SmartAssoc: Decentralized Access Point Selection Algorithm to Improve Throughput (Supplementary File)

Fengyuan Xu, *Member, IEEE,* Xiaojun Zhu, Chiu C. Tan, *Member, IEEE,*
Qun Li, *Member, IEEE,* Guanhua Yan, and Jie Wu, *Fellow, IEEE*

In this supplementary file, we examine the network performance in terms of competitive ratio and convergence if selfish strategy is applied, and provide detailed proofs of theorems stated in the main manuscript.

1 Selfish User Strategy

One natural alternative to solving the association 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 performs the 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.

- *F. Xu and Q. Li. are with the Department of Computer Science, College of William and Mary, Williamsburg, VA 23187. E-mail: {fxu,liqun}cs.wm.edu .*
- *X. Zhu is with the Department of Computer Science and Technology, Nanjing University, Jiangsu 210093, China. E-mail: gxjzhu@gmail.com .*
- *C.C. Tan and J. Wu are with the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122. E-mail: {cctan,jiewu}@temple.edu .*
- *G. Yan is with the Information Sciences Group, Los Alamos National Laboratory, Los Alamos, NM 87545. E-mail: ghyan@lanl.gov .*

Under these assumptions, we will show that such selfish user game converges to a Nash Equilibrium often having non-optimal performance. More complicated scenarios, even, cannot guarantee the existence of the Equilibrium state [1], [2].

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 where 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*[1](PoA) using 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; after that, it will give the competitive ratio of the protocol.

1.1 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 [1].

Given the Lemma 1, and assumptions we made, we describe this game 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

1. the ratio of the worst-case social cost, among all Nash Equilibria over the optimal cost

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 (i.e. the load imposed by the user) is defined as the reciprocal of the transmission rate, $\mathbb{L}_{ij} = \frac{1}{R_{ij}}$.

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 5 *$\overrightarrow{\mathcal{T}}$ 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. □

The above theorem is also applicable to the extended scenario that there are new users coming into the network.

Since we have shown that users' migration always increases the potential - $\overrightarrow{\mathcal{T}}$, this gives us an upper bound (Theorem 6) on the convergence time, in general.

Theorem 6 *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. □

We have shown that the network will finally converge to an equilibrium within bounded steps. However, the number of steps may be exponentially growing with respect to the network size, which is presented in the following Theorem 7. The basic idea is to send light weight users to the network first, which will incur a large amount of AP associations. Note that whenever an AP accepts a new user (maybe users from other APs), we count it as a new AP association.

Theorem 7 *For a network with m identical APs and m^2 users, there exists a scenario where the number of AP associations is at least 2^m.*

Proof: The basic idea is to construct a scenario where light weight users move first, which will incur a large amount of AP associations.

The network is as follows. The users are divided into m groups $G_1, G_2, \ldots, G_m$, each of which has m users. Users in the same group have the same weight. We set the weight of each group recursively. For G_1, its weight is 1. For G_i, its weight is set as more than the sum of the weights of previous groups' users, say $w_{G_i} = m\sum_{j=1}^{i-1} w_{G_j} + 1$. Such weight setting can guarantee that two APs have the same load, if and only if, they serve the same number of users from each group. In the equilibrium after all users join the network, each AP serves m users, each of which comes from a different group. Since each user's choice is deterministic, i.e., it chooses the lightest load AP (note here APs are identical), the number of AP associations is determined by the order in which users join the network, and the order in which they make adjustments. In the following, we construct an adversary that controls such orders to cause more than 2^m AP associations. Whenever the adversary *releases* a user, the user can join the network, or adjust APs, based on the best-reply policy.

Let $F(m)$ be the resulting number of AP associations for an m-AP network, under the adversary's strategy $\mathcal{S}(m)$ described as follows. We will prove $F(m) \geq 2^m$ by induction.

For $m = 2$, the adversary's strategy $\mathcal{S}(2)$ is to release users in an arbitrary order (e.g., light weight first). It

holds trivially that $F(2) \geq 2^2$ since there are 4 users, and each of them invokes at least one AP association.

Suppose $F(k) \geq 2^k$ and this is resulted from the adversary's strategy $\mathcal{S}(k)$. Consider the case when $m = k+1$. Now each group has $k+1$ users. The adversary will use the following strategy $\mathcal{S}(k+1)$. First, it applies the following strategy, creating a k-AP network subproblem.

- Release one user from G_{k+1}.
- Select k users from each G_i where $i = 1, 2, \ldots, k$, making k^2 users. Apply $\mathcal{S}(k)$ to them.

The user from G_{k+1} will occupy an AP. Denote this AP by AP_a. The later released k^2 users will only consider the other k APs, because even the sum of their weights is less than AP_a's current load. Therefore, these k^2 users, together with the k available APs, create a k-AP instance, and the number of involved AP associations is $F(k)$.

In the next, the adversary releases several users, whose associations we ignore, but will lead to an interesting equilibrium.

- Release the rest users from $G_1, G_2, \ldots, G_k$ (i.e., one user per group).
- Release $k-1$ users from G_{k+1}. Wait until equilibrium.

Consider the equilibrium. There must be k APs, each of which serves exactly one user from G_{k+1} and does not serve other users. All the $k(k+1)$ users from the first k groups are crowded in one AP (denote it by AP_b).

At last, the adversary creates another k-AP subproblem.

- Release the last user. It is from G_{k+1}, and it will choose AP_b.
- For users at AP_b, select k users from each G_i where $i = 1, 2, \ldots, k$, making k^2 users. Apply $\mathcal{S}(k)$ to them.

The released user from G_{k+1} will choose AP_b due to lightest load. After it associates with AP_b, all the other users at AP_b have the incentive to move to the other k APs. None of the selected k^2 users will associate back with AP_b at any time until the equilibrium, because AP_b has at least $k+1$ users (one user per group) so that there must be some AP that is less loaded. (Consider i from $i = k+1$ to $i = 1$. If there exists an AP serving two users from G_i, then there is at least one AP serving no user from G_i. This AP is less loaded than AP_b.) The k^2 users and the other k APs create a k-AP instance, contributing to $F(k)$ AP associations.

We can see that, under strategy $\mathcal{S}(k+1)$, the resulting $F(k+1)$ is composed of at least two $F(k)$, so we have $F(k+1) \geq 2F(k) \geq 2^{k+1}$. The theorem follows immediately. □

Note that this theorem considers identical APs, a special case of our model that assumes unrelated APs. The implications of the theorem are two-fold. First, it is impossible to bound the convergence time by polynomials of m and n. Second, some user may need to change AP exponentially many times ($2^m/m^2$), which significantly degrades performance.

1.2 Competitive Ratio

In the following, we obtain the competitive ratio with respect to the minimum 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 minimum available transmission rate is R_{min}. Recall that 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

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

This bounds the PoA from above. In the following, we bound the worst-case PoA from below.

Theorem 8 *There exists a scenario where the competitive ratio is no smaller than $(m-1)/(1+\epsilon)$ where m is the number of APs, and ϵ is a small positive constant.*

Proof: The example below first appears in [3], and recently in [4] for slightly different purposes (centralized greedy load balancing and sequential load balancing game). We show that this example also works in our context (distributed selfish user association).

Suppose there are m APs and $m-1$ users. Each user $i \in U$ can only associate with two APs, i and $i+1$. The weights are set as follows. For $i \in U$, $L_{i,i+1} = i$ and $L_{i,i} = 1+\epsilon$. For this example, the maximum load of the optimal solution is $1+\epsilon$.

The adversary can release users sequentially as $1, 2, \ldots, m-1$. It is easy to see that user i will associate with AP $i+1$ in the equilibrium, causing the maximum load to be $m-1$. Therefore, its competitive ratio is $(m-1)/(1+\epsilon)$. □

The theorem shows that the price of anarchy is $\Omega(m)$. In the following section, we use an online algorithm with competitive ratio $O(\log m)$, which is an exponential improvement.

Furthermore, the selfish strategy has worse performance if it is examined under the realistic wireless LAN model that SmartAssoc considers. There may not exist an equilibrium. To see this, we encode the classic game, matching pennies, in the model. Consider the 2-user and 2-AP scenario. Let $\delta_{1,1,1} = \delta_{1,2,2} = \delta_{2,1,2} = \delta_{2,2,1} = 2$,

	AP 1	AP 2
AP 1	3,3	4,2
AP 2	4,2	3,3

TABLE 1: The cost matrix for two users under different AP associations (left for user 1, and top for user 2). Each entry (a, b) is the cost for users 1 and 2 respectively.

and all others are set as 1. The resulting cost matrix is in Table 1. We can see that user 1 prefers to have the same choice as user 2 does, while user 2 prefers to differ. In distributed scenarios, they will keep switching APs forever.

In summary, the selfish user protocol has a high convergence time and a poor performance in some network scenarios.

2 Theorem Proofs

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:

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

□

Theorem 1 *To solve problem GAS, algorithm [5] is $(m+1)$-competitive, where m is the number of APs. The bound is almost tight, in that there exist a class of GAS instances that the algorithm gives m-competitive solutions.*

Proof: We first show how the algorithm works, then prove its competitive ratio, and then construct the set of bad input instances. The competitive ratio is proved by following the same procedure as in [5], where they proved the 2-approximation ratio for the traditional load balancing problem.

The algorithm performs binary search for the optimum load. In each iteration, it checks whether the optimal solution could yield maximum load T. To answer this question, it drops the integral restrictions and checks the feasibility of the following linear program.

$$\begin{cases} \sum_{i,k} x_{i,k}\delta_{i,j,k} \le T, & \forall j \in M \\ \sum_{k} x_{i,k} = 1, & \forall i \in U \\ x_{i,k} \ge 0, & \forall i \in U, k \in M \\ x_{i,k} = 0 & \text{if } \exists j \in M, \delta_{i,j,k} > T \end{cases}$$

It is easy to see that, if the optimal solution yields maximum load T, then the linear program above is feasible. Suppose the linear program is feasible. We can find an extreme solution $\mathbf{x}$ (a vertex of the polytope defined by the constraints). The algorithm then rounds fractional solution to integral assignment by finding a maximum matching for the fractionally assigned users (users correspond to fractional variables in $\mathbf{x}$). We relax the maximum matching requirement to an arbitrary semi-matching if no maximum matching exists.

We now prove the competitive ratio. We will prove that the rounding of the fractional extreme solution $\mathbf{x}$ produces an integer solution such that

$$\begin{cases} \sum_{i,k} x_{i,k}\delta_{i,j,k} \le T + mT, & \forall j \in M \\ \sum_{k} x_{i,k} = 1, & \forall i \in U \\ x_{i,k} \in \{0, 1\} & \forall i \in U, k \in M \end{cases}$$

The extreme solution $\mathbf{x}$ contains at most $m + n$ non-zero variables, among which the non-integer variables correspond to, at most, m users. To see this, note that by definition, an extreme solution must have mn^2 (the total number of variables) linearly independent constraints satisfied with equality. Among the constraints appeared in the linear program, at most $m+n$ of them could yield a non-zero solution (the m constraints for APs and the n constraints for users). Therefore, there are at most $m+n$ non-zero variables in $\mathbf{x}$. Since each user who corresponds to non-integer variables could contribute at least 2 non-zero variables, there are at most m such users by a counting argument.

Now we assign users with integer variables in $\mathbf{x}$ to the corresponding APs. After this assignment, the load at any AP is, at most, T. Then we assign users with non-integer variables. Recall that we assign fractionally assigned users to any machine that they have positive fractional assignment to, thus each fractionally assigned user contributes at most T load to each AP. Otherwise, we should have $x_{i,k} = 0$ due to the existence of j such that $\delta_{i,j,k} > T$. Given that there are at most m fractionally assigned users, the additional load to each AP can not exceed mT. Thus the claim is proved.

In the following, we construct the class of bad instances. On these instances, the algorithm gives m-competitive ratio (even with maximum matching strategy). Consider the network with the same number of

users and APs, i.e., $n = m$. We set

$$\delta_{i,j,k} = \begin{cases} s_{i,j} & \text{if } k = i \\ \frac{1-s_{i,j}}{m-1} & \text{if } k = i+1 \\ \infty & \text{otherwise} \end{cases}$$

where $s_{i,j}$ are to be defined later and $i+1$ is defined to be 1 for $i = m$. In other words, we set each user i to be able to associate with two APs, APs i and $i+1$. For $s_{i,j}$, we set

$$s_{i,j} = \begin{cases} 1 & \text{if } i + j \leq m+1 \\ 0 & \text{otherwise} \end{cases}$$

Now we have finished constructing the set of instances. We show an example of the inequalities imposed by APs for $m = 3$.

$$\begin{pmatrix} 1 & 0 & \infty & \infty & 1 & 0 & 0 & \infty & 1 \\ 1 & 0 & \infty & \infty & 1 & 0 & \frac{1}{2} & \infty & 0 \\ 1 & 0 & \infty & \infty & 0 & \frac{1}{2} & \frac{1}{2} & \infty & 0 \end{pmatrix} \begin{pmatrix} x_{1,1} \\ x_{1,2} \\ \vdots \\ x_{3,3} \end{pmatrix} \leq \begin{pmatrix} \mathrm{T} \\ \mathrm{T} \\ \vdots \\ \mathrm{T} \end{pmatrix}$$

For the constructed instance, the optimal solution is to assign user i to AP $i+1$, yielding maximum load 1. In the following, we show that the algorithm may assign user i to AP i, resulting in maximum load m, which is a m-competitive solution.

Set $T = m$. Consider the solution

$$x_{i,k} = \begin{cases} \frac{1}{m} & \text{if } k = i \\ \frac{m-1}{m} & \text{if } k = i+1 \\ 0 & \text{otherwise} \end{cases}$$

It satisfies with equalities all the non-zero constraints imposed by the APs and users. We can check that it is an extreme solution (note that it is the unique solution to the $2m$ non-zero constraints). Since the algorithm only requires an arbitrary maximum matching from the fractional assignment, we can see that setting $x_{i,i} = 1$ gives a maximum matching. □

Theorem 3 *If the protocol is to minimize the $\mathcal{L}_p$ norm of the loads (rather than to minimize the maximum load), then the online protocol gives a $r \leq \frac{1}{2^{1/p}-1}$-competitive ratio.*

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. Let $\mathcal{L}_{ij}$ be the resulting load of AP j if users 1 to i all follow our protocol (i may not select j). Let $x_{i,k}$ be the indicator variable for the optimal choice such that $x_{i,k} = 1$, if and only if, user i chooses AP k in the optimal solution. With a little abuse of notation, we let $\delta_{i,j} = \sum_k x_{i,k}\delta_{i,j,k}$. Let $\mathcal{L}_j^*$ be the load on AP j in the optimal solution for minimizing the $\mathcal{L}_p$ norm. We have $\mathcal{L}_j^* = \sum_i \delta_{i,j}$. If users 1 to $i-1$ follow our protocol, but user i chooses the optimal AP, then the load of the current system is $(\mathcal{L}_{i-1,1} + \delta_{i,1}, \mathcal{L}_{i-1,2} + \delta_{i,2}, \ldots, \mathcal{L}_{i-1,j} + \delta_{i,j}, \ldots)$.

Following the main idea of [6], we derive that

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

The first inequality is true because, in this step, client i tries to minimize the $\mathcal{L}_p$ norm. The second one is true because $(x+\delta)^p - x^p$ is increasing with x when $p > 1$ and $\delta \geq 0$.

$$\begin{aligned} &\sum_j \mathcal{L}_{nj}^p \\ = &\sum_i \sum_j \left(\mathcal{L}_{ij}^p - \mathcal{L}_{i-1,j}^p\right) \\ \leq &\sum_i \sum_j \left((\mathcal{L}_{n,j} + \delta_{ij})^p - \mathcal{L}_{n,j}^p\right) &(5) \\ = &\sum_j \sum_i \left((\mathcal{L}_{n,j} + \delta_{ij})^p - \mathcal{L}_{n,j}^p\right) \\ \leq &\sum_j \left(\left(\mathcal{L}_{nj} + \sum_i \delta_{ij}\right)^p - \mathcal{L}_{nj}^p\right) &(6) \\ = &\sum_j \left(\mathcal{L}_{nj} + \mathcal{L}_j^*\right)^p - \sum_j \mathcal{L}_{nj}^p \\ \leq &\left(\left(\sum_j \mathcal{L}_{nj}^p\right)^{\frac{1}{p}} + \left(\sum_j \mathcal{L}_j^{*p}\right)^{\frac{1}{p}}\right)^p - \sum_j \mathcal{L}_{nj}^p &(7) \end{aligned}$$

where (5) is due to (4); (6) 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 $p > 1$, $x \geq 0$, and $\delta_i \geq 0$;[2] (7) is due to Minkowski Inequality. Then

$$2\sum_j \mathcal{L}_{nj}^p \leq \left(\left(\sum_j \mathcal{L}_{nj}^p\right)^{\frac{1}{p}} + \left(\sum_j \mathcal{L}_j^{*p}\right)^{\frac{1}{p}}\right)^p$$

Let $r = (\sum_j \mathcal{L}_{nj}^p)^{\frac{1}{p}} / (\sum_j \mathcal{L}_j^{*p})^{\frac{1}{p}}$. 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. □

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 minimum throughput).*

Proof: Let the heaviest load among all APs running our protocol be $\mathcal{L}_m$, and the heaviest load among all APs in the optimal minimizing heaviest load protocol be $\mathcal{L}_m^*$. Thus, $m^{1/p}\mathcal{L}_m^* = (m \cdot \mathcal{L}_m^{*p})^{1/p} \geq (\sum_j \mathcal{L}_j^{*p})^{\frac{1}{p}} \geq \frac{1}{r}(\sum_j \mathcal{L}_{nj}^p)^{\frac{1}{p}} \geq \frac{1}{r}(\mathcal{L}_m^p)^{1/p} = \frac{1}{r}\mathcal{L}_m$. In other words, the

2. It can be proved by considering the function $f(s+t) - f(s) - f(t)$ where $f(s) = (x+s)^p - x^p$.

$\mathcal{L}_p$ norm protocol is a $rm^{1/p}$-competitive online algorithm for minimizing the heaviest load on all APs. $rm^{1/p} \leq \frac{m^{1/p}}{2^{1/p}-1} \leq m^{1/p}\frac{p}{\ln 2}$. When $p = \ln m$, $m^{1/p}\frac{p}{\ln 2}$ reaches its minimum value, $e \log m$, in the positive real number domain $\mathcal{R}^+$. Thus, we choose $\mathcal{L}_{\ln m}$ norm, and the competitive ratio, correspondingly, is $e \log m$. □

REFERENCES

[1] E. Even-Dar, A. Kesselman, and Y. Mansour, "Convergence time to Nash equilibria," *Lecture Notes in Computer Science*, pp. 502–513, 2003.

[2] I. Milchtaich, "Congestion games with player-specific payoff functions," *Games and economic behavior*, vol. 13, no. 1, pp. 111–124, 1996.

[3] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, "On-line routing of virtual circuits with applications to load balancing and machine scheduling," *J. ACM*, May 1997.

[4] R. P. Leme, V. Syrgkanis, and E. Tardos, "The curse of simultaneity," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM.

[5] J. K. Lenstra, D. B. Shmoys, and E. Tardos, "Approximation algorithms for scheduling unrelated parallel machines," *Math. Program.*, vol. 46, February 1990.

[6] I. Caragiannis, "Better bounds for online load balancing on unrelated machines," in *SODA 2008*.