Cell Breathing Techniques for Load Balancing in Wireless LANs

Yigal Bejerano, *Member*, *IEEE*, and Seung-Jae Han, *Member*, *IEEE*

Abstract—Maximizing network throughput while providing fairness is one of the key challenges in wireless LANs (WLANs). This goal is typically achieved when the load of access points (APs) is balanced. Recent studies on operational WLANs, however, have shown that AP load is often substantially uneven. To alleviate such imbalance of load, several load balancing schemes have been proposed. These schemes commonly require proprietary software or hardware at the user side for controlling the user-AP association. In this paper we present a new load balancing technique by controlling the size of WLAN cells (i.e., AP's coverage range), which is conceptually similar to *cell breathing* in cellular networks. The proposed scheme *does not* require any modification to the users neither the IEEE 802.11 standard. It only requires the ability of dynamically changing the transmission power of the AP beacon messages. We develop a set of polynomial time algorithms that find the optimal beacon power settings which minimize the load of the most congested AP. We also consider the problem of network-wide min-max load balancing. Simulation results show that the performance of the proposed method is comparable with or superior to the best existing association-based methods.

Index Terms—IEEE 802.11 network, cell breathing, load balancing, fairness, combinatorial optimization.

1 Introduction

Recent studies [2], [3] on operational IEEE 802.11 wireless LANs (WLANs) have shown that traffic load is often unevenly distributed among the access points (APs). In WLANs, by default, a user scans all available channels and associates itself with an AP that has the strongest received signal strength indicator (RSSI), while being oblivious to the load of APs. As users are, typically, not evenly distributed, some APs tend to suffer from heavy load, while their adjacent APs may carry only light load. Such load imbalance among APs is undesirable as it hampers the network from fully utilizing its capacity and providing fair services to users. In this paper, we present a novel load balancing scheme that reduces the load of congested APs by forcing the users near the boundaries of congested cells to move to neighboring less congested cells. We achieve this via cell size dimensioning by controlling the transmission power of the AP beacon messages. In this paper, a WLAN *cell* is defined as a region in which the AP beacon signal has the strongest RSSI. Our approach is conceptually similar to *cell breathing* in cellular networks [4], [5]. We present an optimal algorithm that finds deterministic min-max load balancing solutions. Informally, a WLAN is called *min-max load balanced*, if it is impossible to reduce the load of any AP without increasing the load of other APs with equal or higher load. Our approach is practical since it does not require either user assistance or standard modification.

- *Y. Bejerano is with Bell Laboratories, Alcatel-Lucent, 600 Mountain Avenue, Murray Hill, NJ 07974-0636. E-mail: bej@research.bell-labs.com.*
- *S.-J. Han is with the Department of Computer Science, Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea. E-mail: sjhan@cs.yonsei.ac.kr.*

Manuscript received 8 May 2008; revised 10 Dec. 2008; accepted 10 Feb. 2009; published online 18 Feb. 2009.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2008-05-0179. Digital Object Identifier no. 10.1109/TMC.2009.50.

1.1 Related Work

Currently, the IEEE 802.11 standard does not provide any mechanism to resolve load imbalance. To overcome this deficiency, various load balancing schemes have been proposed. These methods commonly take the approach of directly controlling the user-AP association by deploying proprietary client software or hardware. For instance, vendors can incorporate certain load balancing features in their device drivers, AP firmwares, and WLAN cards. In these proprietary solutions, APs broadcast their load levels to users via modified beacon messages and each user chooses the least-loaded AP.

Several studies [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] have proposed a variety of association metrics instead of using the RSSI as the sole criterion. These metrics typically take into account such factors as the number of users currently associated with an AP, the mean RSSI of users currently associated with an AP, and the bandwidth that a new user can get if it is associated with an AP, e.g., [6], [7]. Balachandran et al. [8] proposed to associate a user with an AP that can provide a minimal bandwidth required by the user. In [9], Velayos et al. introduced a distributed load balancing architecture where the AP load is defined as the aggregated downlink and uplink traffic through the AP. In [10], [11], Kumar and coworkers proposed an association selection algorithms which are based on the concept of proportional fairness to balance between throughput and fairness. In [12], Kauffmann et al. provided a mathematical foundation for distributed frequency allocation and user association for efficient resource sharing. Recently, in [13], Shakkottai et al. considered a noncooperative multihoming approach and showed that under appropriate pricing, the system throughput is maximized. In [15], a strong relation between fairness and load balancing is shown. Most of these work determine only the association of newly arrived users. Tsai et al. [14] is an exception, in which Tsai and Lien

1536-1233/09/$25.00 © 2009 IEEE Published by the IEEE CS, CASS, ComSoc, IES, & SPS

proposed to reassociate users when the total load exceeds a certain threshold or the bandwidth allocated to users drops below a certain threshold. While the existing load balancing schemes achieved considerable improvement in terms of throughput and fairness, they require certain support from the client side. In contrast, the proposed scheme does not require any proprietary client support.

Cell breathing has been studied mostly in the context of CDMA cellular networks. The coverage and capacity of a CDMA cell are inversely related with each other [16]. The increase of the number of active users in a cell causes the increase of the total interference sensed at the base station. Therefore, in congested cells, users need to transmit with higher power to maintain a certain signal-to-interference ratio at the receiving base station. As the users in a congested cell increase their transmission power, they also increase their interference to the neighboring cells since all cells use the same frequency in CDMA networks. As a result, the overall network capacity may decrease. Furthermore, since the maximal transmission power of the users is bounded, the users who are far from the base station may experience poor services. To overcome these problems, the *cell breathing* approach was proposed by Togo et al. [4] and Jalali [5]. In essence, they reduce the size of congested cells.

Some studies have explored the benefit of combining the cell breathing methods with other interference mitigation methods. For instance, in [17], Yang and Ephremides presented a solution, which is based on the combination of cell breathing and bandwidth space partitioning. Du et al. [18] proposed a distributed load balancing technique that utilizes a bobble oscillation algorithm. In [19], Sang et al. proposed a method that coordinates the packet level scheduling with cell breathing techniques. These schemes are different from our scheme mainly in two aspects. First, they utilize probabilistic local optimization heuristics. Therefore, they do not provide any guarantee on the quality of the solutions, while our approach finds the optimal solution. Second, these schemes cannot be easily applied to IEEE 802.11 WLANs, e.g., they require the change of scheduling mechanism or bandwidth partition.

Recently in [20], Bahl et al. proposed cell breathing algorithms for WLANs that attempt to maximize the overall system throughput. In this paper, the authors formulate the cell breathing problem as a mixed Integer-Linear program, assuming the continuous power level assignment. In this paper, we present a cell breathing method which finds deterministic global optimal solutions for providing fairness under the assumption that only discrete set of power levels are feasible. The scheme in [20] and our scheme can be viewed as complementary.

1.2 Our Contributions

WLAN fairness needs to be considered in two aspects, *intra-AP fairness* and *inter-AP fairness*. Intra-AP fairness requires each AP to provide fairness to its associated users based on a given objective or metric. The scope of this task is locally confined to each AP cell and it can be obtained relatively easily, e.g., by controlling the traffic. Inter-AP fairness addresses a task of determining user association for ensuring fairness across all APs, assuming that each AP provides intra-AP fairness. Inter-AP fairness is obtained when the load of APs is balanced. Since the load of APs frequently changes as a result of varying channel conditions and the burstiness of user traffic, short-term load balancing causes instability of the system as users will be constantly shifted between APs. To prevent system instability, it is desirable to consider only the long-term channel condition and user traffic. In this paper, *we target at the long-term inter-AP fairness assuming that intra-AP fairness is provided.*

Our scheme provides inter-AP fairness by adjusting the cell sizes. The WLAN cell size is changed by controlling the transmission power of the AP beacons. Note that we do not change the transmission power of the data traffic messages. *The proposed algorithms are not tied to a particular load definition, but support a broad range of load definitions.* We treat the load of an AP as the aggregation of the load contributions of its associated users. The load contribution may be as simple as the number of users associated with an AP or can be more sophisticated to consider factors like transmission bit rates and traffic demands. Consequently, various load balancing and max-min fairness objectives can be achieved, such as bandwidth fairness, time fairness, and weighted fairness.[1] Our scheme *does neither* require any special assistance from users nor any change in the 802.11 standard. It only requires the ability of dynamically changing the transmission power of the AP beacons. Today, commercial AP products already support multiple transmission power levels, so we believe that this requirement can be relatively easily achieved.

Depending on the extent of available information, we consider two knowledge models. The first model assumes *complete knowledge*, in which user-AP association and AP load are known a priori for all possible beacon power settings. Since such information may not be readily available, we also consider the *limited knowledge* model, in which only information on the user-AP association and AP load for the current beacon power setting is available. Below, we describe an overview of our algorithms.

At first, we address the problem of minimizing the load of the congested APs. Let us call the AP with the maximal load as *congested AP* and its load as *congestion load*. We designed two polynomial time algorithms that find optimal solutions, one for the complete knowledge model and the other for the limited knowledge model. These results are intriguing, because similar load balancing problems, e.g., [15], are known to be strong NP-hard. It is particularly interesting that a polynomial time optimal algorithm exists for the limited knowledge model.

Second, we address the problem of min-max load balancing. This is a strong NP-hard problem. In [1], it is proved that there exists no algorithm that guarantees any coordinatewise approximation ratio, and the approximation ratio of any prefix-sum approximation algorithm is at least $\Omega(\log n)$, where n is the number of APs. In this paper, we solve a variant of this min-max problem, termed *min-max priority load balancing*, whose optimal solution can be calculated in polynomial time for both knowledge models. Here, the AP load is defined as an ordered pair of the

1. In the case of weighted max-min, fairness users may have different weights. Such fairness, for instance, enables us to give higher priorities to users that transmit with high bit rates over users with low rates.

aggregated load contributions of its associated users and a unique AP priority.

Our algorithms can be efficiently embedded into adaptive online strategy. Through extensive simulations, we show that the performance of the proposed methods is comparable with or superior to the existing association control methods, irrespective of network load patterns. In particular, we could achieve such performance even with a small number of power levels.

2 Network Model

We consider a WLAN with a set of APs, denoted by $\mathcal{A}$. $|\mathcal{A}|$ denotes the number of APs. All APs are directly attached to a wired infrastructure. Each AP can support several transmission power levels. Without loss of generality, each AP can use one of $K+1$ transmission power levels, denoted by $\{P_k | k \in [0..K]\}$, where the minimal and maximal levels are denoted by $P_{min} = P_0$ and $P_{max} = P_K$, respectively. A power level P_k is identified by its *power index* k and its transmission power is γ times stronger than its predecessor P_{k-1}, where γ is defined as $\gamma = \sqrt[K]{P_{max}/P_{min}}$, i.e., $\gamma > 1$. Because $P_k = \gamma \cdot P_{k-1}$, it follows that $P_k = P_{min} \cdot \gamma^k$ for every $k \in [0..K]$. This assumption is consistent with the transmission power level configurations supported by commercial AP products. We refer to the transmission power level of an AP $a \in \mathcal{A}$ as the *power index of AP* a and it is denoted by p_a. The *network coverage area* is defined as the union of the transmission ranges of all APs in $\mathcal{A}$. Initially, we assume that the AP deployment ensures a high degree of overlaps between the range of adjacent APs, so that every user is covered by at least one AP even when all APs are transmitting at the minimal power level P_{min}. We later relax this *strong coverage* assumption.

$\mathcal{U}$ denotes the set of all users in the network coverage area and $|\mathcal{U}|$ denotes their number. For optimal algorithms, we assume that users have a quasi-static mobility pattern. In other words, users can move freely, but they tend to stay in the same locations for a long period. This assumption is backed up by recent analysis of mobile user behavior [2], [3]. User mobility is handled by the online strategy described in Section 7. At any given time, each user is associated with a single AP. When a user enters a WLAN, it scans all channels (i.e., listening for the beacon messages) for identifying all APs in its reach. Then, based on the RSSI's of the beacon messages, the user associates itself with an AP that has the strongest RSSI. The RSSI depends on the transmission power of the beacon messages and the signal attenuation between AP and user.

Generally, the channel quality is time-varying. For the user-AP association decision, a user performs multiple samplings of the channel quality, and only the signal attenuation that results from long-term channel condition changes are utilized. Based on the RSSI, we can divide the network coverage area into $|\mathcal{A}|$ disjoint cells. The transmission bit rate for a user-AP pair is determined by the Signal-to-Noise Ratio (SNR). Users associated with the same AP may transmit at different bit rates, and each user may contribute a different amount of load to its serving AP. We assume that appropriate channel allocation was made, so that adjacent cells do not interfere with each other.

TABLE 1
Notations

Symbol	Semantics
$\mathcal{A}$	The set of all access points (APs).
$\mathcal{B}$	The bottleneck set of APs.
D (d)	The set of congested APs.
$\mathcal{F}$	The set of fixed APs.
K	The maximal transmission power index.
$l_{a,u}$	Load contribution of user u to AP a.
p_a	Transmission power index of AP a, $p_a \in [0..K]$.
$R_{u,a}$	Signal strength of AP a received by user u.
$\mathcal{S}$	A network state, $\mathcal{S} = \{(a, p_a)\}$.
$\tilde{\mathcal{S}}$	A recorded network state.
$\mathcal{U}$	The set of all users.
$\mathcal{U}_a$	The set of users associated with AP a.
y_a	The load of AP a.
Y	The network congestion load.
$\tilde{Y}$	The congestion load of the recorded state.
$\vec{Y}$	The AP load vector, $\vec{Y} = \{y_1, \cdots, y_{\lvert\mathcal{A}\rvert}\}$.

Our algorithms do not require any particular load definition. We only assume that the load of an AP a, denoted by y_a, is the sum of the load contributions of its associated users. We use $l_{a,u}$ to denote the *load contribution* (also termed the *weight*) of a user u on an AP a. We assume that this contribution is constant,[2] so that the load of each AP $a \in \mathcal{A}$ is $y_a = \sum_{u \in \mathcal{U}_a} l_{a,u}$, where $\mathcal{U}_a$ denotes the set of users associated with a. The APs that experience the maximal load are called the *congested* APs and their load, termed *congestion load*, is denoted by Y. Other APs with lower load are called *noncongested* APs.

Our load model can accommodate various additive load definitions such as the number of users associated with an AP. It can also deal with the multiplicative user load contributions, i.e., $y_a = \Pi_{u \in \mathcal{U}_a} l_{a,u}$, by applying $\log$ to both sides. By using different metrics for evaluating the load contribution of a user on its AP, $l_{a,u}$, different load balancing objectives can be achieved. For instance, by allocating a weight of one to all users, time fairness can be achieved. In such case, each AP serves each one of its associated users for the same time duration but does not necessarily provide the same bandwidth. Weights can be set based on the users transmission bit rates to achieve bandwidth fairness. Weights for different objectives can be combined, so that tradeoffs between different fairness objectives can be made. Table 1 summarizes the key notations used in this paper.

3 Cell Breathing Approach

In this section, we present the basic concept that underlies our approach and also address the algorithmic challenge.

3.1 Basic Concept

We reduce the load of congested APs by reducing the size of the corresponding cells. Such cell dimensioning can be obtained, for instance, by reducing the transmission power of the congested APs. This forces users near the

2. In Section 3, we explain why we consider constant load contributions of the users although we allow changes of the AP transmission power.

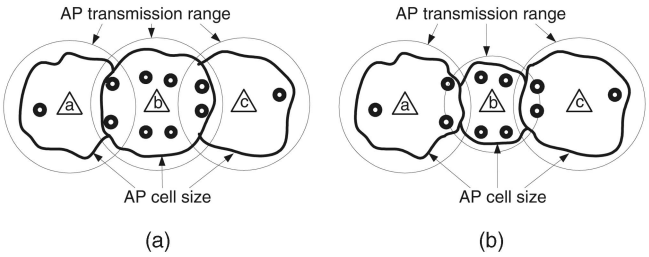


Fig. 1. Balancing AP load by adjusting transmission power. (a) All APs transmit with the same power level. (b) AP **b** transmit with lower power level than APs **a** and **c**.

congested cells' boundaries to shift to adjacent (less-congested) APs.

Example 1. Consider a WLAN with three APs, a, b, and c that transmit with maximal power P_{max} and let us assume that they are associated with 1, 8, and 1 users, respectively, as depicted in Fig. 1a. In this example, we define the load of an AP to be the number of its associated users. Clearly, b has much higher load than the other two APs. Now, by reducing the transmission power of b, the cell size of b is also reduced and four of the users associated with b suffer from low signal quality. These users initiate scanning operations that cause them to shift to adjacent APs. As a result, the number of users associated with the three APs are now 3, 4, and 3, respectively, as illustrated in Fig. 1b, and the AP load becomes more evenly distributed.

Reducing the transmission power of an AP affects the channel quality of all of its associated users, and this effect is not limited to those users that we intend to shift. The users who remain associated with the considered AP also experience lower channel quality and may have to communicate at a lower bit rate than before. This may result in longer transmission time of user traffic, which effectively increases the user load contributions on the AP, if the AP load is determined by considering not by the number of users but by the effective user throughput. Thus, we may end up with increasing the load of APs.

We overcome this problem by the separation between the transmission power of the data traffic and that of the AP beacon messages. On one hand, the transmission bit rate between a user and its associated AP is determined by the quality of the data traffic channel. Transmitting the data traffic with maximal power[3] maximizes the AP-user SNR and the bit rate. On the other hand, each user determines its association by performing a scanning operation, in which it evaluates the quality of the beacon messages of the APs in its vicinity. By reducing the beacon messages' power level of congested APs, we practically shrink the size of their cells, and consequently discourage user association. This concept of controlling the cells' dimensions by adapting power levels of the beacon messages is termed *cell breathing*. *The separation between the power levels of the data traffic and the beacon messages is the only modification that we require from APs*. In the remainder of this paper, when we say transmission power, we mean only the transmission power of beacon messages.

3. Power control of the AP data traffic can be done, separately, for reducing inter-AP interferences. Such power allocation is beyond the scope of this paper.

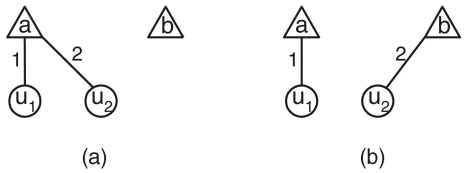


Fig. 2. Example results of a simple greedy algorithm. (a) AP **a** and AP **b** have the same power level. (b) AP **a** has lower power level than AP **b** ($p_a = p_b - 1$).

3.2 Algorithmic Challenges

One may consider a *greedy algorithm*that reduces the power level of the congested APs until any of the congested APs reaches to the minimal power level. This will shift users from congested APs to their neighbors, and the set of congested APs and their load may change during the execution of the algorithm. As we demonstrate in Example 2, even in a very simple case, the greedy algorithm may fail to find the optimal solution. Moreover, it can be shown that in some cases, the final congestion load is even higher than the initial congestion load. We need more sophisticated algorithms.

Example 2. Consider a WLAN with two APs, denoted as a and b, and two users u_1 and u_2. u_1 can only be attached to a and it yields a load of 1. u_2 produces load of 2 on its associated AP. It can be attached to both APs and it chooses an AP with higher RSSI. In case of a tie, u_2 chooses a, e.g., the RSSI of a may be slightly higher than the RSSI of b. It produces load of 2 on its associated AP. We assume that, initially, both APs transmit with the maximal power level, i.e., $p_a = p_b = K$, and therefore, both users are associated with a whose load becomes 3, as shown in Fig. 2a. To balance the load, the greedy algorithm reduces the power level of a and as a result u_2 changes the association to b. Now the loads on the two APs are 1 and 2, respectively, as depicted in Fig. 2b. At the next moment, the algorithm will reduce the power of b, which is now the congested AP, and u_2 will be shifted back to a. The algorithm continues to reduce the power levels of the APs, until they both transmit with the minimal power level. In the final setting, the load on a will be 3 and b has no load, which is obviously not the optimal solution.

4 Congestion Load Minimization

This section presents two algorithms for minimizing the AP congestion load: one for the *complete knowledge* (CK) model another for the *limited knowledge* (LK) model.

4.1 Problem Statement

Definition 1 (A Network State). *A network state $\mathcal{S}$ is defined as the beacon message power indices p_a of all the APs $a \in \mathcal{A}$, i.e., $\mathcal{S} = \{(a, p_a) | \forall a \in \mathcal{A} \wedge p_a \in [0..K]\}$. For simplicity we denote a state by $\mathcal{S} = \{(a, p_a)\}$.*

A network state determines the ranges of all AP cells and the user-AP association. Thus, a user may change its association only when the network state changes, which is termed a *state transition*. As the load of an AP is the aggregated load contribution of its associated users, the network state also determines the load of the APs. We refer to the set D of APs that suffer from the maximal load in a given network state as the *congested APs* and their load, termed the *congestion load*, is denoted by Y.

Definition 2 (AP Congestion Load Minimization). *The AP congestion load minimization problem seeks for a network state that minimizes the congestion load.*

Definition 3 (The Complete Knowledge Model). *A network has complete knowledge when the load contribution, $l_{a,u}$ for every user-AP pair, $u \in \mathcal{U}$ and $a \in \mathcal{A}$, is available.*

A complete knowledge model is feasible when all users collect the RSSI information from all of the nearby APs. Such a feature is suggested, for instance, in the IEEE 802.11-k proposal [22]. Unfortunately, this feature is currently not available in most existing WLANs. We use this model as a building block for the *limited knowledge* solution.

Definition 4 (The Limited Knowledge Model). *A network has limited knowledge when the available information comprises only the set of users that are currently associated with each AP and the load contributions, $l_{a,u}$, of each user u on its serving AP a.*

In the complete knowledge model, the algorithm can a priori determine the user-AP association in all possible states without actually changing the network state. This allows the offline calculation of a desired state. Such offline calculation is impossible in the limited knowledge model.

4.2 Preliminary Observations

Definition 5 (A Set A' Power Reduction). *A set A' power reduction ($A' \subset \mathcal{A}$) causes a state transition, in which all APs in A' reduce their power indices by one level and other APs maintain their current power levels.*

Lemma 1. *For a set A' power reduction, the only possible association changes are for the users who are associated with APs in A' to shift to the APs in the set $\mathcal{A} - A'$. That is, there are no association changes within the set A' nor the set $\mathcal{A} - A'$.*

Proof. This is a direct result from our power level definition given in Section 2. Essentially, a user associates itself with the AP that its beacon message has the strongest RSSI. Consider a user u that is associated with an AP $b \in \mathcal{A} - A'$. Thus, AP b has the strongest RSSI as detected by user u. Obviously, AP b still has the strongest RSSI also after the APs in A' have reduced their transmission power. From this, it follows that during a power reduction operation, a user will not disassociate itself from an AP that has not reduced its beacon message power. Thus, only users associated to APs in the set A' may change their associations.

When an AP reduces the transmit power of its beacon message, then the reduction of its beacon messages RSSI's experienced by the users is proportional to transmission power reduction. In our model, all the APs in A' reduce their transmission power with the same factor. Thus, after a set A' power reduction operation, the AP with the strongest RSSI *in the set* A', stay the same AP as before the set reduction operation. From this, it can be shown that after a set A' power reduction operation, a user will not change its association between two APs included in the set A' and this completes our proof. □

Corollary 1. *A set power reduction of all APs does not change the user-AP association.*

We define that a state $\mathcal{S}^{init}$ *dominates* a state $\mathcal{S}^{new}$ if $p_a^{init} \geq p_a^{new}$ for every AP $a \in \mathcal{A}$. By definition, a state dominates itself. The state $\mathcal{S} = \{(a, p_a = K)\}$, in which the power indices of all APs is K, dominates all other states. This state is called as the *maximal power state*. We define the network state $\mathcal{S}^{opt}$ as a *dominated optimal state*, if it is dominated by $\mathcal{S}^{init}$ and also minimizes the congestion load of the APs among all network states dominated by $\mathcal{S}^{init}$. An AP $a \in \mathcal{A}$ is termed *anchor* if its power index p_a^{opt} in the optimal state $\mathcal{S}^{opt}$ is the same as its power index p_a^{init} in $\mathcal{S}^{init}$. Finally, we denote the load on an AP a in state $\mathcal{S}^{init}$ by y_a^{init}.

Consider an initial state $\mathcal{S}^{init}$ and let $\mathcal{S}^{opt}$ be its dominated optimal state. From Lemma 1, it is "safe" to perform a set power reduction operation as long as the considered set does not contain any anchor AP. This observation is formulated below.

Lemma 2. *Consider an initial network state $\mathcal{S}^{init}$ and let $\mathcal{S}^{opt}$ be one of its dominated optimal states. Let C be the set of the anchor nodes in $\mathcal{S}^{init}$. Then, for every subset $A' \subseteq \mathcal{A} - C$, the state $\mathcal{S}^{new}$, obtained by performing a set A' power reduction operation, dominates the state $\mathcal{S}^{opt}$.*

Proof. This is a direct result from our power level definition given in Section 2 and the definition of anchor nodes. The state $\mathcal{S}^{init}$ dominates the set $\mathcal{S}^{opt}$. Thus for every AP $a \in \mathcal{A}$, it holds that $p_a^{init} \geq p_a^{new}$. Moreover, for every nonanchor node $a \in \mathcal{A} - C$, it follows that $p_a^{init} > p_a^{new}$. Therefore, the state $\mathcal{S}^{new}$ that results from a power reduction operation of any set $A' \subseteq \mathcal{A} - C$ dominates the state $\mathcal{S}^{opt}$. □

Corollary 2. *As long as $\mathcal{S}^{init}$ is not optimal, there exists a sequence of set power reduction operations that converges to an optimal state by reducing the power levels of nonanchor APs. The set of nonanchor APs can be obtained by taking the set of congested APs.*

Since $\mathcal{S}^{opt}$ is not a priori known, we cannot tell when an optimal solution is obtained. Thus, continuing the set power reduction operations on the congested APs may increase the load of some other APs to Y or even higher, which may prevent convergence to an optimal state, as demonstrated in Example 2. To overcome this problem, we use the following two approaches. The first approach, termed *bottleneck set power reduction*, guarantees that the congestion load never increases. We use this approach in the complete knowledge case to perform a sequence of set power reduction operations that *monotonically* converges to an optimal state. The second approach, termed *optimal state recording*, keeps

records of the best state found so far. We use this method in the limited knowledge case.

4.3 Bottleneck Set

We define a *bottleneck set* $\mathcal{B} \subseteq \mathcal{A}$ as the minimal set of APs that contains all congested APs (with load Y) as well as all APs whose load may elevate to Y or above as a result of a set power reduction operation. Formally, the set $\mathcal{B}$ is defined in a recursive manner as follows:

Definition 6 (The bottleneck set). *Consider an initial state $\mathcal{S}^0 = \mathcal{S}^{init}$, a load y_a^0 for each AP $a \in \mathcal{A}$, and a congestion load Y. We define B_0 as a set of congested APs in $\mathcal{S}^0$, i.e., APs with load Y in $\mathcal{S}^0$*

$$B_0 = \{a | a \in \mathcal{A} \wedge y_a^0 = Y\}.$$

Given a state $\mathcal{S}^{i-1}$ and a set B_{i-1}, the state $\mathcal{S}^i$ is obtained by a set B_{i-1} power reduction operation from the initial state $\mathcal{S}^0$, if such a reduction is feasible. The set B_i comprises the set B_{i-1} and all the other APs whose load have changed from $\mathcal{S}^{i-1}$ and is equal to Y or higher at $\mathcal{S}^i$

$$B_i = B_{i-1} \bigcup \{a | a \in \mathcal{A} \wedge y_a^i \geq Y\}.$$

The bottleneck set $\mathcal{B}$ of the state $\mathcal{S}^0$ is defined as $\mathcal{B} = B_i$ for the first index i such that B_i contains any AP with minimal transmission power, or $B_i = B_{i-1}$.

As we show in Section 4.4, the bottleneck set is a useful means for deciding whether the system congestion load can be further reduced by power reduction operation. Obviously, if the bottleneck set contains APs that transmit with minimal power, power reduction operation cannot be performed. We illustrate the bottleneck set concept in Example 3.

Example 3. Consider the WLAN used in Example 2. When two APs have the same power level, as depicted in Fig. 2a, reducing the power level of the AP a decreases its load from 3 to 1, while it increases the load of the AP b to 2. The bottleneck set at that moment contains only a. When a has one power level lower than b, as illustrated in Fig. 2b, the power reduction of b reduces its load from 2 to 0 and increases as load to 3. In this case, the bottleneck set contains both APs.

In the case of complete knowledge, the bottleneck set of a given network state can be easily calculated. First, we calculate the RSSI between each AP a and each user u, denoted by $R_{u,a}$. This information allows us to determine the initial user-AP association, the load y_a of each AP a, and the maximal load Y. From these, we can also calculate the set B_0. For a given B_{i-1}, we can calculate a reduced $R'_{u,a}$ for every AP $a \in B_{i-1}$ and every user u associated with AP a, assuming that the power index of every AP $a \in B_{i-1}$ is one level lower than the one in the initial state $\mathcal{S}^{init}$. From these calculations, we identify all users who will change their association and we evaluate the load of the APs which are not included in B_{i-1}. Then, we construct the set B_i that contains the set B_{i-1} and all APs (not in B_{i-1}) whose new load is Y or higher. We execute this process iteratively until any termination condition is met. Upon termination, the bottleneck set $\mathcal{B}$ is obtained. A formal description of the $Calc\_Bottleneck\_Set$ routine is given in Fig. 3. Recall that this routine *does not* actually change the network state and just calculates the bottleneck set of the initial state $\mathcal{S}^{init}$ by simulating the network state changes.

```
Alg CK_Min_Congestion_Load_Alg(A,U)
    for every AP a ∈ A let p_a = K
    End_Flag = FALSE
    while (End_Flag = FALSE) do
        // Find congested APs and their load.
        Y = max_{a∈A} y_a
        D = {a | a ∈ A ∧ y_a = Y}
        B = Calc_Bottleneck_Set({(a, p_a)}, D, Y)
        if ((B = A) ∨ (exist a ∈ B s.t. p_a = 0)) then
            End_Flag = TRUE
        else
            for every AP a ∈ B let p_a = p_a − 1
        end if
    end while
    return {(a, p_a) | a ∈ A}
end
```

Fig. 3. A formal description of the complete knowledge congestion load minimization algorithm.

Lemma 3. *Consider a state $\mathcal{S}$ with a bottleneck set $\mathcal{B}$ and its congestion load Y. The set $\mathcal{B}$ is the minimal set of APs that contains all congested APs and the load of every AP $a \in \mathcal{A}$ stays the same or strictly less than Y after the set $\mathcal{B}$ reduction operation, if such a reduction is feasible. Moreover, any other set A' that has the above property must include $\mathcal{B}$.*

Proof. In this proof, we assume that the bottleneck set does not contain any AP with minimal transmission power, because otherwise a bottleneck set reduction operation cannot be done. From the definition of the set $\mathcal{B}$, it follows that there is no noncongested AP whose load is elevated to Y or higher due to a set $\mathcal{B}$ power reduction. Now, we just need to show that any other A', which contains all congested APs and has the same property, must also contain $\mathcal{B}$. Suppose that such A' which does not contain $\mathcal{B}$ exists. Let B_i be the largest set as defined in Definition 6 that satisfies $B_i \subseteq \mathcal{B} \bigcap A'$. Recall that such B_i exists since B_0 is included in both $\mathcal{B}$ and A'. From this, it follows that there is an AP $a \in B_{i+1}$ that is not included in A'. However, from the definition of $\mathcal{B}$, it can be shown that reducing the power of set B_i causes the load on a to become Y or higher, which contradicts the above assumption. Therefore, the set A' must include AP a as well and this completes our proof. □

Lemma 4. *Let $\{\mathcal{S}^j\}$ be a sequence of states generated by sequential bottleneck set power reduction operations and let Y_j be the congestion load of state $\mathcal{S}^j$. $\{Y_j\}$ is a monotonic nonincreasing sequence.*

Proof. From Lemma 3, it follows that a bottleneck set reduction operation does not increase the load on any noncongested AP to Y or higher. From Lemma 1, it follows that users change their association to APs that are not included in the bottleneck set. Therefore, the APs' congestion load never increases and may only decrease. □

Theorem 1. *Consider any suboptimal state $\mathcal{S}^{init}$ that dominates an optimal state $\mathcal{S}^{opt}$ and let $\mathcal{B}$ be its bottleneck set. Then, the state $\mathcal{S}^{new}$ obtained by a set $\mathcal{B}$ power reduction operation on $\mathcal{S}^{init}$ also dominates $\mathcal{S}^{opt}$.*

Proof. Since $\mathcal{S}^{init}$ is suboptimal, its congestion load is strictly higher than the congestion load of $\mathcal{S}^{opt}$. From Lemma 3, it follows that the set $\mathcal{B}$ is the smallest set of APs that contains the congested APs and its power reduction operation does not increase the load of any other AP to Y or higher. Consequently, $\mathcal{S}^{new}$ is either $\mathcal{S}^{opt}$ or it dominates $\mathcal{S}^{opt}$. □

4.4 Complete Knowledge Algorithm

We now present the complete knowledge algorithm. Recall that in this model, the user-AP association and the corresponding AP load are known a priori for all possible beacon power assignments.

The algorithm starts with the *maximal power state* in which all APs transmit with the maximal power. Clearly, this state dominates all other state and, in particular, the optimal states. The algorithm, iteratively, calculates the bottleneck set $\mathcal{B}$, as described in Section 4.3. Based on the calculated set $\mathcal{B}$, the algorithm determines whether it needs to apply another set power reduction operation or an optimal state is found. To this end, it utilizes two termination conditions. The first condition checks if $\mathcal{B} = \mathcal{A}$. Recall that from Corollary 1, it follows that reducing the transmission power of all APs does not change the user-AP association. Consequently, it cannot reduce the maximal AP load. In reality, this condition is satisfied when the load of all APs are balanced and any power reduction operation will cause some APs to be congested. The second condition checks if the bottleneck set $\mathcal{B}$ contains an AP that transmits with the minimal transmission power. In such case, the power level of all APs in $\mathcal{B}$ cannot be equally decreased and the algorithm halts. Such a case, typically, occurs when the AP load is not balanced and the algorithm attempts to reduce the maximal load by repeatedly reducing the power of the congested APs. A formal description of the algorithm is given in Fig. 3 and Example 4 illustrates a typical execution.

Example 4. Consider a WLAN with three APs, denoted as a, b, and c, and four users $\{u_1, u_2, u_3, u_4\}$. Let $K = 2$, i.e., three power level. All possible user-AP association are depicted in Fig. 4a, in which solid lines indicate the default association when all APs have the same power level and dotted lines indicate other possible association. The number on each line indicates the load contribution by a user to the load of the associated AP. For example, u_1 can only be associated with a and it yields a load of 4. u_3 can be associated with all APs and its load contribution is 2. It chooses an AP with the highest power level, but in case of a tie, it prefers c and if c has lower power level than a and b it prefers b. An asterisk at the numbers indicates that a gap of two power levels is required for shifting the corresponding user, e.g., u_2 changes its association from a to b only if $p_a = 0$ and $p_b = 2$.

At the initial state, all APs have the same power index 2, and the initial user-AP association are given in Fig. 4b, where the load of each AP is 7, 0, and 12, respectively. In the first iteration, the bottleneck set $\mathcal{B} = \{c\}$ and the power index of c is reduced to 1. Consequently, u_3 changes its association to b. The new user-AP association and AP load are shown in Fig. 4c. Note that c is still the congested AP. However, an additional power reduction of c will change the association of u_4 to b and the load of b will become 17. Thus, in the second iteration, the bottleneck set contains both c and b, as illustrated in Fig. 4d. This is the last iteration since now $p_c = 0$. Clearly, the final state, presented in Fig. 4d, minimizes the congestion load.

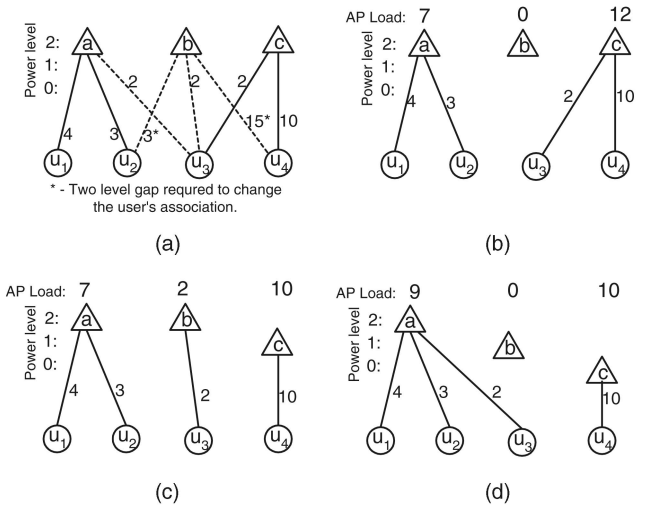


Fig. 4. Example of the complete knowledge algorithm. (a) The considered WLAN. (b) The initial state. (c) After first iteration, $\mathrm{B} = \{\mathrm{c}\}$. (d) After second iteration, $\mathrm{B} = \{\mathrm{b}, \mathrm{c}\}$.

Theorem 2. *The complete knowledge algorithm always finds an optimal state that minimizes the congestion load.*

Proof. The algorithm starts with the maximal power state that dominates the optimal solution. From Theorem 1, it follows that as long as the algorithm has not reached the optimal state, the resulting state, obtained by a bottleneck set power reduction operation, also dominates the optimal solution. From Lemma 4, it can be shown that during the execution of the algorithm, the congestion load never increases. So, now we just have to show that the algorithm stops with an optimal state. Since the number of possible set power reduction operation is limited, The algorithm must stop after at most $K \cdot |\mathcal{A}|$ set power reduction operations. Now suppose that the final state is not optimal. Since the sequence of maximal load Y_j is nonincreasing, we conclude that the algorithm must have stopped before finding an optimal state. The algorithm stopped because $\mathcal{B}$ contains an AP a with $p_a = 0$ or $\mathcal{B} = \mathcal{A}$. Recall that in the first case, set $\mathcal{B}$ power reduction operation cannot be done and in the second case, from Corollary 1 results that such operation does not reduce the congestion load. Thus, there is a set A' that does not contain the bottleneck set $\mathcal{B}$ and its set power reduction operation reduces the congestion load. However, from Lemma 3, we know that such a set A' does not exist. □

```
Alg LK_Min_Congestion_Load_Alg(A,U)
    for every AP a ∈ A let p_a = K
    // Set power indices of all APs to K
    // and evaluate AP load.
    S̃ = {(a, p_a)}
    Ỹ = max_{a∈A} y_a
    End_Flag = FALSE
    while (End_Flag = FALSE) do
        Y = max_{a∈A} y_a
        D = {a | a ∈ A ∧ y_a = Y}
        if (exist a ∈ D s.t. p_a = 0) then
            End_Flag = TRUE
        else
            // Decrease power indices of all APs in D by 1
            // and evaluate AP load.
            for every AP a ∈ D let p_a = p_a − 1
            Y = max_{a∈A} y_a
            if (Y < Ỹ) then
                S̃ = {(a, p_a)}
                Ỹ = Y
            end if
        end if
    end while
    return S̃.
end
```

Fig. 5. A formal description of the limited knowledge congestion load minimization algorithm.

The computational complexity of the algorithm is $O(K \cdot |\mathcal{A}|^3 \cdot |\mathcal{U}|)$. Thus, from theoretical perspective, the algorithm has pseudopolynomial running time. In practice, K is a small value like 10, and therefore, for any practical mean, the algorithm has polynomial running time.

4.5 Limited Knowledge Algorithm

We now present our *limited knowledge algorithm*. Unlike the complete knowledge case, in this model, the algorithm cannot calculate bottleneck set in advance. We overcome this obstacle by using Corollary 2. According to it, as long as a network state is suboptimal and it dominates an optimal solution, a sequence of set power reduction operations of congested APs converges to the optimal state. Thus, after each iteration, the algorithm updates the APs beacon power settings and evaluates the AP loads. This process raises the problem of determining a "termination condition" when an optimal solution is found. Without the termination condition, as demonstrated in Example 2, we may end up with a suboptimal solution. To this end, we use an *optimal state recording* approach that keeps record of the network state with the lowest congestion load found so far. We define two variables for recording. The first is $\tilde{\mathcal{S}}$ that keeps the *recorded state* and the second is $\tilde{Y}$ that keeps the congestion load value of state $\tilde{\mathcal{S}}$, termed the *recorded congestion load*.

The algorithm works as follows. It starts with the maximal power state and it initializes the recorded state $\tilde{\mathcal{S}}$ and the recorded congestion load $\tilde{Y}$ accordingly. Then, the algorithm, iteratively, calculates the set D of congested APs and, as long as the set D does not contain any AP with minimal power level, it performs a set D power reduction operation. After each iteration, the algorithm evaluates the congestion load of the new state and if that load is lower than the recorded congestion load, then the algorithm updates its variables $\tilde{\mathcal{S}}$ and $\tilde{Y}$ correspondingly. At the end, the algorithm sets the AP power levels according to the recorded network state. A formal description of the limited knowledge algorithm is given in Fig. 5 and a typical execution is illustrated in Example 5.

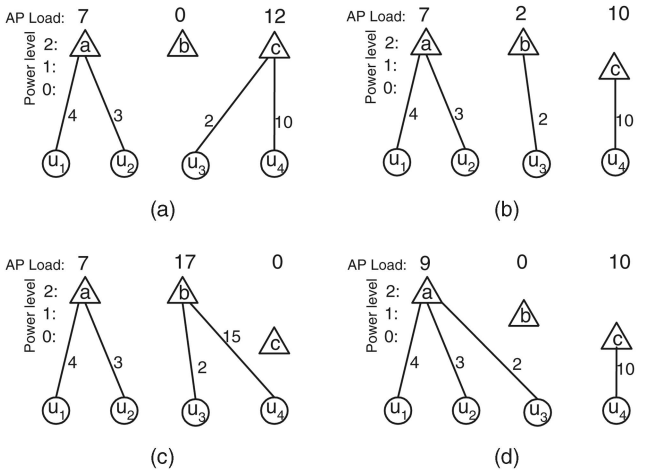


Fig. 6. Example of the limited knowledge algorithm. (a) Initial state, $\mathrm{D} = \{\mathrm{c}\}$. (b) After first iteration, $\mathrm{D} = \{\mathrm{c}\}$. (c) After second iteration, $\mathrm{D} = \{\mathrm{b}\}$. (d) After 3rd iteration, $\mathrm{D} = \{\mathrm{c}\}$.

Example 5. Consider the WLAN used in Example 4. At the initial state, all APs have the same power index 2, and the initial user-AP association and the AP load are shown in Fig. 6a. Since c is the congested AP, the algorithm reduces its power index twice in two successive iterations, as shown in Figs. 6b and 6c. After the first iteration, the load on c is reduced from 12 to 10 and the algorithm keeps a record of this state. After the second iteration, b becomes the congested AP with congestion load 17. In the third iteration, the algorithm reduces its power index and u_3 changes its association accordingly. As a result, c becomes the congested AP again. Since it transmits with minimal power, the iteration loop ends. At the end, the algorithm configures the APs with the recorded state, as shown in Fig. 6b.

Theorem 3. *The limited knowledge algorithm always finds an optimal state that minimizes the congestion load.*

Proof. The algorithm starts with the maximal power state that dominates any optimal solution. Since the algorithm keeps record of the state with the minimal congestion load, we just have to show that it reaches an optimal state before it stops. Now suppose in contrast that the algorithm have not found an optimal state during its execution. Recall that Corollary 2 claims that as long as the algorithm have not reached an optimal state, the sequence of states obtained by iteratively reducing the power of the congested APs also dominates the optimal state. Consequently, the algorithm stops with a suboptimal state that dominates the optimal state. However, the algorithm halts when any congested AP transmits with minimal power. Thus, the load on this AP cannot be reduced by further power reduction operations. This implies that either the final state is optimal or it does not dominate an optimal state, which contradicts our

assumption that the algorithm stopped before finding an optimal state. □

5 Min-Max Load Balancing

The algorithms presented in Section 4 minimize the load of the congested AP, but they do not necessarily balance the load of the noncongested APs, as demonstrated in Examples 4 and 5. In this section, we consider min-max load balancing approach that not only minimizes the network congestion load but also balances the load of the noncongested APs. As mentioned earlier, the proposed approach can be used for obtaining various max-min fairness objectives by associating each users with appropriate load contributions. Unfortunately, min-max load balancing is NP-hard problem and it is hard to find even an approximated solution. In this paper, we solve a variant of the min-max problem, termed *min-max priority-load balancing problem*, whose optimal solution can be found in polynomial time. We present our algorithm for this problem in the limited knowledge case. Obviously, it can be used for the complete knowledge case as well.

5.1 Problem Statement

A commonly used approach to evaluate the quality of a load balancing method is whether it generates a min-max load balanced solution [15], [21]. Informally, we say that a network state is *min-max load balanced* if there is no way to reduce the load of any AP without increasing the load of another AP with same or higher load. We define the *load vector* $\vec{Y} = \{y_1, \ldots, y_{|\mathcal{A}|}\}$ of a state $\mathcal{S}$ to be the $|\mathcal{A}|$-tuple consisting of the load of each AP sorted in decreasing order.

Definition 7 (Min-Max Load Balanced Network State). *A feasible network state $\mathcal{S}$ is called min-max load balanced if its corresponding load vector $\vec{Y} = \{y_1, \ldots, y_{|\mathcal{A}|}\}$ has the same or lower lexicographical value than any other load vector $\vec{Y'} = \{y'_1, \ldots, y'_{|\mathcal{A}|}\}$ of any other feasible state $\mathcal{S}'$. In other words, if $\vec{Y} \neq \vec{Y'}$, where $\vec{Y}$ is the load vector of a min-max load balanced state, there exists an index j such that $y_j < y'_j$ and for every index $i < j$, it follows that $y_i = y'_i$.*

We can show that the problem of finding a min-max load balanced state is NP-hard. Furthermore, we can prove that even a simpler problem, i.e., the problem of identifying the minimal set of congested APs for a known minimal congestion load, is by itself NP-hard.

Theorem 4. *Consider a WLAN and let Y be a known lower bound on its congestion load that can be obtained by the cell breathing approach. Then, identifying a network state that minimizes the number of congested APs is NP-hard, even for instances with only two power levels.*

Corollary 3. *The problem of finding a min-max load balanced state is NP-hard.*

The proofs of the above claims are based on the reduction from the minimum dominating set (MDS) problem [24]. Recall the MDS problem is not just hard to calculate, it is also hard to approximate and there is no algorithm that can find an approximation ratio smaller then $c\log(|V|)$, for some $c > 0$, unless $P = NP$ [23]. By using this reduction and the hardness property of the MDS problem, it can be shown that our min-max load balancing problem is also hard to approximate. The full proof can be found in [1].

We now present a variant of the min-max load balancing problem. In this problem, we assume that each AP $a \in \mathcal{A}$ has a *unique priority*, also termed a *weight*, $w_a \in [1..|\mathcal{A}|]$, which may indicate the AP's importance. In this study, we do not address the problem of allocating priority to the APs. In practice, the APs' priorities can be determined according to the APs' locations and the users that they serve. In the following, we give a new AP load definition, termed as a *priority load*.

Definition 8 (A priority load of an AP). *Consider an AP $a \in \mathcal{A}$ with priority w_a and let $l_a = \sum_{u \in \mathcal{U}_a} l_{a,u}$ be the aggregated load of all of its associated users. The priority load of AP a, denoted by y_a, is defined as the ordered pair $y_a = (l_a, w_a)$.*

For simplicity, we refer to an AP priority load by the *AP load*. We say that AP a has higher load than AP b if $y_a = (l_a, w_a)$ has a higher *lexicographically* value than $y_b = (l_b, w_b)$, i.e., one of the following two condition satisfied: 1) $l_a > l_b$, or 2) $l_a = l_b$ and $w_a > w_b$. Thus, our new objective is finding a network state that provides a *min-max priority load balanced*solution. Since there are no two APs with the same priority, it can be shown that there are no two APs with the same (priority) load. This ensures the following useful property.

Property 1. *With the priority load definition, at any network state, the set of congested APs always contains a single AP.*

Property 1 significantly simplifies the calculation of a min-max solution since at each iteration, the congested AP is uniquely defined.

5.2 Min-Max Algorithm

We now present our min-max algorithm for the limited knowledge model. The algorithm iteratively finds a *min-max priority-load-balanced state* that yields the *optimal load vector* $\vec{Y}$. At any iteration m, $m \in [1..|\mathcal{A}|]$, we call a routine to calculate a network state that minimizes the priority load of the mth coordinate of the load vector. The routine needs to satisfy two requirements:

Requirement 1. The initial state of each iteration, m, must dominate the optimal state.

Requirement 2. The calculated network state at the mth iteration should not affect (increase) the load of the APs that their load have already been determined by the previous iterations.

To meet Requirement 1, the algorithm starts with the maximal power state in the first iteration and we need to ensure that each iteration ends with dominating state of the optimal solution. Moreover, to meet Requirement 2, we define a set of *fixed APs*, $\mathcal{F}$, whose load have already been determined by previous iterations. Initially, the set $\mathcal{F}$ is empty and at each iteration, a new AP is added to it, until $\mathcal{F}$ contains all the APs. We refine the definition of the *congestion load* Y as the maximal load on any nonfixed AP. From Property 1, it follows that at any given time, there

Alg Min_Max_Priority_Load_Balancing_Alg($\mathcal{A}$,$\mathcal{U}$)
  $\mathcal{S} = \{(a, p_a = K) | \forall a \in \mathcal{A}\}$
  $\mathcal{F} = \emptyset$
  while ($\mathcal{F} \neq \mathcal{A}$) **do**
    $(\mathcal{S}, d)$ =LK_Minimize_m_Coordinate($\mathcal{S}, \mathcal{F}$)
    $\mathcal{F} = \mathcal{F} \bigcup \{d\}$
  end while
  return $\mathcal{S}$
end

Routine LK_Minimize_m_Coordinate($\mathcal{S}^{init}, \mathcal{F}$)
  $\tilde{\mathcal{S}} = \mathcal{S}^{init}$
  $\tilde{Y} = Y = \max_{a \in \mathcal{A}-\mathcal{F}} y_a$
  $\tilde{d} = d = a \;\; s.t. \; a \in \mathcal{A} - \mathcal{F} \wedge y_a = Y$
  $End\_Flag = FALSE$
  while ($End\_Flag = FALSE$) **do**
⋆   **if** ($p_d = 0$) **then**
      $End\_Flag = TRUE$
    else
      // Power reduction and AP load evaluation.
      $p_d = p_d - 1$
      $Y = \max_{a \in \mathcal{A}-\mathcal{F}} y_a$
      $d = a \;\; s.t. \; a \in \mathcal{A} - \mathcal{F} \wedge y_a = Y$
      // Check if fixed AP load was increased.
      if (exist $a \in \mathcal{F}$ s.t. $\tilde{y}_a < y_a$) **then**
        $End\_Flag = TRUE$
      // Check if a better network state was found.
      else if ($Y < \tilde{Y}$) **then**
        $\tilde{\mathcal{S}} = \{(a, p_a)\}$
        $\tilde{Y} = Y$
        $\tilde{d} = d$
      end if
    end if
  end while
  Return ($\tilde{\mathcal{S}}$, $\tilde{d}$).
end

Fig. 7. A formal description of the Min-Max priority load balancing algorithm.

is only a single nonfixed AP, termed the *congested AP*, d, which carries the congestion load.

At each iteration m, the algorithm invokes the $LK\_Minimize\ m\_Coordinate$ routine for minimize the mth coordinate of the load vector. The routine uses three recording variables. The *recorded congestion load* $\tilde{Y}$ keeps the congestion load value of the optimal state found so far. The *recorded state* variable $\tilde{\mathcal{S}}$ records the first discovered state with congestion load $\tilde{Y}$. While the *recorded congestion AP*, $\tilde{d}$, identifies the congested AP with this load.

At the beginning, the routine initializes the recording variables $\tilde{\mathcal{S}}$, $\tilde{Y}$, and $\tilde{d}$ with the iteration initial state, its congestion load, and the congested AP, accordingly. Then, the routine, iteratively, calculates the congested AP d and it stops if AP d is already transmitting with minimal power level. If not, the routine performs power reduction operation of d and evaluates the congestion load Y as well as the load of the fixed APs in the new state. It stops if one of the fixed APs suffers from elevated load. This ensures Requirement 2 to preserve the load of the fixed APs. Otherwise, if a state with a lower congestion load is discovered, the routine keeps record of this state by updating the recording variables. At the end, the routine sets the AP power levels

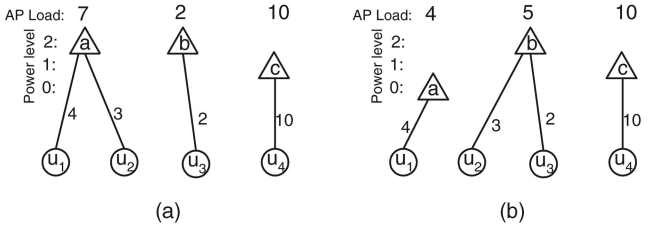


Fig. 8. Example of the min-max priority load balancing algorithm. (a) After first iteration, $\mathrm{F} = \{\mathrm{c}\}$. (b) After second iteration, $\mathrm{F} = \{\mathrm{c}, \mathrm{b}\}$.

according to the recorded network state and returns the recorded state $\tilde{\mathcal{S}}$ and the corresponding congested AP $\tilde{d}$. The later is added to the set of fixed APs and the algorithm invokes the routine again for minimizing the load of the $(m+1)th$ coordinate. A typical execution is illustrated in Example 6 and a formal description of the algorithm is given in Fig. 7.

Example 6. Consider the WLAN used in Example 4. In this case, the first invocation of the $LK\_Minimize\_m\_Coordinate$ routine returns the network state depicted in Fig. 8a. As demonstrated in Example 5, this state dominates any other state that minimizes the first coordinate of the load vector. The second invocation of the $LK\_Minimize\_m\_Coordinate$ routine returns the state shown in Fig. 8b, which is the only min-max load balanced state of this network (regardless of the AP priorities).

We now prove that the proposed algorithm finds the optimal load vector.

Lemma 5. *Consider any iteration m that satisfies Requirement 1. Then, at the end of the iteration, the recorded state $\tilde{\mathcal{S}}$ minimizes the mth coordinate of the load vector, without changing the load of the fixed APs. It also satisfies Requirement 1 at the beginning of the $(m+1)$th iteration.*

Proof. Each iteration starts by preserving the initial state and it stop when the load of a fixed AP has changes. Thus, at any given time, $\tilde{\mathcal{S}}$ maintains the load of the fixed APs. Since $\tilde{Y}$ is the smallest detected congestion load, $\tilde{\mathcal{S}}$ also minimizes the m coordinate. Thus, we have only to show that $\tilde{\mathcal{S}}$ preserves Requirement 1 at the beginning of the $(m+1)$th iteration. From our assumption, it can be found that the m iteration starts with a state that dominates the optimal solution. $\tilde{\mathcal{S}}$ is the first detected state that minimizes the congestion load and satisfies Requirement 2. Thus, it dominates any state that minimizes the first m coordinates, and consequently, it satisfies Requirement 1. □

Theorem 5. *The algorithm always finds a min-max priority-load-balanced solution.*

Proof. A direct result from inductive application of Lemma 5 for all of the m coordinates of the optimal load vector. □

At a final point, it can be shown that the complexity of the algorithm is $O(K \cdot |\mathcal{A}|^4 \cdot |\mathcal{U}|)$.

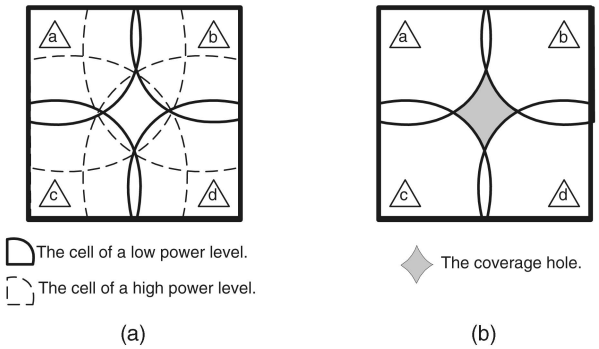


Fig. 9. Coverage hole in sparse AP deployment. (a) The transmission ranges of the two power levels of the APs. (b) A coverage hole when all the APs transmit with low power.

6 Coverage Hole Issue

The algorithms presented so far assume that every user is covered by at least one AP even when all APs are transmitting at the minimal power level P_{min}. This strong coverage requirement may not be satisfied in sparse AP deployments. In such deployment, coverage holes might exist if low transmission power is used for the beacons, as we demonstrate in Example 7.

Example 7. Consider a WLAN with four APs. Each AP support two power levels $p \in [0..1]$ and the corresponding transmission ranges of each AP are depicted in Fig. 9a. In this example, when all the APs transmit with minimal power level P_{min}, the network has a coverage hole, as shown in Fig. 9b. Thus, to prevent the coverage hole, at least one AP has to transmit with high transmission power.

Notice that the minimal power level that an AP may use without causing coverage holes depends on the power levels of its adjacent APs. We specify the minimal transmission power of AP a as a function of the transmission power of its neighbors N_a. That is, each AP $a \in \mathcal{A}$ maintains *minimal admissible power level function* denoted by $p_{min,a} = f_a(N_a)$. For instance, in Example 7, the function of any AP a is $f_a(N_a) = 1$ if all the neighbors of a transmit with minimal power. Otherwise, $f_a(N_a) = 0$.

Obviously, the minimal admissible power level function f_a of an AP a depends on its neighborhood N_a. To calculate these functions, we rely on a radio coverage survey method. For the coverage survey, special mobile terminals with dedicated monitoring software are placed at the strategic locations and monitor the quality of the received beacon messages, as the APs try different power settings. Then, the collected readings are used for calculating the minimal admissible power level function of each AP. This radio coverage survey can be performed upon the WLAN deployment or can be periodically repeated.

Once the minimal admissible power level functions are determined, the algorithms presented in Sections 4 and 5 need to be slightly modified as follows. Since these algorithms are based on set power reduction operations, it is required to verify before performing such operation that the new network state does not contain coverage holes. Otherwise, the algorithms preclude the power reduction operation. In the case of minimizing the congestion load, the algorithms terminate and it is easy to prove that Theorems 2 and 3 still hold. In the case of min-max priority load balancing, the $LK\_Minimize\_m\_Coordinate$ routine stops before producing coverage holes and the algorithm turns to optimize the next coordinate of the load vector. This can be done by replacing the "if" condition marked by $\star$ in Fig. 7 with the condition, "**if**$(p_d \leq f_a(N_a))$, **then**$\cdots$". Theorem 5 is still valid with this modification.

7 Online Strategy

Executing the optimization algorithm every time a user arrives or departs may cause frequent association changes and potential disruption of ongoing user sessions. To avoid this, we propose an online strategy that strikes a balance between the frequency of the association changes and the optimality in terms of load balancing. The online strategy is a combination of global optimization and location optimization. The local optimization deals with dynamic user arrivals and departures, while the global optimization is invoked periodically (or when the local optimization fails to maintain load balancing). The online strategy uses three configuration parameters, which are a *minimal load threshold* Ω, a *cell adaptation threshold* Δ, and a *time threshold* Θ. Ω and Δ determine the invocation condition of the local optimization to prevent the local optimization from unnecessarily invoked for negligible gains, where frequent invocations may cause service interruption to active users. Θ controls the frequency of the periodic invocation of the global optimization.

The local optimization conducts power adjustment in both directions, i.e., decrease and increase. Each AP $a \in \mathcal{A}$ maintains N_a which is a set of its neighboring APs. Let $\bar{y}_a$ be the average load of the APs in N_a. When the load of an AP a reduces, the algorithm checks whether the new load y_a satisfies the *cell enlargement condition*, which is if any AP $b \in N_a$ has $y_b > \Omega$ and also $y_a < (1 - \Delta) \cdot \bar{y}_a$. If this condition is met and the power index p_a of the AP a is not maximal, i.e., $p_a < K$, the algorithm increases the AP a's power level by one. On the flip side, when the load of the AP a increases, the algorithm checks the *cell reduction condition*, which is if $y_a > \Omega$ and also $y_a > (1 + \Delta) \cdot \bar{y}_a$. If the condition is met and the power index of the AP a is not minimal, the algorithm reduces the AP a's power level by one. Power adjustment is conducted until both conditions are satisfied or no further power adjustment is feasible.

8 Simulation Results

We compare the performance of our scheme with three existing methods, which are the Strongest-Signal-First (SSF) method, Least-Loaded-First (LLF) method, and the association control method proposed in [15]. The SSF method is the default user-AP association method in the IEEE 802.11 standard and is the same as our scheme with single power level. The LLF method is a typical load balancing heuristic, in which a user chooses the least-loaded one among the APs that he can reach. The last method determines the user-AP association to achieve the max-min fair bandwidth

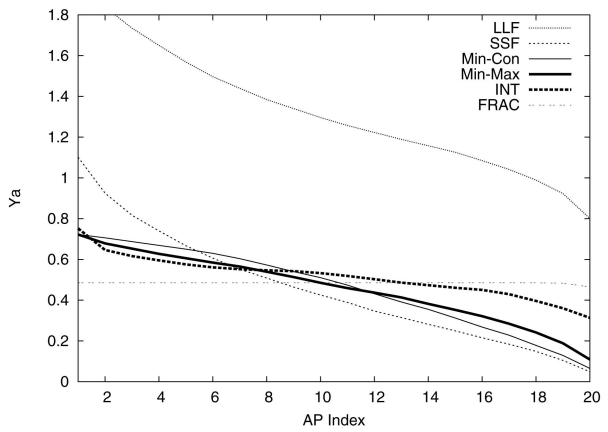


Fig. 10. Load comparison in networks with 100 random users.

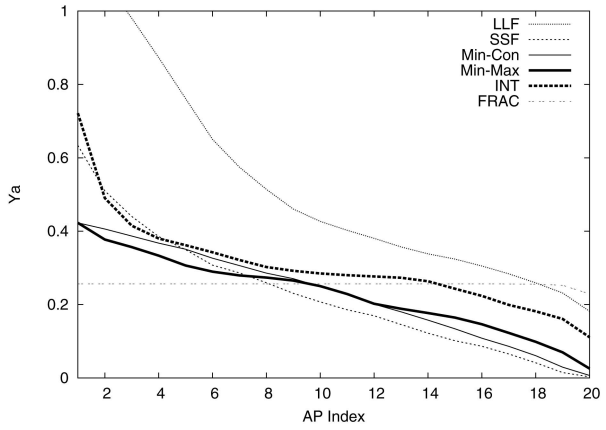


Fig. 11. Load comparison in networks with 50 random users.

allocation, which has two versions. The first version generates a fractional optimal solution (which we call FRAC) under the assumption that a user can simultaneously associate with multiple APs, which violates the single AP association constraint of WLANs. The second version is the integral solution (which we call INT). The FRAC and INT methods are chosen as a performance benchmark because their solution characteristics are known. The FRAC solution provides the strict performance upper bound ($i.e.$, lowest possible congestion load) and the INT method guarantees a two-approximation solution.[4] *One can not expect the proposed cell breathing scheme to outperform the optimal association control method that has significantly higher degrees of freedom than the former. Note that the cell breathing method does not rely on special association control for each mobile.* Surprisingly, however, the simulation results indicate that the cell breathing scheme outperforms not only SSF and LLF but also INT in various load conditions.

The simulation setting is as follows. A total of 20 APs are located on a 5×4 grid, where the distance between two adjacent APs is set to 100 meters and each AP is equipped with a 10 megabits per second backhaul link. Assuming that appropriate frequency planning was made to eliminate the interference among APs, we simulate the IEEE 802.11b wireless link as follows. To determine the bit rate between a user and an AP, SNR is computed and the bit rate is chosen accordingly. The 11 Mbps bit rate is used when SNR ≥ 9 dB, 5.5 Mbps when SNR ≥ 5 dB, 2 Mbps when SNR ≥ 3 dB and 1 Mbps when SNR $\geq$ 1dB. We set the maximal transmission power to 20 dBm and the minimal power to 10 dBm. Unless specified otherwise, 10 power levels are used. To simulate the indoor environment, we chose the path loss exponent [25] of 3.3, so that the path loss is computed by $PL(d) = 40 - 10 \cdot 3.3 \cdot \log d$, where d is the distance between a user and an AP. The background noise level is set to -93 dBm. Under this channel model, the range of a cell with the maximal power level is 150 meters and that with the minimal power level is 75 meters. Therefore, even with the minimal power level, the network has no coverage hole. In the initial simulations for baseline performance analysis, all users are assumed completely back-logged (i.e., the saturate traffic model) and quasi-static. We borrow the system performance metric from [15] and define the load of AP a under the saturate traffic model by

$$y_a = \sum_{u \in U_a} \frac{1}{r_{u,a}},$$

where $r_{u,a}$ is the bit rate with which the user u communicates with AP a and U_a is the set of users that are associated with AP a.

Fig. 10 shows the simulation results when 100 users are randomly distributed within the rectangular box that connects the boundary APs. The ratio of APs to *active users* is 5. The Y-axis represents y_a (i.e., the AP load) and the X-axis represents the AP index. The APs are sorted by their y_a values in decreasing order. Each y_a value is obtained by averaging 300 simulation runs. That is, the y_a for the AP index x is computed by averaging the y_a of the xth highest loaded AP of each run. Only the points corresponding to the integer x indices are meaningful (continuous lines are drawn only for the sake of visualization). The thick solid line represents our Min-Max method and the thin solid line represents our Min-Congestion method. While both methods always generate the same maximal y_a value, it is shown that the Min-Max method generates a load vector with a lower lexicographical order than the Min-Congestion method. In the Min-Max method, the priority of each AP was randomly chosen. The thick dotted line represents the INT solution and the horizontal thin dotted line represents FRAC. Both the proposed schemes and INT are clearly better than SSF, while LLF[5] performs poorly. Each of the proposed schemes (i.e., Min-Congestion and Min-Max) can be implemented in two ways, which are the complete knowledge algorithm and the limited knowledge algorithm. Both version of algorithms produce the same results (i.e., the same AP power settings). The main difference is that the limited knowledge algorithms incur longer convergence time and higher overhead. We will address this issue later.

We also simulated the case of 50 users and the result is shown in Fig. 11. Now, the gap between INT and FRAC is

4. Due to integrality gap, two-approximation is guaranteed only when the AP load is above a certain threshold (which is 1 in our setting). As shown in [15], INT converges to FRAC as the number of users increases.

5. In the LLF method, when there are multiple APs with the same load level, the AP with the strongest signal is chosen.

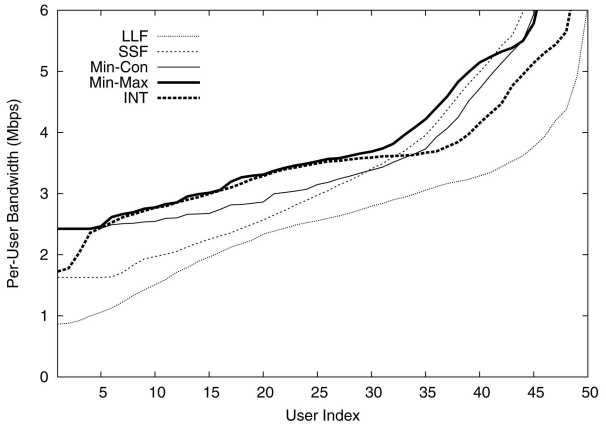


Fig. 12. Per-user bandwidth of 50 random users.

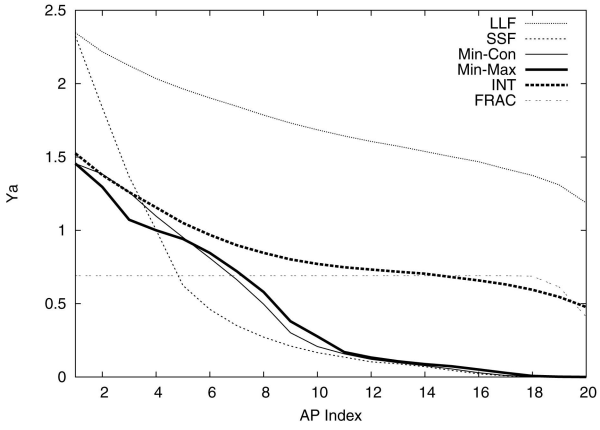


Fig. 13. Load comparison in hot spot networks with 100 users.

bigger than in the case of 100 users, and our Min-Max method outperforms INT with a significant margin. It is because the performance of INT depends on the number of users. Generally, less users make INT farther from FRAC due to the bigger rounding error. Interestingly, the relative performance of our Min-Max method against FRAC and SSF, i.e., the gap between Min-Max and FRAC and the gap between Min-Max and SSF, is not drastically affected by the number of users. The same trend was observed in other simulation settings, such as the case of 200 users (not shown due to the space limit). The steady relative performance against FRAC independent of the network load conditions is a key strength of the proposed schemes.

Fig. 12 depicts the per-user bandwidth allocation in the same simulation setting. We calculate per-user bandwidth by fairly dividing the total AP throughput, assuming the saturate traffic model. A noteworthy observation is that the proposed schemes outperform other schemes in terms of network-wide global throughput, which corresponds to the size of area under each curve in Fig. 12. It is possible because our scheme shifts users from heavily loaded cells to lightly loaded cells (i.e., load balancing) more effectively than other schemes. However, since global throughput and fairness have mutually conflicting relationship in general, one cannot expect that the proposed scheme always maximizes both global throughput and fairness simultaneously.

Now, let us consider the case of unbalanced user distributions. Only 20 percent of the users are randomly distributed and the rest are concentrated on two hot spots which do not overlap with each other. Each hot spot is a circle-shape area with 75 meter radius, and one hot spot contains twice more users than the other hot spot. This setting causes the heavy load condition in the hot spots. Fig. 13 show the results when the total number of users is 100. It shows that the Min-Max method performs even better in the presence of hot spots (i.e., heavy load condition) and clearly beats the INT method.

So far, we have assumed 10 power levels for beacon transmission. To examine the impact of the number of power levels, we tried four different cases. The simulation results in the 100 random user case are shown in Fig. 14. The impact of the number of power levels becomes marginal beyond a certain number of power levels, which is between 5 and 10 in the current simulation setup.

Now, let us examine the case of nonsaturate traffic users. To simulate dynamic traffic pattern, the amount of traffic is differently generated for each user. Under this dynamic traffic model, the load of AP a is defined by

$$y_a = \sum_{u \in U_a} \frac{w_u}{r_{u,a}},$$

where w_u represents the amount of traffic generated by user u during a unit time. While $w_u = 1$ represents a totally backlogged user, $w_u = 0.5$ means that a user u generates, on average, only half of traffic as compared to a backlogged user. Fig. 15 shows the simulation results for the case of 100 random users whose w_u were randomly chosen between 0 and 1. Similar trends to the case of the saturate traffic (Fig. 10) are observed.

While the complete knowledge algorithms compute the optimal solutions at one shot, the limited knowledge algorithms gradually change the power setting until they find the optimal solutions. To capture the overhead of the limited knowledge algorithms, we counted the frequency of power adjustments and consequent association changes during the simulation. The number of power adjustments

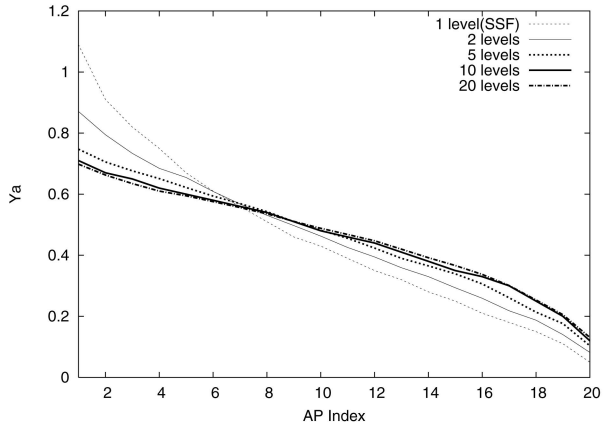


Fig. 14. Impact of the number of power levels on the performance of the Min-Max method.

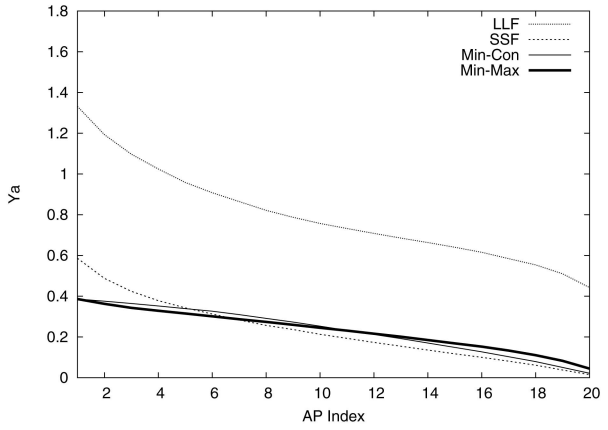


Fig. 15. Load comparison under dynamic traffic model.

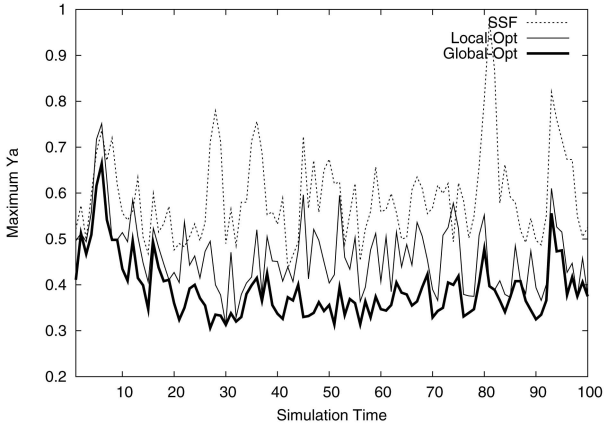


Fig. 16. Online strategy simulation results.

TABLE 2
Runtime Statistics of Limited Knowledge Algorithms
(*format:* Min-Max (Min-Congestion))

Case	Power adjustments	Association changes
100 random	102.9(33.3)	130.7(53.5)
200 random	84.9(39.5)	177.2(92.5)
100 hotspot	119.2(17.9)	94.6(34.3)
200 hotspot	101.6(17.5)	143.6(57.3)

determines the time to take before the system converges on the optimal state and the number of association changes decides the handoff overhead. The simulation results are summarized in Table 2. For each table entry, two numbers are given: the first number is for the limited knowledge Min-Max method and the second number is for the limited knowledge Min-Congestion method. Generally, the Min-Congestion method converges fairly quickly, while the Min-Max method takes a little longer. For instance, if the power adjustment interval is 1 second, the load balancing in a 100 random user network takes about 33 seconds by the Min-Congestion method, and less than 2 minutes by the Min-Max method. The increase of the user number seems not necessarily increasing the convergence time. The presence of hot spots does not necessarily mean longer convergence time either, and in the case of Min-Congestion, it even takes less time to converge.

Finally, we evaluate the online strategy. To simulate the dynamic user arrival/departure (or the user mobility), at each time slot, a certain portion of existing users is taken out and the same numbers of user are injected into randomly chosen locations. The simulation result for 20 percent user replacement at each time slot is shown in Fig. 16. One hundred random users with dynamic traffic model were simulated. The y-axis represents the maximal AP load (i.e., congestion load). The performance of the global optimization and the local optimization is compared with that of SSF. For the global optimization, the optimization algorithm is run every time slot (i.e., $\Theta = 1$). Interestingly, the local optimization clearly outperforms the SSF. In the simulation, both Ω and Δ were set to 0, so that power adjustment was conducted whenever possible. We analyzed the impact of Δ on the performance of the

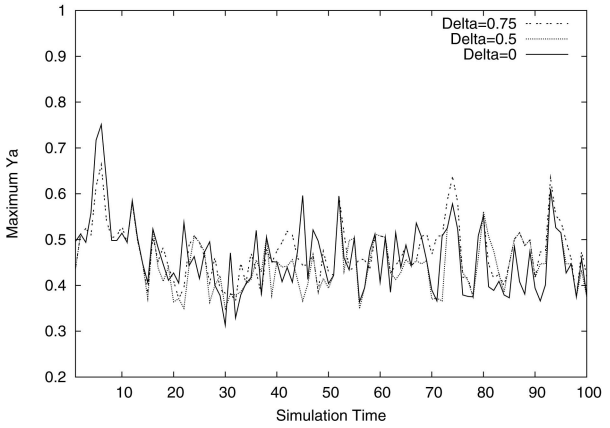


Fig. 17. Impact of Δ on the performance of local optimization.

TABLE 3
Runtime Statistics of Online Local Optimization

Δ	0.75	0.5	0.25	0.0
Number of power adjustments	159	436	975	1366
Number of association changes	139	390	818	841

local optimization by varying its value from 0 to 0.75. The results shown in Fig. 17 indicate that its impact is marginal. Note that the bigger Δ is, the less-likely the power adjustment is. The overhead statistics during the above simulations are summarized in Table 3. From the simulation results, we can draw the following observation: by setting Δ high, the overhead of power adjustment (and consequent user association changes) can be reduced without sacrificing the load balancing performance.

9 Conclusion

We presented a novel scheme for optimal load balancing in IEEE 802.11 WLANs. We provided rigorous analysis of the problem and presented two algorithms that find deterministic optimal solutions. The first algorithm minimizes the load of the congested AP(s) in the network, and the second algorithm produces an optimal min-max (priority) load balanced solution. These optimal solutions are obtained only with the minimal information which is readily available without any special assistance from the

users or modification of the standard. We assume only the control on the transmission power of the AP beacon messages. The simulations show that even a small number of power levels, e.g., between 5 and 10, is enough to achieve near optimal results.

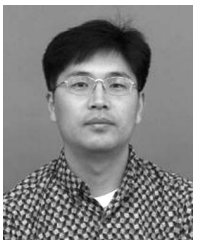
ACKNOWLEDGMENTS

Part of this paper was published in [1]. This research was partially supported by the Ministry of Knowledge Economy, Korea, under the ITRC program supervised by the IITA (IITA-2009-C1090-0902-0006).

REFERENCES

[1] Y. Bejerano and S.-J. Han, "Cell Breathing Techniques for Load Balancing in Wireless LANs," *Proc. IEEE INFOCOM,* 2006.
[2] M. Balazinska and P. Castro, "Characterizing Mobility and Network Usage in a Corporate Wireless Local-Area Network," *Proc. USENIX Int'l Conf. Mobile Systems, Applications, and Services (MobiSys '03),* 2003.
[3] T. Henderson, D. Kotz, and I. Abyzov, "The Changing Usage of a Mature Campus-Wide Wireless Network," *Proc. ACM MobiCom,* pp. 187-201, 2004.
[4] T. Togo, I. Yoshii, and R. Kohno, "Dynamic Cell-Size Control According to Geographical Mobile Distribution in a DS/CDMA Cellular System," *Proc. IEEE Personal, Indoor, and Mobile Radio Comm. Symp. (PIMRC '98),* pp. 677-681, 1998.
[5] A. Jalali, "On Cell Breathing in CDMA Networks," *Proc. IEEE Int'l Conf. Comm. (ICC '98),* pp. 985-988, 1998.
[6] I. Papanikos and M. Logothetis, "A Study on Dynamic Load Balance for IEEE 802.11b Wireless LAN," *Proc. Int'l Conf. Comm. Control (COMCON '01),* 2001.
[7] I. Tinnirello and G. Bianchi, "A Simulation Study of Load Balancing Algorithms in Cellular Packet Networks," *Proc. ACM/IEEE Int'l Workshop Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '01),* pp. 73-78, 2001.
[8] A. Balachandran, P. Bahl, and G.M. Voelker, "Hot-Spot Congestion Relief and Service Guarantees in Public-Area Wireless Networks," *SIGCOMM Computing Comm. Rev.,* vol. 32, no. 1, pp. 59-59, 2002.
[9] H. Velayos, V. Aleo, and G. Karlsson, "Load Balancing in Overlapping Wireless LAN Cells," *Proc. IEEE Int'l Conf. Comm. (ICC '98),* 1998.
[10] A. Kumar and V. Kumar, "Optimal Association of Stations and APs in an IEEE 802.11 WAN," *Proc. Nat'l Conf. Comm.,* 2005.
[11] K. Premkumar and A. Kumar, "Optimal Association of Mobile Wireless Devices with a WLAN-3G Access Network," *Proc. IEEE Int'l Conf. Comm. (ICC '06),* 2006.
[12] B. Kauffmann, F. Baccelli, A. Chaintreau, K. Papagiannaki, and C. Diot, "Self Organization of Interfering 802.11 Wireless Access Networks," INRIA Research Report RR-5649, 2005.
[13] S. Shakkottai, E. Altman, and A. Kumar, "The Case for Non-Cooperative Multihoming of Users to Access Points in IEEE 802.11 WLANs," *Proc. IEEE INFOCOM,* 2006.
[14] T.-C. Tsai and C.-F. Lien, "IEEE 802.11 Hot Spot Load Balance and QoS-Maintained Seamless Roaming," *Proc. Nat'l Computer Symp.,* 2003.
[15] Y. Bejerano, S.-J. Han, and L.E. Li, "Fairness and Load Balancing in Wireless LANs Using Association Control," *Proc. ACM MobiCom,* pp. 315-329, 2004.
[16] V.V. Veeravalli and A. Sendonaris, "The Coverage-Capacity Tradeoff in Cellular CDMA Systems," *IEEE Trans. Vehicular Technology,* pp. 1443-1451, Sept. 1999.
[17] S.-T. Yang and A. Ephremides, "Resolving the CDMA Cell Breathing Effect and Near-Far Unfair Access Problem by Bandwidth-Space Partitioning," *Proc. IEEE Vehicular Technology Conf. (VTC '01),* pp. 1037-1041, 2001.
[18] L. Du, J. Bigham, and L. Cuthbert, "A Bubble Oscillation Algorithm for Distributed Geographic Load Balancing in Mobile Networks," *Proc. IEEE INFOCOM,* 2004.
[19] A. Sang, X. Wang, M. Madihian, and R. Gitlin, "Coordinated Load Balancing, Handoff/Cell-Site Selection, and Scheduling in Multi-Cell Packet Data Systems," *Proc. ACM MobiCom,* pp. 302-314, 2004.
[20] P. Bahl, M.T. Hajiaghayi, K. Jain, V.S. Mirrokni, L. Qiu, and A. Saberi, "Cell Breathing in Wireless LANs: Algorithms and Evaluation," *IEEE Trans. Mobile Computing,* vol. 6, no. 2, pp. 164-178, Feb. 2007.
[21] J.M. Kleinberg, Y. Rabani, and E. Tardos, "Fairness in Routing and Load Balancing," *Proc. IEEE Ann. Symp. Foundations of Computer Science (FOCS '99),* pp. 568-578, 1999.
[22] D. Simone, *802.11k Makes WLANs Measure Up.* Network World, Mar. 2004.
[23] R. Raz and S. Safra, "A Subconstant Error-Probability Low-Degree Test, and Subconstant Error-Probability PCP Characterization of NP," *Proc. ACM Symp. Theory of Computers,* pp. 475-484, 1997.
[24] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.
[25] T.S. Rappaport, *Wireless Communications: Principle and Practice.* Prentice Hall, 1996.

Yigal Bejerano received the BSc degree (summa cum laude) in computer engineering in 1991, the MSc degree in computer science in 1995, and the PhD degree in electrical engineering in 2000 from the Technion—Israel Institute of Technology, Haifa, Israel. He is currently a member of the technical staff (MTS) at Bell Labs, Alcatel-Lucent. His research interests are mainly management aspects of high-speed and wireless networks, including the areas of mobility management, network monitoring, topology discovery, Quality-of-Service (QoS) routing, wireless LAN, and wireless mesh networks. He was on the technical program committee of numerous conferences. He is a member of the IEEE.



Seung-Jae Han received the BS and MS degrees in computer science and engineering from Seoul National University, Korea, in 1989 and 1991, respectively, and the PhD degree in computer science and engineering from the University of Michigan, Ann Arbor, in 1998. From 1999 to 2005, he was a member of the technical staff at Bell Labs, Lucent Technologies. He is currently an associate professor in the Department of Computer Science, Yonsei University, Seoul, Korea. His research interests are wireless networks, multimedia communication, and embedded systems. He is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.