This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Designing a Practical Access Point Association Protocol

Fengyuan Xu*, Chiu C. Tan*, Qun Li*, Guanhua Yan†, and Jie Wu‡
*College of William and Mary, VA
†Los Alamos National Laboratory, NM
‡Temple University, PA
*{fxu,cct,liqun}cs.wm.edu,†ghyan@lanl.gov,‡jiewu@temple.edu

***Abstract*—In a Wireless Local Area Network (WLAN), the Access Point (AP) selection of a client heavily influences the performance of its own and others. Through theoretical analysis, we reveal that previously proposed association protocols are not effective in maximizing the minimal throughput among all clients. Accordingly, we propose an online AP association strategy that not only achieves a minimal throughput (among all clients) that is provably close to the optimum, but also works effectively in practice with a reasonable computational overhead. The association protocol applying this strategy is implemented on the commercial hardware and compatible with legacy APs without any modification. We demonstrate its feasibility and performance through real experiments.**

I. Introduction

Wireless networks are increasingly used to provide ubiquitous Internet access. A crucial determinant of quality of service in wireless networks is the problem of access point (AP) selection in the distributed manner. A user selecting an inappropriate AP will experience bad service, or even hurt other users' throughputs. The current technique of AP selection is for the user to selfishly pick the AP with the strongest signal, or RSSI value. The intuition is that factors like multipath effect and path loss which reduce throughput will have a smaller effect when the user is communicating with an AP with a larger RSSI.

This simple strategy might fail when there is a large number of users crowded together. Consider the case when we have two APs on orthogonal channels, one with much stronger signal strength than the other, and a collection of users. All the users will simply pick the same AP (with the largest RSSI), so that the actual throughput of each user is very small because of channel contention between users. Based on this observation, alternative criteria such as selecting the AP which yields the largest throughput have been suggested.

However, it is unclear how well this *selfish* strategy will perform when every user attempts to connect to the AP which is able to maximize their own throughput. Unlike an AP's RSSI value measured by a user which is not affected by additional users associating with that AP, the AP's throughput will change as more users join in. Therefore, a user selecting an AP based on throughput may have to switch APs constantly, hence lowering overall performance.

In practice, we believe a good performance is to achieve the maximized minimal throughput for all clients. Using this simple but reasonable metric, we seek to design a practical distributed protocol for AP association. We theoretically analyze the worst-case performance of the selfish strategy, and introduce an online algorithm that achieves a better worst-case performance. Incoming user employing this algorithm determines an irrevocable association, only making use of the load information on the nearby APs, in order to minimize the $\mathbb{L}_p$ norm of the loads on all APs at the moment. Based on our online algorithm, we have implemented an association protocol for commodity hardware driver at the client side. This protocol works well with current legacy 802.11 APs. Using a combination of real experiments and extensive simulation, we demonstrate that our online association protocol performs better than the RSSI-based and selfish AP selection.

Our main contributions are:

- We have designed a distributed online algorithm for AP association based on the L_p norm of the loads on APs in proximity, and demonstrated theoretically its performance compared with selfish strategy.
- A light-weight method is introduced to estimate one user's throughput on the target AP without association, reducing the implementation overhead.
- We demonstrate our protocol's practicability using real experiments, as well as provide extensive simulations for large scale networks.
- Our solution is practical and does not require any modification on APs, making our technique applicable to existing wireless networks.

II. Related Work

AP association plays an important role in improving wireless performance [1]–[10]. Work by [11] demonstrated why the use of signal-to-noise ratios in selecting APs is not appropriate. Both [11] and [12] considered techniques to allow the client to estimate the AP workload before connecting, while [13] considered the use of available bandwidth in AP selection. A more holistic solution encompassing factors such as the number of connected clients and mean RSSI was proposed in [14].

The selfish behavior of users in a congestion game has been studied theoretically. A special case where each user's decision is a singleton set is considered in [15], while [16] describes a class of congestion game where the payoff function associated

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

with each resource is user specific. The convergences under different load balancing scenarios are provided in [17]. In this work we model the decentralized AP selection with selfish users as an extension of the weighted singleton congestion game in which the weight of a user varies as the associated AP changes. Other modelings of the wireless infrastructure selection include [18] and [19], targeting at different scenarios and goals. The major distinction of our work is that we design and implement on the commercial chipset an online greedy AP selection protocol in the distributed manner. The performance is supported by theoretical proof and demonstrated both in real experiments and simulations.

Compared with decentralized methods, work by [20] and [21] uses the idea that better AP association decisions can be obtained by relying on a global view of the entire WLAN, or an extra centralized controller. AP side system is modified in [21] to aggregate workload information and provide association control according to it. In [20], a more complicated central scheme for AP association is discussed.

There are also other papers discussing how to multiplex multiple APs. In [22], it created multiple virtual interfaces based on one single wireless card, and made them communicate with associated APs like simultaneously. The paper [23] built a multi-interface association mechanism to distribute a client's data traffic on multiple accessible APs in a scenario where the backhaul link is the bandwidth bottleneck.

III. Motivation

We consider an IEEE 802.11 infrastructure network [24], in which there are m APs and n stationary clients or users. Given no central controller and local information, All the clients are allowed to freely choose an AP within the transmission range to associate with. The goal of this paper is *to maximize the minimum throughput over all clients*.

The AP association protocol currently employed in IEEE 802.11 networks lets a client associate with the AP that gives the strongest signal. We term this the Best-RSSI strategy. However, the Received Signal Strength Indication (RSSI) may not be a good indicator of throughput changes. To illustrate, we present a simple case study on certain stationary wireless LAN client under an environment where there are a lot of interference and congestion randomly generated by co-existed others sharing the same AP with this client. Figure 1 records during a time interval the associated AP's RSSI measured at this client side and the MAC layer throughput samplings. Through this paper, we consider the MAC layer throughput if no specification.

It is easy to observe that the relatively stable state of the RSSI does not reflect the relatively intensive fluctuation of the sampled throughput. Thus, RSSI is not suitable and accurate enough to evaluate an AP's performance. It is possible to end up with a situation in which all clients connect to a single AP. In this case, the competitive ratio*, in terms of average client throughput, can be as bad as $1/m$. When m is large, this plummeting performance becomes unacceptable.

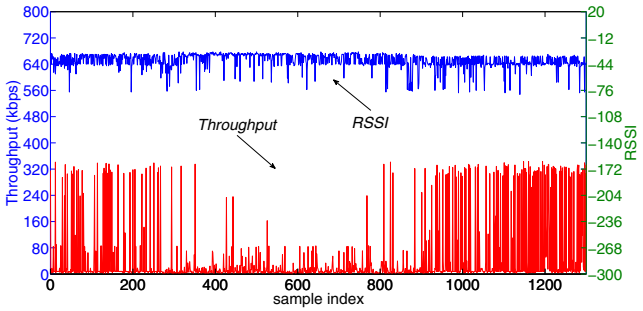

Fig. 1: RSSI versus throughput. The sample index increases along with time

IV. Selfish User Strategy

One natural alternative to solving the above problem, with respect to our goal, is to let the clients behave myopically by applying in decentralized AP selection the *best-reply* policy. Explicitly, it means that every user keeps moving to associate with the AP that could offer it the best throughput *until no user can gain higher throughput by unilaterally deviating from its current decision (Nash Equilibrium)*.

To simplify the analysis for selfish users, we make two assumptions in this section. In the next section, we will use a more realistic assumptions. First, we assume that the interference between the communications of two APs is not considered, i.e., the nearby APs operate on orthogonal channels. Second, the association procedure of a user is considered as an atomic operation, so only one user perform association at a time. The time at which a user makes a decision to change APs is marked as a *decision step*. However, we do not require users to follow a certain decision order, which means in each decision step the user who is picking a new AP could be any one. Under these assumptions, we will show that such selfish user game always converges to a Nash Equilibrium with non-optimal performance. More complicated scenarios even cannot grantee the existence of the Equilibrium state [16], [17].

We denote by U_a the set of users connecting with AP a. So let $n_a = |U_a|$ represent the cardinality of this set. We designate by st_u the percentage of service time the user u gains from associated AP, and T_u corresponds to the throughput of u. And for any user u and AP a, we use R_{ua} to denote the *transmission rate* under the situation only u is associating with a. R_{ua} varies even for the same user. For the rest of this section, unless otherwise specified, the transmission rate refers to the effective transmission rate, which considers the overhead caused by retransmissions, random backoff and so on.

To examine the performance of this protocol, we consider two aspects: convergence and competitive ratio. The competitive ratio here is equivalent to the *price of anarchy*† using

*competitive ratio is the performance ratio, with respect to some metric, between worst outcome of certain association protocol and the optimal strategy case.

†the ratio of the worst-case social cost among all Nash Equilibria over the optimal cost

minimum user throughput as social cost. The following subsections show first whether the selfish user protocol will eventually stabilize and how fast the protocol will achieve convergence in general, and after that give the competitive ratio of the protocol.

A. Convergence of the Selfish Strategy

In this subsection, we will show how to model this selfish throughput strategy as a special case of the weighted congestion game, where the weight of a user varies as the associated AP set, which is singleton, changes. This game is proved to be converged with not ideal speed by leveraging the technique similar to [17]. We start with a lemma to characterize the throughput calculation.

Lemma 1: All the users on an AP a have the same throughput T_a

$$T_a = \frac{1}{\sum_{i \in U_a} \frac{1}{R_{ia}}} \tag{1}$$

Proof: Owing to *Carrier-Sense Multiple Access with Collision Avoidance* (CSMA/CA) protocol, all the users associated to the same AP, no matter what their transmission rates are, have a fair chance to seize the channel for packet transmission.

Therefore, given the same packet size z, any user u connecting to AP a with a transmission rate R_{ua} is assigned a portion of service time from a:

$$st_u = \frac{\frac{z}{R_{ua}}}{\sum_{i \in U_a} \frac{z}{R_{ia}}} = \frac{\frac{1}{R_{ua}}}{\sum_{i \in U_a} \frac{1}{R_{ia}}} \tag{2}$$

It is obvious that every user on the same AP has throughput:

$$T_a = T_u = st_u \times R_{ua} = \frac{1}{\sum_{i \in U_a} \frac{1}{R_{ia}}} \tag{3}$$

■

Given the Lemma 1 and assumptions we made, the selfish AP selection can be modeled as an extension of the congestion game. For sake of clarity and consistence in terminology, we describe this model in the wireless LAN scenario. Consider a set M of APs, each having a load function depending on the total weight of the users associated (Definition 1), and a set U of users, each of whom only can choose one AP from a permissible subset of M (in the absence of a coordinating authority). The weight of a user i on AP j is defined as $\mathbb{L}_{ij} = \frac{1}{R_{ij}}$. Accordingly, maximizing the minimum throughput over all clients is equivalent to minimizing the maximum load over all APs.

Definition 1: The load of an AP a, $\mathbb{L}_a$, is

$$\mathbb{L}_a = \sum_{i \in U_a} \mathbb{L}_{ia} = \sum_{i \in U_a} \frac{1}{R_{ia}}$$

For the convergence proof, we introduce a sorted vector (in ascending order) of all users' throughput as the potential function. According to $Lemma$ 1, we can simplify this vector to a new one $\overrightarrow{\mathcal{T}}$ by using T_a above to represent respectively the current throughput of every user associated with AP a (for $\forall\ a\ \in M$, where M is the set of all APs). Put differently, given a user i associated with AP a, its throughput is replaced with T_a in the $\overrightarrow{\mathcal{T}}$.

The following defines the *lexicographic order* on different vectors ($\overrightarrow{\mathcal{T}}$).

Definition 2: One vector $\overrightarrow{\mathcal{T}}$ defined above is called lexicographically larger than the other one $\overrightarrow{\mathcal{T}'}$ if $\overrightarrow{\mathcal{T}}$'s first unequal element is larger than its corresponding position index one in $\overrightarrow{\mathcal{T}'}$, where both vectors are in ascending order.

Definition 3: In $\overrightarrow{\mathcal{T}}$, $T_a(s)$ denotes the throughput T_a at the decision step s.

We show the convergence of the protocol by identifying a potential function and showing that this potential strictly increases after each step. Consider the vector $\overrightarrow{\mathcal{T}}$, in an ascending order of all users' throughput.

Theorem 1: $\overrightarrow{\mathcal{T}}$ lexicographically increases when a user i moves from AP j to k for better throughput.

Proof: Based on the assumption that interference is not considered in this section, we know this migration only influences two components of $\overrightarrow{\mathcal{T}}$: one corresponding to the throughput for AP j that user i just left, while the other corresponds to the AP k that i has joined. Other components remain unchanged. Suppose this is the $s+1$-th decision step. Because user i moves for a higher throughput, $T_k(s+1) > T_j(s)$; and because AP j has one less client, its throughput increases: $T_j(s+1) > T_j(s)$. In a word, if $T_j(s)$ is the p^{th} component in $\overrightarrow{\mathcal{T}}$ at step s, $T_j(s+1)$ and $T_k(s+1)$ reside at two positions whose indexes are no smaller than p^{th} (at the right side of p^{th} position including p).

Assume in $\overrightarrow{\mathcal{T}}$ at step s, $(m-q)^{th}$ (recall m is the number of APs) is the first position in which the value is larger than the one in position p, i.e., there are q throughput larger than $T_j(s)$ in $\overrightarrow{\mathcal{T}}$. Note that only if no throughput is equal to $T_j(s)$ in $\overrightarrow{\mathcal{T}}$ at step s, $m-q=p$. After step $s+1$, this number q increases by 1 for the reason mentioned above (T_j moves to the right). Thus, the $(m-q-1)^{th}$ position becomes the first position that holds different values for step s and step $s+1$ in $\overrightarrow{\mathcal{T}}$. Obviously, according to the definition of lexicographical order, the vector $\overrightarrow{\mathcal{T}}$ at step $s+1$ is larger. ■

Since we have shown that users' migration always increases the potential - $\overrightarrow{\mathcal{T}}$, this gives us an upper bound (Theorem 2) on the convergence time in general.

Theorem 2: Without specifying the concrete underlying configuration, this network (m APs and n clients) reaches the equilibrium in at most m^n steps.

Proof: It is equal to the number of different sorted vectors, which is bounded by the number of network topology snapshots. In other words, after performing at most m^n steps, this potential function $\overrightarrow{\mathcal{T}}$ will come to a state at which it will not be larger any more. This means that no matter which user has the chance to make a decision at the next step, it will stay on its current AP from its selfish point of view. ■

B. Competitive Ratio

In the following, we obtain the competitive ratio with respect to the minimal throughput over all clients in the selfish protocol.

We still assume that the number of APs is m and the number of clients (or users) is n. We also assume that, among all users, the maximal available transmission rate is R_{max} and the minimal available transmission rate is R_{min}. For convenience, we define the load a client imposes to an AP as the reciprocal of the transmission rate of a client when connecting to an AP. Therefore, we define $\mathbb{L}_{max} = \frac{1}{R_{min}}$, and $\mathbb{L}_{min} = \frac{1}{R_{max}}$.

In a Nash Equilibrium of the selfish strategy, suppose the most loaded AP is k, which has a load $\mathbb{L}^S$, i.e., $\mathbb{L}_k$. Since this selfish user game reaches the equilibrium, any client connecting to AP k is not willing to move to any other AP j. That is, $\mathbb{L}^S \leq \mathbb{L}_{max} + \mathbb{L}_j$ for $\forall j \in \{a | a \in M \& a \neq k\}$. Thus, $\mathbb{L}^S \cdot m \leq \mathbb{L}_{max} \cdot (m-1) + \sum_{j=1}^{m} \mathbb{L}_j \leq \mathbb{L}_{max} \cdot (m-1) + \mathbb{L}_{max} \cdot n$

$$\therefore \quad \mathbb{L}^S \leq \frac{\mathbb{L}_{max} \cdot (n+m-1)}{m}$$

Then in the optimal strategy, the maximal load over all the APs is $\mathbb{L}^O \geq \frac{n \cdot \mathbb{L}_{min}}{m}$ In sum, the price of anarchy is

$$\frac{\mathbb{L}^S}{\mathbb{L}^O} \leq \frac{n+m-1}{n} \cdot \frac{\mathbb{L}_{max}}{\mathbb{L}_{min}}$$

In summary, the selfish user protocol has a high convergence time and a poor performance when the number of APs is large and the ratio between the maximal and minimal rates is large.

V. Online Algorithm

We have shown that both the Best-RSSI and selfish user protocols perform poorly under certain scenarios. In this section, we introduce our practical online association strategy. Our online algorithm considers merely communication load including interference and congestion, which provides a more realistic model.

The protocol is simple without assuming a complex interaction among clients and APs. We merely assume that, when a new client joins the network, it can measure the loads (recall the Definition 1) of all APs within its hearing range. If the client does not affect an AP, or does so with negligible influence, it does not need to know the load on that AP. For example, if a client is far away from an AP, the interaction between them or the influence would be marginal and the client will not consider the information on that AP when it makes its AP association decision. We will show by implementation in section VI that this assumption can be approximately achieved through a practical and low-overhead measurement method. We do not even assume how the loads will be changed when a client joins – although we do assume the load on each AP will be non-decreasing when a client joins.

In the following, we examine several scenarios to show the ramifications of our assumptions and demonstrate how much our assumptions conform to the reality.

- **Interference with APs**: When client i joins the network, it might interfere with the transmission and reception on several APs. We denote the loads imposed on AP j after i makes its association decision as $\mathbb{L}_{ij}$. Note that j may not be the AP that client i associates with.
- **Interference with clients**: When client i joins the network, it might interfere with another client k. Even though i may not directly interfere with the AP (say AP j) that k is associated with (possibly due to being out of transmission range), the interference of i on k's communication may change the load on AP j. If the load on AP j is visible to client i, this scenario is amenable to our analysis; otherwise, we will ignore the load imposed by this indirect influence because the load change due to this rippling effect is marginal.
- **Myopic network configuration**: When a client i joins the network, it may not see all the APs because of the limited communication range. If the client does not see an AP, we assume the load change on that AP owing to the joining of client i is negligible. Furthermore, in this case, client i will not be able to associate with that AP because there is no usable bidirectional link between the client and the AP.

In summary, we study a more practical and complicated wireless LAN model here than the one used in the previous section. The weight (communication load) a user adds on the associated AP might vary as the local network configuration changes or new incoming clients appear.

The online AP selection algorithm runs as follows. When a new client appears (in online fashion), it will make an irrevocable association with one of the visible AP so that the $\mathbb{L}_p$ norm of the loads on all the APs within its transmission range, after its join, will be minimized at the moment. Since the client is unable to affect the other APs that are not in its hearing range, this algorithm will minimize the $\mathbb{L}_p$ norm of all the APs' loads in the system $(\mathbb{L}_1^p + \mathbb{L}_2^p + \cdots + \mathbb{L}_m^p)^{1/p}$ in each new association event. Then we have

Theorem 3: The online protocol gives a $r \leq \frac{1}{2^{1/p}-1}$-competitive ratio if the protocol is to minimize the $\mathbb{L}_p$ norm of the loads.

Theorem 3 is proven in Appendix using the main idea of [25]. From theorem 3, we derive that

Theorem 4: The online protocol is a $e \log m$ competitive protocol for minimizing the maximal load (or $\frac{1}{e \log m}$, w.r.t. maximizing the minimal throughput).

Proof: Let the heaviest load among all APs running our protocol be $\mathbb{L}_m$ and the heaviest load among all APs in the optimal minimizing heaviest load protocol be $\mathbb{L}_m^*$. Thus, $m^{1/p}\mathbb{L}_m^* = (m \cdot \mathbb{L}_m^{*p})^{1/p} \geq (\sum_j \mathbb{L}_j^{*p})^{\frac{1}{p}} \geq \frac{1}{r}(\sum_j \mathbb{L}_{nj}^p)^{\frac{1}{p}} \geq \frac{1}{r}(\mathbb{L}_m^p)^{1/p} = \frac{1}{r}\mathbb{L}_m$. In other words, the L_p norm protocol is a $rm^{1/p}$-competitive online algorithm for minimizing the heaviest load on all APs. $rm^{1/p} \leq \frac{m^{1/p}}{2^{1/p}-1} \leq m^{1/p}\frac{p}{ln2}$. When $p = lnm$, $m^{1/p}\frac{p}{ln2}$ reaches its minimum value, $e \log m$, in the positive real number domain $\mathbb{R}^+$. Thus, we choose $\mathbb{L}_{lnm}$ norm, and the competitive ratio, correspondingly, is $e \log m$. ■

Instead of being related to the number of APs and the ratio between the maximal and minimal rates, the competitive ratio of this protocol is linear to the logarithm of the number of APs, an almost constant competitive ratio for a small number of APs, which is deemed very promising since a constant competitive

ratio algorithm usually gives a very good practical performance.

Furthermore, this algorithm has the advantage of computational simplicity and feasibility for practical implementation. The expected performance bound, for each client joined, is ensured just by the local network information at the moment it was coming as a new client. It is not necessary to reconsider its decision once a new association event occurs. In other words, our online algorithm takes exactly n steps to finish. In the next section, we demonstrate that this algorithm can be employed as a practical and light-weighted association protocol for off-the-shelf wireless LAN adapters.

VI. Protocol Implementation for Online Algorithm

We implement aforementioned online algorithm on the popular commercial wireless LAN adapter by taking advantage of the legacy standard 802.11 protocol. The implementation is shown using the Click Modular Router toolkit [26]. The association functionality of the MadWifi driver v0.9.4 [27] is directly taken over by our module. It does not require any change at the AP side, so our implementation can be used in any open 802.11 networks or networks the client is able to access.

We consider the scenario that wireless link is the bottleneck of a communication connection. The discussion of other cases is out of scope of this paper. Thus, the workload of an AP is reflected by the wireless traffic on air for this AP. Here we monitor the uplink stream traffic, ignoring the downstream, because accuracy improvement of throughput measurement is small compared with the extra complexity in the implementation experience of [23]. Every channel is considered to be interference-free with others, as this type of interference is ignorable compared to interference inside the channel. Thus, the computation of the $\mathbb{L}_p$ norm of the AP's workload can be reduced to per-channel based computation, while the comparison is still among all channels.

In order to realize our online algorithm in the driver, an efficient implementation is required to measure every AP's load on the same channel when a client i joins a candidate AP j [‡]. A natural way to obtain this information is to let the client i perform an association operation with j, and generate traffic on j while at the same time capturing an uplink data stream for each AP by passive listening. The packet retransmission and duplication does not count. However, the association process consumes a lot of time, especially for encrypted wireless networks. When the set of association candidates is not small, the user is not able to bear waiting so long. Nevertheless, sending data packets without association will lead to rejection from the object AP because of the IEEE 802.11 standard. Thus, we find a more light-weight way to obtain the equivalent load information without association. Currently 802.11 standards require an AP to respond to *probe requests*, even if the request is sent by a station not associated with the AP [§]. We leverage this to create a packet type to replace the real data packet in the MAC layer. The intuition is to generate modified probe request traffic to the object AP j, similar to the data traffic, to estimate other APs' loads as if the client i is associating with j. The detail modifications of the probe request packet are made as follows.

- We make the probe request uni-cast, forcing the target AP to return an ACK upon receiving the packet. This behavior is similar to a station transmitting data packets to an AP. And this process is important as well for calculating the throughput.
- We change the subtype flag in the packet header to prevent the AP from returning a probe response, in order not to introduce unnecessary traffic to the network.
- The packet size, transmission rate, and inter-arrival time of modified probe requests are packet-wisely customizable by the user, which is able to provide more accurate throughput information for specific estimation based on upper-layer applications. This feature is implemented in the probing generator module. Its performance is shown in the next section.

We also implement an AP filter to make the candidate AP list programmable. The user can select a preferred channel, network, and minimal RSSI threshold to customize the list. Only qualified APs will be considered for estimation to reduce overhead.

The implemented protocol is described as pseudo code (Algorithm 1). A list of candidate APs is determined in the first place for estimation according to the beacons. In the list, the candidates from the same channel i group as a set $\mathcal{C}_i$. For each target AP j in $\mathcal{C}_i$, the user injects a probing traffic, which consists of modified probe request packets, to the AP j, while measuring all members' loads. After these measurements, the user can calculate the $\mathbb{L}_p$ norm loads for all members within this channel set, $(\sum_{k\in\mathcal{C}_i}\mathbb{L}_k^p)^{1/p}$, where p is the natural logarithm of the number of APs. This $\mathbb{L}_p$ norm value, influenced by association with the target AP, will be compared with the current best candidate AP among all channels. The comparison strategy applied in the best candidate updating stage is controlled through two programmable parameters. The first one is the norm-difference threshold T_{nd}, and the second one is RSSI-difference threshold T_{rd}. If the norm for the target AP are at least T_{nd} smaller than the norm for the current best AP, the target AP will be the best candidate AP instead of current one. Otherwise, the algorithm continues to check whether this norm is just smaller than the current best candidate's, as well as whether the target AP's RSSI is at least T_{rd} more than the best candidate's. If so, the target AP will be the best; for all other cases, we keep the current best AP without updating. The user treats every member of a channel set as the target member respectively, and repeats this process for each channel set. After the evaluation of all candidate APs, the user will associate with the final best candidate AP.

[‡]Assume i always has some communication demand after association
[§]It is done automatically by the firmware, transparent to the upper layers

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Algorithm 1 Protocol Description

```
Discovers all available APs
Applies the filter on discovered APs
Puts all filtered APs into candidate list
for each C_i do
  for each AP j in C_i do
    Generates a probing traffic to j
    Estimates the new load of each AP within C_i
    Calculates the L_p norm change
    Updates the best candidate AP according to T_nd and T_rd
  end for
end for
Associates with the best candidate AP
```

VII. Evaluation

We verify the feasibility of our online algorithm and demonstrate the performance of our protocol implementation in this section. Each client is powered by a 1.66GHz CPU with 1 GB RAM, running on Linux kernel 2.26.24. A D-Link WNA-2330 with Atheros 5312 chipset wireless card is used.

A. Application Aware Probing

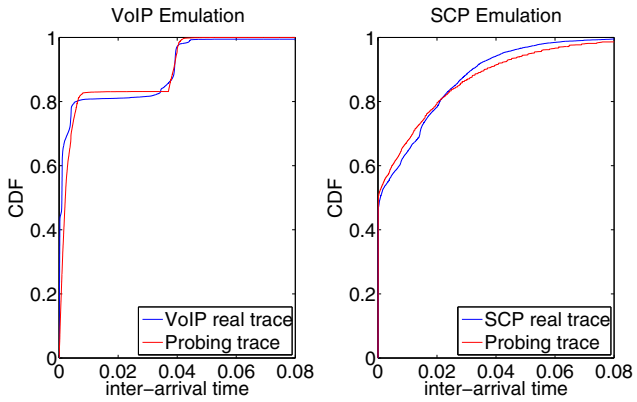

Fig. 2: Upper-layer application stream emulations.

Since our modified probing stream, used to emulate the real data stream, is programmable in terms of the packet size, inter-arrival time, and transmission rate, it is easy to generate specific streams to mimic the data stream of a certain application. Thus, the client is able to find out the "best" association AP with respect to the application it wants to use. Figure 2 demonstrates how similar our probing can be to the secure copy (SCP) and VoIP protocols, respectively. The SCP protocol used is the Unix *scp* command line program. SCP transfers a single file from a laptop to a remote desktop on the Internet through a commercial AP on the channel 6 with RSSI -58. The packet size of the probe emulating SCP is 1500 bytes, and the inter-arrival time is presented at the right of Figure 2. On the left side of Figure 2, we choose Skype as the VoIP application to set up a communication between two laptops through the same commercial AP on the channel 1 with RSSI -33. The probing packet used in this experiment is 200 bytes on average. For both experiments, we first brought up our driver module to create virtual interfaces in the kernel for all upper-layer applications. Then a script was executed to associate with the target AP in each experiment. After association and IP assignment, we ran the application and our probing generator to generate the traffic, respectively. The traffic traces are captured by Wireshark for cumulative distribution function (CDF) calculation of the inter-arrival time. In these two experiments, the transmission rate for all streams is fixed at 36Mbps.

B. Measurement Accuracy

The APs' load information needed in our protocol is derived by monitoring the wireless channels. Although it is not necessary for monitoring to capture all packets on the air, a relatively accurate measurement can help to make a better association decision. Thus we conducted a series of experiments to investigate the capture missing, which is the main factor to cause the measurement error of the load. In this paper, we are focusing on the Atheros 5212 chipset, while other chipsets can be easily studied like this as well. We set up two laptops with a distance of 10 feet between them. One is the target laptop, which is used to generate a data stream for measurement. The other laptop acts as the monitor to estimate the data throughput from the target laptop. To make our experiment comprehensive, we use different transmission rates when transmitting the data streams. The rates used are 1Mbps, 2Mbps, 5.5Mbps, 11Mbps, 18Mbps, 36Mbps, and 54Mbps. We also use different inter-packet times of 5ms, 10ms, 15ms, and 20ms at each rate, respectively. The packet size in all trials is 1000 bytes, and every trial last 5 seconds for data stream generating and capturing. The experiment results are shown in Figure 3. The x axis presents the captured packet number in each trial at the monitor laptop side, while the y axis presents the number of transmitted packets counted at the target laptop side. It is clear that if there is no error, all points (star or circle) should fall on the green dash line $y = x$. Based on these experimental results (excluding the six outliers), we are able to calculate the best linear fitting by using the Least Square method, which is shown as the red line, $y = 0.8806 \times x$. 0.8806 is used for estimation calibration with respect to the Atheros 5212 chipset, i.e., $estimation = \frac{measurement}{0.8806}$

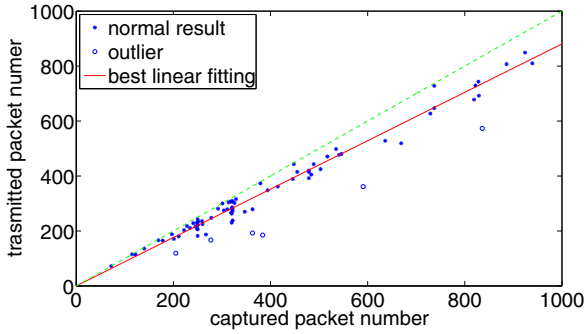

Fig. 3: Calibration Experiments for Atheros 5212 chipset

TABLE I: Comparison Experiment Settings

APs				
ID	Model		Channel	
AP-1	LinkSys WRT54G		CH 11	
AP-2	D-Link DI-713P		CH 11	
AP3	LinkSys WRT54G		CH 1	
Clients				
ID	Adapter	Pkt payload (byte)	Inter-pkt time(ms)	Transmission rate(Mbps)
STA-1	internal	1000	5	2
STA-2	internal	1000	5	11
STA-3	external	1000	5	11
STA-4	external	1000	5	11

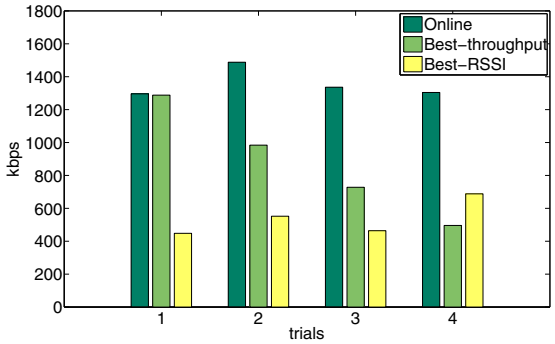

Fig. 4: Comparison Experiment Results

C. Comparison Experiment

We conduct an experiment to compare our association method with other practical ones, the Best-throughput and Best-RSSI strategies. Best-throughput here is one special case of the selfish strategy, of which the convergence speed can be bad. It means every client will make an irrevocable association decision to maximize its own throughput. In the experiment, there are three APs consisting of an *extended service set* (ESS) and four wireless LAN clients. Two of the APs (AP-1 and AP-3), whose process capabilities are relatively stronger than the thirds , are set in channels 1 and 11, respectively, and the third one (AP-2) is set in channel 11 as well. Three of the four clients, STA-1, STA-2 and STA-4, are put close to each other. Detailed settings are shown in Table I.

The experiment includes four trials. In each trial, clients came to join the ESS one by one by using the same association strategy. It is reasonable because that, in real world, the time to perform the association operation is statistically much smaller than the inter-arrival time between new users, so the possibility that two clients will want to join the ESS at the same time is low. After joining, the client will generate traffic using the configuration in table I to the associated AP. A trial was repeated three times, one for each association strategy.

The minimum client throughput for each strategy in the four trials are presented in Figure 4. Our algorithm performs better than the other two because it can balance the APs' workloads and reduce the interference among all wireless nodes. The performance of the Best-throughput is unstable, and it also indicates that the selfish strategy cannot compete with ours under similar overhead costs. Meanwhile the Best-RSSI strategy often makes all clients associate with the same AP.

D. Overhead

All three protocols mentioned above need an AP discovery phase. However, Best-RSSI does not have any other overhead, whereas Best-throughput and our protocol need extra time to evaluate every discovered AP. For each channel, our protocol spends a fixed amount of extra time, 500 ms, on passive listening, compared with the Best-throughput one. Nevertheless, the Best-throughput protocol requires real de-association and re-association operations, including the IP assignment, before every measurement. These operations consume a lot of time, from 3 sec to 8 sec. When the AP number grows, the Best-throughput protocol spends much more time than ours. Therefore, the overhead of the selfish protocol, which is equivalent to multiple runs of the Best-throughput protocol, is even worse. Thus, our protocol is the most efficient one.

VIII. Simulation Result

We use simulations to evaluate our association protocol on a larger scale with more wireless nodes and various configurations. We use NS2 version 2.33 as our simulator. The multiple channel feature is patched into the NS2 wireless portion following the instructions of [28]. The MAC layer type is 802.11, while the radio propagation model is two-ray-ground. Ad-Hoc routing protocol is disabled since we are focusing on the infrastructure type of wireless LAN. The RTS/CTS mechanism is also disabled. The data traffic for users is a UDP stream with a packet size of 1000 bytes and average inter-arrival time of 1 ms. The transmission rate is set to 11 Mbps. The selected channels include 1, 4, 5, 6 , and 11 for covering the orthogonal and adjacent channel cases. The throughput measurements are between the wireless nodes.

We implement the following three practical association protocols for comparison.

1) **Online**. This is our proposed online algorithm. It is implemented as described in Algorithm 1.
2) **Selfish**. The behavior of *Selfish* is defined in section IV. For the same reason that the convergence speed of selfish strategy can be very bad, we demonstratively run the protocol for 5 rounds. In the first round, the clients come to join the wireless LAN one by one. Each client will associate with every AP to measure the UDP throughput and pick the one who is able to offer the highest value. In each of the next four rounds, every client will repeat the above process to adjust its association based on current wireless LAN association topology. Finally, every client will keep its association with the AP it picks in the last round.
3) **Ideal**. This is the globally optimal association solution in terms of maximizing the minimum throughput over all clients. *Ideal* is obtained by enumerating all possible association topologies, given a specific scenario setting

only including the location and channel assignment information. In the real world, it is not practical because of its complexity.

Every experiment conducted below consists of many trials. Each trial has its own scenario configuration. The configuration provides the locations of all wireless nodes and the APs' channels. Both of these two information are randomly generated. Every throughput measurement, no matter whether it is for a data stream or probing stream, takes 3 seconds in the simulation.

The first simulation is to study the competitive ratio of *online* compared with *ideal* from the perspective of empirical experiments. The experimental value of the competitive ratio is a good indicator of the performance gap on average between the *online* and *ideal*. The statistically stable performance is also a concerned issue to the users in the real world. This experiment randomly picked 50 scenarios for testing. For every scenario, the competitive ratio in terms of the minimal client throughput, shown in Figure 5, is calculated based on the test result of *online* and *ideal*. The theoretical upper bound is also provided for comparison. The simulation results shows that about 86 percent of competitive ratio is above 0.47, and 70% are quit stable, just around 0.5. The worse competitive ratio among these 50 trials is 0.313, while the theoretical upper bound, computed from $\frac{1}{e \log m}$, is 0.232.

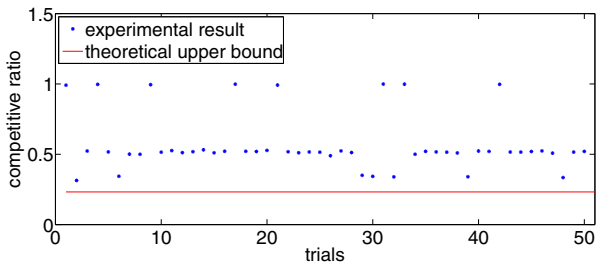

Fig. 5: Competitive ratio results, w.r.t. minimal client throughput, for 5 clients and 3 APs within 20×20. The simulation repeated 50 times with different configurations.

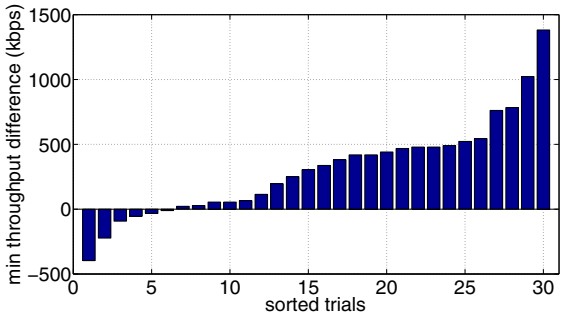

Fig. 6: Simulation result for 10 clients and 3 APs within 30×30. The simulation repeated 30 times. Each bar is the representative of minimum throughput difference for each trial. The results are sorted in ascending order

Next, we conducted a scale-up comparison simulation between *online* and *selfish*, which includes three experiments to show the performance in large scale deployments. In the first experiment, there are 10 clients and 3 APs within a rectangle of 30×30; the second one has 20 clients and 6 APs located in a rectangle of 90×90; 30 clients and 9 APs are involved in the third experiment within a rectangle of 150×150. Each experiment ran 30 trials. For each trial scenario, both our strategy and the selfish strategy were applied for the association processes of all clients on this setting, respectively. After finishing all users' association processes, we measured the UDP traffic throughput for every client and found the minimum, T_{online} for *online* and $T_{selfish}$ for the *selfish* strategy. Then the minimal client throughput difference, shown in Figures 6, 7, and 8, is calculated by using $diff = T_{online} - T_{selfish}$. These figures show that, even through the selfish protocol is allowed to consume more time, our strategy is more often to perform better in terms of maximizing the minimum client throughput.

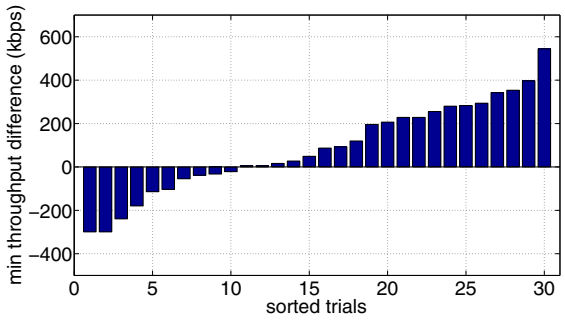

Fig. 7: Simulation result for 20 clients and 6 APs within 90×90. The simulation repeated 30 times. Each bar is the representative of minimum throughput difference for each trial. The results are sorted in ascending order

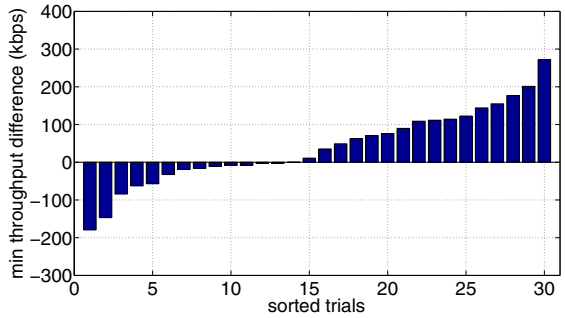

Fig. 8: Simulation result for 30 clients and 9 APs within 150×150. The simulation repeated 30 times. Each bar is the representative of minimum throughput difference for each trial. The results are sorted in ascending order

In the *online* strategy, since every client only needs to run our association once, the following clients in the future will not affect the behaviors of current associated clients. Meanwhile, for the *selfish* protocol, the unexpected new clients can easily break the current equilibrium into an unstable state, which will interrupt the usage of users. Thus, the *online* is more practical and less-intrusive. From the figures, it is shown that ours can, despite not knowing who will come to join the network, reduce the performance downgrade for the client who has the minimum throughput.

IX. Conclusion

In this paper, we consider the problem of AP association in WLAN. We present a theoretical analysis of the performance of two commonly used AP selection protocols, and propose an online algorithm with a provable good competitive ratio. The association protocol based on this algorithm is implemented on the real testbed in a light-weight way. We evaluated our scheme using a combination of implementation on commodity hardware and extensive simulation. We demonstrate that it is promising that by combining our theoretical understanding and real system implementation experience, a new, practical, and better AP association protocol is possible.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-0721443, CNS-0831904, CAREER Award CNS-0747108, CNS-0626240, CCF-0830289, and CNS-0948184.

References

[1] V. Mhatre and K. Papagiannaki, "Using smart triggers for improved user performance in 802.11 wireless networks," in *MobiSys 2006*.
[2] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall, "Improved access point selection," in *MobiSys 2006*.
[3] K. Sundaresan and K. Papagiannaki, "The need for cross-layer information in access point selection algorithms," in *IMC 2006*.
[4] S. Shakkottai, E. Altman, and A. Kumar, "Multihoming of users to access points in wlans: A population game perspective," *IEEE JSAC 2007*.
[5] T. Korakis, O. Ercetin, S. Krishnamurthy, L. Tassiulas, and S. Tripathi, "Link Quality based Association Mechanism in IEEE 802.11h compliant Wireless LANs," in *RAWNET 2005*.
[6] H. Wu, K. Tan, Y. Zhang, and Q. Zhang, "Proactive scan: Fast handoff with smart triggers for 802.11 wireless lan," *INFOCOM 2007*.
[7] A. Mishra, M. Shin, and W. Arbaugh, "An empirical analysis of the ieee 802.11 mac layer handoff process," *SIGCOMM Computer Communication Review 2003*.
[8] I. Ramani and S. Savage, "Syncscan: practical fast handoff for 802.11 infrastructure networks," in *INFOCOM 2005*.
[9] M. Lu and J. Wu, "Localized Access Point Association in Wireless LANs with Bounded Approximation Ratio," in *ICCCN 2008*.
[10] H. Han, B. Sheng, C. C. Tan, Q. Li, and S. Lu, "A measurement based rogue ap detection scheme," in *INFOCOM*, 2009.
[11] G. Judd and P. Steenkiste, "Fixing 802.11 access point selection," *ACM SIGCOMM Computer Communication Review*, 2002.
[12] Cisco System Inc., "Data sheet for cisco aironet 1200 series," 2004.
[13] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley, "Facilitating access point selection in ieee 802.11 wireless networks," in *IMC 2005*.
[14] I. Papanikos and M. Logothetis, "A study on dynamic load balance for ieee 802.11b wireless lan," in *COMCON 2001*.
[15] S. Suri, C. Tóth, and Y. Zhou, "Selfish load balancing and atomic congestion games," *Algorithmica*, vol. 47, no. 1, pp. 79–96, 2007.
[16] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and economic behavior*, vol. 13, no. 1, pp. 111–124, 1996.
[17] E. Even-Dar, A. Kesselman, and Y. Mansour, "Convergence time to Nash equilibria," *Lecture Notes in Computer Science*, pp. 502–513, 2003.
[18] K. Mittal, E. Belding, and S. Suri, "A game-theoretic analysis of wireless access point selection by mobile users," *Computer Communications*, 2008.
[19] M. Cesana, I. Malanchini, and A. Capone, "Modelling network selection and resource allocation in wireless access networks with non-cooperative games," in *IEEE MASS*, 2008.
[20] Y. Bejerano, S.-J. Han, and L. Li, "Fairness and load balancing in wireless LANs using association control," *ACM/IEEE Transactions on Networking*, 2007.
[21] R. Murty, J. Padhye, A. Wolman, and B. Zill, "Designing high performance enterprise wi-fi networks," in *NSDI 2008*.
[22] R. Chandra, P. Bahl, and P. Bahl, "Multinet: connecting to multiple ieee 802.11 networks using a single wireless card," in *INFOCOM 2004*.
[23] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi, "Fatvap: Aggregating ap backhaul capacity to maximize throughput," in *NSDI 2008*.
[24] G. Matthew, *802.11 wireless networks: the definitive guide*, 2002.
[25] I. Caragiannis, "Better bounds for online load balancing on unrelated machines," in *SODA 2008*.
[26] "http://read.cs.ucla.edu/click/."
[27] "http://madwifi-project.org/."
[28] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005*.

Appendix

The proof of Theorem 3

Proof: Suppose client $1, 2, 3, \cdots, i, i+1, \cdots, n$ join the network sequentially. Consider the situation when client i is joining the network. Our protocol shows that client i will associate with AP h. Now let the load on AP j after client i associates with AP h be $\mathbb{L}_{ij}$ in our protocol (where h may not be j). In the optimal solution, however, client i may associate with AP k. Now at this point, if client i associates with AP k instead of h, the load on any AP will not decrease, but increase with a value (say δ_{ik}).

$\mathbb{L}_j^*$ is the load on AP j in the optimal solution for minimizing the $\mathbb{L}_p$ norm.

$$\begin{aligned} & \sum_j (\mathbb{L}_{ij}^p - \mathbb{L}_{i-1,j}^p) \\ \leq & \sum_j ((\mathbb{L}_{i-1,j} + \delta_{ij})^p - \mathbb{L}_{i-1,j}^p) \quad (4) \\ \leq & \sum_j ((\mathbb{L}_{n,j} + \delta_{ij})^p - \mathbb{L}_{n,j}^p) \quad (5) \end{aligned}$$

(4) is true because in this step client i tries to minimize the $\mathbb{L}_p$ norm. (5) is true because $(x+\delta)^p - x^p$ is increasing with x when $p > 1$ and $\delta > 0$.

$$\begin{aligned} \sum_j \mathbb{L}_{nj}^p &= \sum_i \sum_j (\mathbb{L}_{ij}^p - \mathbb{L}_{i-1,j}^p) \\ &\leq \sum_i \sum_j ((\mathbb{L}_{n,j} + \delta_{ij})^p - \mathbb{L}_{n,j}^p) \quad (6) \\ &= \sum_j \sum_i ((\mathbb{L}_{n,j} + \delta_{ij})^p - \mathbb{L}_{n,j}^p) \\ &\leq \sum_j ((\mathbb{L}_{nj} + \sum_i \delta_{ij})^p - \mathbb{L}_{nj}^p) \quad (7) \\ &= \sum_j (\mathbb{L}_{nj} + \mathbb{L}_j^*)^p - \sum_j \mathbb{L}_{nj}^p \\ &\leq ((\sum_j \mathbb{L}_{nj}^p)^{\frac{1}{p}} + (\sum_j \mathbb{L}_j^{*p})^{\frac{1}{p}})^P - \sum_j \mathbb{L}_{nj}^p \quad (8) \end{aligned}$$

where (6) is due to (5); (7) is due to the fact that $\sum_{i=1}^k ((x+\delta_i)^p - x^p) \leq (x + \sum_{i=1}^k \delta_i)^p - x^p$ for $p > 1$, $x > 0$, and $\delta_i > 0$; (8) is due to Minkowski Inequality. Then

$$2\sum_j \mathbb{L}_{nj}^p \leq ((\sum_j \mathbb{L}_{nj}^p)^{\frac{1}{p}} + (\sum_j \mathbb{L}_j^{*p})^{\frac{1}{p}})^P$$

Let $r = (\sum_j \mathbb{L}_{nj}^p)^{\frac{1}{p}} / (\sum_j \mathbb{L}_j^{*p})^{\frac{1}{p}}$. We then have $2r^p \leq (r+1)^p$ and $r \leq \frac{1}{2^{1/p}-1}$.

Thus, our protocol is a $r \leq \frac{1}{2^{1/p}-1}$-competitive online algorithm to minimize the L_p norm. ∎