. Here, as practical situations, this extension considers that at most one host may join or leave at the same time, and any communicating AP cannot be suspended and any communicating host cannot change its associated AP. Through numerical experiments using the *WIMNET simulator* in two network instances, the effectiveness of the proposal is demonstrated. Furthermore, it is implemented in the elastic WLAN system testbed using *Raspberry Pi* for the AP. The performance of this implementation is verified through experiments in four scenarios.

Keywords: Elastic WLAN system, host join/leave, access point configuration, testbed, Raspberry Pi

1 Introduction

Recently, a *Wireless Local Area Network (WLAN)* has become increasingly popular. The wireless communication between an *Access Point (AP)* and a host make a WLAN more flexible, scalable, and accessible than a wired LAN. Due to the characteristics of easy installations, flexible coverages areas, and low costs, WLANs have been commonly deployed around the world to provide the Internet access in various places, including airports, shopping malls, stations, and hotels [1][2]. Actually,

WLANs have become the most common ways for the Internet access in governments, companies, and educational institutes.

In a WLAN, APs are often installed and used in the service field randomly. It can cause poor network performances due to overlapping of the same frequency signals between them [3]. Therefore, the configuration of these APs should be properly managed according to the traffic demands in the field, where redundant APs should be turned off for energy saving and interference preventions.

In a WLAN, the distribution of users is often non-uniform [4], and the number of users or traffics is fluctuating depending on time and day of the week [2], which is often unpredictable [5]. In addition, the conditions of network devices and communication links may be influenced by various factors, such as power shortages, device failures, bandwidth controls by authorities, and even weather changes [6]. Actually, many developing countries including Bangladesh and Myanmar often suffer from the unreliable and slow Internet access due to discontinuities of electricity supplies for the time being [7][8]. To solve the abovementioned problems, we have studied the *elastic WLAN system* using heterogeneous AP devices [9][10]. In our studies of this system, three types of APs, namely *dedicated APs (DAPs)*, *virtual APs (VAPs)*, and *mobile APs (MAPs)*, have been considered. Figure 1 shows a simple topology of the elastic WLAN system. It dynamically controls the number of active APs according to traffic demands and device conditions. For the control of this system, we have proposed the *active AP configuration algorithm* that selects the minimum number of active APs in the network, satisfying the constraints of the throughputs for the hosts, and assigns one channel to every active AP from the limited number of the non-interfered channels in a way to minimize the overall interference in the network.

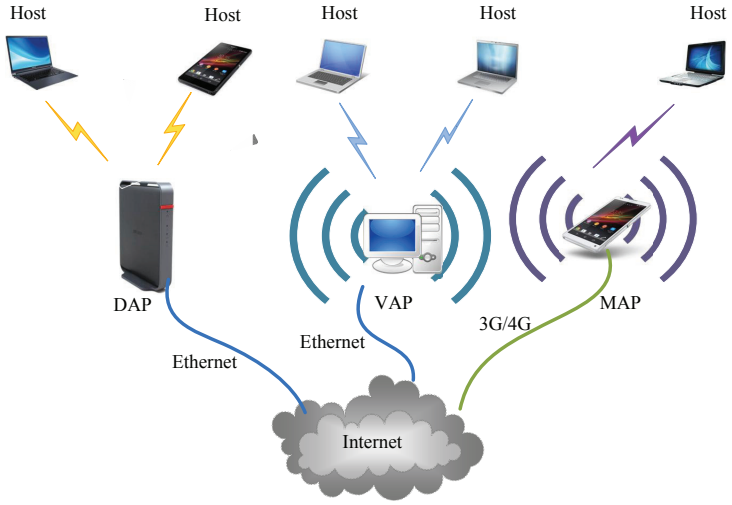


Figure 1: Overview of Elastic WLAN.

Unfortunately, this algorithm can only find the whole solution of all the active APs at the same time, with their associated hosts and assigned channels for a given state of the traffic demands and the device conditions in the network. Even if only one host joins or leaves the network, it can generate the totally new configuration without considering the current state. If some hosts are currently communicating with the Internet, they cannot be stopped to change the associated APs. However, in the elastic WLAN system, the hosts and the APs must be suspended for a while when their associations or assigned channels are changed. Thus, the new configuration must avoid this discontinuity of communicating services. In this paper, we propose the extension of the *AP configuration algorithm* to overcome the discontinuity problem in the current algorithm. In this extension, the currently communicating APs and hosts are designated as *communicating APs* and *communicating hosts* for convenience. Any *communicating AP* is not deactivated and any *communicating host* does not change the associated AP by the algorithm. Through numerical experiments using the *WIMNET simulator* [11] in two network instances, the effectiveness of the

proposed extension is demonstrated.

Furthermore, the proposed algorithm extension is implemented on the elastic WLAN system testbed to verify the practicality in the real system. In this testbed implementation, *Raspberry Pi* [12] is adopted for the AP and Linux PC is for the host. *Raspberry Pi* is a small-size low-cost computer, and has become popular in academics and industries around the world. Therefore, the use of Raspberry Pi in the elastic WLAN system is significant for its disseminations in developing countries. The performance of this elastic WLAN system testbed is evaluated through experiments in four scenarios, where the measured throughputs by the proposal are always higher than those by the comparison methods.

The rest of the paper is organized as follows: Section 2 describes related works. Section 3 reviews our previous works. Section 4 presents the extension of the active AP configuration algorithm. Section 5 evaluates the proposal through simulations. Section 6 describes the implementation of the elastic WLAN system testbed. Sections 7 evaluates the testbed implementation. Finally, Section 8 concludes this paper with some future directions.

2 Related works

In this section, we briefly show some related works to this paper.

[4] adopts a game theoretic approach to balance the loads among the APs. Users associated with highly congested APs are moved to less loaded APs for better throughputs. Although this approach can balance the load among APs, the user movement from one AP to another AP makes it impractical for real applications, because a host has to find another suitable AP and move to the place, if it cannot get the required data rate or service from the associated AP.

[13][14] propose a load balancing approach among APs to mitigate congestions and fair distributions of users. Each host monitors the wireless channel qualities that it experiences from nearby APs and reports them to the network control center that determines the host and AP associations. Since the objective function of the proportional fairness is non-linear, its implementation is much more challenging, where detecting the bottleneck users and finding their normalized bandwidth is NP-hard.

[15] provides an algorithm to place several *extension points* for cellular networks such that the throughput capacity of a WLAN is maximized. These extension points act as relay points between the APs and the hosts located in longer distances. Each extension point can receive the packets from the APs and deliver them to the hosts. In a conventional WLAN, the extension point is not used, because the multi-hop communications degrade the performance due to strong interferences.

[16] presents an AP association approach to improve the network throughput while balancing the loads among APs. The authors investigate and evaluate a measurement driven framework with three objective functions in a dense WLAN; (i) the frequency/channel selection across the APs to minimize the noise or interference between neighboring APs, (ii) the user association by considering the load of the AP and the receiving signal strength, and (iii) the power control for each AP. While each of the three objective functions can achieve its optimization objective independently, in some cases, to apply all of them (channel selection, user association and power control) can lead to the suboptimal performance. Actually, the authors found that if all of them are blindly applied, it can degrade the performance. A host is initially associated with the AP that provides the highest RSSI. Then, any host from an overloaded AP is migrated to a lightly loaded AP for balancing the load in the network. Although this may improve the user performance, they do not consider the current communicating services of active hosts that may suffer from packet losses during the change of associations. Besides, they do not consider the host leave operation to optimize the network performance in two testbeds, where they only consider the host join operation.

On the other hand, our algorithm considers both the host join and leave operations. For a newly joining host, it finds the association with the AP that can provide the best performance. If the current active APs in the network cannot satisfy the required throughput condition, some inactive APs are newly activated and the loads of the APs are balanced. Besides, even in such a case, the current communicating hosts do not change the AP associations, or any communicating

AP is not deactivated to avoid packet losses. At the host leave operation, our algorithm may deactivate the unnecessary APs to satisfy the required throughput condition that are not currently communicating, to reduce the interference and save the energy. [17] presents a heuristic algorithm for the AP placement based on the user distribution in an indoor WLAN system. The authors have proposed a fuzzy C-clustering-based AP deployment strategy to maximize the user coverage and balance the load among the APs.

[18] presents an AP selection algorithm to maximize the throughput for a newly joining host in the multi-rate WLAN by moving the host towards the location of the desired AP. Unfortunately, it does not consider the host leave operation to optimize the network performance. However, if a small move of a host in the network field can improve the throughput, this idea should be adopted in the elastic WLAN system, which will be in our future works. [19][20][21] focus on the highest received signal strength indicator (RSSI) to select the AP association. However, the AP with the highest RSSI can be overloaded by a number of users, and thus may not ensure the proper load balancing among APs. [22] presents a smart AP solution to balance the loads across multiple Wi-Fi interfaces. However, the implementation of this approach is difficult since each AP must be equipped with multiple interfaces operating on different channels simultaneously.

3 Review of Previous Works

In this section, we review our previous works. First, we discuss the topology and the behavior of the elastic WLAN system. Then, we describe the formulation of the active AP configuration problem and the procedure of the active AP configuration algorithm.

3.1 Elastic WLAN System

The elastic WLAN system consists of three types of APs, namely, the *Dedicated Access Point (DAP)*, the *Virtual Access Point (VAP)*, and the *Mobile Access Point (MAP)*. In this paper, a *DAP* implies a commercial AP that adopts the dedicated software/hardware for the AP functions and offers higher communications, a *VAP* does a virtual AP using a personal computer (PC) that adopts an open software for the AP functions, and a *MAP* does a mobile router that accesses to the Internet through a cellular network. Generally, a VAP offers slower communications than a DAP, and a MAP does the slowest among them. The elastic WLAN system dynamically controls the number of active APs by activating or deactivating APs according to the output of the active AP configuration algorithm. Figure 1 shows an example of this system. The implementation of the elastic WLAN system [10] adopts a server to manage and control the APs and the hosts with the administrative access to them. The server controls the system by the following three steps:

- The server explores the devices in the network and collects the necessary information for the input to the AP configuration algorithm.
- The server executes the AP configuration algorithm using the input derived in the previous step. The output contains the list of the active APs and the hosts associations.
- The server applies the algorithm output to the network configuration by activating or deactivating the specified APs and changing the specified host associations.

3.2 Network Environment

In our previous studies [23][24], we considered the *indoor network environment* inside a building as the target field for our proposal, because WLANs are usually used there. In such an environment, users use their personal computers (PCs) to access the Internet at fixed positions where chairs or tables are available. The mobility of WLAN users is much lower than that of cellular system users, because PCs are much larger and heavier than smart phones and often require the use of both hands. Thus, WLAN users connect with WLANs while sitting on chairs and putting their PCs on

tables. Sometimes, the hosts can be regularly distributed, since for example, students have their fixed positions in laboratories.

3.3 Objective of AP Configuration Algorithm

The objective of active AP configuration algorithm is to minimize the number of active APs in the elastic WLAN system while satisfying the minimum host throughput constraint, in order to minimize the power consumption and the network interference while offering the sufficient throughput performance. When the current active APs cannot satisfy the constraint, this algorithm newly activates the necessary inactive APs, and then, balances the loads among the active APs. If all the DAPs are activated, VAPs are activated if available. If the constraint is still not satisfied, MAPs are activated if available.

However, it can happen that the current elastic WLAN system does not satisfy the minimum host throughput constraint, even if all the APs are active, since it does not have the admission control to limit the number of hosts in the system. In this situation, it activates all the APs and gives up satisfying the constraint. In future works, the admission control will be adopted in the system.

3.4 Formulation of Active AP Configuration Algorithm

The active AP configuration problem was formulated as follows:

1. **Inputs**
   - Number of hosts: H
   - Number of APs: $N = N^D + N^V + N^M$. where N^D , N^V , and N^M respectively represent the number of DAPs, VAPs, and MAPs
   - AP ID: $i = 1$ to N
   - Host ID: $i = 1$ to H
   - Link speed of the ith AP to the jth host, s_{ij} ($i = 1$ to N, $j = 1$ to H): it can be estimated by the formula in [10].
   - Number of non-interfered channels
2. **Outputs :**
   - The set of active APs (DAPs, VAPs, and MAPs)
   - The set of hosts associated with each active AP
   - The assigned channel to each active AP
3. **Objectives :**
   - To minimize E_1
   - Holding the first objective, to maximize E_2

   The cost function E_1 represents the number of active APs (DAPs, VAPs and MAPs) in the network:

$$E_1 = E_D^1 + E_V^1 + E_M^1 \tag{1}$$

   where E_D^1 represents the number of active DAPs, E_V^1 does the number of active VAPs, and E_M^1 does the number of active MAPs respectively. The cost function E_2 is defined as follows:

$$E_2 = \min_j [TH_j] \tag{2}$$

where TH_j represents the average host throughput for the jth AP and is defined by:

$$TH_j = \frac{1}{\sum_k \frac{1}{s_{jk}}} \tag{3}$$

where s_{jk} represents the link speed between the jth AP and the kth host.

- The total interfered communication time E should be minimized.

$$E = \sum_{i=1}^{N} [IT_i] = \sum_{i=1}^{N} \left[\sum_{\substack{k \in I_i \\ c_k = c_i}} T_k \right] \tag{4}$$

where IT_i represents the interfered communication time for AP_i, T_i does the communication time for AP_i, I_i does the set of interfered APs for AP_i, and c_i does the assigned channel to AP_i.

4. **Constraints:**
   - *Minimum host throughput constraint:* every host must enjoy the given threshold G on average when all the hosts are communicating simultaneously.
   - *Bandwidth limit constraint:* the bandwidth of the wired network to the Internet must be less than or equal to the total available bandwidth of the network B_a.
   - *Channel assignment constraint:* every active AP must be assigned one channel.

3.5 Procedure of Active AP Configuration Algorithm

The procedure of the active AP configuration algorithm is described here. Due to the space limitation, the details are omitted here. Please refer [10].

3.5.1 First Phase: Active AP Selection and Host Association

The first phase of the algorithm selects the active APs and the associated hosts.

(1) **Preprocessing:** The link speed for each possible pair of an AP and a host is estimated using the signal strength of the host from the AP. Then, this phase initializes the variables for the next phases.

(2) **Initial Solution Generation:** An initial solution E_1 is derived using a greedy algorithm [25].

(3) **Host Association Improvement:** The cost function E_2 is calculated for the greedy solution using Eq. (2). Then, this solution is improved.

(4) **AP Selection Optimization:** The cost functions E_1 and E_2 are further jointly optimized in this phase by the *local search* [9][26] under the constraints mentioned before.

(5) **Link Speed Normalization:** The fairness criterion is applied here when the total expected bandwidth exceeds B^a. Then, the link speed is normalized.

(6) **Termination Check:** When either of the following conditions is satisfied, go to the second phase in 3.5.2:
   (a) The *minimum host throughput constraint* is satisfied.
   (b) All the APs in the network have been activated.

(7) **Additional VAP Activation:** If VAPs are not selected for candidate APs, they are selected as candidate APs, and go back to Step (3).

(8) **Additional MAP Activation:** If MAPs are not selected for candidate APs, they are selected as candidate APs. The locations of hosts are considered as the locations of the candidate MAPs, and go back to Step (3).

3.5.2 Second Phase: Channel Assignment

The second phase of the algorithm assigns a channel to each active AP.

(1) **Preprocessing:** The interference and delay conditions of the network are represented by a graph.

(2) **Interfered AP Set Generation:** The set of APs that are interfering with each other is found for each AP.

(3) **Initial Solution Construction:** An initial solution is derived using a greedy algorithm.

(4) **Solution Improvement by Simulated Annealing:** The initial solution is improved by the simulated annealing (SA) procedure with the constant *SA temperature* T^{SA} for the *SA repeating times* R^{SA}, where T^{SA} and R^{SA} are given algorithm parameters [10].

3.5.3 Third Phase: Channel Load Averaging

The third phase of the algorithm averages the load among the different channels.

(1) **Initialization:** The *AP flag* is initialized by $0(= OFF)$ for every AP. This flag is used to avoid processing the same AP again.

(2) **AP Selection:** One OFF flag AP is selected to move its associated host to a different AP that is assigned a different channel.

(3) **Host Selection:** One host associated with the selected AP is selected for the AP movement.

(4) **Change Application:** Finally, the new associated AP is selected for this selected host.

4 Extensions of Active AP Configuration Algorithm

In this section, we present the extensions of the active AP configuration algorithm to consider the currently communicating APs and hosts.

4.1 Definitions of Terms

In the current elastic WLAN system implementation, any host needs to suspend the operation for about 60 seconds when it changes the associated AP. It should be avoided for a host that is currently communicating with the Internet. In the extensions, any *communicating host* does not change the associated AP to ensure the continuous Internet service, while any *communicating AP* is not deactivated. For the algorithm extension, we introduce the following terms to describe the states of the network and simulations:

- **joining host** is a host that newly joins the network to be associated with an active AP for the Internet access. The algorithm must find the associated active AP for the joining host.
- **leaving host** is a host that leaves the network after completing the Internet service.
- **communicating host** is a host that is currently communicating with the Internet. The algorithm cannot change the associated AP to avoid the communication suspension.
- **communicating AP** is an AP that is currently associated with a communicating host. The algorithm cannot stop this AP to avoid the communication suspension.
- **active host** is a host that is currently associated with an active AP.
- **inactive host** is a host that is not associated with an active AP.

4.2 Algorithm Extension for Joining Host

First, we present the algorithm extension for one *joining host*. When a new host joins the network, the associated active AP must be selected for the *joining host* such that the constraints in subsection 3.4 are satisfied. Here, the selected active AP must satisfy the *minimum host throughput constraint* when the following *normalized link speed* is considered if $B^a < B^e$:

$$\hat{s}_{jk} = s_{jk} \times \frac{B^a}{B^e} \tag{5}$$

where B^a and B^e represent the available and expected total bandwidth respectively, s_{jk} represents the link speed between the jth AP and the kth host, and $\hat{s}_{jk}$ does its normalized link speed. Then, the *average host throughput* (TH_j) using the normalized link speeds is calculated for each active AP (jth AP) when the *joining host* is associated with it:

$$TH_j = \frac{1}{\sum_k \frac{1}{\hat{s}_{jk}}} \tag{6}$$

where TH_j represents the *average host throughput* for the jth AP. If $TH_j \geq G$, the *minimum host throughput constraint* is satisfied, where G represents the threshold for this constraint. If multiple active APs can satisfy it, the AP that maximizes the E_2 defined in Eq. (2) is selected. On the other hand, if any active AP cannot satisfy the constraints for the *joining host*, this host is first associated with the active AP that maximizes the E_2 defined in Eq. (2), and the active AP configuration algorithm in the previous section is applied from *AP Selection Optimization* (Step (4) in Section 3.5.1). Here, to avoid the suspensions of the currently communicating hosts, the two modifications are considered: 1) any *communicating AP* is not deactivated in *AP Selection Optimization*, and 2) any *communicating host* does not change the associated active AP in *Host Association Improvement* (Step (3) in Section 3.5.1) and in *AP Selection Optimization*.

4.3 Algorithm Extension for Leaving Host

Next, we present the algorithm extension for one *leaving host*. When an existing host leaves the network, it is only necessary to remove this *leaving host* from the associated active AP. Then, if the AP is not associated with any host, this AP is deactivated. Otherwise, to minimize the number of active APs and improve the host associations, the active AP configuration algorithm is applied from *AP Selection Optimization* (Step (4) in Section 3.5.1) with the two modifications for the *joining host*.

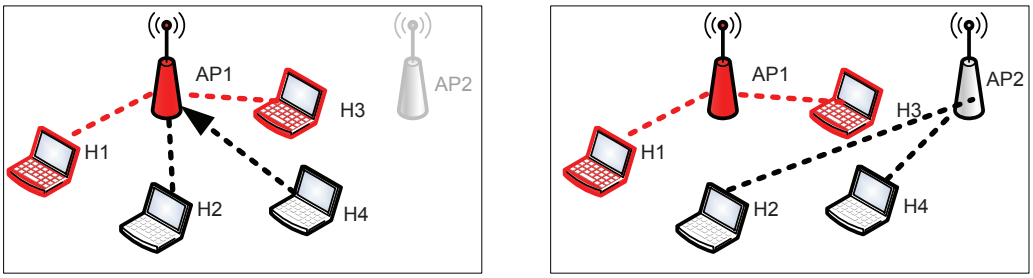
4.4 Examples of Host Join and Leave Operations

In this paper, it is assumed that at most one host may join or leave the network at each time. First, the host join operation in Figure 2 is discussed. When H4 joins the network, it newly activates AP2 to satisfy the minimum host throughput constraint. Then, to balance the loads between AP1 and AP2, H2 changes the associated AP from AP1 to AP2, because H3 cannot change the associated AP as the communicating host.

Then, the host leave operation in Figure 3 is discussed. When H4 leaves the network, if H3 changes the associated AP to AP1, it can deactivate AP2 while satisfying the minimum host throughput constraint. However, H3 cannot change the associated AP as the communicating host, and AP2 cannot be deactivated.

5 Evaluations by Simulations

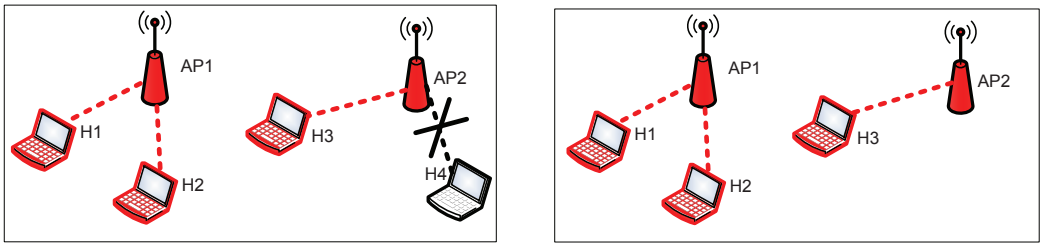
In this section, we evaluate the algorithm extensions through simulations using the *WIMNET simulator* [11].



(a) Before H4 join

(b) After H4 join

Figure 2: Host join operation.



(a) Before H4 leave

(b) After H4 leave

Figure 3: Host leave operation.

5.1 Host Behavior Model

In our simulations, it is assumed that a host randomly joins and leaves the network by following the *Poission* distribution, and a host continues the communication with the Internet during the time derived by the *Exponential* distribution. In each simulation in this paper, at the first, $N1$ hosts are randomly selected from all the N hosts in the network for $N1 < N$ as *active hosts*, and the original active AP configuration algorithm is applied to the $N1$ active hosts so that the active APs and the associated active hosts are obtained. For each active host, the communication duration time is calculated by the following equation:

$$F(p, \mu) = \frac{-ln(1-p)}{\mu} \tag{7}$$

where p is $0-1$ random number and μ is the parameter for the *Exponential* distribution. Then, every time the constant time Td is elapsed, the *communicating host* is maintained if the communication duration time is not over since the arrival to the network. Here, $Td = 10sec$ is used in this paper. Besides, the number of newly joining hosts from *inactive hosts* and the number of leaving hosts from *active hosts* are calculated here by the following procedure:

1. Calculate the probability to select each number k by the *Poission* distribution:

$$F(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{8}$$

   where λ is the parameter for the *Poission* distribution.

2. Calculate the cumulative probability $C(k, \lambda)$ for each number k by:

$$C(k, \lambda) = \sum_{i=0}^{k} \frac{\lambda^i \cdot e^{-\lambda}}{i!} \tag{9}$$

3. Generate a $0-1$ random number p.
4. Find k such that $C(k,\lambda) \leq p < C(k+1,\lambda)$.

Then, the corresponding number of joining hosts are randomly selected from *inactive hosts*. For each newly joining host, the communication duration time is calculated by Eq. (7).

5.2 Simulation Platform

In simulations, we adopt the hardware and software in Table 1. For the performance comparison, the

Table 1: Simulation environment.

simulator	WIMNET Simulator
interface	IEEE 802.11n
CPU	Intel Core i7
memory	4GB
OS	Ubuntu LTS 14.04

nearest AP association approach (NAP) is considered as the simple method for AP-host associations since NAP can offer the fastest link speed between an AP and a host when other hosts are not associated with the AP, where the active APs are selected by our algorithm. Table 2 summarizes the simulation parameters that are used for the simulation using WIMNET simulator.

Table 2: Simulation parameters for WIMNET simulator.

parameter	values
packet size	2360 bytes
max. transmission rate	150 Mbit/s
propagation model	log distance path loss model
rate adaptation algorithm	link speed estimation model [9]
carrier sense threshold	-85 dBm
transmission power	19 dBm
collision threshold	10
RTS/CTS	yes

5.3 Evaluation of Small Topology

Firstly, we evaluate our proposal using a small-size network topology.

5.3.1 Small Topology

The *small topology* in Figure 4 is first considered for simulations, where 30 hosts, 4 DAPs are distributed in the $100m \times 50m$ rectangular area. The circles and squares represent the APs and hosts respectively. The minimum host throughput threshold $G = 10Mbps$ and the bandwidth limit constraint $B^a = \infty$ are examined. As the host behavior model in this topology, $N1 = 18$ hosts are randomly selected as initial *active hosts* from $N = 30$ hosts, and $\lambda = 1$ and $\mu = .02$ are adopted.

5.3.2 Results

Figure 5 (a) shows the changes of the number of active hosts and the number of communicating active APs by the host behavior model, the number of active APs by the proposed algorithm, and the least number of APs required to satisfy the *minimum host throughput constraint* (ideal APs), when the simulation time elapsed. The number of ideal APs is calculated by $G \times N1/AT$. AT represents the average total throughput for one AP that is given by the throughput at the average

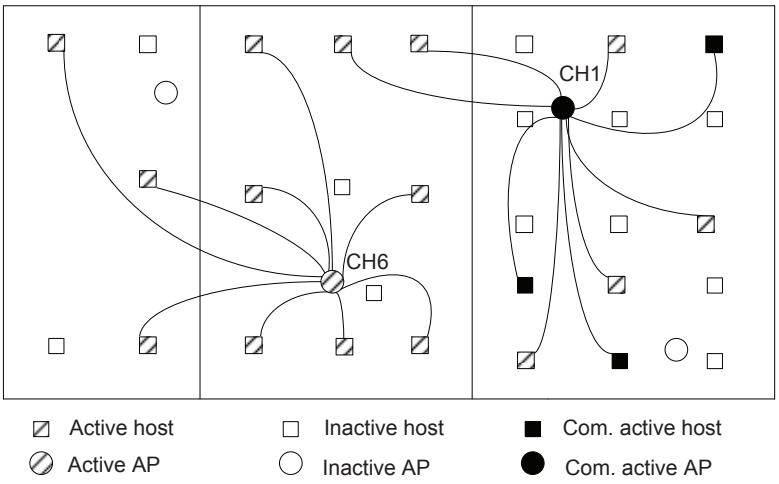
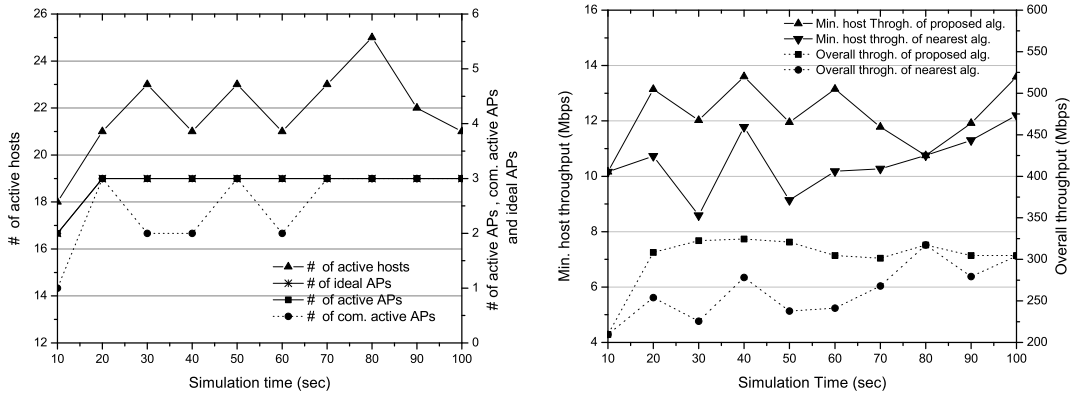


Figure 4: Solution for small topology with 30 hosts and 4 APs.



(a) Change of active hosts and APs

(b) Change of min. host and overall throughputs

Figure 5: Performance graph for small topology.

least distance with each host in the network. This graph indicates that the number of active APs is dynamically controlled by the proposed algorithm depending on the number of active hosts. Figure 5 (b) shows the changes of the minimum host throughput and the overall throughput by the WIMNET simulator for the AP-host association by the proposal and for that by the compared NAP. This graph indicates that the minimum host throughput by the proposal is always larger than the threshold $G = 10Mbps$ whereas that by the comparison is sometimes smaller than the threshold. Besides, the overall throughput by the proposal is larger than that by the comparison at most of the simulation time.

Table 3 compares the average minimum host throughput and overall throughput by the two methods. This table also indicates that the performance by the proposal is better than that by the comparison.

Table 3: Throughput comparison between two methods for both topology.

instance	small topology		large topology	
method	proposed	compare	proposed	compare
ave. min. host throughput (Mbps)	12.21	10.51	11.53	10.31
ave. overall throughput (Mbps)	301.81	261.47	450.70	407.69

5.4 Evaluation of Large Topology

Secondly, we evaluate our proposal using a large-size network topology.

5.4.1 Large Topology

The *large topology* in Figure 6 is then considered for simulations, which basically models the third floor of Engineering Building-2 in Okayama University. 70 hosts and 10 DAPs are regularly distributed in the six rooms with two different sizes, $7m \times 6m$ and $3.5m \times 6m$. The same minimum host throughput constraint $G = 10Mbps$ and bandwidth limit constraint $B^a = \infty$ are examined. As the host behavior model in this topology, $N1 = 35$ hosts are randomly selected as initial *active hosts* from $N = 70$ hosts, and $\lambda = 1.5$ and $\mu = .02$ are adopted.

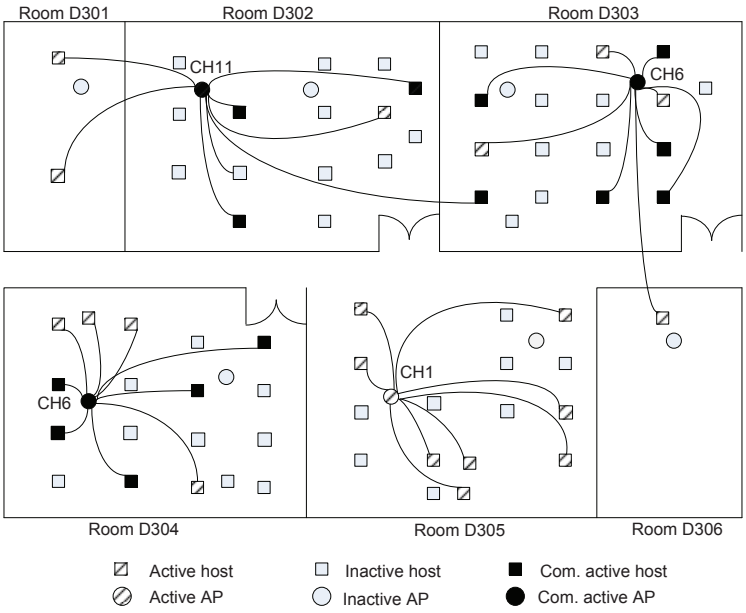
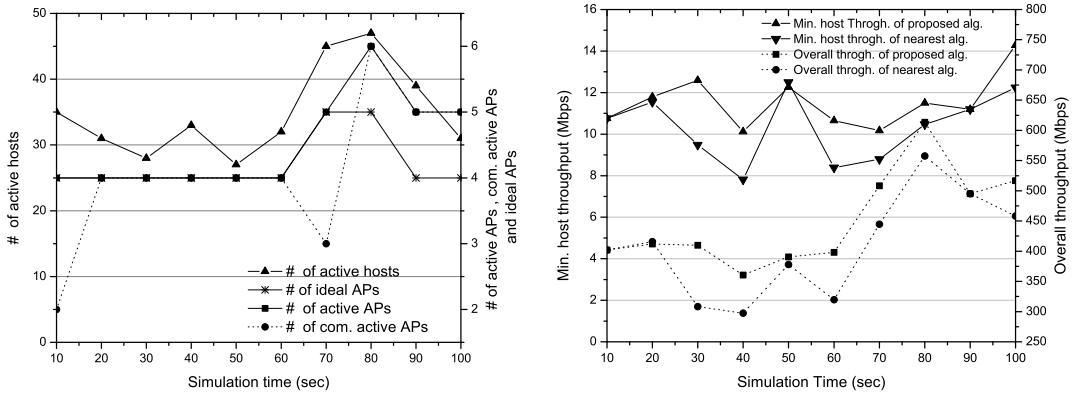


Figure 6: Solution for large topology with 70 hosts and 10 APs.



(a) Change of active hosts and APs

(b) Change of min. host and overall throughputs

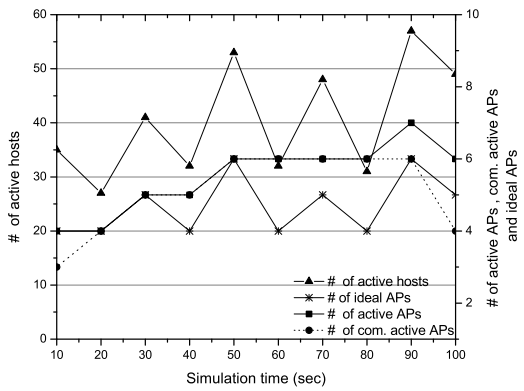
Figure 7: Performance graph for large topology.

5.4.2 Results

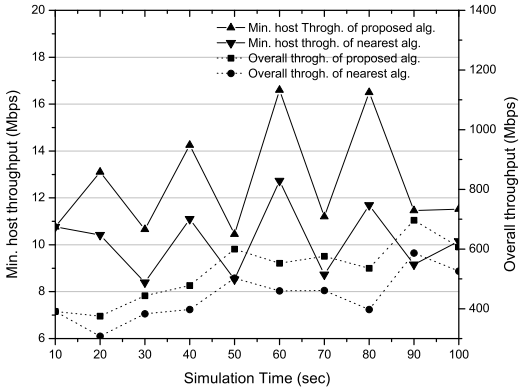
Figure 7 (a) shows the changes of the number of active hosts and the number of communicating active APs by the host behavior model, and the number of active APs by the proposed algorithm, when the simulation time elapsed. Figure 7 (b) shows the changes of the minimum host throughput and the overall throughput by the WIMNET simulator for the AP-host association by the proposal and for that by the compared NAP. From them, the same performance results can be observed for the large topology as for the small topology.

5.4.3 Results for Larger λ

In some networks, the number of users can be more widely fluctuated by time. For example, at public WLANs in a station or an airport, a lot of users may arrive there by a train or an airplane and join the network at the same time. They often stay there for a short time. To simulate such cases, the larger value of $\lambda = 4$, 5 and $\mu = .05$ are adopted. Figures 8 (a) and 9 (a) show the changes



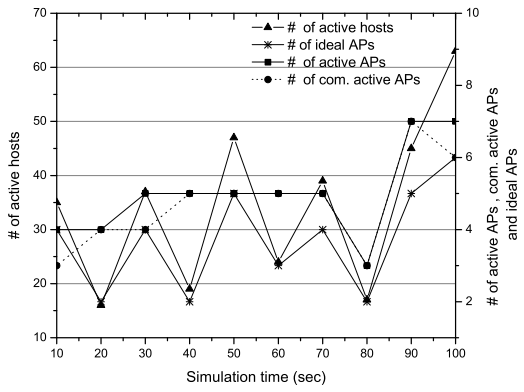
(a) Change of active hosts and APs



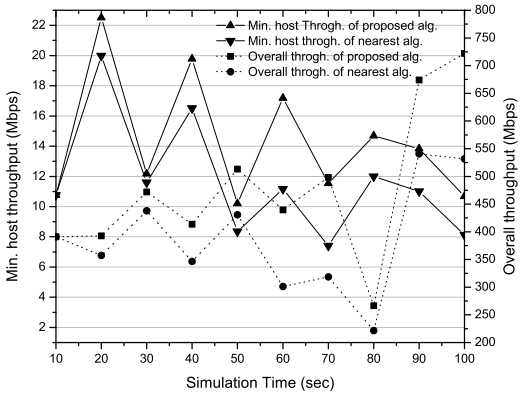
(b) Change of min. host and overall throughputs

Figure 8: Performance graph for large topology with $\lambda = 4$.

of the number of active hosts and the number of communicating active APs and the number of active APs for each λ respectively. Figures 8 (b) and 9 (b) indicate that by the proposal, the minimum host throughput is always larger than the threshold $G = 10Mbps$ and the overall throughput is larger than that of the comparison. Table 4 compares the average minimum host throughput and overall throughput between the two methods. This table indicates that the performance by the proposal is better than that by the comparison.



(a) Change of active hosts and APs



(b) Change of min. host and overall throughputs

Figure 9: Performance graph for large topology with $\lambda = 5$.

Table 4: Throughput comparison between two methods for large λ.

instance	$\lambda = 4$		$\lambda = 5$	
method	proposed	compare	proposed	compare
ave. min. host throughput (Mbps)	12.65	10.16	14.33	11.70
ave. overall throughput (Mbps)	525.39	441.21	478.29	387.73

5.4.4 Results for Moving Hosts

As described before, this study assumes that WLAN users usually use PCs on fixed positions where desks or chairs are available. However, users may move to different positions in the network field if the new positions are better for users. This move of a host to a different position can be represented by a pair of the leave operation from the current position and the join operation at the new position. Thus, when a host move appears, basically, the algorithm extension for a leaving host and that for a joining host are applied sequentially. However, if the host moves to a near position from the current one, it can keep the same associated AP if the minimum host throughput constraint is satisfied. In this case, the proposed extension is not applied, where only the link speed is updated.

To consider moving hosts, we simulated the proposal with them for the *large topology*. Here, $\lambda = 1.5$ is used for the parameter of joining and leaving hosts, $\mu = .02$ is for the parameter of the communication duration time, and 10% is for the communicating host selection probability. At the time hosts may join or leave the network, the moving hosts are randomly selected from non-communicating hosts with the fixed probability, where 0%, 5%, 10%, and 15% are used for this probability. The selected moving host is moved to a randomly selected host position instantly. Table 5 summarizes the simulation results. The throughput performance is similar in any case. Therefore, our proposal is effective for the moving hosts in the network field.

Table 5: Simulation results for moving hosts.

parameters	results			
moving host selection probability	0%	5%	10%	15%
avg. # of active APs	4.54	4.54	4.58	4.54
avg. # of moving hosts	0	1.86	3.70	5.40
avg. # of communicating hosts	3.38	3.34	3.44	3.36
avg. min. host throughput (Mbps)	11.32	11.16	11.34	11.33
avg. overall throughput (Mbps)	452.44	448.22	446.95	443.50

5.4.5 Comparisons with Previous Study

To clarify the effectiveness of the proposed algorithm extensions, we simulate the cases when the rate of the communicating hosts among the hosts is set to 25% and 50%, and compare the number of the active APs, the minimum host throughput, the total throughput, and the number of hosts changing the associated APs between the previous algorithm and the proposed one. It is noted that as the number of hosts changing the associated APs increases, the number of suspending hosts to change the associated APs increases.

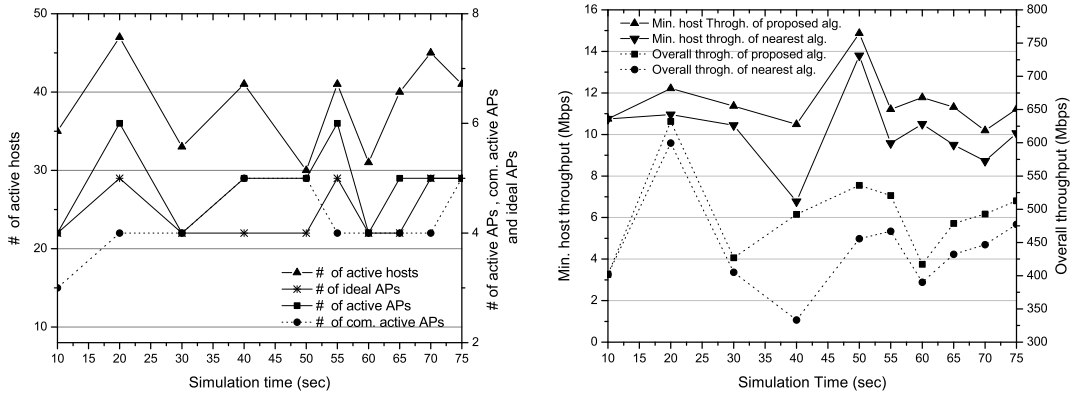
Table 6 shows the results when the rate of the communicating hosts is set to 25% and 50%. The network load is changed where the number of active hosts is set to 41 on average, to take the average values of the minimum host throughput and the overall throughput for each case. It is found that the average values of the both throughputs are similar between the two algorithms, while 37.5% or 41.2% hosts must change the associated APs while communicating, which can increase the network management overheads and suspend the Internet services for them.

Table 6: Comparisons of performances between proposal and previous.

simulation cases		results	
rate of com. host	parameters	proposal	previous
25%	avg. # of active APs	5.00	5.00
	avg. min. host throughput (Mbps)	10.96	10.97
	avg. overall throughput (Mbps)	487.45	480.81
	avg. rate of AP change hosts among com. hosts	0%	37.5 %
50%	avg. # of active APs	6.00	6.00
	avg. min. host throughput (Mbps)	12.31	13.03
	avg. overall throughput (Mbps)	562.74	568.98
	avg. rate of AP change hosts among com. hosts	0%	41.2%

5.5 Discussions

The simulation results show that at any situation, the extended algorithm can minimize the number of active APs while the minimum host throughput is maintained, and the overall throughput is better than the nearest AP association approach. They confirm the effectiveness of the proposal.



(a) Change of active hosts and APs

(b) Change of min. host and overall throughputs

Figure 10: Performance graph for large topology with network parameter change.

In Figure 5 (a), the number of active APs and the number of ideal APs are always same. In others, they are sometimes different by 1 or 2 because of the communicating APs that cannot be deactivated. Furthermore, in order to see the performance of the extended algorithm when important parameters in the host behavior model are changed during the simulation, another simulation is conducted, where Td is changed from $10sec$ to $5sec$ and λ is changed from 3 to 2 when $50sec$ is elapsed. As shown in Figure 10, our algorithm can adapt to the change of the network parameters immediately.

The results in Section 5.4.5 show that our proposal can avoid the suspensions of communicating host due to the change of the associated APs. The host mobility results in subsection 5.4.4 also show that the proposed algorithm can improve the network performance similar to related work [18]. Furthermore, by considering host leaving from the network, this algorithm deactivates unnecessary APs to save energy and reduce interferences. On the whole, the proposed extension exhibits dynamic feature of real time WLAN system and outperform all the other algorithms.

6 Elastic WLAN System Implementation Using Raspberry Pi

In this section, we present the implementation of the elastic WLAN system testbed using *Raspberry Pi* for the dynamic AP configuration approach. Appendix A explain how to configure *Raspberry Pi* for AP using hostapd daemon [27][28].

6.1 System Topology

Figure 11 shows the simple network topology of the elastic WLAN system. *Raspberry Pi* is used for the AP and a *Linux laptop PC* is for the server and the host. The server can manage and control all the APs and the hosts by using the administrative access to them. The APs are connected to the server through wired connections. The hosts and the APs are connected through wireless connections.

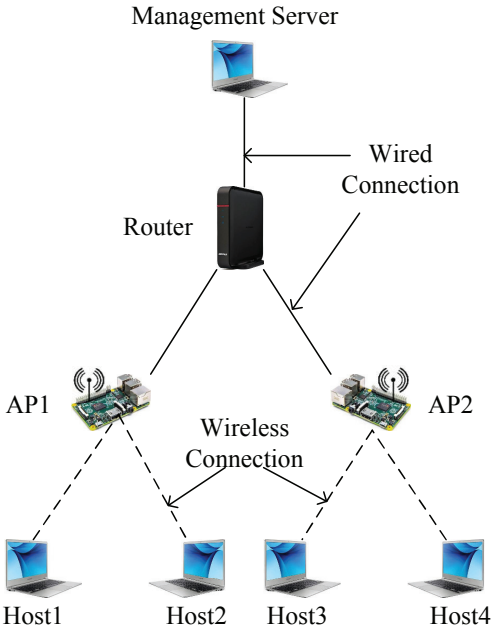


Figure 11: Elastic WLAN system topology.

6.2 Execution Flow of Elastic WLAN for Dynamic Approach

Figure 12 shows the execution flow of the elastic WLAN system testbed. Previously, it had six steps of steps 1-6 [10]. To deal with dynamic behaviors of joining and leaving hosts, steps 7-10 are added here. The following section describes, how the server can detect communicating APs and hosts. All the steps (step 7 $\sim$ step 10) are moved to Appendix B with detail operations.

6.3 Detection of Communicating AP and Host

The AP association of any host that currently has the Internet service, must be continued even after the algorithm output application. To detect such communicating APs and hosts, the server inspects the packets that are exchanged between the host and the AP, using *tcpdump* [29]. The *tcpdump* is a powerful network debugging tool and command-line packet analyzer on a network interface. The server collects the packets between the active hosts and their associated active APs, and detects the

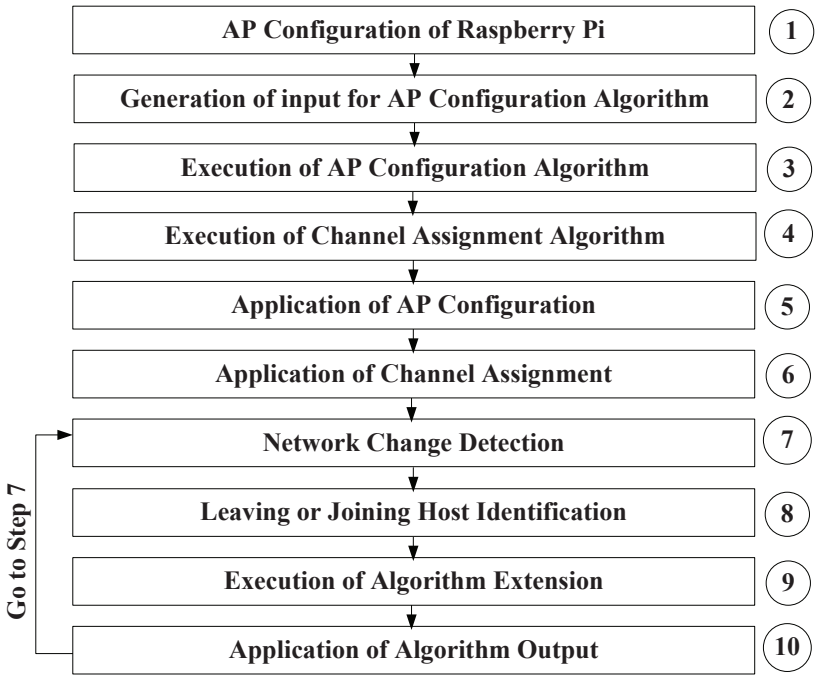


Figure 12: Elastic WLAN system execution flow for Dynamic AP Configuration.

hosts and APs that are currently communicating from them. The following commands are used in this step:

Linux commands for communicating hosts and APs dectection

```
#/bin/bash
# to inspect packets exchange between hosts and APs
01: timeout 50 tcpdump -i wlan0 host ip
# analyze and detect Com. APs and hosts
02: ./ckcomap tcppacket.txt
```

The command in `01` inspects the TCP packets between the APs and their associated hosts for $50sec.$ using *tcpdump*. Here, *host ip* represents the IP address of any host connecting to any active AP. Then, a C program is executed to filter and analyze the packets. The server detects the communicating hosts and APs using the command in `02`.

7 Evaluations by Testbed

In this section, we evaluate the implemented elastic WLAN system testbed using four network scenarios in our university.

7.1 Devices and Software

The devices and software in Table 7 are used for throughput measurements using the testbed in real network environments. The IEEE 802.11n protocol is used for any communication link with the channel bonding.

Table 7: Devices and Software used in the testbed.

Devices and Software		
server PC	OS	Ubuntu LTS 14.04
	model	Lesance W255HU
	Processor	Intel(R), Core(TM)-i3
client PC (type-1)	OS	Ubuntu LTS 14.04
	Model	Toshiba Dynabook R731/B
	Processor	Intel(R), Core-i5
client PC (type-2)	OS	Ubuntu LTS 14.04
	Model	Fujitsu Lifebook S761/C/SSD
	Processor	Intel(R), Core-i5
access point	Raspberry Pi 3	
	OS	Raspbian
	Processor	1.2 GHz
software/tools	openssh	to access remote PC and AP [30][31]
	hostapd	to prepare and configure AP
	nmcli	for association change [32][33]
	nm-tool	to measure signal strength [34][35]
	tcpdump	to analyze packets
	arp-scan	to discover active network devices [36]

7.2 Two Comparison Methods

To evaluate the throughput performance by our proposal, the two simple methods for selecting active APs and host associations are considered to be compared.

7.2.1 Comparison Method 1 (COMP-1)

To compare the throughput results in various network scenarios, a simple comparison method (COMP-1) is adopted. In this method, the same number of APs is activated in the network by randomly selecting them from available APs, where the channel is assigned by our algorithm. For any newly joining host, the host is associated with the AP that provides the highest RSSI from the active APs.

7.2.2 Comparison Method 2 (COMP-2)

As another comparison method (COMP-2), the active APs are also selected by the RSSI to the joining host. For any joining host, if the number of active APs is smaller than the algorithm, the AP that provides the highest RSSI is newly activated, and the host is associated with it. Again, for any AP, the channel is assigned by the algorithm.

7.3 Network Scenarios

For evaluations, four network scenarios are prepared for the elastic WLAN system testbed. For each AP, one of the three bonded channels, 1+5, 4+8, and 7+11, is assigned by the proposed algorithm.

7.3.1 3×4 Scenario in One Room

In the first scenario, three *Raspberry Pi* devices for APs and four Linux PCs for hosts are prepared in a room of size $7m \times 6m$. Figure 13 shows the distance between the hosts and APs. Any access point is connected to the server using the wired connection.

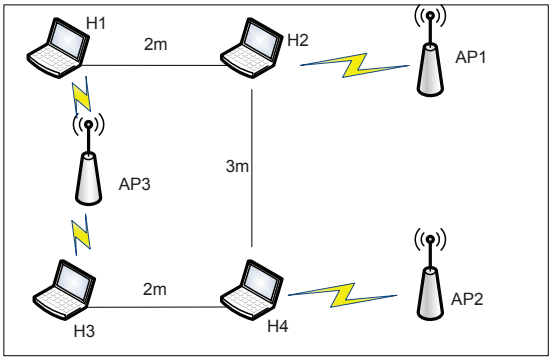


Figure 13: Testbed for 3×4 scenario in one room.

7.3.2 3×4 Scenario in Different Rooms

In the second scenario, three *Raspberry Pi* devices for APs and four Linux PCs for hosts are prepared in two rooms with the size of $7m \times 6m$ separated by the wall and one corridor at the third floor of Engineering Building-2 in Okayama University. As shown in Figure 14, any AP is $5m - 6m$ away from another AP in the different room and corridor to reduce the interference.

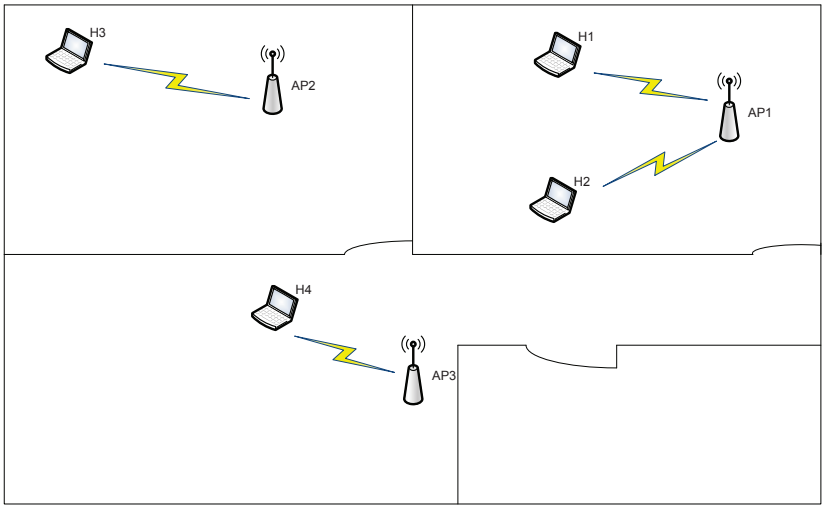


Figure 14: Testbed for 3×4 scenario in different rooms.

7.3.3 3×6 Scenario

In the third scenario, three *Raspberry Pi* devices for APs and six Linux PCs for hosts are placed in the same field, as shown in Figure 15.

7.3.4 4×8 Scenario

In the fourth scenario, four *Raspberry Pi* devices for APs and eight Linux PCs for hosts are distributed in the three rooms and the corridor as shown in Figure 16 at the second floor of Graduate School Building in Okayama University. The size of each room is $9m \times 5.5m$, $3.5m \times 5.5m$, and $7m \times 5.5m$ respectively. Any AP is $4m$ away from another AP in the same room.

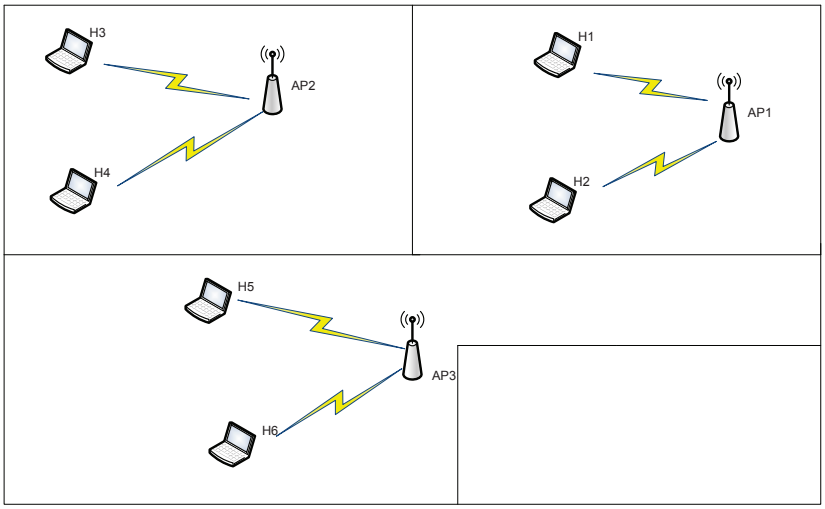


Figure 15: Testbed for 3×6 scenario.

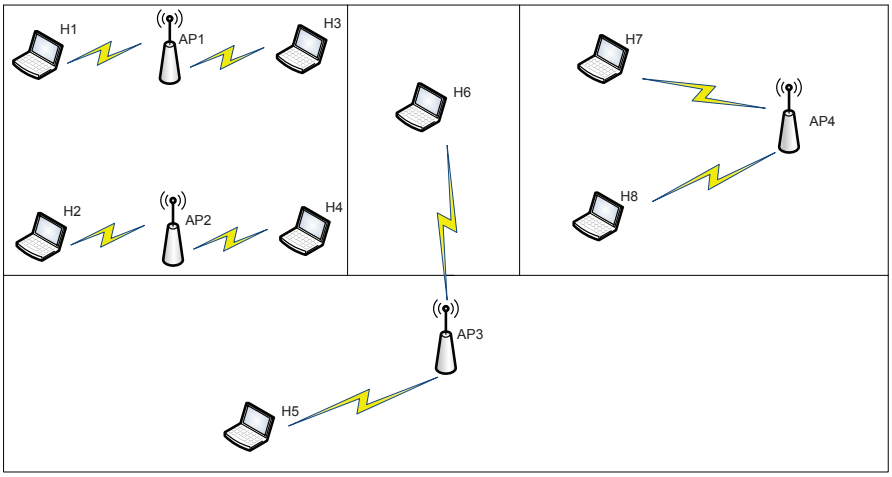


Figure 16: Testbed for 4×8 scenario.

7.4 Host Join/Leave Dynamics

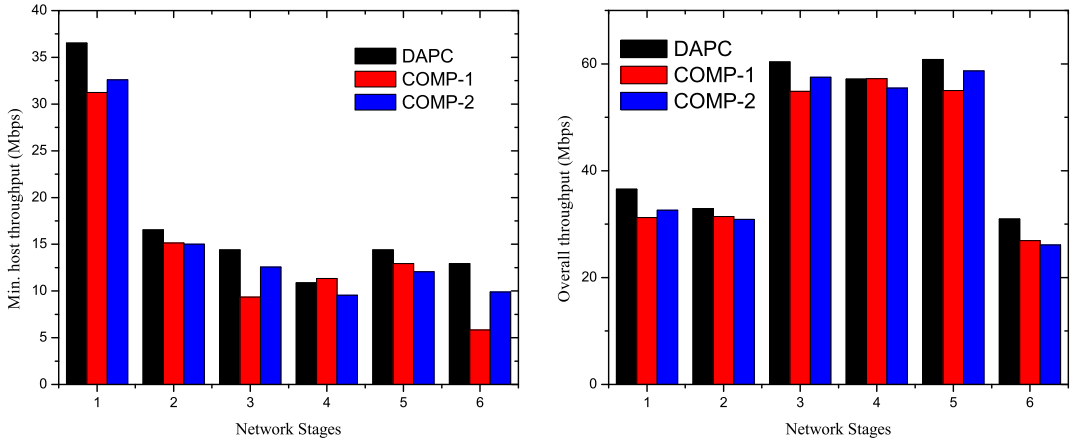
In each scenario, the host join/leave dynamics in the network are represented by a sequence of stages. At each stage, 1) one host joins or leaves the network, 2) Steps 7-10 in Section 6.2 are executed, and 3) the throughputs of all the active hosts are measured when they are concurrently communicating with the *iperf* [37] server through the associated APs. By following the host behavior model in 5.1, the joining and leaving hosts are randomly selected for each network stage with $\lambda = 1$ and $\mu = .02$.

7.5 Throughput Measurement Results

For each scenario, the throughputs at each stage are measured and compared.

7.5.1 3×4 Scenario in One room

Figure 17 (a) and (b) show the minimum host throughput results and the overall throughput results in the testbed for the 3×4 scenario in one room, by the proposal, by COMP-1, and by COMP-2 at each stage, where the number of active hosts is changed from 1 to 4. Except for the minimum host throughput at stage 4, our proposal always provides the better performance than COMP-1.



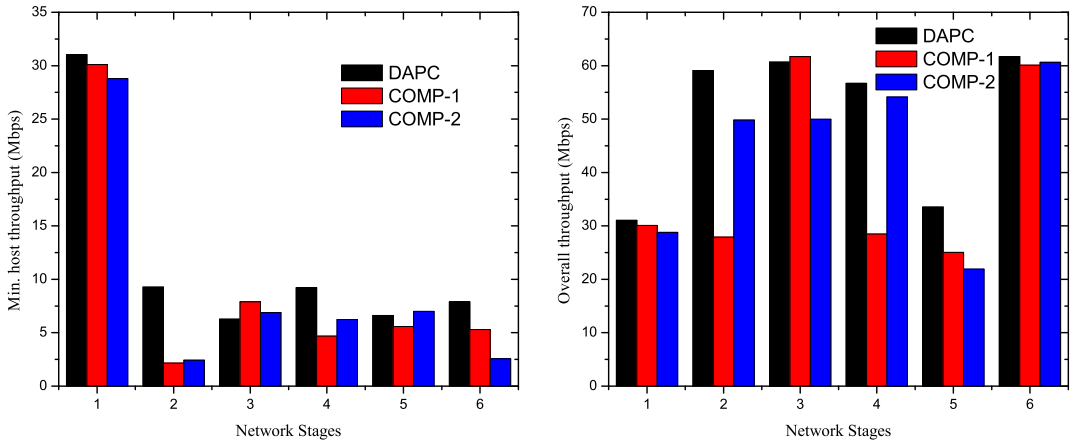
(a) Minimum host throughput (b) Overall throughput

Figure 17: Throughput results for 3×4 scenario in one room.

7.5.2 3×4 Scenario in Different Rooms

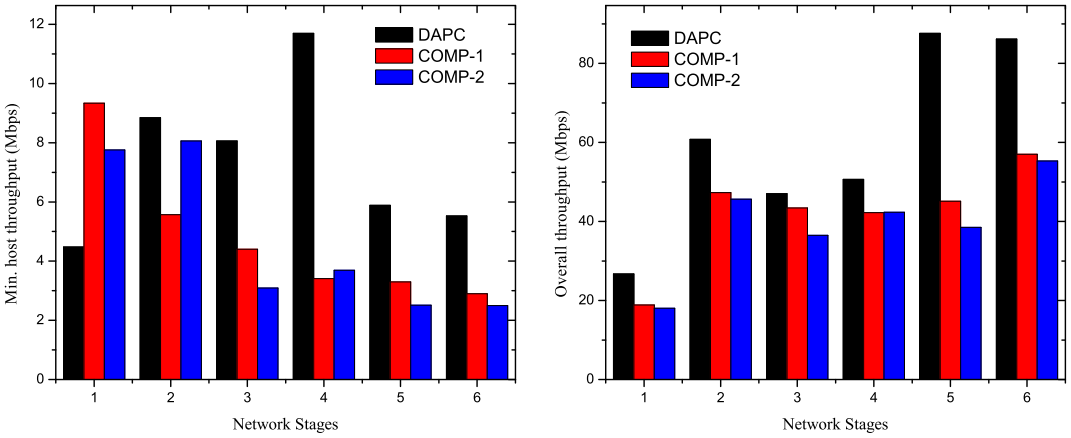
Figure 18 (a) and (b) show the minimum host throughput results and the overall throughput results in the testbed for the 3×4 scenario in different rooms, by the proposal, by COMP-1, and by COMP-2 at each stage, where the number of active hosts is changed from 1 to 4. Except for the minimum host throughput at stages 3, 5 and the overall throughput at stage 3, our proposal always provides the better performance than COMP-1 and COMP-2.

It can be observed that the minimum host throughput becomes lower at any stage than that in one room case. Here, since a host is connected to an AP in a different room, the RSS at such a host from the AP becomes smaller due to the wall attenuation, and thus, the throughput becomes lower.



(a) Minimum host throughput (b) Overall throughput

Figure 18: Throughput results for 3×4 scenario in different rooms.



(a) Minimum host throughput (b) Overall throughput

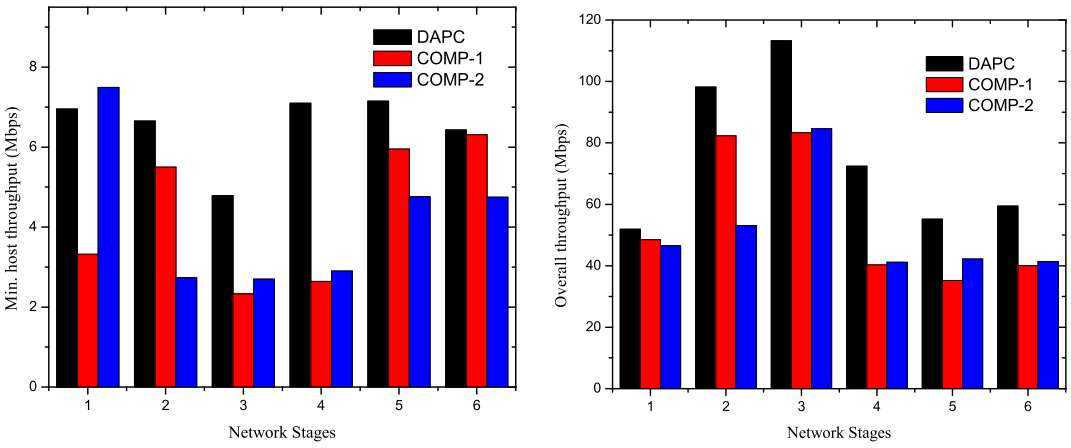
Figure 19: Throughput results for 3×6 scenario.

7.5.3 3×6 Scenario

Figure 19 (a) and (b) show the minimum host throughput results and the overall throughput results in the testbed for the 3×6 scenario, by the proposal, by COMP-1, and by COMP-2 at each stage, where the number of active hosts is changed from 2 to 6. Except for the minimum host throughput at stage 1, our proposal always provides the better performance than COMP-1 and COMP-2.

7.5.4 4×8 Scenario

As the largest topology, Figure 20 (a) and (b) show the minimum host throughput results and the overall throughput results in the testbed for the 3×6 scenario, by the proposal, by COMP-1, and by COMP-2 at each stage, where the number of active hosts is changed from 3 to 8. Except for the minimum host throughput at stage 1, our proposal always provides the better performance than COMP-1 and COMP-2.



(a) Minimum host throughput (b) Overall throughput

Figure 20: Throughput results for 4×8 scenario.

8 Conclusion

In this paper, we proposed the extension of the *active AP configuration algorithm* to deal with the dynamic nature of the *elastic WLAN system* and implemented the testbed using *Raspberry Pi*. In this extension, the number of active APs is minimized such that the minimum host throughput constraint is satisfied, while any communicating AP is not suspended and any communicating host does not change the associated AP. Through numerical experiments in simulations using the WIMNET simulator and throughput measurements in real environments using the testbed, the effectiveness of our proposal was demonstrated. Our future works include further enhancements of the dynamic features of the active AP configuration algorithm and the elastic WLAN system implementation by considering host mobility and admission control, and their evaluations in various practical scenarios.

Acknowledgments

This work is partially supported by JSPS KAKENHI (16K00127).

Appendix-A : AP Configuration of Raspberry Pi

1. Install the *hostapd* using the following command [27][28]:

   ```
   $ sudo apt-get install hostapd
   ```

2. Modify the configuration file */etc/hostapd/hostapd.conf* with the desired SSID and PASSWORD. A simple example of the configuration file is given below:

   ```
   interface=wlan0
   ssid=SSID
   channel=1
   wpa_passphrase=PASSWORD
   ```

3. Uncomment and set *DAEMON_CONF* to the absolute path of a hostapd configuration file to start hostapd during system boot:

   ```
   DAEMON_CONF="/etc/hostapd/hostapd.conf"
   ```

4. Setup the *wlan0* interface to have a static IP address in the network interface configuration file */etc/network/interfaces*. An example of the interface file is given below:

   ```
   auto wlan0
   iface wlan0 inet static
   address 192.168.1.11
   netmask 255.255.255.0
   network 192.168.1.0
   ```

5. Finally, install the DHCP server for assigning the dynamic IP addresses to the hosts.

Appendix-B : Details of Step 7 to Step 10 of Execution Flow of Elastic WLAN

Before start the execution of the Step 7 to Step 10 , shown in Figure 12, the server compiles all the C programs at once to save the CPU time and load. The following commands perform the compilation of each C program.

Linux commands for compilation of each C program

```
#/bin/bash
# to detect the network change and identify joining or leaving host
01: g++ -o chkD CheckDevicesForjoinleave.cpp
02: g++ -o Inputforleavinghost GenAlgmInputforLeaving.cpp
03: g++ -o ckcurap CheckWithCurrentAP.cpp
04. g++ -o dcs Decisioncurjoin.cpp
05. g++ -o CrIn CreateInputforAllHost.cpp
06: g++ -o ckcomap Comaphost.cpp
```

After that, the server uses the executable files and the necessary input files generated by the each step to perform the whole execution process. What follows, we describe each of the steps in details.

7. **Network Change Detection**
   In this step, the system detects whether the network state is being changed due to joining of new hosts or leaving of existing hosts in the network. To detect it, the server uses the following commands:

   Linux commands for network change detection

   ```
   #/bin/bash
   # to explore the connected devices in network
   01: sudoarp-scan --interface=eth0 192.168.11.0/24
   # to detect the network change
   02: ./chkD netoutpre.txt netoutrecent.txt
   ```

   The command in `01` explores the connected devices in the network using *arp-scan* [36]. The output consists of the IP and MAC addresses of the hosts and the APs that are available in the network. The command in `02` identifies network changes. Here, *netoutpre.txt* and *netoutrecent.txt* contain the device information of the network according to previous and current status. The C Program will check the devices and can detect the network changes. After detecting the network state change, the server takes the following steps to control the number of active APs.

8. **Leaving or Joining Host Identification**
   After detecting the network state change, the server identifies the hosts that newly joined or left the network and finds their IP addresses by running the same C program.

   Linux commands for leaving or joining host identification

   ```
   #/bin/bash
   # identify joining or leaving hosts/hosts
   01: ./chkD netoutpre.txt netoutrecent.txt
   ```

   The command in `01` takes the two files, *netoutpre.txt* and *netoutrecent.txt*, as the input. After analyzing the IP and MAC addresses of the devices, the server identifies and prepares the list of the changed hosts. If there is no change in the network, the server goes back to the previous step in Figure 12.

9. **Execution of Algorithm Extension**
   According the host identifications, the active AP configuration algorithm is applied for joining hosts or leaving hosts. The output of the algorithm contains the information of the network changes, including the associations between the hosts and APs and the list of APs for activation or deactivation.

   (a) **Steps in Leaving Operation**
   Before executing the algorithm for the leaving host, the server prepares the input by the

following commands:

Linux commands for leaving host operation

```
#/bin/bash
# generate input for the Modified APC Alg.
01: ./Inputforleavinghost leavinghosts.txt
alg_out.txt previousinput.txt group_info.txt newinput.txt
```

The command in `01` takes the input from *previousinput.txt* and prepares the new input for the leaving host extension. Before applying the algorithm, the server detects the communicating hosts and APs, as described in Section 6.3.

(b) **Steps in Joining Operation**
The server collects the signal strengths of the newly joining hosts and the active APs in the network. Then, it applies the following commands:

Linux commands for joining host operation

```
#/bin/bash
# to inspect signal strength for joining host
01: sudo nm-tool
02: sh ./nm_scriptjoinhost.sh
03: ./ckcurap JoinHosts.txt
# decision on current APs
04: ./dcs ch_association.sh ap_activation.sh
# generate input for the Modified APC Alg
# if we need to activate new AP
05: ./CrIn CurHostsall.txt CurAPall.txt
```

The commands in `01` and `02` find the receiving signal strength of each joining host from every AP using *nm-tool* [34][35]. The *ssh* [30][31] protocol is used to execute the command in `01` remotely in each host. The output consists of the currently associated APs, the list of associable APs, and the receiving signal strength of each host from all the associable APs. After this, the server converts the receiving signal strength to the estimated link speed using the sigmoid function in [38], and generates the input for the AP configuration algorithm. By using the commands in `03` and `04`, the server knows whether the currently activated APs can satisfy the minimum host throughput constraint for the hosts or not. It also detects the changes of their AP associations, if necessary. If they cannot satisfy the constraint, the server applies the algorithm to activate some deactivated APs, after preparing the algorithm input using the command in `05`. Again, the communicating hosts and APs are detected using the procedure in Section 6.3.

10. **Application of Algorithm Output**
Then, the server changes the corresponding associations and deactivates all the unused APs by following the algorithm output. For the joining hosts, some additional APs are activated and are assigned the channels, if necessary. Besides, the channels of some active APs are also changed, if necessary. The following commands are used in this step. The commands in `01` and `02` activate or deactivate the *Raspberry Pi* AP respectively. The server adjusts the number of active APs according to the algorithm output by activating or deactivating APs in the network. The command in `03` connects a host to a new AP using *nmcli* [32][33]. Here, *NewSSID* represents the new AP for the host and *PASSWORD* does the security key of the AP. The server modifies the AP-host association according to the algorithm output using this command.

Linux commands for application of algorithm output

```
#/bin/bash
# for activation of a Raspberry Pi AP
01: sudo /etc/init.d/hostapd start
# for deactivation of a Raspberry Pi AP
02: sudo /etc/init.d/hostapd stop
# to change the association of a host to a new AP
03: sudo -s nmcli dev wifi connect NewSSID password PASSWORD
# to change the channel of a raspberry Pi AP
04: sed -i -e 's/.*channel.*/channel='$NewChannel'/'/etc/hostapd/
hostapd.conf
# to restart the service of hostapd daemon
05: sudo /etc/init.d/hostapd restart
```

The command in `04` assigns the new channel to the AP using *sed* [39]. For this, the server modifies the configuration file */etc/hostapd/hostapd.conf* with the channel number. Here, *'s'* represents the substitution command and *NewChannel* does the channel to be assigned in the *hostapd.conf* file of the AP. The command in `05` restarts the *hostapd* daemon [27][28]. After the assignment of the new channel, the server restarts it to make the change take effect. It takes $20 \sim 30sec.$ to stop the *hostapd* service, and takes $40 \sim 60sec.$ to change the channel of an active AP. To restart the *hostapd* daemon, it takes $20 \sim 30sec$ on average. The server changes the channel of an active AP, only if (i) the AP is not the *communicating AP* and (ii) the algorithm changes the channel because of the joining or leaving hosts.

References

[1] B. P. Crow, I. Widjaja, J. G. Kim, and P. T. Sakai, "IEEE 802.11 wireless local area networks," IEEE Commun. Mag., vol. 35, no. 9, pp. 116–126, Sept. 1997.

[2] M. Balazinska and P. Castro, "Characterizing mobility and network usage in a corporate wireless local-area network," Proc. Int. Conf. Mobile Systems, Appl. Services, pp. 303-316, 2003.

[3] S. Lanzisera, B. Nordman, and R. E. Brown, "Data network equipment energy use and savings potential in buildings," Energy Efficiency, vol. 5, no. 2, pp. 149-162, May 2012.

[4] K. Mittal, E. M. Belding, and S. Suri, "A game-theoretic analysis of wireless access point selection by mobile users," Computer Commun., vol. 31, no. 10, pp. 2049-2062, Jan. 2008.

[5] D. Kotz and K. Essien, "Analysis of a campus-wide wireless network," Wireless Networks, vol. 11, no. 1-2, pp. 115-133, Jan. 2005.

[6] F. Nadeem, E. Leitgeb, M. S. Awan, and S. Chessa, "Comparing the life time of terrestrial wireless sensor networks by employing hybrid FSO/RF and only RF access networks," Proc. Int. Conf. Wireless. Mobile Commun., pp. 134-139, 2009.

[7] "Electricity sector in Bangladesh," Internet: `https://en.wikipedia.org/wiki/Electricity_sector_in_Bangladesh`, Access Jan. 20, 2017.

[8] "Electricity to transform rural Myanmar," Internet: `http://www.worldbank.org/en/news/feature/2015/09/16/electricity-to-transform-rural-myanmar`, Access Jan. 20, 2017.

[9] M. S. A. Mamun, M. E. Islam, and N. Funabiki, "An active access-point configuration algorithm for elastic wireless local-area network system using heterogeneous devices," Int. J. Netw. Comput., vol. 6, no. 2, pp. 395-419, July 2016.

[10] M. S. A. Mamun, N. Funabiki, K. S. Lwin, M. E. Islam, and W.-C. Kao, "A channel assignment extension of active access-point configuration algorithm for elastic WLAN system and its implementation using Raspberry Pi," Int. J. Netw. Comput., vol. 7, no. 2, pp. 248-270, July 2017.

[11] N. Funabiki ed., "Wireless mesh networks," InTech-Open Access Pub., Jan. 2011, http://www.intechopen.com/books/wireless-mesh-networks, Access Jan. 5, 2018.

[12] "Ultimate guide to Raspberry Pi," Comp. Shop., vol. 324, pp. 104-119, Feb. 2015, Internet: http://micklord.com/foru/Raspberry%20Pi%20Pages%20from%20Computer%20Shopper%202015-02.pdf, Access Jan. 5, 2018.

[13] Y. Bejerano, S.-J. Han, and L. E. Li, "Fairness and load balancing in wireless LANs using association control," IEEE/ACM Trans. Networking, vol. 15, no. 3, pp. 560-573, June 2007.

[14] Y. Bejerano, S.-J. Han, and L. E. Li, "Fairness and load balancing in wireless LANs using association control," Proc. Int. Conf. Mobile Comput. Netw., pp. 315-329, 2004.

[15] A. So and B. Liang, "An efficient algorithm for the optimal placement of wireless extension points in rectilineal wireless local area networks," Proc. Int. Conf. Quality. Service. Hetero. Wired/Wireless Netw., 2005.

[16] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. P. Mhatre, "Measurement-driven guidelines for 802.11 WLAN design," IEEE/ACM Trans. Networking, vol. 18, no. 3, pp. 722-735, June 2010.

[17] S. Tang, L. Ma, and Y. Xu, "A novel AP placement algorithm based on user distribution for indoor WLAN system," China Commun., vol. 13, no. 10, pp. 108-118, Oct. 2016.

[18] S. Miyata, T. Murase, and K. Yamaoka, "Novel access-point selection for user QoS and system optimization based on user cooperative moving," IEICE Trans. Commun., vol. 95, no. 6, pp. 1953-1964, 2012.

[19] S. K. Lundsgaard, M. S. Bhally, G. M. Di Prizio, and T. Lee, "Selection of a prepared access point from among a plurality of access points," US Patent 8,654,741, Feb. 18, 2014.

[20] Y. H. Seok, "Method for providing information of access point selection," US Patent 9,008,210, Apr. 14, 2015.

[21] H. Gong and J. Kim, "Dynamic load balancing through association control of mobile users in WiFi networks," IEEE Trans. Consumer Electronics, vol. 54, no. 2, pp. 342-348, May 2008.

[22] X. Chen, Y. Zhao, B. Peck, and D. Qiao, "SAP: Smart access point with seamless load balancing multiple interfaces," Proc. INFOCOM, pp. 1458-1466, 2012.

[23] M. E. Islam, N. Funabiki, and T. Nakanishi, "An access-point aggregation approach for energy-saving wireless local area networks," IEICE Trans. Commun., vol.E96-B, no.12, pp.2986-2997, Dec. 2013.

[24] M. E. Islam, N. Funabiki, and T. Nakanishi, "Extensions of access-point aggregation algorithm for large-scale wireless local area networks," Int. J. Netw. Comput., vol.5, no.1, pp.200-222, Jan. 2015.

[25] L. A. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," Combinatorica, vol.2, no.4, pp. 385-393, Dec. 1982.

[26] D. P. Williamson and D. B. Shmoys, "The design of approximation algorithms," Cambridge Univ. Press, Apr. 2011.

[27] "Hostapd: the linux way to create virtual Wifi access point," Internet: nims11.wordpress.com/2012/04/27/hostapd-the-linux-way-to-create-virtual-wifi-access-point/, Access Jan. 5, 2018.

[28] "Setting up a Raspberry Pi as a WiFi access point," Internet: https://cdn-learn.adafruit.com/downloads/pdf/setting-up-a-raspberry-pi-as-a-wifi-access-point.pdf, Access Jan. 5, 2018.

[29] "Tcpdump and libpcap," Internet: http://www.tcpdump.org, Access Jan. 5, 2018.

[30] T. Ylonen and C. Lonvick, "The secure shell (SSH) protocol architecture," 2006.

[31] D. J. Barrett, R. E. Silverman, and R. G. Byrnes, "SSH, the secure shell: the definitive guide," O'Reilly Media Inc., 2005.

[32] "nmcli - command-line tool for controlling NetworkManager," Internet: http://manpages.ubuntu.com/manpages/precise/man1/nmcli.1.html, Access Jan. 5, 2018.

[33] "Using the network manager command line tool: nmcli," Internet: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Networking_Guide/sec-Using_the_NetworkManager_Command_Line_Tool_nmcli.html, Access Jan. 5, 2018.

[34] "nm-tool," Internet: http://linux.die.net/man/1/nm-tool, Access Jan. 5, 2018.

[35] "View your current network settings with nm-tool," Internet: https://nfolamp.wordpress.com/2010/05/21/view-your-current-network-settings-with-nm-tool/, Access Jan. 5, 2018.

[36] "Arp-scan user guide," Internet: http://www.nta-monitor.com/wiki/index.php/Arp-scan_User_Guide, Access Jan. 5, 2018.

[37] "iPerf - the TCP, UDP and SCTP network bandwidth measurement tool," Internet: https://iperf.fr/, Access Jan. 5, 2018.

[38] M. E. Islam, K. S. Lwin, M. S. A. Mamun, N. Funabiki, and I. Lai, "Measurement results of three indices for IEEE802.11n wireless networks in outdoor environments," The 17th IEEE Hiroshima Section Student Symposium, pp. 410-414, Nov. 2015.

[39] "Learning Linux Commands: sed," Internet: https://linuxconfig.org/learning-linux-commands-sed, Access Jan. 5, 2018.