

Association Control in Mobile Wireless Networks

Minkyong Kim, Zhen Liu, Srinivasan Parthasarathy, Dimitrios Pendarakis, Hao Yang

IBM T.J. Watson Research Center

19 Skyline Drive, Hawthorn, NY 10532

Email: {minkyong, zhenl, spartha, dimitris, haoyang}@us.ibm.com

Abstract—As mobile nodes roam in a wireless network, they continuously associate with different access points and perform *handoff* operations. However, these handoffs can potentially incur interruptions for interactive applications and increase the energy usage. To alleviate such negative impacts, we present novel association control algorithms that can minimize the frequency of handoffs occurred to mobile users. Specifically, we show that a greedy `LookAhead` algorithm is optimal in the offline setting, where the user’s future mobility is known. Inspired by such optimality, we further propose two online algorithms, namely `LookBack` and `Track`, that operate without any future mobility information. Instead, they seek to predict the lifetime of an association using randomization and statistical approaches, respectively. We evaluate the performance of these algorithms using both analysis and trace-driven simulations. The results show that the simple `LookBack` algorithm has surprisingly a competitive ratio of $(\log k + 2)$, where k is the maximum number of APs that a user can hear at any time, and the `Track` algorithm can achieve near-optimal performance in practical scenarios.

I. INTRODUCTION

As mobile users roam within a wireless infrastructure network, they continuously associate with different access points (APs) to sustain their wireless connectivity. The selection of user-AP association, known as *association control*, plays a key role in the Quality of Service perceived by the mobile users. For example, many wireless devices intend to associate with the AP that currently has the strongest signal [1] or the least load [2]. In addition to such local greedy decisions, there are advanced association control schemes for achieving load balancing among different APs and max-min bandwidth fairness among different users [3]. However, all these schemes focus on the network performance at each snapshot, yet the stability of the association decisions is largely overlooked. On the other hand, frequent association changes can result in not only unacceptable delays for interactive applications, such as VoIP, during the transition period, but also potential interruption of real-time communication services and subsequent system-component failures. To address these problems, it is highly desirable that the association decisions can lead to not only instantaneous high-quality links but also smooth connectivity over a long period of time.

There have been many research efforts to alleviate the disruptive impact of handoffs by reducing the latency of individual handoffs. Pack *et al.* [4] attempts to minimize the channel scanning time during a handoff, while other approaches, [5], [6], [7], attempt to minimize the handoff latency by propagating a mobile user’s context information (e.g., security and authentication information) proactively and aggressively

within the network. While these approaches provide significant savings in handoff *latencies* over naive methods, *frequent* handoffs could still be problematic for interactive applications even with latencies of the order of tens to hundreds of milliseconds.

In this paper, we present a new approach for association control that minimizes the handoff *frequency*; our approach complements the existing strategies for minimizing handoff *latency*. In typical wireless LAN deployments, a mobile device is often within the communication range of multiple APs, and can choose one of them to associate with at any point in time. By carefully selecting APs that are likely to maximize the duration of association, the device can significantly minimize the number of handoffs. Our main contribution is the *design, analysis, and evaluation of novel association control techniques which minimize the number of handoffs over time*. The specific contributions of our work are as follows:

First, we introduce the handoff minimization problem for wireless mobile devices. We consider both the *offline* and *online* settings of this problem; in the former setting, the future mobility pattern of the wireless device is known in advance, while in the latter setting, future mobility is unknown. To the best of our knowledge, we are the first to consider the problem of association control for wireless mobile devices with the goal of minimizing handoff frequency.

Second, we present `LookAhead`, an optimal algorithm for the off-line handoff minimization problem. Although the precise mobility patterns of wireless users may not be known in advance in practical scenarios, the `LookAhead` algorithm is significant due to two reasons: (1) The key insights from `LookAhead` guide us in the design of our online algorithms, which do not require any knowledge of users’ future mobility. (2) We use `LookAhead` as a natural baseline for the performance comparison of various association control algorithms.

Third, we present `LookBack`, an *online* algorithm which carefully applies randomization to select one AP among the currently available set of APs. We show that `LookBack` is not only computationally efficient, but also has an expected competitive ratio of $(\log k + 2)$, where k is the maximum number of APs available to the user at any time. This is indeed a non-trivial performance guarantee which cannot be obtained through other natural algorithms. For example, consider the natural strategy of choosing an AP *uniformly* at random from the set of currently available APs whenever a handoff needs to occur. This strategy has a competitive ratio of $\Theta(k)$ *instead of* $O(\log k)$.

Fourth, we present *Track*, an online heuristic for association control based on mobility history. The intuition behind *Track* is that if the mobile users exhibit repetitive mobility patterns (or pseudo-periodicity) over long periods of time, then such patterns can be efficiently leveraged for intelligent AP selection in order to minimize the frequency of handoffs. This is indeed a common scenario in practice and confirmed by our simulation studies that indicate the performance of *Track* is near-optimal in such scenarios.

To evaluate our proposed algorithms, we perform *trace-driven* simulations based on real data set containing associations between a large set of mobile users and APs. The traces used were collected at the University of California, San Diego over a period of 11 weeks and record the associations, as well as the set of APs heard by mobile users in regular 20 second intervals. Our simulation shows that the trace contains twice as many handoffs as that can be achieved by *LookAhead*, the off-line optimal algorithm. This suggests a big opportunity for improvement. Indeed our *LookBack* and *Track* algorithms both outperform the default algorithm used in the trace, and *Track* is close to optimal. These results indicate that our techniques can significantly reduce the number of connectivity interruptions with small additional complexity.

This paper is organized as follows. In Section II, we present an overview of the handoff minimization problem. In Section III, we describe the off-line optimal algorithm, *LookAhead*. We then present two online algorithms, *LookBack* and *Track*, in Section IV. In Section V, we present the results of trace-driven simulations of our algorithms, as well as their comparison with the existing approach of the trace. We survey related work in Section VI, and present our concluding remarks in Section VII.

II. THE HANDOFF MINIMIZATION PROBLEM

In this paper, we consider a wireless network which provides ubiquitous network access for mobile devices. Many organizations, companies and universities have already deployed 802.11-based wireless networks inside their campuses. In the near future, wireless mesh networks promise to provide even larger coverage, e.g., in a metropolitan area or even an entire city. Due to the limited transmission range of 802.11 devices, these networks consist of a large number (e.g., hundreds or even thousands) of APs. Moreover, in most deployment scenarios, these APs provide redundant coverage for performance and robustness. As such, at any given time, a mobile device may be within the transmission range of multiple APs, and the device must choose one of them to associate so that it can properly send and receive traffic. This AP selection mechanism is called *association control*, which is typically implemented in the device driver.

There is no specific association control scheme specified in the 802.11 standards, thus the vendors can freely implement their own solutions, which are typically based on perceived signal strengths [1]. For example, a device can stay with the currently associated AP until its signal strength drops below a certain threshold. Then the device re-associates with the

AP that has the strongest signal at that time. Since signal strength can fluctuate due to many factors such as path fading, obstacles and mobility, the devices may experience frequent handoffs. During each association change, a handoff protocol must be invoked among the device, the previously used AP and the new AP, so that the device can keep its current application sessions alive. Unfortunately, this handoff process can take hundreds of milliseconds or even a few seconds (with 802.1X authentication in place) [6] which presents noticeable interruption for interactive applications such as VoIP. While there have been many research efforts on reducing the handoff latency [4], [5], [6], [7], these schemes often require changes to the 802.11 standards, thus facing significant deployment hurdles.

In this paper, we take a standard-compliant approach and develop novel association control schemes to minimize the number of handoffs that occur to a mobile device. In what follows, we formally define the handoff minimization problem and comment on its practical implications.

Problem Formulation: An instance of the handoff minimization problem is defined as follows. We are given a mobile user and a collection of APs. Time is divided into discrete slots $1, 2, 3, \dots, T$. At each slot t , the user is given a candidate set of APs, $C(t)$, that can satisfy the user's minimum quality-of-service requirement (e.g., a minimal signal strength threshold)¹. The user must *associate* with exactly one AP in $C(t)$ at the beginning of slot t . For any AP $a \in C(t)$ we say that a is available to the user at slot t . If the AP associated with the user at slot $t + 1$ is different from the one she associated with during slot t , then the user is said to have undergone a *handoff* at slot $t + 1$. We need to design an association control mechanism that chooses an AP for each slot. Our objective is to minimize the total number of handoffs that the user undergoes over all slots.

We consider both offline and online settings of the handoff minimization problem. In the offline setting, we are given the $C(t)$ values for each slot t as part of the problem input; i.e., the candidate set of APs at each slot is revealed *upfront*. Intuitively, the offline version models the setting where the mobility pattern of the user is known in advance. In the online setting, each $C(t)$ value is revealed only during slot t ; at every slot t , we need to *instantaneously* select an AP in the set $C(t)$ and associate it with the user, *without* any knowledge of the future sets: $C(t + 1), C(t + 2), \dots, C(T)$.

Competitive Ratio: We use the notion of *competitive ratio under the oblivious adversary model* in analyzing the performance of an online algorithm. Given a problem instance \mathcal{I} , let $C(\mathcal{I})$ denote the expected cost incurred by the *online* algorithm; here, the expectation is taken over (possibly) the internal random choices made by the algorithm itself. Let $OPT(\mathcal{I})$ denote the cost incurred by the optimal *offline* algorithm for \mathcal{I} . The competitive ratio of the online algorithm

¹In other words, before invoking the association control mechanism, the user may first filter out those APs with poor performance, measured by signal strengths, current loads, etc. However, such application-specific filtering policies are out of the scope in this work.

is the maximum possible value of the ratio $\frac{C(\mathcal{I})}{OPT(\mathcal{I})}$, where the maximum is taken over all possible instances of the problem.

In the competitive ratio analysis of LookBack, due to a minor technicality, we define the cost of an algorithm as the number of associations performed by the algorithm, rather than the number of handoffs. Note that at the very first slot, an association occurs but no handoff occurs. However, at every other slot, a handoff occurs whenever an association occurs. Hence, the number of associations is exactly one plus the number of handoffs in any algorithm. We also note that the mobile device obtains the candidate AP set $C(t)$ by scanning the wireless channel at time t and listening for beacons from nearby APs. In reality, the device may scan the channel only when the currently associated AP becomes unavailable or its signal strength drops below a threshold. Hence, the next time instant when a channel scan occurs is itself a function of the current AP association.

Solution framework: A mobile node can initiate a scanning whenever the channel condition (signal strength) with the currently associated AP, s_c , becomes equal to or less than a certain threshold, T_s . For example, if T_s is set to zero, the node will scan only when the currently associated AP can no longer be reached. If T_s is set to infinity (or a very large value), the node can scan the channels continuously or periodically.

When the channel condition with the currently associated AP, s_c , falls below a certain threshold, T_a , the node tries to switch to another AP. (Note that the threshold for handoff, T_a , can be different from the threshold for scanning, T_s .)

Whenever the signal strength with the currently associated AP, s_c becomes less than or equal to T_s or T_a , the node scans wireless channels to discover the set of currently available APs ($C(t)$), and calls an association control function, which returns an AP that the node should associate with; the association control algorithms are described in the following sections.

III. LOOKAHEAD: OPTIMAL OFFLINE ALGORITHM

We now describe algorithm LookAhead, an optimal algorithm for the offline handoff minimization problem. Consider a slot t . Suppose the user was associated with AP ‘a’ during the *previous* slot; if this AP is available during slot t , then the user continues its previous association with ‘a’ during this slot. In this case, no new association or handoff is needed at slot t . Otherwise, if ‘a’ is unavailable during slot t , LookAhead selects an AP from the set $C(t+1)$ as follows. For any AP, define its duration to be the number of slots for which it is *contiguously* available starting from slot $t+1$. For example, if AP b is available during slot $t+1, t+2, t+3, t+5$, but not $t+4$, then b has a duration of three. LookAhead selects an AP in the set $C(t+1)$ with the maximum duration and associates the user with this AP during slot $t+1$ (if there are multiple APs with the maximum duration, one of them is selected arbitrarily). We observe that, with the exception of the first slot, whenever a new association occurs it results in a handoff. At the first slot, by definition, an association occurs but not a handoff. The following theorem proves the optimality of LookAhead.

Theorem 1: Given an instance of the handoff minimization problem, let OPT denote the number of handoffs in algorithm LookAhead. Every algorithm incurs at least OPT handoffs for this instance.

Proof: Assume the contrary. Suppose there exists an algorithm \mathcal{A} which incurs strictly less than OPT handoffs. Define $length(i)$ as the number of slots for which association i persisted in algorithm \mathcal{A} . Specifically, $length(i)$ is the number of slots that elapsed in algorithm \mathcal{A} starting from the i^{th} association until the $i+1^{st}$ association. For example, consider slots 1, 2, 3, 4, 5, 6, 7, and 8; suppose the user associated with access points $a_1, a_1, a_2, a_2, a_2, a_3, a_1$ and a_1 during these slots respectively in algorithm \mathcal{A} ; then the first association occurred at the first slot², the second association occurred at the third slot, the third association occurred at the sixth slot, and the fourth association occurred at the seventh slot. Further, $length(1) = 2, length(2) = 3, length(3) = 1,$ and $length(4) = 2$. Analogously, let $optlength(i)$ denote the number of slots that elapsed in algorithm LookAhead between its i^{th} association until its $i+1^{st}$ association.

Clearly, the total length of all associations in any algorithm is equal to T , which is the total number of slots in the problem instance. The average length of an association in any algorithm is ratio of T and the total number of associations that occurs in that algorithm. The total number of associations in any algorithm is one more than the number of handoffs performed by that algorithm. Since LookAhead performs OPT handoffs, and algorithm \mathcal{A} performs strictly less than OPT handoffs, the average length of an association in Algorithm \mathcal{A} is strictly greater than the average length of an association in Algorithm LookAhead. Hence, there exists an integer $i \geq 1$ such that $length(i) > optlength(i)$. Let $i_{\min} \geq 1$ denote the minimum integer such that $length(i_{\min}) > optlength(i_{\min})$. Since $length(j) = optlength(j)$ for any $j < i_{\min}$, the i_{\min}^{th} association occurred during the same slot t in both algorithm \mathcal{A} and LookAhead. Hence, the candidate set of APs available for the user during the i_{\min}^{th} association is the same in both the algorithms. Since LookAhead always chooses the AP with the maximum duration $optlength(i_{\min}) \geq length(i_{\min})$ which is a contradiction. This completes the proof of the theorem. ■

IV. ONLINE ALGORITHMS

To minimize the number of handoffs, it is important to understand and predict users’ mobility patterns. If we have accurate knowledge about the path of a user and the set of access points (APs) available to the user, we can obtain an optimal handoff schedule using the algorithm presented in Section III. However, in practice, we do not know users’ future mobility patterns. Therefore, we have developed two online algorithms, LookBack and Track, which use the past trajectory of individual users to intelligently select the next AP. Two algorithms are different in how they use the past trajectory information.

²By definition, the first association always occurs during the first slot

A. LookBack: Randomized Online Algorithm

We now describe algorithm `LookBack`, a provably good algorithm for the *online* handoff minimization problem. Recall that an online algorithm needs to determine the association between the user and the APs during each slot t , *without* any knowledge of the future sets $C(t+1), C(t+2), \dots, C(T)$. Algorithm `LookBack` broadly works as follows. For each slot t , `LookBack` maintains a set of APs $B(t)$ which is a *subset* of the candidate set $C(t)$. Suppose the user was associated with AP ‘a’ during the *previous* slot; the user continues its previous association with ‘a’ during slot t if this AP is available during slot t . In this case, no new association or handoff is needed at slot t . Otherwise, if ‘a’ is unavailable during slot t , then `LookBack` selects an AP from the set $B(t)$ *uniformly at random* and associates the user with this AP.

We now complete the description of `LookBack` by specifying how $B(t)$ is updated during slot t . During the first slot, $B(1)$ is set to $C(1)$ which is the candidate set of APs available during the first slot. At time t , if $B(t-1) \cap C(t) \neq \Phi$, i.e., if the intersection between the current $B(\cdot)$ and the candidate set $C(t)$ is non-null, then $B(t) = B(t-1) \cap C(t)$. Else, if this intersection is null, then $B(t) = C(t)$. Intuitively, $B(t)$ is the set of APs that are available currently and that were in the set $B(t-1)$; if there are no such APs, then $B(t)$ is the set of APs that are available currently. We emphasize that `LookBack` selects a random AP in the set $B(t)$ at slot t if and only if the AP with which the user was associated during the previous slot is unavailable during slot t . Let k denote the maximum number of APs available to the user at any slot. We now prove a surprising result that the competitive ratio of Algorithm `LookBack` is at most $2 + \log k$.

1) *Analysis:* We define a *marking* process for algorithm `LookBack` as follows. Recall the rules for updating the set $B(t)$ in `LookBack`. During the first slot, $B(1)$ is set to $C(1)$ which is the candidate set of APs available during the first slot. We will always *mark* the first slot. At slot t , if $B(t-1) \cap C(t) \neq \Phi$, i.e., if the intersection between the current $B(\cdot)$ and the candidate set $C(t)$ is non-null, then $B(t) = B(t-1) \cap C(t)$. Else, if this intersection is null, then $B(t) = C(t)$. In this latter case, we mark slot t . The following claim holds.

Claim 2: Let OPT denote the optimal number of associations for the given instance. Let ℓ denote the number of slots marked by algorithm `LookBack` for this instance. Then, $OPT \geq \ell$.

Proof: Consider any two slots t_1 and t_2 that were marked successively by `LookBack`. By definition of the marking process, there does not exist any AP which is available contiguously during all the slots t_1, t_1+1, \dots, t_2 . Hence, in any algorithm, the AP associated with the user during a marked slot is different from the AP associated with the user during the previously marked slot. Thus, the minimum number of associations needed for the instance is at least the number of marked slots, which is ℓ . This proves the claim. ■

Claim 3: Consider any two slots t_1 and t_2 that were marked successively by `LookBack`. The expected number

of associations performed by `LookBack` during the slots t_1, t_1+1, \dots, t_2-1 is at most $2 + \log k$.

Proof: We first observe that by definition of the marking process, there exists at least one AP which is available during all the slots t_1, t_1+1, \dots, t_2-1 , but there is no AP which is available during all these slots as well as during slot t_2 . Since slot t_1 was marked, we have $B(t_1) = C(t_1)$. Let $m = |B(t_1)|$. As in the algorithm `LookAhead`, define the duration of an AP in the set $B(t_1)$ to the number of contiguous slots for which the AP is available starting at slot t_1 . Let a_1, a_2, \dots, a_m denote the ordered list of APs in $C(t_1)$ sorted in the decreasing order of their durations. If the random AP in the set $C(t_1)$ that is first associated with the user luckily turns out to be a_1 , then this AP will be available until time t_2 and no more associations are required until then.

More generally, let α denote the number of APs in the set $C(t_1)$ with the maximum duration. The user first selects a random AP $a_{i_1} \in B(t_1)$ and associates with a_{i_1} . This association lasts for d_{i_1} slots, where d_{i_1} is the duration of a_{i_1} . Crucially, the update rule for the set $B(\cdot)$ ensures that at time $t = d_{i_1} + 1$, when the user needs to associate with a new AP, *none* of the APs with duration $\leq d_{i_1}$ will be part of the set $B(t_1 + d_{i_1})$. Hence, the second AP a_{i_2} which the user associates with is selected at random among all APs in the set $C(t_1)$ whose durations are strictly greater than d_{i_1} ; the third AP is a random choice among those APs in the set C_{t_1} whose durations are strictly greater than d_{i_2} , and so on. This selection process terminates when the user associates with one of the first α APs $a_1, a_2, \dots, a_\alpha$ which possess the maximum duration. From the above description, the randomized AP selection process can be seen to be analogous to the randomized binary search algorithm. We now show using an analysis similar to that of randomized binary search that the expected number of associations performed by the user until she selects one of the α APs in the set $C(t_1)$ with the maximum duration is at most $2 + \log(k)$.

Let $Z(j)$ denote the expected number of associations performed by the user until she selects one of first α APs, *given that only the j APs with the longest duration are currently in the set $B(\cdot)$* . We have,

$$Z(j) = 1 \quad (\text{if } j = \alpha) \quad (1)$$

$$Z(j) \leq \frac{\alpha}{j} + \sum_{q=\alpha+1}^j \frac{1 + Z(q-1)}{j} \quad (\text{if } j > \alpha) \quad (2)$$

The rationale behind Eqn. (1) is as follows: if the current APs in the set $B(\cdot)$ are exactly the first α APs with the longest duration, then with probability one, the association process will terminate in a single step. Hence, $Z(j) = 1$ in this case. The rationale behind Eqn. (2) is as follows: when $j > \alpha$, we have two scenarios; in the first scenario, with probability $\frac{\alpha}{j}$, the user’s random choice will coincide with the first α APs and $Z(j) = 1$. In the second scenario, with probability $\frac{1}{j}$ the user will choose a fixed AP among the last $j - \alpha$ APs; if the user chooses the AP a_q for some $q \in \alpha + 1, \dots, j$, then this incurs one association; further, during the slot at which the

next association occurs, there are at most $q - 1$ APs in the set $B(\cdot)$ and the expected number of associations at that stage is at most $Z(q - 1)$.

Given Eqns. (1) and (2), we now show using induction that:

$$\forall j \geq \alpha: Z(j) \leq 1 + \frac{1}{\alpha} + \frac{1}{\alpha+1} + \dots + \frac{1}{j-1} \quad (3)$$

W.l.o.g., we will first assume that for all j , Eqn. (2) holds with equality instead of \leq . Hence, we have:

$$Z(j) = \frac{\alpha}{j} + \sum_{q=\alpha+1}^j \frac{1+Z(q-1)}{j} \quad (\text{if } j > \alpha) \quad (4)$$

Eqn. (3) clearly holds for $j = \alpha$ from Eqn. (1). It is also easy to verify that Eqn. (3) holds for $j = \alpha + 1$. Let $u \geq \alpha + 1$. Assume that Eqn. (3) holds for all j in α, \dots, u . We now prove that Eqn. (3) holds for $j = u + 1$. By Eqn. (4), we have: $Z(u + 1) = \frac{\alpha}{u+1} + \sum_{q=\alpha+1}^{u+1} \frac{1+Z(q-1)}{u+1} = \frac{\alpha}{u+1} + \frac{u}{u+1} \cdot \frac{1}{u} \sum_{q=\alpha+1}^u (1 + Z(q-1)) + \frac{1+Z(u)}{u+1} = \frac{\alpha}{u+1} + \frac{u}{u+1} \cdot (Z(u) - \frac{\alpha}{u}) + \frac{1+Z(u)}{u+1} = \frac{1}{u+1} + Z(u)$.

The induction claim follows by summing up the Harmonic series yielded by the above expression. The summation $\frac{1}{\alpha} + \frac{1}{\alpha+1} + \dots + \frac{1}{j-1}$ is at most $1 + \log(j - 1)$ for any $j > 1$; further, k is the maximum number of APs available to the user during any slot and hence j is at most k . Hence, the claim follows. ■

Theorem 4: Given an instance of the handoff minimization problem, let R denote the number of associations performed by algorithm `LookBack`, and let OPT denote the optimal number of associations required for this instance. We have $\mathbf{E}[R] \leq (2 + \log k) \cdot OPT$.

Proof: By Claim 2, $OPT \geq \ell$, where ℓ is the number of slots marked by `LookBack`. By Claim 3, $R \leq (2 + \log k) \cdot \ell$. These facts together prove the theorem. ■

B. Track: Mobility-Based Algorithm

We now describe our mobility-based algorithm, `Track`. Figure 1 shows an example scenario with two mobile wireless nodes moving through an intersection. Initially, Node 1 scans wireless channels and hears access point A and B, and associates with B. As it enters the intersection, it loses the connectivity to B. It scans again and discovers two APs: E and F. Given Node 1's trajectory, Node 1 can be associated longer with E than F, and thus should choose E. Node 2 in Figure 1 initially hears C and D, and associates with D. As it loses the connectivity, it scans and discovers E and F. Given its trajectory, it can be associated longer with F than E, so it should choose F. Note that even though both Node 1 and Node 2 heard E and F at the intersection, Node 1 should choose E and Node 2 should choose F. The optimal AP is different because their trajectories are different.

Since the set of APs available to a node depends on the node's location, we use it to represent the node's location. We then use the list of sets of APs to denote the node's trajectory. Our `Track` algorithm predicts future trajectories using the user's past movement trajectory. Given a predicted trajectory

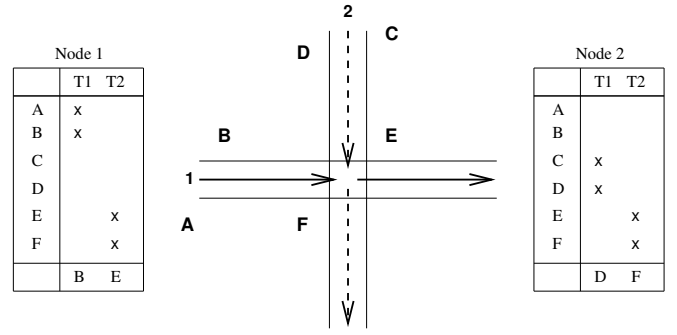


Figure 1. Example AP association scenario. Solid arrows represent trajectory of Node 1, and dotted arrows denote that of Node 2. 'A' through 'F' show the APs. In the tables, 'x' shows the AP that the node heard at each scan, and the last row shows the AP with which the node associated.

and a set of currently available APs, `Track` determines the AP with which the node may be associated longest. We describe below how `Track` maintains the user's movement trajectory (i.e., *states*) and expected association durations if a node were to choose a particular AP among the available ones.

`Track` defines a state to be a list of available AP sets. The algorithm with history of n uses n sets, starting from the most current set, $C(t)$, to define state $ST(p)$:

$$ST(p) = (C(t), C(t-1), \dots, C(m)) \quad (5)$$

where $m = \max[t - n, 0]$ and p is the state identifier. Each state represents a unique trajectory. For convenience of description, we call $C(t)$ of $ST(p)$ as V_p . For each state $ST(i)$, we maintain the expected association duration of APs in V_i . If the available AP at scan $t-1$ continues to be available at scan t , we add the elapsed time between two scans to the *session* association duration of that AP. However, if the node no longer hears the AP at t , we do not know exactly when the AP became unavailable; it could have been anytime between the two scans. In this case, given no other knowledge about the distribution of the association times, we assume that the expect time that the node continued to hear the AP is equal to half the elapsed time between the two scans. At scan t , we need to update session duration of the APs not only in the current state, but also all other states (called *liveStates*) whose APs have been continuously available up to scan $t-1$. A state, $ST(j)$, is removed from the *liveStates* set only when none of its APs in V_j is in $C(t)$.

Once an AP stops from being continuously available, we update the *expected* association duration with the session duration. To compute the expected duration, we use the exponentially weighted moving average (EWMA):

$$E_t = \frac{1}{4}D_t + \frac{3}{4}E_{t-1} \quad (6)$$

where E_t is the expected duration and D_t is the session duration. In summary, the session association duration is updated after each scan, while the expected association duration is updated only when an AP is no longer available.

Consider an example scenario shown in Table I. This example illustrates how the `Track` algorithm with history of

	T0	T1	T2	T3
A	×		×	×
B		×	×	
C	×	×	×	×
	↔ ST(1) ↔			
		↔ ST(2) ↔		
			↔ ST(3) ↔	

Table I

TRACK-1: EXAMPLE SCENARIO FOR STATE UPDATES. ‘A’, ‘B’, AND ‘C’ DENOTE APs, AND ‘×’ SHOWS THE APs DETECTED AT EACH SCAN. ST(1), ST(2), AND ST(3) REPRESENT STATES.

State	AP	T1	T2	T3
1 (B,C),(A,C)	B	0	20	30
	C	0	20	40
2 (A,B,C),(B,C)	A		0	20
	B		0	10
	C		0	20
3 (A,C),(A,B,C)	A			0
	C			0

Table II

TRACK-1: STATE UPDATES OVER TIME. THIS TABLE SHOWS HOW THE EXPECTED ASSOCIATION DURATIONS FOR APs ARE UPDATED OVER TIME FOR THE EXAMPLE SCENARIO ILLUSTRATED IN TABLE I. WE ASSUME THAT THE ELAPSED TIME BETWEEN EACH SCAN IS 20 SECONDS.

1 (Track-1) updates the session association duration for a mobile node. Initially at T0, the node hears A and C. At T1, it discovers B and C. At this point, ST(1) can be defined using $C(1)=(B,C)$ and $C(0)=(A,C)$. ST(2) and ST(3) are defined similarly. Table II shows how session association durations are updated for this particular example. Assume that the elapsed time between each scan (or time slot) is 20 seconds. ST(1) keeps track of the expected duration of B and C. At T2, B and C are still available, so we add 20 seconds to both duration in ST(1). At T3, we add 10 seconds to B since it is no longer available, while we add 20 to C since it is still available. The session duration in other states is maintained in the same manner. Note that at each scan, we update the durations not only in the current state, but also in other states that contain APs that have been available continuously.

In our current implementation, states do not expire; we assume that we have an unlimited storage space for the states. This assumption was reasonable since the number of new states converges to a constant after a certain period of time (as shown later). However, for real deployment of this algorithm, we can imagine using a limited storage space for maintaining states and setting an expiration time for each state.

Algorithm 1 shows the pseudo code of the TRACK algorithm. Given the set of currently available APs, P_t , this function returns the AP with the longest expected duration among P_t .

V. EVALUATION

We have evaluated the performance of our proposed association control algorithms using trace-driven simulations. The results show that our algorithms can significantly reduce the number of handoffs that occur to a mobile device. We first describe our evaluation methodology in Section V-A and then present the simulation results in Section V-B.

Algorithm 1 TRACK(P_t)

Require: P_t : available APs

```

for  $A \leftarrow$  each AP in  $P_t$  do
  for  $S \leftarrow$  each state in  $A$ 's liveStates do
    UpdateSessionDuration( $A,S$ ,elapsedTime)
  end for
end for
for  $B \leftarrow$  each AP in  $P_{t-1} - P_t$  do
  for  $T \leftarrow$  each state in  $B$ 's liveStates do
    UpdateSessionDuration( $B,T$ ,elapsedTime/2)
    UpdateExpectedDuration( $B,T$ ) /* Eq. 6 */
  end for
end for
if  $P_t$  contains currentlyAssociatedAP then
  return currentlyAssociatedAP
end if
for  $C \leftarrow$  each AP in  $P_t$  do
  UpdateExpectedDuration( $C$ ,CurrentState)
end for
return AP with the longest expected duration among  $P_t$ 

```

A. Methodology

In order to evaluate the performance of our algorithms in a realistic setting, we drove our simulations by an extensive set of mobility trace collected from 275 college-freshman users over an 11-week period at UCSD [8]. In this trace, each user carried a PDA equipped with a Symbol Wireless Networker 802.11b card and freely roamed in the UCSD campus, which provided extensive 802.11b coverage. When the PDA was powered on, it sampled and recorded every 20 seconds the list of APs it overheard, as well as their respective signal strengths. Among these APs, the device also recorded the AP with which the device associated.

While the trace was originally collected for the purpose of wireless topology discovery, it naturally provides a good sample for our association control study because it contains all necessary information needed for the operation of our algorithms. In particular, at any given time, the trace provides not only the set of APs that a device can be potentially associated to, but also the history of how long the device has stayed in each AP's vicinity. Moreover, the trace also provides realistic mobility profiles for a campus environment, which are critical in understanding the impact of handoffs on these mobile users.

During our study, however, we also found several limitations in the trace. First, the trace contains many gaps where no samples are available for an extended period of time. This may be because the device had run out of battery or the user simply turned the device off. Second, the device may temporarily lose its connectivity and may not be associated with any AP even though it could hear at least one AP. From the trace itself, the reasons for such phenomena are not clear. Third, each line in the trace denotes a reading for a particular user for one of the APs that the user heard during a scan. Although each scan

is supposed to be 20 seconds apart, some readings (or lines of traces) are only several seconds apart. We expect that this is mostly due to clock synchronization issues. To address this problem, we considered consecutive trace lines that are less than or equal to 3 seconds apart to be readings from one scan.

To ensure a meaningful and fair comparison, we pre-processed the trace as follows before we fed it into our simulator. First, we removed all samples in which the device was not associated with any AP. In other words, we ignored the intervals of connection loss in the trace. Secondly, whenever the samples were missing for 30 minutes or longer, we assumed that the device was turned off, and we did not count its first association after the wakeup as a handoff. We chose 30 minutes since it is often the default duration after which APs disassociate users if they have heard no messages from the users.

For each user, our simulator for on-line algorithms—LookBack and Track—reads in each scan, which happened every 20 seconds, and counts the number of handoffs that happened during individual days. For Track algorithm, we use the history of 1 and history of 0, denoted by Track-1 and Track-0, respectively.

Because the UCSD trace was collected to include as much information as possible for future studies, it contains the set of APs that each mobile device scanned every 20 seconds. However, in reality, periodic scanning consumes power and, therefore, is not desirable for battery-powered mobile devices. To apply our Track algorithm to realistic settings, we also simulated our Track algorithm with limited scans, instead of with every scan in the trace. We pretended that the mobile device scans only when it needs to change its associated AP. In our simulation, we included Track algorithm with limited scans using history of 1 and 0: Track-1S and Track-0S.

For comparison purposes, we also counted the number of handoffs that occurred in the trace, which indicates the performance of the particular wireless cards in use. These cards periodically scan the channels and re-associate with the AP that has the strongest signal, unless the card is configured otherwise [8]. Such signal-based heuristics are also used by many existing WLAN devices on the market.

In practice, a user may lose her wireless connectivity due to moving to an area without coverage. She may then later roam to another location with connectivity and re-connect to the network. Since handoff from one AP to another cannot happen in this scenario, we do not count the re-connection as handoff.

We evaluate the performance of different algorithms on a *user-day* basis. Our primary metric is the number of handoffs that occurred for a particular user in a particular day. To assess the benefits of our algorithms for highly mobile users, we also extracted a subset of *mobile user-days* from the trace. A mobile user-day is defined as a user-day in which at least 20 handoffs occurred in the trace. As we shall see, such mobile user-days typically reflect a higher degree of mobility, and hence benefit more from our algorithms, as compared to the rest of the trace.

Algorithms	All user-days	Mobile user-days
Trace	5.99	44.89
Optimal	3.09	22.62
LookBack	4.28	28.41
Track-1	3.62	26.15
Track-0	3.65	26.19
Track-1S	3.82	24.44
Track-0S	3.89	24.77

Table III
AVERAGE NUMBER OF HANDOFFS FOR DIFFERENT ALGORITHMS.

B. Simulation Results

The UCSD trace contains 7028 unique user-days. Out of 7028, only 613 user-days (8.7%) have handoffs greater than or equal to 20. This low percentage for mobile users is not surprising and goes along with the analysis presented by Henderson *et al.* [9] that users are not mobile most of time. For Track algorithm with limited scans, we have 6716 user-days, out of which 606 are mobile user-days. The reason that these user-days are less than ones in the original trace is because we assume that if a node does not need to change the currently associated AP, it does not scan wireless channels and therefore some user-days trace are considered to be empty; these omitted user-days are stationary days since there were no need for new scans.

To understand the overall performance, we computed the average number of handoffs for different algorithms, shown in Table III. Not surprisingly, Optimal generated the minimum average number of handoffs. The Track algorithms performed closed to optimal, while LookBack performed worse than Track but still better than Trace. Among variations of Track algorithms, history of 1 performed better than history of 0 for both the original Track algorithm and the Track algorithm with limited scans. This performance improvement for history of 1 comes in added complexity, which we will analyze below. For all user-days, the original Track (Track-*) outperformed the Track with limited scans (Track-*S) as expected. However, the difference is small. For mobile user-days, Track-*S outperformed Track-*; we currently do not have a clear explanation for this particular result.

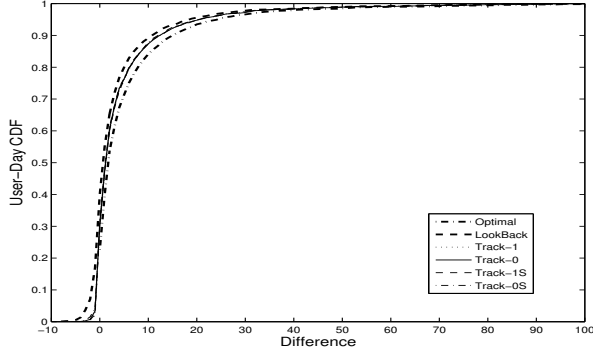
Table IV shows the average number of scans per user-day. From the trace where scanning was performed every 20 seconds, the average is 1214.3 scans per user-day. If devices scan only when they need to handoff, the average falls down to 6.7 scans per user-day. Although periodic scanning is commonly used in many 802.11-based localization systems [10], we expect that aggressive scanning nonetheless reduces the battery life of mobile devices. In the future, we plan to explore the power consumption of scanning to better understand overhead of our approaches. Another concern in aggressive scanning is its effect on data transmission [11]. In light of these concerns, it is significant that the performance of the Track algorithm using only limited scans is not much different from the Track algorithm with periodic scans. Thus, Track with limited scans is a better choice for battery-powered mobile devices.

While the average handoffs provides a good summary of

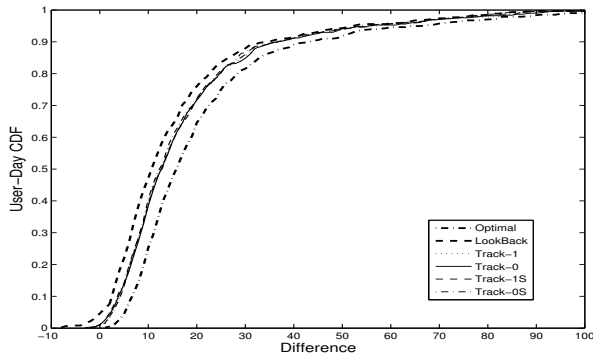
	Average scan count
All scans	1214.3
Limited scans	6.7

Table IV

AVERAGE NUMBER OF SCANS PER USER-DAY. THE FIRST ROW SHOWS THE NUMBER OF SCANS IN THE TRACE (USED BY THE ORIGINAL TRACK ALGORITHM), AND THE SECOND ROW SHOWS THE NUMBER OF SCANS THAT THE TRACK ALGORITHM WITH LIMITED SCANS USED.



(a) All user-days



(b) Mobile user-days

Figure 2. Difference in handover counts. The x-axis shows the difference between the handoffs detected in the trace and those using different algorithms. The y-axis shows the CDF of user-days. (a) and (b) show the difference for all user-days and that for mobile user-days, respectively. Although the maximum difference is 232 handoffs, of which Optimal produced, the x-axis for both figures are truncated at 100 for better viewing.

performance, we now consider the performance gain in more detail. We computed the number of handoffs reduced compared to those in the trace for each user-day, shown in Figure 2. The x-axis shows the difference in handoffs, and the y-axis shows the CDF of all user-days. (a) and (b) show the difference for all user-days and that for mobile user-days, respectively. Although the maximum difference is 232 handoffs, of which Optimal produced, the x-axis for both figures are truncated at 100 for better viewing. We also excluded user-days of which the trace has zero handoffs; we ended up with 3328 user-days and 613 mobile user-days.

Figure 2 shows that Track algorithms performed better than LookBack but worse than the off-line optimal. Using Track algorithms, roughly 12%-13% of all user-days reduced more than 10 handoffs per day. For mobile user-days, roughly 60% reduced more than 10 handoffs, and 9% reduced more than 40 handoffs per day.

Heuristic	Track-1	Track-0	Track-1S	Track-0S
States	8.5	3.7	4.1	2.2

Table V

AVERAGE NUMBER OF NEW STATES GENERATED PER DAY.

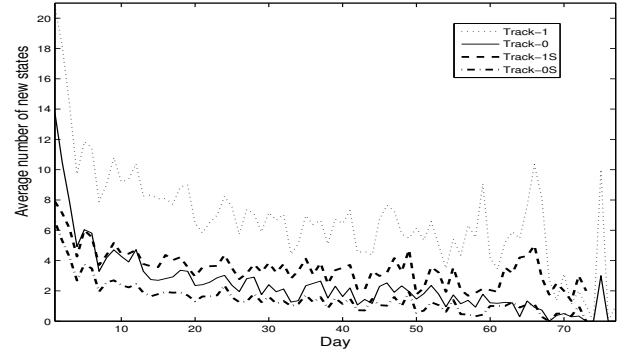


Figure 3. Average number of new states over time.

We now want to consider the overhead or complexity of variations of our Track algorithm. As Track encounters new states, it needs to keep adding them to the existing set of states. We computed the average number of new states generated per day for each variations. The result is shown in Table V. The algorithms using history of 0 reduced the average states by more than half (56%) for the original Track and by 46% for the Track with limited scans.

We also extracted the average number of newly generated states over time, shown in Figure 3. The x-axis shows days and the y-axis shows the average new states for each day. Days on the x-axis does not represent calendar days. Instead, they denote i th day each user connected to the wireless network. All four variations of Track algorithm showed a decreasing trend over time. However, Track-1 fluctuated by a large amount, meaning that for certain days, many new states needed to be generated even after the algorithm was run for many days. Track-1S also fluctuated even though the amount was less than Track-1. Both Track-0 and Track-0S eventually approached zero, meaning that no new states were encountered.

C. Summary of results

Here we summarize our results for all user-days:

- Compared to the trace, Optimal algorithm reduced the number of handoffs nearly by half (48%). Although this algorithm would not be applicable in many cases, it can still be used in the cases where the future mobility pattern is known, such as trains.
- LookBack algorithm provided a worse-case boundary of $(\log k + 2)$, although its performance gain was moderate.
- Compared to the trace, Track algorithm (Track-1S) reduced the number of handoffs by 36%, using only a small number of scans (6.7) and a small number of states (4.1) per user-day.

VI. RELATED WORK

Association control is an important component of wireless mobile networks that affects the system performance, espe-

cially when the access points are densely deployed. Numerous association control schemes have been proposed in the research literature [12], [13], [3], [14], while various vendors of WLAN products also have their own proprietary solutions [2], [1]. In a broad context, the association decisions have direct impact on the network performance in various perspectives, e.g., load balancing across APs, network utilization, throughput fairness among different users, and user-perceived interruptions. The existing solutions all attempt to optimize along one dimension or the other.

A client can choose the AP with the highest signal strength, so that it can maximize its own expected throughput [1]. Alternatively, in order to balance the AP loads, each AP can broadcast its current load in the Beacon message and each client can choose the least loaded AP in its vicinity [2]. In order to achieve certain QoS guarantees, the association schemes can take into account the available bandwidth on each AP. For example, in [12], a user can request a minimal amount of bandwidth and will subsequently be associated with an AP that can satisfy such a bandwidth requirement. Tsai et al. [13] further propose to re-associate the existing users when the bandwidth requests are violated. Bejerano et al. [3] also propose efficient association algorithms to achieve max-min bandwidth fairness among different users. To address the flash-crowd problems in a heavily utilized network, IQU [14] was recently proposed to queue the association requests from the users and grant network access to them in a round-robin and preemptive manner. Our work differs from all these existing designs in that we focus on the user-perceived connection interruptions by minimizing the number of handoffs occurring to a mobile client. To the best of our knowledge, this problem has not been studied before.

There are many previous efforts to provide seamless connectivity for mobile users by minimizing the handoff latency. For example, Pack et al. [4] propose to reduce the channel scanning time by using topographical knowledge of the deployed APs. However, this approach requires centralized maintenance and dissemination of the entire network topology. Neighbor Graph techniques were proposed in [5], [6] to update the network topology in a distributed and adaptive manner. Recently, SyncScan [7] was proposed to avoid the need for such topology information by continuously monitoring the nearby APs at each client. Furthermore, many of these designs make extensive use of proactive caching, so that the time spent on re-authentication and re-association can be minimized. While each of these fast handoff mechanisms has its own merits, they all require significant changes to the 802.11 protocol specifications and hence face various deployment hurdles in practice. Given that fast handoff is not available in most deployed 802.11 networks, we take a different approach in this work by minimizing the number of handoffs. As such, our proposed association control schemes are orthogonal and complimentary to the existing fast handoff designs. Furthermore, our solution requires no changes to the 802.11 protocol or the deployed APs, thus it can be readily used in today's 802.11 networks.

VII. CONCLUSIONS

Maintaining seamless connectivity for mobile users is a major research challenge in wireless networks. To this end, we have presented novel association control algorithms to minimize the frequency of handoffs and thus alleviate the interruptions experienced by mobile users. The fundamental insight in our algorithms is that the association decisions should take into account the longevity of the associations, which is ignored by the existing solutions. The association lifetime depends on how the users move in the future. We propose two online algorithms, *LookBack* and *Track*, which use randomization and a statistical approach, respectively, to estimate the longevity of associations. Our trace-driven simulations confirm the effectiveness of the proposed algorithms. In particular, the *Track* algorithm can achieve near-optimal performance in realistic settings, and can be readily deployed in the existing 802.11 wireless networks.

REFERENCES

- [1] Symbol Technologies, "Wireless Networker CF Radio Card Data Sheet," 2006.
- [2] Cisco Systems Inc., "Aironet 802.11 a/b/g WLAN Client Adapter Data Sheet," 2006.
- [3] Y. Bejerano, S.-J. Han, and L. Li, "Fairness and Load Balancing in Wireless LANs Using Association Control," in *Proc. ACM Mobicom*, Philadelphia, PA, USA, September 2004.
- [4] S. Pack and Y. Choi, "Fast Inter-AP Handoff using Predictive Authentication Scheme in a Public Wireless LAN," in *Proc. IEEE Networks Conference*, Atlanta, GA, USA, August 2002.
- [5] M. Shin, A. Mishra, and W. Arbaugh, "Improving the Latency of 802.11 Hand-offs using Neighbor Graphs," in *Proc. ACM MobiSys*, Boston, MA, USA, June 2004.
- [6] A. Mishra, M. Shin, and W. Arbaugh, "Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network," in *Proc. IEEE Infocom*, Hong Kong, China, March 2004.
- [7] I. Ramani and S. Savage, "SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks," in *Proc. IEEE Infocom*, Miami, FL, USA, March 2005.
- [8] M. MaNett and G. M. Voelker, "Access and Mobility of Wireless PDA Users," *Mobile Computing Communications Review*, vol. 9, pp. 40–55, April 2005.
- [9] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *Proceedings of the Tenth Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM Press, September 2004, pp. 187–201.
- [10] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *Proceedings of the Tenth ACM International Conference on Mobile computing and networkign (MOBICOM)*, Philadelphia, PA, Sept. 2002.
- [11] T. King, T. Kaenselmann, S. Kopf, and W. Effelsberg, "Overhearing the wireless interface for 802.11-based positioning systems," in *Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom'07)*, White Plains, NY, 2006, pp. 145–150.
- [12] A. Balachandran, P. Bahl, and G. Voelker, "Hot-Spot Congestion Relief in Public-area Wireless Networks," *SIGCOMM Computer Communication Review*, vol. 32, pp. 59–59, January 2002.
- [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] A. Jardosh, K. Mittal, K. Ramachandran, E. Belding, and K. Almeroth, "IQU: Practical Queue-Based User Association Management for WLANs," in *Proc. ACM Mobicom*, Los Angeles, CA, USA, September 2006.