International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 7, Number 2, pages 248–270, July 2017

A Channel Assignment Extension of Active Access-Point Configuration Algorithm for Elastic WLAN System and Its Implementation Using Raspberry Pi

Md. Selim Al Mamun, Nobuo Funabiki, Kyaw Soe Lwin
Department of Electrical and Communication Engineering, Okayama University
3-1-1 Tsushimanaka, Okayama 700-8530, Japan

Md. Ezharul Islam
Department of Computer Science and Engineering, Jahangirnagar University
Bangladesh

Wen-Chung Kao
Department of Electrical Engineering, National Taiwan Normal University
162, Section 1, Heping E. Rd., Taipei, 106, Taiwan

Received: February 13, 2017
Revised: May 5, 2017
Accepted: June 7, 2017
Communicated by Yoshiaki Kakuda

Abstract

Recently, *Wireless Local-Area Network (WLAN)* has become prevailing as it provides flexible Internet access to users with low cost through installation of several types of *access points (APs)* in the network. Previously, we proposed the *active AP configuration algorithm* for the *elastic WLAN system* using heterogeneous APs, which dynamically optimizes the configuration by activating or deactivating APs based on traffic demands. However, this algorithm assumes that any active AP may use a different channel from the other ones to avoid interferences, although the number of non-interfered channels in *IEEE 802.11* protocols is limited. In this paper, we propose the extension of the AP configuration algorithm to consider the channel assignment to the active APs under this limitation. Besides, AP associations of the hosts are modified to improve the network performance by averaging loads among channels. The effectiveness of our proposal is evaluated using the WIMNET simulator in two topologies. Finally, the elastic WLAN system including this proposal is implemented using *Raspberry Pi* for the AP. The feasibility and performance of the implementation are verified through experiments using the testbed.

Keywords: Elastic WLAN system, interference minimization, channel assignment, active access-point configuration algorithm, testbed, Raspberry Pi

1 Introduction

Recently, *Wireless Local Area Networks (WLANs)* have been widely applied for the Internet access due to the characteristics of uncomplicated installations, flexible coverages, and low costs [1].

Wireless connections between *access points (APs)* and hosts make WLANs inexpensive, flexible, and scalable. In WLANs, APs are often installed and used in a service field randomly, which can cause poor network performances due to overlapping of the same frequency signals [2]. Therefore, the configuration of these APs should be properly arranged according to the communication demands in the field, while redundant APs should be turned off for energy saving and interference prevention.

Practically, the number of users or traffics in a network fluctuates depending on time and day of the week. For example, in a university, the number of students increases in the afternoon during weekdays, while it decreases in the morning or evening, and in whole days on weekends. Besides, conditions of network devices or communication links may be changed for the factors of power shortages, device failures, bandwidth controls by the authorities or even with the weather changes [3].

That is, developing countries such as Bangladesh or Myanmar suffer from the unreliable and slow Internet access due to discontinuities of electricity supplies for the time being [4][5]. Generally, the irregular flow of electricity results in damage to network devices. Also, such countries often meet fluctuations of the allocated bandwidths to individual organizations by authorities. In such uncertain cases, only the critical APs in the network should be activated by supplying electricity using generators or batteries as backup power sources.

To solve the abovementioned problems, we have studied the *elastic WLAN system* using heterogeneous AP devices [6][7]. Figure 1 shows a simple topology of the elastic WLAN system. This system dynamically controls the number of active APs according to communication demands and network device conditions. For this purpose, we proposed the *active AP configuration algorithm* to activate the minimum number of APs in the network while satisfying the constraints of the throughputs for the hosts.

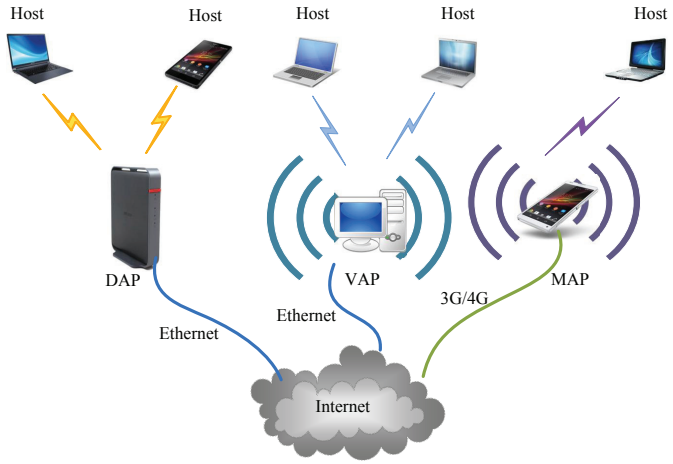
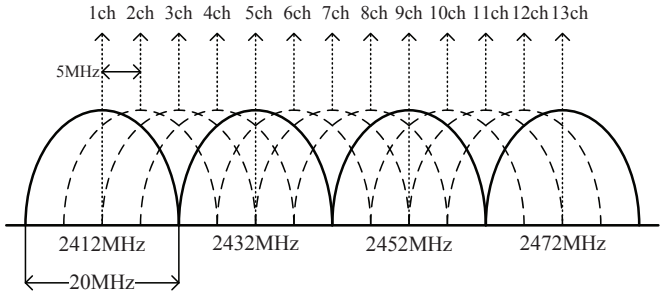


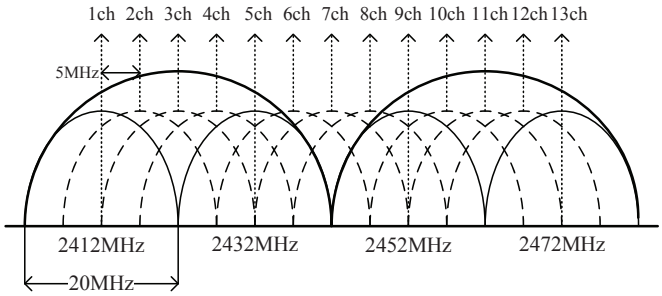
Figure 1: Illustration of elastic WLAN system with heterogeneous devices.

In the proposed algorithm, three types of APs, namely *dedicated APs (DAPs)*, *virtual APs (VAPs)*, and *mobile APs (MAPs)* are considered. Virtual APs use personal computers (PCs) of users installing software for AP functions. Mobile APs are often called *mobile routers*. To connect to the backbone network in the Internet, DAPs and VAPs use wired connections, whereas MAPs use cellular networks. For the positions of MAPs, the same locations as the hosts are considered, because the host owners may use MAPs for the Internet access.

However, the previous algorithm assumes that each AP can use a different non-interfered channel from the other APs to avoid interferences among them. In reality, the number of non-interfered channels available in *IEEE 802.11* protocols is limited [8]. Therefore, some APs have to share the same channel if the number of APs is larger than the number of available non-interfered channels. Therefore, the proper channels should be assigned to the APs so that it can avoid the interference as much as possible to improve the network performance.



(a) 20MHz Channels



(b) 40MHz Channels

Figure 2: Available channels in IEEE802.11n for 2.4GHz band.

WLANs using IEEE 802.11n [9] operate in two unlicensed frequency spectrum bands: the 2.4 GHz *Industrial, Scientific, and Medical (ISM)* band and the 5 GHz *Unlicensed National Information Infrastructure (U-NII)* band. The IEEE 802.11n standard defines the limited number of channels for use in APs and clients. One channel has 22 MHz width and is separated by 5 MHz from the adjacent channel, which is conventionally referred as the 20 MHz channel. Figure 2 illustrates the available channels in IEEE 802.11n for the 2.4 GHz spectrum.

In this 2.4 GHz band, 13 channels exist where only four channels can be non-interfered. If the channel bonding technique [10] is adopted, which is common in IEEE 802.11n, the 40 MHz channel is derived by bonding two adjacent 20 MHz channels. It achieves more than twice data throughputs, however it reduces the number of non-interfered channels to only two. Since the 2.4 GHz band becomes crowded, clients start using the 5 GHz band. This can carry up to nine non-interfered 40 MHz channels, but it has a shorter range than the 2.4 GHz.

In this paper, we propose the *channel assignment extension* of the active AP configuration algorithm for the elastic WLAN system to solve the above mentioned problem. One channel is assigned to every active AP from the limited number of the non-interfered channels in a way to minimize the overall interference in the network, after the active APs and the AP-host associations are found in the previous algorithm.

Then, the *channel load averaging* is applied to balance the loads among the channels by changing some AP-host associations. After the channel assignment, the communication loads among the channels may become imbalanced, because the previous algorithm finds the AP-host associations assuming the infinite number of the non-interfered channels.

The proposed extension of the algorithm is implemented on the elastic WLAN system testbed to verify the practicality in real systems. In this testbed implementation, *Raspberry Pi* [11] 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 important for its dissemination in developing countries.

The proposed algorithm extension is evaluated by simulations in two network topologies using the WIMNET simulator. The comparisons with the random channel assignment and the existing

algorithm confirm the effectiveness of the proposal. Then, the elastic WLAN system testbed is evaluated through experiments, where the measured throughputs are higher for the algorithm channel assignment than those for the random one. In short, the contributions of this research work are summarized as below:

1. The channel assignment extension to the active APs from the limited number of non-interfered channels in a way to minimize the overall interference in the network,
2. The load averaging extension among the channels for the throughput improvement after the channel assignment, and
3. The elastic WLAN system testbed extension for the channel assignment with the load averaging and the Raspberry Pi implementation for the AP.

A variety of studies have been addressed to the channel assignment problem in WLAN. In [12], the *Least Congested Channel Search (LCCS)* approach was proposed. The network administrator conducts the on-site survey of radio frequencies to determine the number of APs and their locations that are required for the coverage. Then, these APs are configured manually with non-interfered channels to avoid interferences as much as possible. This approach is manual, whereas our proposal is automatic.

In [13], a centralized traffic and interference-aware channel assignment approach was proposed for a mesh network. Channels are assigned to APs based on the ranking that is generated from important parameters such as the aggregate traffic, the distance from the gateway node, and the number of interfaces at each node. This approach focuses on finding the optimal route in a mesh network allocating non-interfered channels, whereas our proposal assigns channels to APs in WLAN.

In [14], an interference system model using the interference graph was proposed. The degree of the interference between two channels is defined using the interference table. Then, the interference minimization problem is formulated, and the channel assignment algorithm is developed to minimize the total interference in the network. This algorithm is a simple greedy one that cannot find optimal solutions, whereas our proposal combines a greedy algorithm and a simulated annealing to seek optimal solutions.

In [15], another interference system model was proposed to consider the channel bonding and frame aggregation mechanisms in IEEE 802.11n. First, the throughput of each host in the system is estimated. Then, a channel assignment problem is formulated and translated into a throughput optimization problem. Finally, a distributed channel assignment algorithm is developed for multi-rate 802.11n WLANs. The approach pays the higher cost to exchange throughput information among hosts, whereas our proposal is the centralized one to avoid it.

The rest of the paper is organized as follows: Section 2 presents brief reviews of our previous works. Section 3 proposes the extension of the active AP configuration algorithm. Section 4 describes the implementation of the elastic WLAN system. Sections 5 and 6 evaluate the algorithm extension and the testbed implementation. Finally, Section 7 concludes this paper with future works.

2 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.

2.1 Elastic WLAN System

The elastic WLAN system dynamically controls the number of active APs on the network by activating or deactivating APs according to the network condition. Figure 3 offers an example topology of the elastic WLAN system.

Our implementation of the elastic WLAN system in [16] adopts a server to manage and control the APs and the hosts. This server has the administrative access to all the devices in the network. The server controls the system by the following three steps:

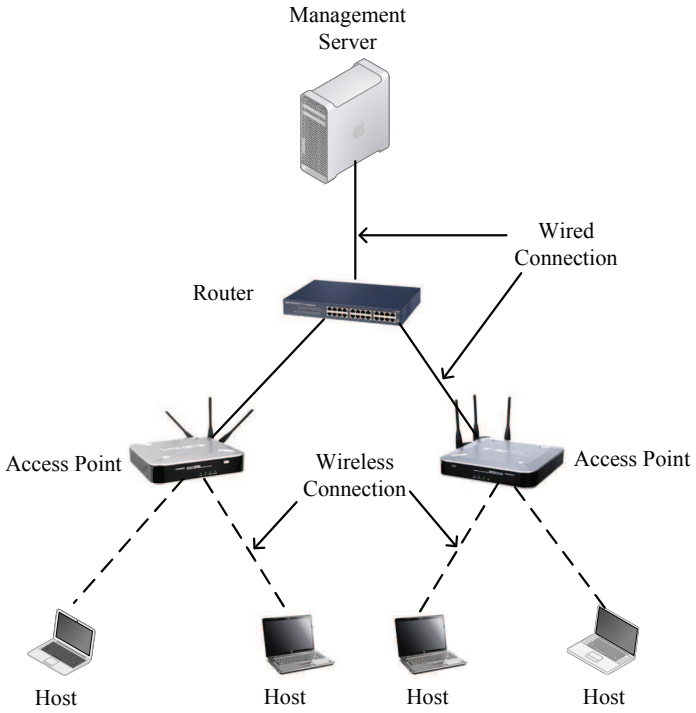


Figure 3: Example elastic WLAN system topology.

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

2.2 Formulation of Active AP Configuration Algorithm

The AP configuration problem in the previous work 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) : The link speed could be estimated by measuring the signal strength of the host from the AP and our derived formula in [6].
2. **Outputs :**
   - The set of active APs (DAPs, VAPs, and MAPs)
   - The set of the hosts associated with 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:

$$E_1 = E_1^D + E_1^V + E_1^M \tag{1}$$

   where E_1^D represents the number of active DAPs, E_1^V does the number of active VAPs, and E_1^M does the number of active MAPs respectively. The cost function E_2 is defined as follows:

$$E_2 = \min_j \left[TH_{min,j} \right] \tag{2}$$

   where $TH_{min,j}$ represents the minimum host throughput for the jth AP and is defined by:

$$TH_{min,j} = \frac{1}{\sum_k \frac{1}{s_{jk}}} \tag{3}$$

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

4. **Constraints :**

   - Minimum host throughput constraint: every host in the network 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.

2.3 Procedure of Active AP Configuration Algorithm

Figure 4 shows the flow for the active AP configuration algorithm. The algorithm consists of eight phases to solve the complex AP configuration problem.

2.3.1 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 as:

1. For each AP, make a list of hosts that can be associated to this AP. It is called the *associable host list for APs*.
2. Initialize each AP as the non-active AP. Initially, only the DAPs are selected as *candidate APs*.

2.3.2 Initial Solution Generation

An initial solution E_1 is derived using a greedy algorithm [17]. It repeats the following procedures:

1. Select the AP which covers the maximum number of uncovered hosts.
2. Activate this AP and increment E_1 by one.
3. Update the number of uncovered hosts in the associable host list for APs.

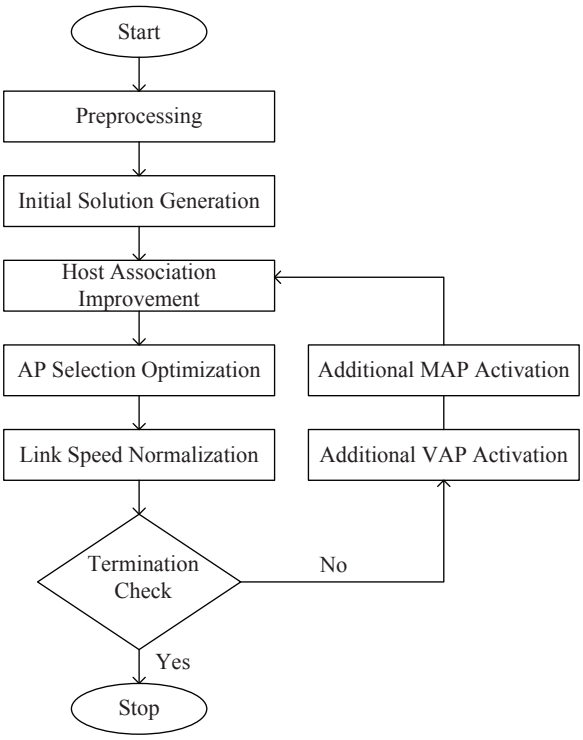


Figure 4: AP configuration algorithm flow.

2.3.3 Host Association Improvement

The cost function E_2 is calculated for the greedy solution using Eq. (2). Then, this solution is improved by repeating the following procedure:

1. Find the AP that gives the lowest host throughput in Eq. (2).
2. Select one host randomly from the associated hosts with this AP, and connect it to another associable active AP that is selected randomly. Then, calculate E_2.
3. Keep the new association only if the new E_2 is higher than the previous E_2. Otherwise, roll back to the previous association.

2.3.4 AP Selection Optimization

The cost functions E_1 and E_2 are further jointly optimized in this phase by the *local search* [6][18] under the constraints mentioned before.

2.3.5 Link Speed Normalization

The fairness criterion is applied here when the total expected bandwidth exceeds B^a. Then, the link speed is normalized as:

1. Calculate the expected total bandwidth B^e by the summation of the throughputs of all the APs.
2. If $B^e > B^a$, adjust each AP-host link speed as:

$$\hat{s}_{ij} = s_{ij} \times \frac{B^a}{B^e} \tag{4}$$

where $\hat{s}_{ij}$ is the normalized link speed.

2.3.6 Termination Check

The algorithm is terminated when either of the following conditions is satisfied:

1. The *minimum host throughput constraint* is satisfied.
2. All the APs in the network have been activated.

2.3.7 Additional VAP Activation

If VAPs are not selected for candidate APs, they are selected as candidate APs. Go back to Phase 3. Otherwise, go to Phase 8.

2.3.8 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. Go back to Phase 3.

3 Channel Assignment Extensions of Active AP Configuration Algorithm

In this section, we present the channel assignment extension of the active AP configuration algorithm to consider the limited number of non-interfered channels in IEEE 802.11n. Specifically, we show the formulation of the channel assignment problem, and present the procedures for the channel assignment algorithm and the channel load averaging.

3.1 Formulation of Channel Assignment Problem

First, the channel assignment problem is formulated as a combinatorial optimization problem to consider the corresponding algorithm, as follows:

1. **Inputs:**
   - Output of the AP configuration algorithm: the set of active APs and their associated hosts
   - Number of non-interfered channels
2. **Outputs:**
   - Channel assignment to the active APs
3. **Objectives:**
   - The total interfered communication time E should be minimized.

$$E = \sum_{i=1}^{N} [IT_i] = \sum_{i=1}^{N} \left[\sum_{\substack{k \in I_i \\ c_k = c_i}} T_k \right] \tag{5}$$

   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. **Constraint:**
   - Every AP must be assigned one channel.

3.2 Procedure of Channel Assignment Algorithm

Then, the proposed channel assignment extension is presented here. Figure 5 shows the flow chart for the proposed channel assignment extension. The outputs of the AP configuration algorithm, namely the list of the active APs, the list of the hosts and the information of AP-host associations, become the inputs to the channel assignment extension. It is composed of the following four phases.

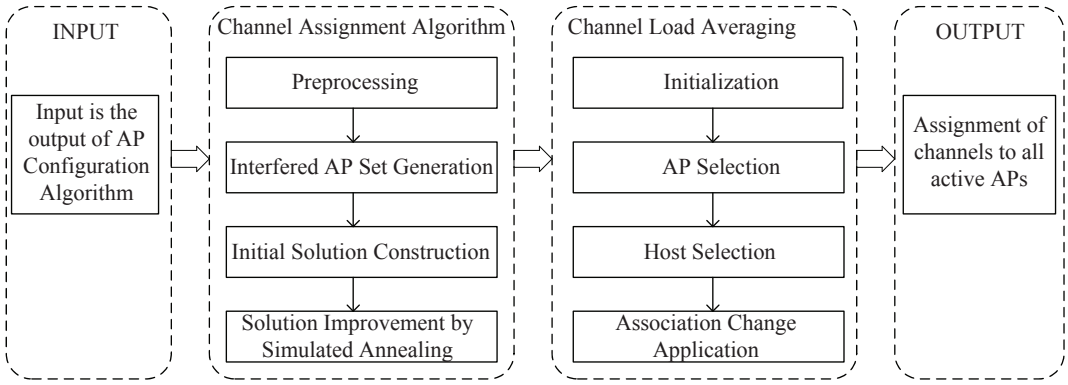


Figure 5: Channel assignment extension flow.

3.2.1 Preprocessing

The interference and delay conditions of the network are represented by a graph.

1. Construct the *interference graph*, $G = (V, E)$, from the APs and the hosts, where the vertex V represents the set of APs and the edge E presents the existence of the interference between two APs. $e(i, j) \in E$ if the ith AP is interfered with the jth AP in the network.

2. Calculate the *communication time* for each AP. The communication time T_i for the ith AP is defined as the total time when the AP transmits 1-bit to all the associated hosts. It is given by:

$$T_i = \sum_j \frac{1}{sp_{ij}}. \tag{6}$$

   where sp_{ij} represents the link speed between the ith AP to the jth host.

3. Calculate the *neighbor interfered communication time* for each AP. The neighbor interfered communication time NT_i for the ith AP is given by:

$$NT_i = \sum_{e(i,k)=1} T_k. \tag{7}$$

3.2.2 Interfered AP Set Generation

The set of APs that are interfering with each other is found for each AP.

1. Sort the APs in descending order of NT_i, where the tie-break is resolved by T_i.

2. Find the interfered AP set for each AP by repeating the following steps:

   (a) Initialize the interfered AP set by $I_i = \{i\}$ for AP_i.

   (b) Expand I_i by checking the APs in the sorted order in (a) whether the AP is interfered with every AP in I_i. If so, include this AP, AP_j into I_i, i.e. $I_i = I_i \cup \{j\}$.

3. Calculate the total interfered communication time AT_i for the ith AP that is given by:

$$AT_i = \sum_{k \in I_i} T_k. \tag{8}$$

3.2.3 Initial Solution Construction

Then, an initial solution is derived using a greedy algorithm.

1. Sort the APs in descending order of the total interfered communication time AT_i, where the tie-break is resolved by NT_i.
2. Assign a channel c to AP_i such that the interfered communication time IT_i is minimized if assigned. IT_i is given by:

$$IT_i = \sum_{\substack{k \in I_i \\ c_k = c}} T_k \tag{9}$$

   where c_k is the assigned channel to AP_k.
3. Repeat 2. until each AP is assigned one channel.
4. Calculate the cost function E using Eq. (5) and save this initial solution as the best solution E^{best}.

3.2.4 Solution Improvement by Simulated Annealing

Finally, the initial solution is improved by repeating the following 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:

1. Randomly select one AP and one new channel for the channel change trial.
2. Calculate the interfered communication time, IT_i after assigning this new channel by:

$$IT_i = \sum_{\substack{k \in I_i \\ c_k = c_i}} T_k. \tag{10}$$

3. Calculate E^{new} using Eq. (5) for the new channel assignment, and $\Delta E = E^{new} - E$.
4. If $\Delta E \leq 0$, accept the new channel assignment, and keep this new solution as the best one.
5. Otherwise, generate a 0-1 random number, *rand*, and if $rand \leq \frac{-\Delta E}{T^{SA}}$, then accept the new channel assignment.

3.3 Channel Load Averaging

After the channel assignment to the APs, the total load can be imbalanced between different channels. That is, some channels may be crowded, and some channels may not. Because the associated hosts with each AP are fixed during the channel assignment, it is impossible to average the load by changing host associations. Thus, to further improve the performance of the network, the load averaging procedure among channels is applied to the solution from the channel assignment.

3.3.1 Initialization

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

3.3.2 AP Selection

One AP is selected to move its associated host to a different AP that is assigned a different channel.

1. Terminate the procedure if every AP has $1(= ON)$ AP flag.
2. Initialize the host flag by $0(= OFF)$ for every host.
3. Select one AP, say AP_i, that satisfies the two conditions:
   (a) The AP flag is OFF.
   (b) The interfered communication time IT_i is largest among the OFF APs.
4. If one AP is selected, set the corresponding AP flag ON.

3.3.3 Host Selection

Then, one host associated with AP_i is selected for the AP movement.

1. Select one host, say H_j, that satisfies the four conditions:
   (a) The host flag is OFF.
   (b) The host is associated with AP_i.
   (c) The host can be associated with another AP that is assigned a different channel from AP_i, or is located out of the interference range of AP_i.
   (d) The link speed of AP_i and the host is the smallest among the hosts satisfying (a)–(c).
2. If one host is selected, set the corresponding host flag ON.
3. Otherwise, go back to 3.3.2 for the new AP selection.

3.3.4 Association Change Application

Finally, the new associated AP is selected for H_j.

1. Select the AP that has the largest link speed among the APs found in $3.3.3 - 1. - (c)$.
2. Calculate the cost function E using Eq. (5) if H_j is associated with this AP.
3. If the new cost function E is equal to or smaller than the previous E, accept the new association, and go back to 3.3.3.
4. Otherwise, select another AP that has the next largest link speed, and go back to 2.
5. If no such AP exists, go back to 3.3.3 for the new host selection.

4 Elastic WLAN System Implementation Using Raspberry Pi

In this section, we describe the implementation of the elastic WLAN system testbed extension for the channel assignment with the load averaging, and present the Raspberry Pi implementation for the AP. Figure 6 shows the execution procedure for the elastic WLAN system.

4.1 System Topology

Figure 7 shows a simple network topology of the elastic WLAN system. Raspberry Pi is used for the AP and a Linux laptop PC is for the host.

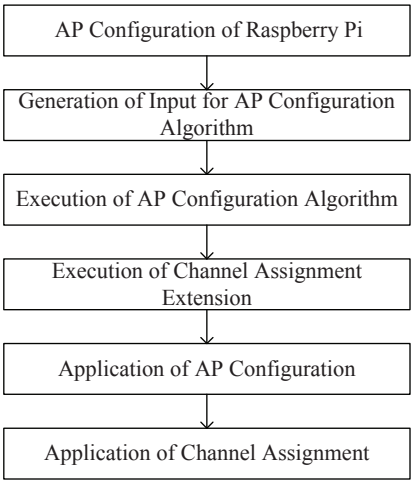


Figure 6: Elastic WLAN system execution flow.

4.2 Linux Commands

The adopted commands in the elastic WLAN system are shown as follows:

Linux commands for system.

```
#/bin/bash
# installation command for hostapd daemon
01: sudo apt-get install hostapd
# to explore the connected devices in network
02: sudoarp-scan --interface=eth0 192.168.11.0/24
# to find the signal strength of a PC
03: sudo nm-tool
# to run AP configuration algorithm
04: g++ -o apc APConfigurationAlgorithm.cpp
05: ./apc input.txt min_host_throughput bw_limit
# to run channel assignment algorithm
06: g++ -o ca ChannelAssignment.cpp
07: ./ca HostAPassociation.txt num_of_channels
# for activation of a Raspberry Pi AP
08: sudo /etc/init.d/hostapd start
# for deactivation of a Raspberry Pi AP
09: sudo /etc/init.d/hostapd stop
# to change the association of a host to a new AP
10: sudo -s nmcli dev wifi connect NewSSID password PASSWORD
# to change the channel of a raspberry Pi AP
11: sed -i -e ‘s/.*channel.*/channel=’$NewChannel‘/’/etc/hostapd/
hostapd.conf
# to restart the service of hostapd daemon
12: sudo /etc/init.d/hostapd restart
```

The command in `01` configures Raspberry Pi as the AP using *Host access point daemon (hostapd)* [19][20]. It is necessary to modify the configuration file of *hostapd* with desired SSID and PASSWORD, and to setup the WLAN interface with a static IP address in the network interface configuration file as shown in Appendix-A.

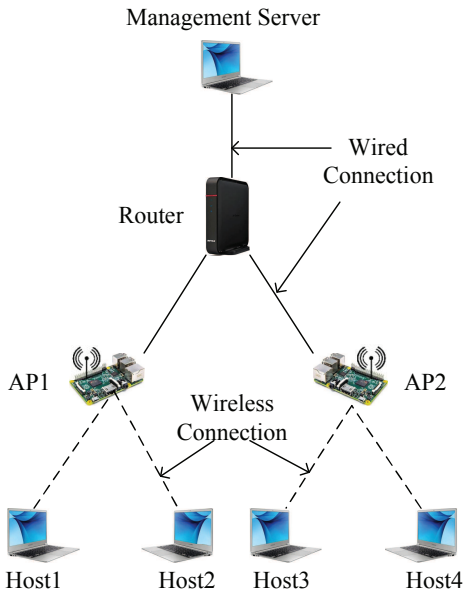


Figure 7: Elastic WLAN system topology.

The command in `02` explores the connected devices in the network using *arp-scan* [21]. The output consists of the IP and MAC addresses of the hosts and the APs that are available in the network. A simple C program is developed to identify the hosts and APs in this system using the MAC addresses of the devices. After this, the server generates the list of permitted APs and the list of permitted hosts.

The command in `03` finds the receiving signal strength of each host from each AP using *nm-tool* [22][23]. *ssh* [24][25] protocol is used to execute the command in `03` 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 associable APs. After this, the server converts the receiving signal strength to the estimated link speed using the sigmoid function in [26], and generates the input for the AP configuration algorithm

The commands in `04` and `05` compile the program for the AP configuration algorithm and execute it respectively. The *minimum host throughput constraint* and the *bandwidth limitation constraint* are specified by the user. *input.txt* presents the input file generated in the previous step, *min_host_throughput* does the minimum host throughput constraint, and *bw_limit* does the bandwidth limitation constraint. After this, the list of active APs and their associations with the hosts are obtained.

The commands in `06` and `07` compile the program for the channel assignment extension and execute it respectively. *HostAPassociation.txt* presents the input file to the channel assignment extension that contains the list of active APs and their associations with the hosts, and *num_of_channels* does the number of available channels.

The commands in `08` and `09` activate and 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 `10` connects a host to a new AP using *nmcli* [27][28]. *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.

The command in `11` assigns the new channel to the Raspberry Pi AP using *sed* [29]. 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 `12` restarts *hostapd* daemon. After the assignment of

the new channel, the server restarts it to make the change take effect.

5 Evaluations by Simulations

In this section, we evaluate the extended active AP configuration algorithm with the channel assignment and the load averaging through simulations in two network topologies using the WIMNET simulator [30]. Here, we describe the simulation platform with necessary parameters for the WIMNET simulator, the simulated network topologies, and the simulation results.

5.1 Simulation Platform

Table 1 and Table 2 summarize the hardware and software platforms, and the simulation parameters that are used for the simulation using WIMNET simulator.

Table 1: Simulation Environment.

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

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 [6]
carrier sense threshold	−85 dBm
transmission power	19 dBm
collision threshold	10
RTS/CTS	yes

5.2 Network Topologies

As the first topology, the *random topology* in Figure 8 is considered, where 30 hosts and 8 DAPs are distributed in a 400m × 200m rectangular area. The circles and squares represent the APs and hosts respectively. The minimum host throughput constraint $G = 5$ and the bandwidth limit constraint $B^a = \infty$ are examined. For simplicity, VAPs and MAPs are not used in this topology.

Then, as the second topology, the *regular topology* in Figure 9 is considered, which basically models the third floor of Engineering Building-2 in Okayama University, Japan. There are six rooms with two different sizes, $7m \times 6m$ and $3.5m \times 6m$. 6 DAPs and 60 hosts are regularly distributed in the field. The minimum host throughput constraint $G = 5$ and the bandwidth limit constraint $B^a = \infty$ are examined.

5.3 Evaluation of Channel Assignment Algorithm

Firstly, the channel assignment algorithm in Section 3.2 is evaluated in the two topologies with two, three, and four channels. Here, the throughput results by the random channel assignment and the algorithm assignment without using SA are also obtained through simulations, in order to compare them with the proposed channel assignment algorithm. To avoid the bias in the random channel

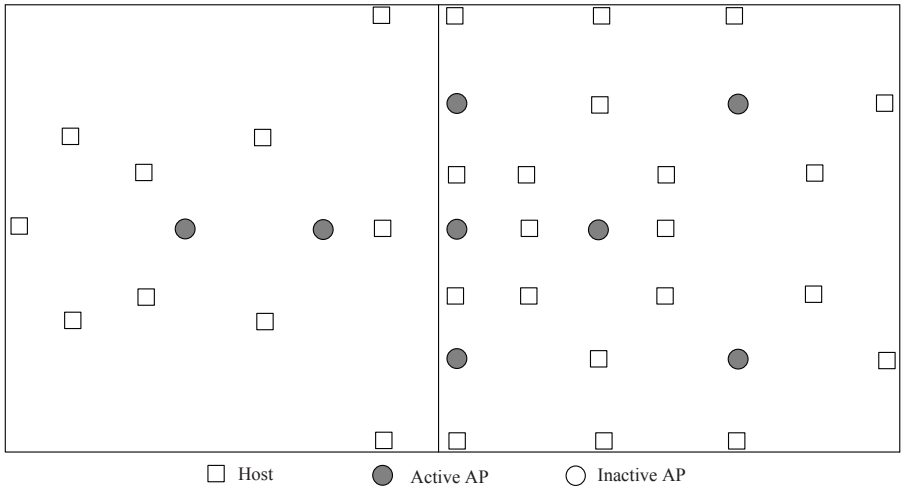


Figure 8: Random topolgy with 30 hosts and 8 DAPs.

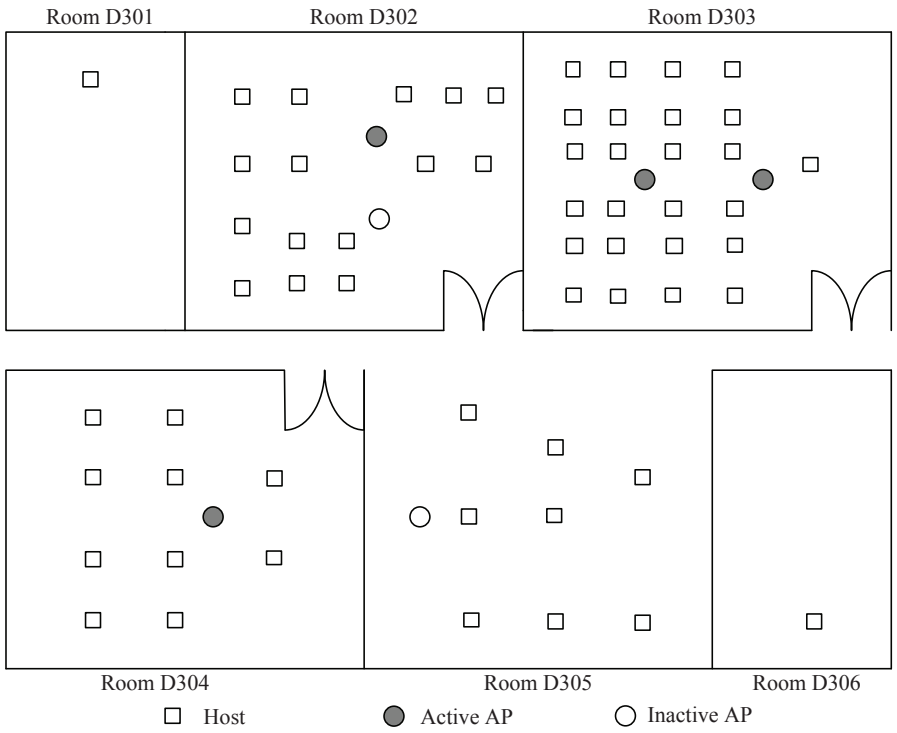


Figure 9: Regular topolgy with 60 hosts and 6 DAPs.

assignment, 20 different channel assignments are generated by using different random numbers, and their average results are used in evaluations. Tables 3 and 4 show the minimum host and overall throughput results in the three cases for the random topology and the regular topology respectively. They indicate that the greedy initial solution in our algorithm provides better results than the random assignment, and SA can further improve them by reducing interferences. With four channels, the greedy initial solution has no interference, which cannot be improved by SA.

Figure 10 compares the channel assignment results before and after applying SA, where different channels are represented by different shapes for APs. They indicate that in the initial solution of the

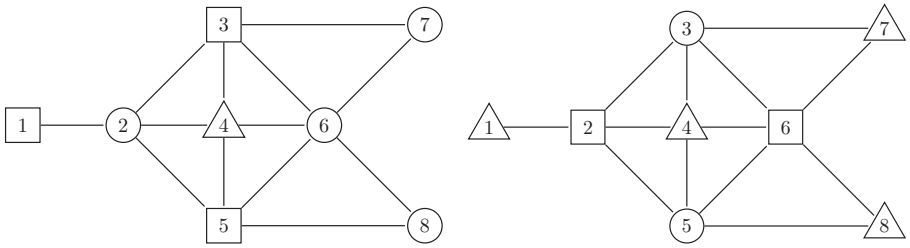
Table 3: Throughput comparisons in three cases of channel assignment for random topology.

# of channels	2			3			4		
case	random	w/o SA	with SA	random	w/o SA	with SA	random	w/o SA	with SA
# of active APs	8	8	8	8	8	8	8	8	8
min. host through.	3.63	3.87	4.73	4.68	5.10	6.61	5.04	6.61	6.61
overall through.	113.76	120.53	147.96	145.06	154.91	201.46	151.74	201.46	201.46

Table 4: Throughput comparisons in three cases of channel assignment for regular topology.

# of channels	2			3			4		
case	random	w/o SA	with SA	random	w/o SA	with SA	random	w/o SA	with SA
# of active APs	4	4	4	4	4	4	4	4	4
min. host through.	3.05	3.17	3.27	3.13	3.24	3.4	3.51	6.17	6.17
overall through.	188.27	200.06	201.67	191.88	201.07	205.07	222.61	402.12	402.12

algorithm, the same channel is assigned to AP#6, AP#7, and AP#8, which causes interferences, and it is resolved by applying SA.



(a) Before simulated annealing (b) After simulated annealing

Figure 10: Channel assignment results by algorithm for random topology with three channels.

5.4 Evaluation of Channel Load Averaging

Secondly, the channel load averaging (CLA) in Section 3.3 is evaluated in the two topologies with two, three, and four channels. Tables 5 and 6 compare the minimum host and overall throughput results between the cases with and without applying CLA for the random topology and the regular topology respectively. They indicate that CLA can further improve the throughput by averaging the loads among the channels, except for two channels case where the loads between two channels are already balanced after SA.

Table 5: Throughput comparisons between two cases of channel load averaging for random topology.

# of channels	2		3		4	
case	w/o CLA	with CLA	w/o CLA	with CLA	w/o CLA	with CLA
# of active APs	8	8	8	8	8	8
min. host through.	4.39	4.39	6.61	6.80	6.61	6.80
overall through.	147.96	147.96	201.46	219.23	201.46	219.23

5.5 Evaluation of Dynamic Network Load Changes

Thirdly, we evaluate the performance of the proposed algorithm when the network load changes dynamically. In simulations, the number of hosts is changed from 20 to 30 for the random topology

Table 6: Throughput comparisons between two cases of channel load averaging for regular topology.

# of channels	2		3		4	
case	w/o CLA	with CLA	w/o CLA	with CLA	w/o CLA	with CLA
# of active APs	4	4	4	4	4	4
min. host through.	3.27	3.27	3.4	4.5	6.17	6.19
overall through.	201.67	201.67	205.07	283.27	402.12	404.34

and from 40 to 60 for the regular topology respectively. Tables 7 and 8 summarize the simulation results. They indicate that depending on the number of hosts, the algorithm dynamically optimizes the number of active APs in the network while satisfying the minimum host throughput constraint.

Table 7: Simulation results under network load changes for random topology.

# of hosts	10		20		30	
algorithm	proposed	random	proposed	random	proposed	random
# of active APs	3	3	5	5	8	8
min. host through.	5.0	4.01	5.78	4.07	6.80	5.04
overall through.	54.09	42.43	124.78	93.17	219.23.47	151.74

Table 8: Simulation results under network load changes for regular topology.

# of hosts	40		50		60	
algorithm	proposed	random	proposed	random	proposed	random
# of active APs	2	2	3	3	4	4
min. host through.	5.09	3.87	5.85	3.84	6.19	3.51
overall through.	209.13	163.73	303.12	209.52	404.34	222.61

5.6 Evaluation of Network with Uncontrolled APs

Fourthly, we evaluate the algorithm for the two network topologies when the number of APs that cannot be activated/deactivated by the algorithm/system is increased from 1 to 4. The uncontrolled APs are randomly selected, and the average results are examined after 10 trials are repeated for each number of uncontrolled APs to avoid the bias in random selections. Here, the APs that cannot be deactivated by the system are categorized as uncontrolled APs in this paper. Thus, the uncontrolled APs are always active where their channels are assigned randomly.

Tables 9 and 10 summarize the simulation results for the random topology and regular topology respectively. They indicate that the network throughput drops when the number of uncontrolled APs increased in the network.

Table 9: Simulation results with uncontrolled APs for random topology.

# of uncontrolled APs	0	1	2	3	4
# of active APs	8	8	8	8	8
min. host through.	6.80	6.43	6.03	5.26	5.14
overall through.	219.23	197.56	189.11	161.79	156.21

5.7 Comparison with Existing Algorithm

Finally, the performance of the proposed channel assignment algorithm is evaluated through comparisons with the representative existing algorithm called the *ADJ-minmax approach* [14]. In the

Table 10: Simulation results with uncontrolled APs for regular topology.

# of uncontrolled APs	0	1	2	3	4
# of active APs	4	4	4	4	4
min. host through.	6.19	6.16	4.94	4.01	3.47
overall through.	404.34	402.12	307.42	282.56	220.51

simulations of both algorithms, the same interference model is adopted for fair comparisons. Tables 11 and 12 compare the minimum host and overall throughput results between the proposed algorithm and the ADJ-minmax approach for the random topology and the regular topology respectively. They indicate that the proposed algorithm increases the overall throughput by 3% and 1% for the random topology with two and three channels, and by 1% and 38% for the regular topology with two and three channels respectively.

Table 11: Throughput comparisons between proposed algorithm and ADJ-minmax for random topology.

# of channels	2		3	
algorithm	proposed	ADJ-minmax	proposed	ADJ-minmax
# of active APs	8	8	8	8
min. host through.	4.7	4.7	6.8	6.8
overall through.	147.97	144.27	219.23	217.28

Table 12: Throughput comparisons between proposed algorithm and ADJ-minmax for regular topology.

# of channels	2		3	
algorithm	proposed	ADJ-minmax	proposed	ADJ-minmax
# of active APs	4	4	4	4
min. host through.	3.31	3.30	4.50	3.39
overall through.	201.67	200.30	283.27	204.98

6 Evaluations by Testbed

In this section, we evaluate the elastic WLAN system testbed including the extended active AP configuration algorithm, using two network scenarios.

6.1 Network Scenarios

For evaluations, two network scenarios for the elastic WLAN system testbed, namely, the 3×3 scenario and the 5×5 scenario, are prepared in our building.

6.1.1 3×3 Scenario

In this scenario, three Raspberry Pi devices for APs and three Linux PCs for hosts are prepared in a 7m $\times$ 6m room. As shown in Figure 11, any AP is located 1m away from its neighbor APs, and any host is 1m away from its associated AP.

6.1.2 5×5 Scenario

In this scenario, three rooms separated by walls are used, where two rooms have the size of $7m \times 6m$ and one room has the size of $3.5m \times 6m$. Five Raspberry Pi devices for APs and five Linux PCs for

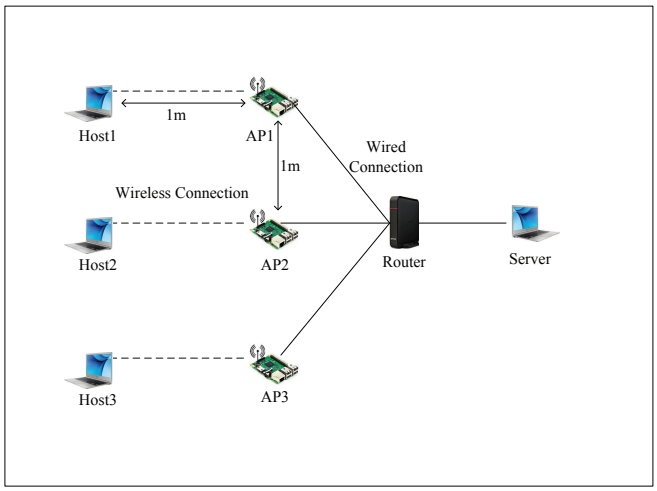


Figure 11: Testbed for 3×3 scenario.

hosts are distributed in the rooms as shown in Figure 12. Any Linux PC is located very close to the associated AP where the distance is less than 50cm. Any AP is $4m$ away from another AP in the same room. Because the signals from the APs in different rooms become small by the walls. Thus, the interferences among the APs can be reduced.

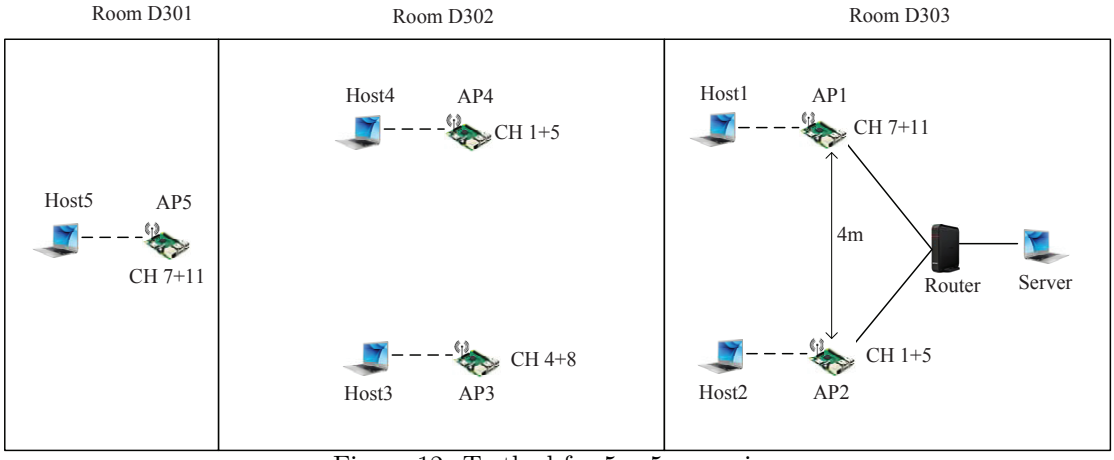


Figure 12: Testbed for 5×5 scenario.

6.2 Single Raspberry Pi AP Throughput Measurement

First, the throughput performance of the single Raspberry Pi AP with the associated Linux PC is measured in our building for reference, using TCP traffics with *iperf* [31]. The average measured throughput is 42 Mbps. It is noted that all the experiments were conducted on weekends to reduce the interferences from other APs in the same building.

6.3 Throughput Results in Testbed

To evaluate the effect of the proposed algorithm extension in the elastic WLAN system implementation, throughputs are measured with and without applying the algorithm using the elastic WLAN system testbed. Here, with the algorithm, the three bonded channels of 1+5, 4+8, and 7+11 are

assigned to the APs by the algorithm, where the channel bonding is adopted for 11n. On the other hands, without the algorithm, one of the three channels is randomly assigned to each AP. Any throughput is measured using *iperf* when all the hosts are communicating in parallel with the server through their connected APs. Besides, the effect of the active AP configuration algorithm is evaluated with the channel assignment extension.

6.3.1 3×3 Scenario Result

Table 13 shows the throughput results in the testbed for the 3×3 scenario. When the number of active APs is not minimized, the overall throughput by the random and algorithm channel assignment is 47.56 Mbps and 68.10 Mbps respectively. Then, by minimizing the number of active APs by the AP configuration algorithm, the overall throughput becomes 32.62 Mbps, 34.63 Mbps, and 33.56 Mbps respectively, when the single bonded channel of 1+5, 4+8, or 7+11 is assigned to the single AP. Thus, the channel assignment algorithm can improve the throughput performance of the small testbed when multiple APs are active.

Table 13: Throughput comparisons between proposed algorithm and random assignment for 3×3 scenario.

AP minimization	no		yes		
channel assignment	random	proposed	1+5	4+8	7+11
# of active APs	3	3	1	1	1
min. host through.	13.2	18.7	9.87	10.21	10.16
overall through.	47.56	68.10	32.62	34.63	33.56

6.3.2 5×5 Scenario Result

Table 14 shows throughput results in the testbed for the 5×5 scenario. When the number of active APs is not minimized, the overall throughput for the random and proposed channel assignment is 121.11 Mbps and 151.32 Mbps respectively. Then, by minimizing the number of active APs by the AP configuration algorithm, the overall throughput for the random and proposed channel assignment becomes 92.05 Mbps and 118.84 Mbps respectively. Thus, the proposed channel assignment algorithm can greatly improve the throughput performance of the larger testbed, while the active AP configuration algorithm can efficiently reduce the number of active APs with keeping the throughput performance.

Table 14: Throughput comparisons between proposed algorithm and random assignment for 5×5 scenario.

AP minimization	no		yes	
channel assignment	random	proposed	random	proposed
# of active APs	5	5	3	3
min. host through.	19.32	27.2	14.61	19.21
overall through.	121.11	151.32	92.05	118.84

7 Conclusions and Future Works

This paper presented the extension of the AP configuration algorithm to consider the channel assignment to the active APs under the limited number of non-interfered channels. Then, AP associations of hosts are improved by averaging loads among channels. The effectiveness of our proposal is evaluated using the WIMNET simulator in two network topologies. The elastic WLAN system including this proposal is implemented using Raspberry Pi for the AP, and the performance is verified through

experiments using the testbed. Our future works include further enhancements of the AP configuration algorithm and the elastic WLAN system implementation, 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:

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

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

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] 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.

[3] 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. Fifth Int. Conf. on Wireless and Mobile Commun. (ICWMC-2009), pp.134-139, 2009.

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

[5] "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.

[6] 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, 2016, Internet: http://www.ijnc.org/index.php/ijnc/article/view/134, Access Jan. 20, 2017.

[7] M. S. A. Mamun, M. E. Islam, and N. Funabiki, "Active access-point configuration algorithm with dynamic mobile router placement for elastic WLAN system," Int. Works. Auto. Self-Organ. Net., Dec. 2015.

[8] M. Elwekeil, M. Alghoniemy, M. El-Khamy, H. Furukawa, and O. Muta, "Optimal channel assignment for IEEE 802.11 Multi-cell WLANs," Signal Processing Conf. (EUSIPCO), Proc. 20th European, pp. 694–698, Aug. 2012.

[9] IEEE, "IEEE Std 802.11-2012, IEEE standard for information technology- telecommunications and information exchange between systems - Local and metropolitan area network- Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications,", Internet: https://standards.ieee.org/getieee802/download/802.11-2012.pdf, Access Jan. 20, 2017.

[10] T.-D. Chiueh, P.-Y. Tsai, and I.-W. Lai, "Baseband receiver design for wireless MIMO-OFDM communications," 2nd ed. Wiley-IEEE Press, 2012.

[11] "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. 20, 2017.

[12] J. Geier, "Assigning 802.11b access point channels," WiFi Planet, Feb., 2002.

[13] H. Skalli, S. K. Das, L. Lenzini, and M. Conti, "Traffic and interference aware channel assignment for multi-radio wireless mesh networks," Proc. of ACM Int. Conf. on Mobile Comput. and Netw (MOBICOM) pp. 15–26, Dec. 2006.

[14] K. Zhou, X. Jia, Y. Chang, and X. Tang, "Partially overlapping channel assignment for WLANs using SINR interference model," Int. J. Commun. Sys. vol. 27, no. 11, pp. 3082–3095, Nov. 2014.

[15] D. Gong, M. Zhao, and Y. Yang, "Channel assignment in multi-rate 802.11n WLANs," Wireless Commun. Netw. Conf., pp. 392–397, Apr. 2013.

[16] M. E. Islam, N. Funabiki, K. S. Lwin, M. S. A. Mamun, S. K. Debnath, and M. Kuribayashi, "Performance evaluations of elastic WLAN testbed," IEICE General Conf., pp. S76-S77, Mar. 2016.

[17] 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.

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

[19] "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. 20, 2017.

[20] "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. 20, 2017.

[21] "Arp-scan user guide," Internet: http://www.nta-monitor.com/wiki/index.php/Arp-scan_User_Guide, Access Jan. 20, 2017.

[22] "nm-tool," Internet: http://linux.die.net/man/1/nm-tool, Access Jan. 20, 2017.

[23] "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. 20, 2017.

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

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

[26] 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.

[27] "nmcli - command-line tool for controlling NetworkManager," Internet: http://manpages.ubuntu.com/manpages/precise/man1/nmcli.1.html, Access Jan. 20, 2017.

[28] "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. 20, 2017.

[29] "Learning Linux Commands: sed," Internet: https://linuxconfig.org/learning-linux-commands-sed, Access Jan. 20, 2017.

[30] N. Funabiki ed., "Wireless mesh networks," InTech-Open Access Pub., Jan. 2011, Internet: http://www.intechopen.com/books/wireless-mesh-networks, Access Jan. 20, 2017.

[31] "iPerf - the TCP, UDP and SCTP network bandwidth measurement tool," Internet: https://iperf.fr/, Access Jan. 20, 2017.