

Cell Breathing Techniques for Load Balancing in Wireless LANs

Yigal Bejerano and Seung-Jae Han
Bell Laboratories, Lucent Technologies

Abstract: Maximizing the network throughput while providing fairness is one of the key challenges in wireless LANs (WLANs). This goal is typically achieved when the load of the access points (APs) is balanced. However, recent studies on operational WLANs have shown that AP load is often substantially uneven. To alleviate such imbalance of load, several load balancing schemes have been proposed. These schemes, essentially, require proprietary client software or specially-designed WLAN cards at the user computers for controlling the user-AP association.

In this paper we present a new technique that achieves load balancing by reducing the cell size of congested APs, which is conceptually similar to the so-called *cell breathing* methods in cellular networks. The proposed scheme *does not* require any modification at the user side neither the standard, but it only requires the ability of dynamically changing the transmission power of the AP beacon messages. Unlike existing cell-breathing methods, which utilize local optimization heuristics, we develop algorithms that guarantee to find the optimal beacon power settings, which minimize the load of the most congested APs, in polynomial time. We then consider the problem of network-wide min-max load balancing. We prove that this problem is *NP*-hard and cannot be easily approximated. In spite of this, we identify a variant of the problem, termed *min-max priority load balancing*, and present polynomial-time algorithms to find optimal solutions. Extensive simulations show that the performance of our cell-breathing methods is comparable with or superior to the existing association-based methods.

Keywords: Wireless Local Area Networks (WLAN), IEEE 802.11, Cell Breathing, Power Control, Load Balancing, Fairness, Combinatorial Optimization.

I. INTRODUCTION

Recent studies [1], [2] on operational IEEE 802.11 wireless LANs (WLANs) have shown that the traffic load is often unevenly distributed among the access points (APs). In WLANs, by default, a user scans all available channels to detect its nearby APs 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 the network capacity and providing fair services to the users. In this paper, we present a novel load-balancing scheme that reduces the load of congested APs by decreasing their cell size and forcing the users near the boundaries of congested cells to move to the neighboring less-congested cells. We achieve such cell dimensioning by controlling the transmission power of the AP beacon messages, which is conceptually similar to the so-called *cell breathing* method in cellular networks [3], [4]. In contrast to

previous studies on cell-breathing that mostly rely on local optimization heuristics, we present an optimal cell dimensioning 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 decreasing the load of other APs with equal or higher load. Our approach is particularly attractive in that it does not require neither user assistance or standard modification, unlike most existing proposals for WLAN load balancing.

A. Load Balancing via User-AP Association Control

Currently the IEEE 802.11 standard [5] does not provide any standard method to resolve the load imbalance. To overcome this deficiency, various load balancing schemes have been proposed by both the academia and the industry. Most of these methods commonly takes the approach of directly controlling the user-AP association by deploying proprietary client software or specially-designed WLAN cards at the user computers. For instance, some vendors have already incorporated certain load-balancing features in their device drivers, AP firmwares, and WLAN cards [6], [7]. In these proprietary solutions, the APs broadcast their load levels to users via modified beacon messages, and each user chooses the least-loaded AP.

Several studies [8]-[14] have proposed a variety of association metrics instead of using the RSSI as the sole association 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.*, [8], [9]. Balachandran *et al.* [10] proposed to associate a user with the AP that can provide a minimal bandwidth required by the user. If there exist many of such APs, the one with the strongest RSSI is selected. In [11], Velayos *et al.* introduced a distributed load balancing architecture where the load of an AP is defined as the aggregated downlink and uplink traffic through the AP. In [12], Kumar *et al.* proposed an association selection algorithm which is based on the concept of proportional fairness to balance between throughput and fairness. Most of these work heuristically determine only the association of newly arrived users. [13], [14] are exceptions. Tsai and Lien [13] proposed to reassociate users when the total load exceeds a certain threshold or the bandwidth allocated to users drops below a certain threshold. In [14], an on-line scheme that periodically optimizes the user-AP association is proposed. This work also proved a strong correlation between fairness and load balancing, *i.e.*, the fair service is obtained when the AP load is balanced.

Although the user-AP association control approach can achieve load balancing in WLANs, the requirement of deploying proprietary client software/hardware on all (or most) users raises an acute question about its practicality. Today, WLAN users frequently move between different WLANs, such as hotels, air-

ports, shopping centers and university campuses.¹ Different networks are managed by different organizations and likely adopt different load balancing mechanisms. It is unrealistic to require the users to have the appropriate client modules for each visiting network. This motivates the need for a new load-balancing scheme that does not require any proprietary client module nor any modification of the standard.

B. Cell Breathing for Load Balancing

In CDMA cellular networks, the coverage and capacity of a cell are inversely related with each other [15]. 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 band in CDMA networks. As a result, the overall network capacity may decrease [3]. Furthermore, since the maximal transmission power of the users is bounded, the users who are far from the base station may experience poor services. This so-called *near-far problem* may result in imbalanced cell handoff boundaries for reverse and forward links, as the latter is determined by the strength of the pilot signal of the base stations, independent of the interference [4]. In other words, the cell handoff boundary of the reverse link is tighter than that of the forward link. To overcome these problems the *cell breathing* approach was proposed by Togo *et al.*[3] and Jalali [4], independently. This approach shrinks the cell size of congested cells and balances the forward and reverse link handoff boundaries by reducing the pilot signal transmission power of the corresponding base stations.

Some studies have explored the benefit of combining the cell-breathing methods with other interference mitigation methods. For instance, in [16], Yang and Ephremides presented a solution for the near-far problem, which is based on the combination of cell-breathing and bandwidth space partitioning. Du *et al.*[17] proposed a distributed load balancing technique that utilizes a bobble oscillation algorithm. In [18], Sang *et al.* proposed a method that coordinates the packet level scheduling with cell-breathing techniques. Generally speaking, the existing cell-breathing techniques utilize probabilistic local optimization methods. Therefore, they do not provide any guarantee on the quality of the solutions. Since the cells of WLANs are much smaller than those of cellular networks, the probabilistic local optimization methods may not work well in WLANs, as we will demonstrate later in the paper with a simple example. Moreover, these techniques cannot be easily applied to IEEE 802.11 WLANs, *e.g.*, they require the change of scheduling algorithms or the knowledge on the user location. This motivates the design for a new cell-breathing method for WLANs which finds deterministic global optimal solutions.

C. Min-Max Load Balancing Algorithms

In principle, our objective can be viewed as a min-max variant of the *unrelated parallel machine scheduling* problem [19]. The latter seeks for a job-machine association that minimizes the maximal processing time of any machine, for a given set of

jobs, machines, and the required running time of each job on each machine². Since there are extensive literature on parallel machine scheduling problems and max-min solutions, we discuss here only the most relevant ones to our study. Most of the work on min-max (or max-min) solutions address the problem of finding a fair bandwidth allocation to a set of pre-determined routes in a wired network [20], [21].

Selecting routes for the max-min fair bandwidth allocation is a much harder problem and has been studied in [22], [23]. Megiddo [22] addressed the single-source fractional flow problem and presented a polynomial time algorithm that finds an optimal max-min fair solution. Extending this work, Kleinberg *et al.* [23] considered the case that a connection is routed along a single path. In particular, their approach can be applied to *load conserving* instances of the unrelated parallel machine scheduling problem, where each job imposes the same load on the subset of machines on which it can be run. They argued that a coordinate-wise constant-factor approximation cannot be found for this problem and presented a prefix-sum 2-approximation algorithm, in which for every integer $k > 0$ the sum of the first k coordinates of the calculated machine load vector sorted in increasing order is at most twice the sum of the first k coordinates of the optimal min-max fractional assignment.

Another important study is the user-AP association control scheme presented in [14]. This study can be mapped to the general unrelated parallel machine scheduling, where each job may have different running time on each machine. It presents a min-max load balancing algorithm that ensures a coordinate-wise 2-approximation ratio as compared to the optimal min-max fractional solution. Notice that all of these methods require a complete control on the job-machine association. This assumption is feasible for the user-AP association control schemes. However, such freedom of association control is unavailable in the cell-breathing approach, which only implicitly controls the user-AP association by adjusting the cells' boundaries. This raises the need for new min-max load balancing algorithms for the cell-breathing approach.

D. Our Contributions

In this paper we present a new load balancing scheme for IEEE 802.11 WLANs. The proposed scheme adjusts the size of cells by changing the transmission power of the AP beacon messages without changing the transmission power of the data traffic channel. While there exist similar approaches in cellular networks, to the best of our knowledge, we are the first who applies the cell breathing concept to IEEE 802.11 WLANs. More importantly, unlike the existing cell breathing studies [3],[4],[16],[17],[18], we tackle the challenge of finding the deterministic global optimum instead of relying on local optimization heuristics. *Our 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 contributions may be as simple as the number of users associated with an AP or can be more sophisticated to take account of factors like transmission bit rates and traffic demands. Our scheme *does not* require any special assistance from users nor any change in the standard. It only requires the ability of dynamically changing the transmission power of

¹In many places even free 802.11 access is offered. Recently, some cities *e.g.*, Philadelphia and San-Francisco, declared their intention to build a free city-wide 802.11 networks.

²When a given job cannot be served by a specific machine, infinite running time is assumed on that machine.

the AP beacon messages. Today, commercial AP products already support multiple transmission power levels, so we believe this requirement can be relatively easily achieved via AP software update.

Our algorithms will be run on a network operation center which collects the load and association information from the APs via such methods as SNMP. Depending on the extent of the available information, we consider two knowledge models. The first model assumes *complete knowledge*, in which the user-AP association and the corresponding AP load are known a priori for all possible beacon power assignments. Since such information is not readily available in current WLANs, we also consider the second model, the *limited knowledge* model, in which only information on the user-AP association and AP load for the current beacon power assignment is available. The algorithms for the complete knowledge model serve as building blocks for the algorithms for the more practical limited knowledge model.

We present our algorithms in two steps. At first, we address the problem of minimizing the load of the most congested APs, whose load is called the *congestion load*. We present two polynomial time algorithms that find optimal solutions, one for the complete knowledge model, another for the limited knowledge model. These results are intriguing, because similar load balancing problems, *e.g.*, [14], are known to be strong *NP*-hard. It is particularly interesting that a polynomial-time optimal algorithm exists for the limited knowledge model. Our algorithms are rooted from a simple observation that as long as the current power setting 'dominates' the optimal setting (*i.e.*, each AP has the same or higher power level than its power level in the optimal solution), an optimal solution can be obtained by a certain sequence of power reduction operations. The algorithms start with the maximal power level at all APs and, iteratively, reduce the power of a selected set of APs. For the complete knowledge case, we use the concept of *bottleneck set*. After reducing the power level of all APs in the bottleneck set, the load of each AP is guaranteed to stay the same or strictly lower than the initial congested load before the power reduction. This property ensures monotonic convergence to the optimal solution. For the limited knowledge case, we take a different approach, termed *optimal state recording*, in which the power levels of the congested APs are gradually reduced until the power cannot be reduced any further, while the best solution found so far is recorded.

Secondly, we address the problem of finding the min-max load balanced solutions. We prove that this is a strong *NP*-hard problem and there exists no good approximation algorithm. More specifically, we prove that there exists no algorithm that guarantees any coordinate-wise 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 the network. In spite of this, we identified 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 aggregated load contributions of its associated users and a unique AP priority. By deploying the optimal state recording method, we were able to construct an algorithm that, iteratively, compute each coordinate of the min-max priority load balanced solutions. We, later, show that our algorithms can be efficiently embedded into adaptive on-line schemes.

Through extensive simulations, we show that the performance

of our cell-breathing methods is overall 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. Our min-max priority load balancing algorithms yield near optimal results even for the non-priority min-max problem by randomly choosing AP priorities. Although we primarily focus on IEEE 802.11 WLANs, our schemes should be applicable to other wireless networks. Due to the space limitation, we omit some proofs.

II. THE NETWORK MODEL

We consider an IEEE 802.11 WLAN that comprises a set of *access points* (APs), denoted by \mathcal{A} . $|\mathcal{A}|$ denotes the number of APs. All APs are attached to a fixed infrastructure, which connects them to wired networks, *e.g.*, the Internet. Each AP has a certain transmission range and it can serve only those users that reside in that range. Each AP is configured to 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. Each 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 [6], [7]. We denote the transmission power of each AP $a \in \mathcal{A}$ by P_a and its corresponding power index by p_a . For the sake of simplicity, we assume that the AP deployment ensures a high degree of overlaps between the range of adjacent APs. Consequently, *every user is covered by at least one AP even when all APs are transmitting at the minimal power level P_{min}* . We define the *network coverage area* to be the union of the transmission ranges of all APs in \mathcal{A} .

We use \mathcal{U} to denote the set of all users in the network coverage area and use $|\mathcal{U}|$ to denote their number. We assume that users have a quasi-static mobility pattern. In other words, users are free to move from place to place, 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 [1], [2]. At any given time, each user is associated with a single AP. Each AP periodically transmits beacon messages for advertising its presence. When a user enters a WLAN, the user initiates a *scanning* operation, in which 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 the AP that has the strongest RSSI. Whenever the channel quality deteriorates below a certain threshold, *e.g.*, due to the user movement, the user initiates a new scanning operation and it may associate itself with a different AP.

The RSSI that a user $u \in \mathcal{U}$ senses for an AP $a \in \mathcal{A}$ is denoted by $R_{u,a}$. It depends on the *transmission power* of AP a , P_a , and the *signal attenuation*, which is denoted by $g_{u,a}$, *i.e.*, $R_{u,a} = g_{u,a} \cdot P_a$. We consider only the signal attenuation that results from long-term channel condition changes, such as path-loss and slow fading. We assume that during the short period of time for executing our algorithms the signal attenuation between each user-AP pair does not change. As the user-AP association depends on the RSSI's, we can divide the network coverage area into $|\mathcal{A}|$ disjoint cells. A *cell of an AP a* defines

the region in which AP a has the strongest RSSI. Note that the cell of an AP a is subsumed in its transmission range and it depends not only on the transmission power of a , but also that of the other APs in a 's vicinity.

The transmission bit-rate for a user-AP pair is determined by the Signal-to-Noise Ratio (SNR), which is the strength of the received signal over the accumulated strength of the other interfering transmissions and the background noises. Users who are associated with the same AP may transmit with different bit rates. Each user contributes a certain amount of load on its serving AP, and the load on an AP is the aggregation of the *load contributions* of its associated users. Our algorithms do not require any explicit load definition and just 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 of a user u on an AP a . We assume that this contribution is constant³, 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 *non-congested* APs.

Our flexible load model can accommodate many commonly-used load definitions, including the number of users associated with an AP as well as more advanced load definitions that may take account of the effective transmission bit-rate or the average traffic demand. It can also deal with the multiplicative user load contributions, *i.e.*, $y_a = \prod_{u \in \mathcal{U}_a} l_{a,u}$, by applying log to both sides. Table I summarizes the key 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.
$g_{u,a}$	Attenuation of AP a 's signal detected by user u .
K	The maximal transmission power index.
$l_{a,u}$	Load contribution of user u to AP a .
P_a	Transmission power of 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, $\mathcal{S} = \{(a, p_a)\}$.
$\bar{\mathcal{S}}$	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 congested load.
\bar{Y}	The congestion load of the recorded state.
\vec{Y}	The AP load vector, $\vec{Y} = \{y_1, \dots, y_{ \mathcal{A} }\}$.

TABLE I
NOTATIONS.

III. THE CELL BREATHING APPROACH

In this section we present the basic concept that underlies our approach. We also address some practical aspects and the algorithmic challenges that our approach encounters. In this study, we assume the presence of a *Network Operation Center* (NOC). APs report to the NOC about their associated users, their load and additional relevant information. The NOC executes our algorithm and configures the APs accordingly.

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

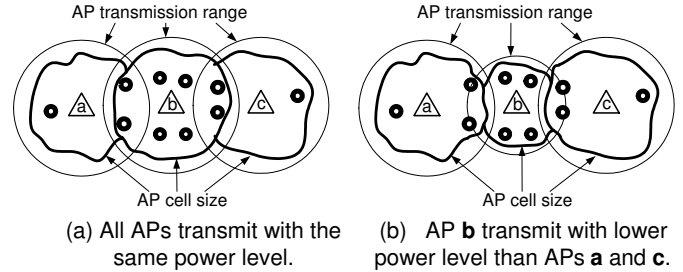


Fig. 1. Balancing the AP load by adjusting their transmission power.

A. The Concept of Cell Breathing

Our scheme reduces the load of congested APs by reducing the size of the corresponding cells. This forces users near the congested cells' boundaries to shift their association to adjacent (less-congested) APs. Such cell dimensioning can be obtained, for instance, by reducing the transmission power of the congested APs, as we illustrate in Example 1.

Example 1: Consider a WLAN with three APs, a , b and c that transmit with maximal power P_{max} and let's assume that they are associated with 1, 8 and 1 users, respectively, as depicted in Figure 1-(a). 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 Figure 1-(b), and the AP load becomes more evenly distributed. \square

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 only the number of users but also the effective user throughput. Thus, we may end up with increasing the load of more APs rather than reducing the load of the congested APs.

We overcome this problem by the segregation 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 power⁴ 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 new user association. This concept of controlling the cells' dimensions by adapting power levels of the beacon messages is termed *cell breathing*. *The segregation between the power levels of the data traffic and the beacon messages is the only modification that*

⁴Power 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.

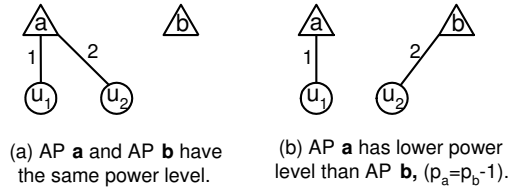


Fig. 2. Example of an execution of the greedy algorithm.

we require from APs. We believe this can be relatively easily achieved by software update.

B. Triggering User-AP Association Changes

In the long run, the method described above balances the AP load by discouraging new user association with congested APs. However, since the cell breathing method does not explicitly control the user-AP association, it may not provide immediate relief to the congested APs. It is because, once the association decision is made, a user stays connected with the same AP as long as it experiences a satisfactory channel quality, regardless of the received beacon message strength. For immediate load reduction, we need to encourage the users in the congested cells to invoke the scanning operations. One method is reducing the data traffic power levels of the congested APs for a short period, which will trigger the scanning mechanism for the users near the boundaries of the congested cells. Another method is sending "dis-association" messages to some or all users who are associated with the congested APs. The latter method provides the flexibility to select particular users to change their association without affecting others. In the remainder of this paper, we assume that a proper method is used to trigger association shifts after the cell range of an AP is altered and *limit our discussion just to the power level of the beacon messages*. That is, when we say transmission power, we mean only the transmission power of beacon messages.

C. 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. Note that since this algorithm attempts to shift users from congested APs to their neighbors, 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.

Example 2: Consider a WLAN with two APs, denoted as a and b , and two users u_1 and u_2 . User u_1 can only be attached to a and it yields a load of 1. User u_2 can be attached to both APs and chooses an AP with a higher power level, while it chooses a in case of a tie. 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 Figure 2-(a). 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 load on the two APs are 1 and 2, respectively, as depicted in Figure 2-(b). 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. \square

Example 2, demonstrates the need for more sophisticated algorithms.

IV. MINIMIZING THE CONGESTION LOAD

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.

A. The 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. When we assume that the users are always associated with the AP whose beacon signal has the strongest RSSI, it also determines the user-AP association. Therefore, it also determines the set of *congested APs*, D , and their *congestion load*, Y . Notice that a user may change its association only when the network state changes, which is termed a *state transition*. Let us now define our objectives.

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

We address this problem in two types of networks depending on the information available.

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

A complete knowledge model is feasible when all users collect the RSSI information from all of the nearby APs and send the information to the NOC. Such a feature is suggested, for instance, in the IEEE 802.11-k proposal [24]. Unfortunately, this feature is currently not available in most existing WLANs. We use this model mainly as a building block of the *limited knowledge* solution.

Definition 4 (The Limited Knowledge (LK) 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 associated AP a .

In the complete knowledge model, the NOC can a priori determine the user-AP association in all possible states without actually changing the network state. This allows the NOC to perform an off-line calculation of a desired state and to directly configure the APs with the corresponding power levels of that state. Such calculation is not possible in the limited knowledge model. Nevertheless, we show in the following that for both models the optimal network states can be found.

B. Preliminary Observations

We present some fundamental observations that relevant to our algorithms. In particular, we study the relationship between the AP power reduction and the network state transition.

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' neither the set $\mathcal{A} - A'$. From Lemma 1 follows Corollary 1.

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 for every AP $a \in \mathcal{A}$ satisfies that $p_a^{init} \geq p_a^{new}$. 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 an *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} and \mathcal{S}^{opt} by y_a^{init} and y_a^{opt} , correspondingly.

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 as follow:

Lemma 2: Consider an initial network state \mathcal{S}^{init} and let \mathcal{S}^{opt} be one of its dominated optimal state. 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} . From Lemma 2 follows Corollary 2.

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 non-anchor APs. The set of non-anchor 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 of the congested APs may increase the load on 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 congested 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 and we prove that this method also finds the optimal solution.

C. The 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 due to 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}^{init} . The set B_i comprises of 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 .

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

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

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

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 enables 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 received signal $R'_{u,a}$ 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 Figure 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. From Definition 6 results Lemma 3.

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. The load of every AP $a \in \mathcal{A}$ stays the same or strictly less than Y after the set \mathcal{B} reduction operation. Moreover, any other set A' that have the above property must include \mathcal{B} .

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 .

Lemma 4: $\{Y_j\}$ is a monotonic non-increasing sequence. Furthermore, Lemma 3 enables us to present in Theorem 1 a strong property of bottleneck sets that is essential for our optimality proofs.

Theorem 1: Consider any sub-optimal 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} .

```

Routine Calc_Bottleneck_Set( $\mathcal{S}^0 = \{(a, p_a^0)\}, B_0, Y$ )
 $B_{-1} = \emptyset$  //used for the termination condition.
 $i = 0$ 
while ( $(B_i \neq B_{i-1})$  and  $(\forall a \in B_i, p_a^i > 0)$ ) do
   $i = i + 1$ 
  //Get the simulated state  $\mathcal{S}^i$  by performing set  $B_{i-1}$ 
  //power reduction operation from state  $\mathcal{S}^{i-1}$ .
  for every AP  $a \in B_{i-1}$  let  $p_a^i = p_a^{i-1} - 1$ 
  for every AP  $a \in \mathcal{A} - B_{i-1}$  let  $p_a^i = p_a^0$ 
  // Evaluate the new user association and
  // compute the load on each AP in state  $\mathcal{S}^i$ .
   $y_a^i =$  The load of AP  $a$  in state  $\mathcal{S}^i$ .
   $B_i = B_{i-1} \cup \{a | a \in \mathcal{A} \wedge y_a^i \geq Y \wedge y_a^i > y_a^{i-1}\}$ 
end while
 $\mathcal{B} = B_i$ 
return  $\mathcal{B}$ 
end

```

Fig. 3. A formal description of the bottleneck set calculation routine.

Proof: Since \mathcal{S}^{init} is sub-optimal its congestion load is strictly higher than the congestion load of \mathcal{S}^{opt} . From Lemma 3 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} . \square

D. The Complete Knowledge Algorithm

We now present the complete knowledge algorithm. 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 IV-C. 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 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 Figure 4 and a typical execution of the algorithm is illustrated in Example 4

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 Figure 5-(a), 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 indi-

```

Alg CK_Min_Congestion_Load_Algo( $\mathcal{A}, \mathcal{U}$ )
for every AP  $a \in \mathcal{A}$  let  $p_a = K$ 
 $End\_Flag = FALSE$ 
while ( $End\_Flag = FALSE$ ) do
  // Find congested APs and their load.
   $Y = \max_{a \in \mathcal{A}} y_a$ 
   $D = \{a | a \in \mathcal{A} \wedge y_a = Y\}$ 
   $\mathcal{B} = Calc\_Bottleneck\_Set(\{(a, p_a)\}, D, Y)$ 
  if ( $(\mathcal{B} = \mathcal{A}) \vee (\text{exist } a \in \mathcal{B} \text{ s.t. } p_a = 0)$ ) then
     $End\_Flag = TRUE$ 
  else
    for every AP  $a \in \mathcal{B}$  let  $p_a = p_a - 1$ 
  end if
end while
return  $\{(a, p_a) | a \in \mathcal{A}\}$ 
end

```

Fig. 4. A formal description of the complete knowledge congestion load minimization algorithm

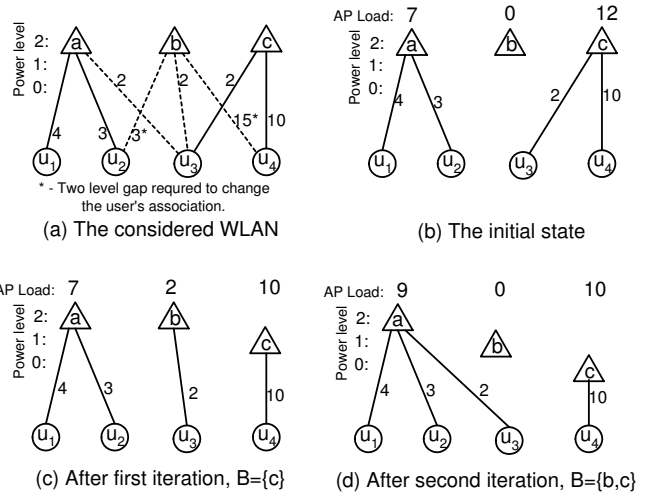


Fig. 5. Example of an execution of the complete knowledge algorithm.

cates 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 Figure 5-(b), 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 Figure 5-(c). Note that b is still the congested AP. However, more 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 Figure 5-(d). This is the last iteration, since now $p_c = 0$. Clearly, the final state, presented in Figure 5-(d), minimizes the congested load, but it does not balance the load of the non-congested APs. \square

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

Proof: The algorithm starts with the maximal power state that dominates the optimal solution. From Theorem 1 follows that as long as the algorithm hasn't reached the optimal state the resulting state, obtained by a bottleneck set power reduction operation, also dominates the optimal solution. From Lemma 4

results that during the execution of the algorithm the congestion load of the WLAN never increases. So now we just have to show that the algorithm stop 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 non-increasing, 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 results that such a set A' does not exist. \square

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

E. The Limited Knowledge Algorithm

We now present our *limited-knowledge algorithm* that finds an optimal state in the limited knowledge case. Unlike the complete knowledge case, we cannot calculate the bottleneck set in advance. We overcome this obstacle by using Corollary 2. According to it, as long as a network state is sub-optimal and it dominates an optimal solution, a sequence of set power reduction operations of congested APs converges to the optimal state. This property 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 sub-optimal 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 congested 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 Figure 6 and a typical execution is illustrated in Example 5

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 Figure 7-(a). Since, c is the congested AP, the algorithm reduces its power index twice in two successive iterations, as shown in Figures 7-(b) and 7-(c). 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

```

Alg LK_Min_Congestion_Load_Alg( $\mathcal{A}, \mathcal{U}$ )
  for every AP  $a \in \mathcal{A}$  let  $p_a = K$ 
  // Set power indices of all APs to  $K$  and evaluate AP load.
   $\tilde{\mathcal{S}} = \{(a, p_a)\}$ 
   $\tilde{Y} = \max_{a \in \mathcal{A}} y_a$ 
   $End\_Flag = FALSE$ 
  while ( $End\_Flag = FALSE$ ) do
     $Y = \max_{a \in \mathcal{A}} y_a$ 
     $D = \{a | a \in \mathcal{A} \wedge y_a = Y\}$ 
    if (exist  $a \in D$  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  $a \in D$  let  $p_a = p_a - 1$ 
       $Y = \max_{a \in \mathcal{A}} y_a$ 
      if ( $Y < \tilde{Y}$ ) then
         $\tilde{\mathcal{S}} = \{(a, p_a)\}$ 
         $\tilde{Y} = Y$ 
      end if
    end if
  end while
  return  $\tilde{\mathcal{S}}$ .
end

```

Fig. 6. A formal description of the limited-knowledge congestion load minimization algorithm

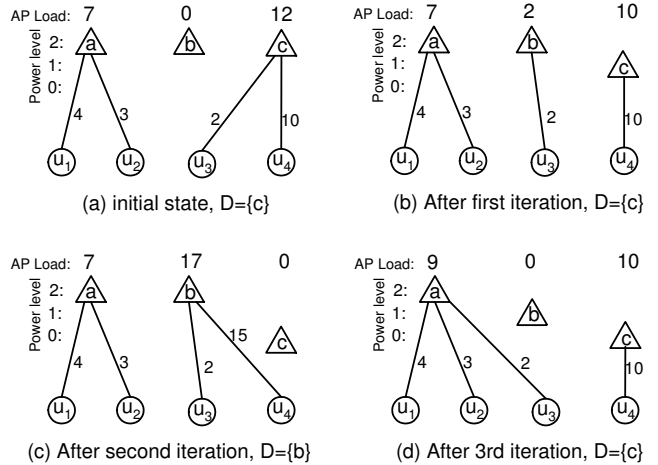


Fig. 7. Example of an execution of the limited knowledge algorithm.

congested 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, shown in Figure 7-(b). \square

Theorem 3: The limited-knowledge algorithm always finds an optimal state that minimizes the network 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 congested 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 haven't reached an optimal state, the sequence of states obtained by iteratively reducing the power of the congested APs also dominates the optimal state. Conse-

quently, the algorithm stops with a sub-optimal 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. \square

V. FINDING A MIN-MAX LOAD BALANCED STATE

The algorithms presented in Section IV minimize the network congestion load but they do not necessarily balance the load of the non-congested APs, as demonstrated in Examples 4 and 5. In this section, we consider the min-max load balancing solutions that not only minimize the network congestion load but also balance the load of the non-congested APs. This objective is formally defined in Section V-A. Unfortunately, this problem is NP -hard and it is hard to find even an approximated solution. In spite of this, we introduce 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 Section V-B. Our solution is given for the limited-knowledge model, obviously, it can be used for complete-knowledge model as well.

A. The 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 [14], [23]. 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*, $\vec{Y} = \{y_1, \dots, y_{|\mathcal{A}|}\}$, 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 $\vec{Y} = \{y_1, \dots, y_{|\mathcal{A}|}\}$ has the same or lower lexicographical value than any other load vector $\vec{Y}' = \{y'_1, \dots, y'_{|\mathcal{A}|}\}$ 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 now show that the problem of finding a min-max load balanced state is NP -hard. Furthermore, we 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.

Proof: For the sake of simplicity, we assume that the load generated by a user on its associated AP may be zero. The problem stays NP -hard also when the load of a user is strictly positive. We prove this theorem by reducing any instance of the *minimal dominating set* (MDS) [26] problem to the considered state selection problem. Recall that the MDS problem of a given graph $G(V, E)$ is a known NP -hard problem that seeks for the smallest subset $Q \subset V$ such that every node in $V - Q$ has at least one neighbor in Q . Consider a graph $G(V, E)$ and let $N_v \subseteq V$

be the set of nodes that contains node v and all its neighbors in the graph G . We construct a WLAN with $|V|$ users, denoted by \mathcal{U} , and $|V|$ APs, denoted by \mathcal{A} , that can transmit in one of two power level, *i.e.*, $K = 1$. For each node $v \in V$, we define an AP $a_v \in \mathcal{A}$ and a user $u_v \in \mathcal{U}$. User u_v can be associated with any AP $a_{v'}$ such that $v' \in N_v$. If the power index of AP a_v is 1 or all APs $a_{v'}$, $v' \in N_v$, have power index 0, then user u_v is associated with AP a_v and produces load of 1. Otherwise, user u_v is associated with one of the other AP $a_{v'}$, $v' \in N_v - \{v\}$ with power level 1 and it does not yield any load on this AP. Note that in our construction the load of an AP may be either 1 or zero.

We claim that the graph $G(V, E)$ has a dominating set of size m if and only if there is a network state such that m APs have load 1 and all other APs have load 0. If the graph G has a dominating set $Q \subseteq V$ of size m , then we assign power index 1 to every AP a_v , $v \in Q$ and 0 to the other APs. Since Q is a dominating set, result that all users will be associated with APs a_v , $v \in Q$. Thus, the number of APs with load 1 is m . Now suppose that there is a power level selection that yields m APs with load 1. Clearly, each user u_v is associated with one AP $a_{v'}$, $v' \in N_v$. Thus we just have to show that the load of this AP must be 1. If u_v is associated with AP a_v then, by definition, the load of a_v is 1. Otherwise, u_v is associated with an AP $a_{v'}$, $v' \in N_v - \{v\}$. Thus, the power index of this AP must be 1 and consequently also user $u_{v'}$ is associated with AP $a_{v'}$. Therefore, the load of AP $a_{v'}$ must be 1. From the above follows that for each set N_v at least one of the corresponding APs has load 1. In other words, the APs with load 1 define a dominating set of size m for the graph G and this completes our proof. \square

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

The above proofs are based on the reduction from the minimal dominating set (MDS) problem. 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 than $c \log(|V|)$, for some $c > 0$, unless $P = NP$ [25]. 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.

Theorem 5: There exists no polynomial algorithm that can ensure γ -coordinate-wise approximation solutions to our min-max load balancing problem for any γ , unless $P = NP$.

Theorem 6: There exists no polynomial algorithm that can ensure γ -prefix-sum approximation solutions to our min-max load balancing problem for $\gamma < c \cdot \log |\mathcal{A}|$, for some $c > 0$, unless $P = NP$.

In spite of the NP -hardness, we now turn to present a variant of the min-max load balancing problem that its optimal solution can be calculated in polynomial time. In this problem, we assume that each AP $a \in \mathcal{A}$ has a *unique priority*, also termed a *weight*, $w_a \in [1..|\mathcal{A}|]$, that indicates the AP's importance. In this study we do not address the problem of allocating priority to the APs. In the following we give a new AP load definition, termed a *priority load*, for this problem.

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, results that there are no two AP 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.

B. The 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, similar to the algorithm presented in Section IV-E, to calculate a network state that minimizes the priority-load of the m -th 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 m -th 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 define the *congesting load*, Y , as the maximal load on any non-fixed AP. From Property 1 follows that at any given time there is only a single non-fixed AP, termed the *congested AP*, d , that carries the congesting load.

At each iteration m the algorithm invokes the *LK_Minimize m_Coordinate* routine for minimize the m -th coordinate of the load vector and let's assume that Requirement (1) is satisfied at the beginning of the iteration. The routine uses three recording variables. The *recorded congestion load*, \tilde{Y} , keeps the congested load value of the optimal state found so far. The *recorded state* variable, \tilde{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{S} , \tilde{Y} and \tilde{d} with the iteration initial state, its congested 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 congested 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 according to the recorded network state and returns the recorded state \tilde{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 $(m + 1) - th$

```

Alg Min-Max-Priority-Load-Balancing-Alg( $\mathcal{A}, \mathcal{U}$ )
 $\tilde{S} = \{(a, p_a = K) | \forall a \in \mathcal{A}\}$ 
 $\mathcal{F} = \emptyset$ 
while ( $\mathcal{F} \neq \mathcal{A}$ ) do
  ( $S, d$ ) = LK_Minimize_m_Coordinate( $S, \mathcal{F}$ )
   $\mathcal{F} = \mathcal{F} \cup \{d\}$ 
end while
return  $\tilde{S}$ 
end

Routine LK_Minimize_m_Coordinate( $S^{init}, \mathcal{F}$ )
 $\tilde{S} = S^{init}$ 
 $\tilde{Y} = Y = \max_{a \in \mathcal{A} - \mathcal{F}} y_a$ 
 $\tilde{d} = d = a$  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
     $p_d = p_d - 1$  // Power reduction & load evaluation.
     $Y = \max_{a \in \mathcal{A} - \mathcal{F}} y_a$ 
     $d = a$  s.t.  $a \in \mathcal{A} - \mathcal{F} \wedge y_a = Y$ 
    // Check if fixed AP load was increased.
    if ( $\exists a \in \mathcal{F}$  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
       $\tilde{S} = \{(a, p_a)\}$ 
       $\tilde{Y} = Y$ 
       $\tilde{d} = d$ 
    end if
  end if
end while
Return ( $\tilde{S}, \tilde{d}$ ).
end

```

Fig. 8. A formal description of the Min-Max priority-load balancing algorithm.

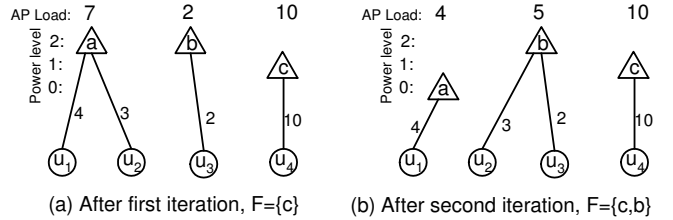


Fig. 9. Example of an execution of the min-max priority load balancing algorithm.

coordinate. A typical execution is illustrated in Example 6 and a formal description of the algorithm is given in Figure 8.

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 Figure 9-(a). 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 Figure 9-(b), which is the only min-max load balanced state of this network. \square

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{S} minimizes the m -th 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{S} maintains the load of the fixed APs. Since \tilde{Y} is the smallest detected congestion load, \tilde{S} also minimizes the m -coordinate. Thus we only have to show that \tilde{S} preserves Requirement (1) at the beginning of the $(m + 1)$ -th iteration. From our assumption, results that the m iteration starts with a state that dominates the optimal solution. \tilde{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). \square

Theorem 7: The algorithm always finds a min-max priority-load balanced solution.

Proof: This is a direct result from inductive application of Lemma 5 for all of the m coordinates of the optimal load vector. \square

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

VI. THE ON-LINE STRATEGY

Execution of the optimization algorithms described in the previous sections each time a user arrives or departs may cause frequent association changes and potential disruption of on-going user sessions. To avoid this, we propose an on-line strategy that strikes a balance between the frequency of the association changes and the optimality of the network state in terms of load balancing. Our on-line strategy is a combination of the global optimization, which are described in the previous sections, and location optimization. The local optimization deals with dynamic user arrivals and departures, while the global optimization is invoked periodically or whenever the local optimization fails to maintain a load-balanced state. The on-line 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 algorithm is fundamentally different from the global optimization algorithm in that it not only decreases the AP power level but also increases it. The details are given below. For each AP $a \in \mathcal{A}$, we define a set N_a of all of its neighboring APs and let \bar{y}_a be the average load of the APs in N_a . When the load of an AP a reduces for whatever reason (e.g., user movements or local optimization operations by other APs), 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. If any of the two conditions is satisfied but the local algorithm cannot adjust the AP power level, the global optimization algorithm is invoked. In addition, the global optimization periodically is called in every Θ time units.

VII. SIMULATION RESULTS

In this section we compare the performance of our scheme with two existing methods, which are the Strongest-Signal-First (SSF) method and the association control method proposed in [14]. 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 method in [14] determines the user-AP association to achieve the max-min fair bandwidth allocation instead of making association decisions purely based on the signal strength. Since the max-min fairness problem is NP-hard, it first calculates the fractional optimal solution (which we call FRAC) under the assumption that a user can simultaneously associate with multiple APs, and then obtains the integral solution (which we call INT) via rounding to satisfy the single association constraint. This particular method is chosen as a performance benchmark because the characteristics of its solutions are known. The FRAC solution provides the strict performance upper bound (i.e., lowest possible congested load) and the INT method guarantees a 2-approximation solution⁵. Furthermore, as shown in [14] the integer solution (INT) converges with the fractional one (FRAC) as the number of users increases. Thus, *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. The objective of the comparison with the INT solution is to show that our cell-breathing achieves comparable performance to the association control method, without requiring special client software for association control on each mobile.* Surprisingly, however, the simulation results indicate that the cell-breathing scheme outperforms the INT solution in various load conditions.

The simulation setting is as follows. A total of 20 APs are located on a 5 by 4 grid, where the distance between two adjacent APs is set to 100 meters and each AP is equipped with a 10 Mbps 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 $\text{SNR} \geq 9$ dB, 5.5 Mbps when $\text{SNR} \geq 5$ dB, 2 Mbps when $\text{SNR} \geq 3$ dB and 1 Mbps when $\text{SNR} \geq 1$ dB. We set the maximal transmission power to 20 dBm and the minimal power to 10 dBm, while the intermediate power levels are determined by equally dividing the 10 dBm gap by the number of power levels simulated. Unless specified otherwise, 10 power levels are used. To simulate the indoor environment, we chose the path loss exponent [27] 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. All users are assumed completely back-logged and quasi-static. The priority of each AP is randomly chosen. We borrow the system performance metric from [14] and define the load of an AP a by,

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

⁵ Due to integrality gap, 2-approximation is guaranteed only when the AP load is above a certain threshold (which is 1 in our setting).

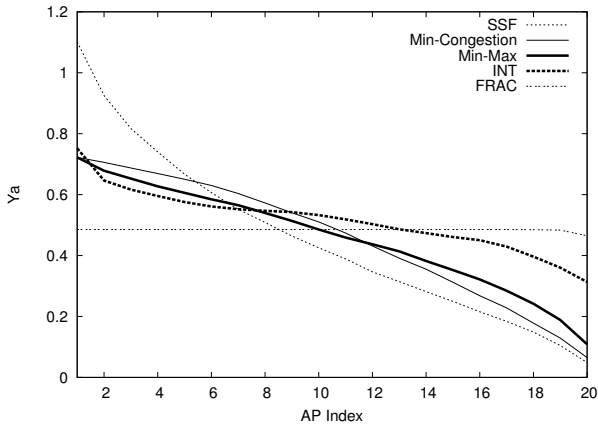


Fig. 10. Load comparison in networks with 100 random users.

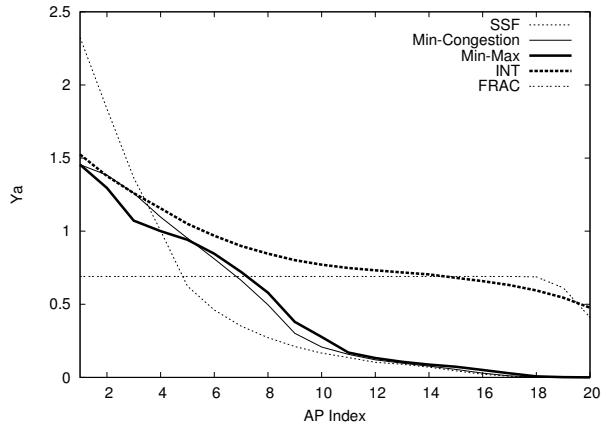


Fig. 12. Load comparison in hotspot networks with 100 users.

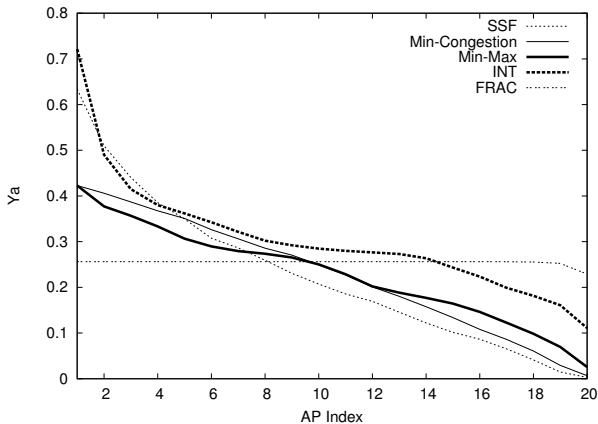


Fig. 11. Load comparison in networks with 50 random users.

where $r_{u,a}$ is the bit rate with which the user u communicates with the AP a and U_a is the set of users that are associated with AP a . Due to space limitation, we provide only few charts with typical results of our simulations.

Figure 10 shows the simulation results when 100 users are randomly distributed within the rectangular box that connects the boundary APs. 100 users were chosen to simulate a moderately-loaded network in which 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. Note that 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 x -th highest-loaded AP of each run. Only the points corresponding to the integer x indices are meaningful (continuous lines are drawn only for the presentation purpose). 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. The thick dotted line represents the INT solution and the horizontal thin dotted line represents the FRAC solution. Both the Min-Max and INT solutions are clearly better than the SSF solution. Both yield very similar maximal y_a values, which are about 35% higher than FRAC.

We also simulated the case of 50 users and the result is shown

in Figure 11. Now the gap between the INT and the FRAC is bigger than in the case of 100 users, and our Min-Max method outperforms the INT method with a significant margin. It is because the performance of the INT method depends on the number of users. Generally less users make the INT solution farther from the FRAC solution due to the bigger rounding error. Interestingly, the relative performance of our Min-Max method against the FRAC method and the SSF method, 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. For example, in the case of 200 users (not shown due to the space limit), the gap between FRAC and the our scheme stays around 35%. The steady relative performance against the FRAC solution regardless of the network load conditions (i.e., number of users) is one of the key strengths of the cell-breathing method.

We then consider the case of unbalanced user distributions. Only 20% of the users are randomly distributed and the rest are concentrated on two hotspots which do not overlap with each other. Each hotspot is a circle-shape area with 75 meter radius, and one hotspot contains twice more users than the other hotspot. This setting causes the heavy load condition in the hotspots. Figure 12 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 hotspots (i.e., heavy load condition) and clearly beats the INT method.

So far, we have assumed 10 power levels. To examine the impact of the number of power levels, we tried 4 different numbers of power levels. The simulation results in the 100 random user case is shown in Figure 13. 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.

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 user movements during the simulation. The number of power adjustments determines the time to take before the system converges on the optimal state and the number of user movements decides the handoff overhead. The collected statistics are summarized in Table II. For each table entry, two numbers are given; the first number is for the limited knowledge Min-

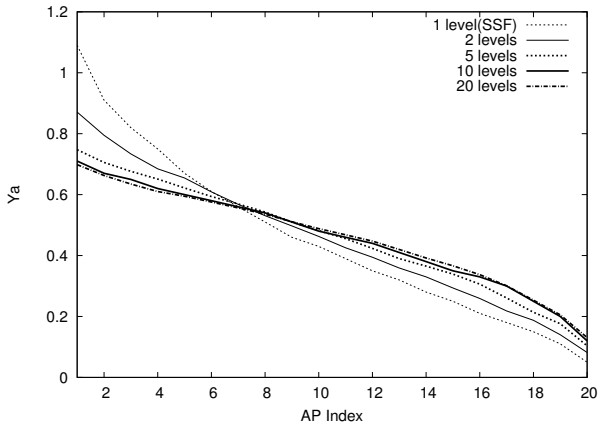


Fig. 13. Impact of the number of power levels on the performance of the Min-Max method

Case	Power adjustments	User movements
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)

TABLE II

RUN-TIME STATISTICS OF LIMITED KNOWLEDGE ALGORITHMS.

format: MIN-MAX (MIN-CONGESTION)

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 hotspots does not necessarily mean longer convergence time either and in the case of Min-Congestion, it even takes less time to converge.

VIII. CONCLUSION

We presented a novel cell breathing scheme for optimal load balancing in IEEE 802.11 networks. We provided rigorous analysis of the problem and presented two algorithms that find network-wide deterministic optimal solutions. The first algorithm minimizes the load of most 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 only assume the control on the transmission power of the AP beacon messages, which should be possible with simple software update of APs. The simulations show that even a small number of power levels, e.g., between 5 to 10, is enough to achieve near optimal load balancing solutions, which is another indication of the practicality of our scheme. In practice, our cell-breathing scheme can be deployed in network management tool of WLANs and activated each time the APs experience unbalanced load.

REFERENCES

- [1] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proc. USENIX MobiSys*, 2003.
- [2] T. Henderson, D. Kotz and I. Abyzov. The Changing Usage of a Mature Campus-wide Wireless Network. In *Proc. ACM MobiCom 2004*, pages 187–201, Philadelphia, PA, USA, September 2004.
- [3] T. Togo, I. Yoshii and R. Kohno. Dynamic cell-size control according to geographical mobile distribution in a DS/CDMA cellular system. In *Proc. IEEE PIMRC'98*, Vol. 2, pages 677–681, Boston, MA, USA, September 1998.
- [4] A. Jalali. On cell breathing in CDMA networks. In *Proc. IEEE ICC'98*, Vol. 2, pages 985 – 988, Atlanta, Georgia, USA, June 1998.
- [5] IEEE Standard 802.11, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications", 1999 Edition (ISO/IEC 8802-11:1999).
- [6] Proxim Wireless Networks. ORINOCO AP-600 data sheet, 2004.
- [7] Cisco Systems Inc. Data sheet for cisco aironet 1200 series, 2004.
- [8] I. Papanikos and M. Logothetis. A study on dynamic load balance for IEEE 802.11b wireless LAN. In *Proc. COMCON*, 2001.
- [9] I. Tinnirello and G. Bianchi. A simulation study of load balancing algorithms in cellular packet networks. In *Proc. ACM/IEEE MSWiM*, pages 73–78, 2001.
- [10] A. Balachandran, P. Bahl, and G. M. Voelker. Hot-spot congestion relief and service guarantees in public-area wireless networks. *SIGCOMM Comput. Commun. Rev.*, 32(1):59–59, 2002.
- [11] H. Velayos, V. Aleo and G. Karlsson, Load balancing in overlapping wireless LAN cells. In *Proc. IEEE ICC'04*, Vol. 7, pages 3833–3836, June 1998.
- [12] A. Kumar and V. Kumar, Optimal Association of Stations and APs in an IEEE 802.11 WLAN. In *Proc. of 11th National Conference on Communication*, January, 2005.
- [13] T-C. Tsai and C-F. Lien. IEEE 802.11 hot spot load balance and QoS-maintained seamless roaming. In *Proc. National Computer Symposium (NCS)*, 2003.
- [14] Y. Bejerano S-J. Han and L. E. Li. Fairness and Load Balancing in Wireless LANs Using Association Control. In *Proc. ACM Mobicom 2004*, pages 315-329, Philadelphia, PA, USA, September 2004.
- [15] V. V. Veeravalli and A. Sendonaris. The Coverage-Capacity Tradeoff in Cellular CDMA Systems. *IEEE Trans. on Veh. Tech.* pages 1443-1451, September, 1999
- [16] S-T. Yang and A. Ephremides, Resolving the CDMA cell breathing effect and near-far unfair access problem by bandwidth-space partitioning. In *Proc. IEEE VTC 2001 Spring*, Vol. 2, pages 1037 - 1041, May 2001.
- [17] L. Du, J. Bigham and L. Cuthbert, A Bubble Oscillation Algorithm for Distributed Geographic Load Balancing in Mobile Networks. In *Proc. IEEE Infocom 2004*, Hong-Kong, March 2004.
- [18] A. Sang, X. Wang, M. Madhian and R. Gitlin Coordinated Load Balancing, Handoff/Cell-site Selection, and Scheduling in Multi-cell Packet Data Systems. In *Proc. ACM Mobicom 2004*, pages 302–314, Philadelphia, PA, USA, September 2004.
- [19] J. K. Lenstra, D. B. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.
- [20] J. M. Jaffe. Bottleneck flow control. *IEEE Trans. on Communications*, 29:954–962, 1981.
- [21] Y. Afek, Y. Mansour, and Z. Ostfeld. Convergence complexity of optimistic rate based flow control algorithms. In *Proc. ACM STOC*, pages 89–98, 1996.
- [22] N. Megiddo. Optimal flows in networks with multiple sources and sinks. *Mathematical Programming*, 7:97–107, 1974.
- [23] J. M. Kleinberg, Y. Rabani, and E. Tardos. Fairness in routing and load balancing. In *Proc. IEEE FOCS*, pages 568–578, 1999.
- [24] D. Simone, 802.11k makes WLANs measure up. *Network World*, March 29, 2004.
- [25] R. Raz and S. Safra A sub-constant error-probability low-degree test, and sub-constant error-probability PCP characterization of NP. In *Proc. of 29th Ann. ACM Symp. on Theory of Comp*, ACM, pp 475-484, 1997.
- [26] M. R. Garey and D. S. Johnson, *Computers and Intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Company, San Francisco, 1979.
- [27] T. S. Rappaport, *Wireless Communications: Principle and Practice*. Prentice Hall, 1996.