Concurrent Wi-Fi for Mobile Users: Analysis and Measurements

Hamed Soroush* Peter Gilbert† Nilanjan Banerjee△
Brian Neil Levine* Mark Corner* Landon Cox †

*Dept. of Computer Science, Univ. of Mass. , Amherst, {*hamed,brian,mcorner*}@*cs.umass.edu*
△ Dept. of CSCE, Univ. of Arkansas, Fayetteville, *nilanb@uark.edu*
† Dept. of Computer Science, Duke Univ. , {*gilbert,lpcox*}@*cs.duke.edu*

ABSTRACT

We present the first in-depth analysis of the performance of attempting concurrent AP connections from highly mobile clients. Previous solutions for concurrent Wi-Fi are limited to stationary wireless clients and do not take into account a myriad of mobile factors. Through an analytical model, optimization framework, and numerous outdoor experiments, we show that connection duration, AP response times, channel scheduling, available and offered bandwidth, node speed, and `dhcp` *joins all affect performance. Building on these results, we design, implement, and evaluate a system, Spider, that establishes and maintains concurrent connections to 802.11 APs in a mobile environment. The system uses multi-AP selection, channel-based scheduling, and opportunistic scanning to maximize throughput while mitigating the overhead of association and* `dhcp`*. While Spider can manage multiple channels, we empirically demonstrate that it achieves maximum throughput when using multiple APs on a single channel. Our evaluation shows that Spider provides a 400% improvement in throughput and 54% improvement in connectivity over stock Wi-Fi implementations.*

1. INTRODUCTION

The concurrent deployment of high-quality wireless networks and large-scale cloud services offers the promise of ubiquitous access to an almost limitless amount of content. Today's mobile users expect to be able to listen to music from Pandora or watch video from Netflix regardless of where they are or what they are doing. However, as users' expectations have grown more demanding, the performance and connectivity failures endemic to existing wireless networks have become more apparent and painful [2]. These problems are exacerbated by mobility in general and *vehicular mobility* in particular [2]. Users want uninterrupted access to rich content from the cloud while driving a car or riding public transportation through a city, but existing wireless networks are not yet up to the task.

Improving Wi-Fi will play a critical role in addressing the frustrations of urban mobile users. Many users rely on Wi-Fi-only devices such as laptops, portable music players, and tablet computers. These devices experience frequent and sustained connectivity lapses due to the limited range and coverage of Wi-Fi access points [3]. At the same time, devices with cellular radios such as smartphones experience poor network performance in dense urban areas, where networks strain under the heavy load of bandwidth-hungry clients. As a result, researchers have proposed supplementing uneven cellular connectivity with Wi-Fi [2], and network carriers are rapidly deploying Wi-Fi access points throughout cities to siphon off traffic from their overburdened cellular networks [21].

In this paper, we ask the following question: how can a truly mobile user best utilize a dense deployment of Wi-Fi APs? One approach is to aggregate connectivity by simultaneously associating with a large number of APs. Virtualized Wi-Fi systems, such as Virtual Wi-Fi [5], FatVAP [12], and Juggler [19], have shown that stationary users connected to multiple access points can achieve up to $3x$ greater bandwidth than users connected to a single access point [12]. These results are promising, but we observe that they are not directly applicable to mobile users (particularly those moving at vehicular speeds) due to the overhead of acquiring `dhcp` leases.

More specifically, acquiring a `dhcp` lease from an AP is a relatively slow process that does not support the power-save mode on which virtualized Wi-Fi depends. This would remain true even if channel-switching delays could be reduced to zero. As a result, clients using virtualized Wi-Fi must dwell on an AP's channel until a lease has been obtained. As we show through analysis, measurement, and experimentation, the dwell times needed to obtain `dhcp` leases from APs can swamp TCP round-trip times, leading to TCP timeouts and greatly reduced bandwidth. Our results show that at higher speeds, mobile users receive better performance by connecting to multiple APs *only* if they appear on the same channel. Only at lower speeds can mobile users recover from the throughput loss induced by dhcp joins on other channels.

Furthermore, while these two extremes are clear, the breaking point between them, which we refer to as the *dividing speed*, is not obvious. One of the main contributions of this paper is a general model of virtualized Wi-Fi that isolates the critical factors for determining an optimal schedule over one or more channels. These factors include the user's speed, the AP's dhcp response time, the AP's offered bandwidth, and the attained bandwidth. For example, in a typical environment, our model suggests that users traveling at an average speed of 10 m/s ($\sim$22 mph) or faster should form concurrent Wi-Fi connections only within a single channel. We also empirically show that link-layer association, dhcp, and TCP performance are negatively affected by multi-channel solutions.

Building from the results of our analytical study, we have designed and implemented a virtualized Wi-Fi system called *Spider* that is practical for high-speed mobile users. Spider benefits both Wi-Fi-only mobile devices (e.g., the majority of laptops, portable music players, and tablets) as well as devices equipped with both Wi-Fi and cellular radios. As an experimental framework, Spider allows us to study the applicability of virtualized Wi-Fi solutions and their challenges in highly mobile scenarios.

Spider solves the practical issues of access-point discovery and dhcp lease acquisition while optimizing bandwidth using a single wireless channel. Through a set of extensive outdoor experiments we show that Spider can maximize bandwidth using multiple APs on a single wireless channel, achieving 122KBps, an improvement of more than 400% over a multi-channel approach. These results demonstrate that Spider can effectively supplement cellular networks for highly mobile clients.

We also empirically investigate trade-offs between throughput and connectivity, which is an important consideration for Wi-Fi-only devices. Spider can be used to manage and schedule associations with APs on multiple channels, though at the cost of achieved bandwidth, as our model predicts. We demonstrate that if connectivity is a greater priority than throughput, then joining multiple APs on multiple channels is best: Spider can provide 44% better connectivity than a single-AP approach (compared to 35% in the multiple-AP, single-channel case), but at an achieved bandwidth of 28KBps.

To summarize, in this paper we make the following contributions: (1) We present a formal model of virtualized Wi-Fi for mobile users suggesting that there is a threshold speed, beyond which throughput can only be maximized by remaining on a single channel due to the delays of acquiring dhcp leases; (2) We present measurements indicating that at high speeds virtualized Wi-Fi systems can trade throughput for improved connectivity; and (3) We present the design, implementation, and evaluation of a virtualized Wi-Fi system called Spider that incorporates the lessons learned from our analytic and empirical measurements.

2. MOBILE MULTI-AP CONNECTIONS: A MODEL AND EXPERIMENTATION

Past research has shown the benefits of associating with multiple Wi-Fi APs in series for mobile clients [2, 8] and the benefits of concurrent associations for stationary clients via virtualized Wi-Fi systems [5, 12, 19]. However, to the best of our knowledge, this paper presents the first investigation of the potential benefits for truly mobile clients of concurrently associating with multiple APs.

Existing virtualized Wi-Fi systems provide a starting point. Virtualized Wi-Fi systems [5, 12, 19] rely on a client to instruct an AP to buffer incoming packets while the client communicates with another AP. A client does this by falsely claiming to an AP that it is entering *power-save mode* (PSM), which causes the AP to buffer the client's packets. Given that the backhaul bandwidth is typically smaller than the wireless bandwidth, such a scheme can lead to higher aggregate throughput for a client when its switching delays are kept very short. Static multi-AP solutions, in particular, are not concerned with the delays caused by associating with new APs, and they do not need to be. In a typical static scenario, the joining process (association and dhcp) happens once, and its duration is negligible compared to the total connection time.

However, in a mobile Wi-Fi environment, connections are relatively brief and clients must continuously associate and obtain dhcp leases from APs as they become available. Importantly, the packets involved in the join process cannot be buffered using a PSM request, and therefore, the client cannot switch away without significantly reducing the likelihood of successfully obtaining a dhcp lease. This is because the time to complete the dhcp process is controlled by the AP rather than the client (unlike the switching schedule). In some circumstances, it may be possible to reduce dhcp delays through coordination across APs, but Wi-Fi APs in urban areas are rarely in the same administrative domain, making this approach impractical for urban mobile users.

To better understand the effect of mobility on the performance of multi-AP systems, this section describes the design and validation of a model that predicts the probability of obtaining a dhcp lease from an AP as a function of the amount of time spent in range. Based on this model, we formulate an optimization framework to determine schedules that maximize aggregate throughput.

Our analytical framework allows us to find the *dividing speed* above which a mobile multi-AP solution should consider APs only on a single channel for throughput maximization. However, the framework is limited since it does not consider the complexities of the multi-phase algorithms running link-layer association, dhcp, and TCP. In order to isolate the effects of multi-channel switching on these protocols, we also perform a series of real experiments (Section 2.2). The bottom line of the model, optimization framework, and em-

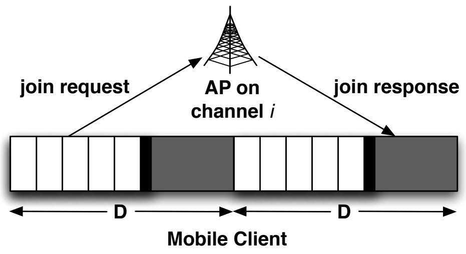


Figure 1: Depiction of a join failure due to untimely arrival of the join response message in our simplified scenario. White regions represent the fraction (f_i) of the scheduling period (D) spent on channel i to communicate with the AP. Gray regions depict the rest of the scheduling period spent away from the AP on another channel. Switching delay (w) is shown in black. For simplicity, the model assumes that join requests are sent in the beginning of intervals of length $\lceil \frac{Df_i}{c} \rceil$ where c is a constant. Join process becomes more complicated in reality, where it involves multiple steps to associate and obtain a `dhcp` lease instead of one.

pirical experiments is that switching channels is detrimental to overall throughput for almost any short-lived connection. This conclusion is further supported by a series of extensive experiments presented in Section 4 using a complete system in an outdoor setting.

2.1 Analytical Framework

We have designed our analytical framework with two key questions in mind: *First, how do channel switching and node speed affect the time to associate and obtain a* `dhcp` *lease?* And *second, at vehicular speeds, when is it beneficial to aggregate throughput from APs on multiple channels?*

2.1.1 Join Model

We model the probability that a mobile node succeeds in joining an AP operating on channel i during the first t seconds that it is in range. We assume a round robin schedule, where the node spends a fraction of time f_i on channel i from the total scheduling duration of D. When switching channels, the node incurs a delay w during which no packets can be received. We approximate $t \simeq s \times D$, where s is a positive integer reflecting the speed of the mobile node (note that s is smaller when the node's speed is higher). We refer to each of the s time intervals as a *round*. In sum, in each round, a duration of Df_i is spent on channel i. For simplicity, we assume that the mobile node switches to channel i as soon as it enters the range of the AP.

In practice, a Wi-Fi join event consists of several complementary steps. However, for simplicity we assume that association involves a single handshake. Further, we assume that in the absence of losses, the time between the transmission of the single join request and the reception of the single join response in a non-virtualized scenario is uniformly distributed in the interval $[\beta_{min}, \beta_{max}]$. During a round, the mobile node can transmit a maximum of $\lceil \frac{Df_i}{c} \rceil$ join requests to the AP, where c is the time between two consecutive requests. In practice, the duration c is determined by `dhcp` and link-layer timeout values, and we set it to a constant in the model. We let $\beta_{m,k} \in [\beta_{min}, \beta_{max}]$ denote the *join time* corresponding to a request transmitted in segment k of round m, where $1 \leq k \leq \lceil \frac{Df_i}{c} \rceil$. We then compute the probability that a request made at the beginning of segment k in round m leads to a successful join. Note that for a successful join, the response from the AP must be received at the mobile node within a time interval Df_i of the current or later a round (see Fig. 1). The above constraint can be mathematically formulated as follows.

$$\begin{aligned} w + (k-1)c + \beta_{m,k} &\leq (n-m)D + Df_i && (1) \\ w + (k-1)c + \beta_{m,k} &\geq (n-m)D && (2) \end{aligned}$$

These can be combined into one equation as

$$(n-m)D + c - w \leq kc + \beta_{m,k} \leq (n-m+f_i)D + c - w \quad (3)$$

where $n \geq m$ is the round in which the response is received. Since $\beta_{m,k} \in [\beta_{min}, \beta_{max}]$, we have

$$kc + \beta_{min} \leq kc + \beta_{m,k} \leq kc + \beta_{max} \quad (4)$$

Using the above equations, we can derive the probability that a request sent during segment k of round m leads to a successful join for a lossless channel as

$$q(m,n,k) = \begin{cases} 0 & \text{, if } \delta^{min}_{m,n} > \alpha^{max}_k \\ 0 & \text{, if } \delta^{max}_{m,n,fi} < \alpha^{min}_k \\ \frac{\min\{\alpha^{max}_k, \delta^{max}_{m,n,fi}\} - \max\{\alpha^{min}_k, \delta^{min}_{m,n}\}}{\alpha^{max}_k - \alpha^{min}_k} & \text{, otherwise} \end{cases} \quad (5)$$

where

$$\begin{aligned} \alpha^{min}_k &= kc + \beta_{min} \\ \alpha^{max}_k &= kc + \beta_{max} \\ \delta^{min}_{m,n} &= (n-m)D + c - w \\ \delta^{max}_{m,n,f_i} &= (n-m+fi)D + c - w \end{aligned}$$

Hence, the probability that *no* request made in round m leads to a successful join in round n in a lossy channel with message loss probability h can be calculated from the previous equation as

$$\overline{q(m,n,h)} = \prod_{k=1}^{\lceil \frac{Df_i - w}{c} \rceil} \left(1 - q(m,n,k)(1-h)^2\right) \quad (6)$$

Thus, given that the mobile node spends a fraction f_i of time on a channel i, the probability of obtaining at least one lease in time t is the following:

$$p(f_i, t) = 1 - \prod_{m=1}^{\lfloor \frac{t}{D} \rfloor} \prod_{n=m}^{\lfloor \frac{t}{D} \rfloor} \overline{q(m,n,h)} \quad (7)$$

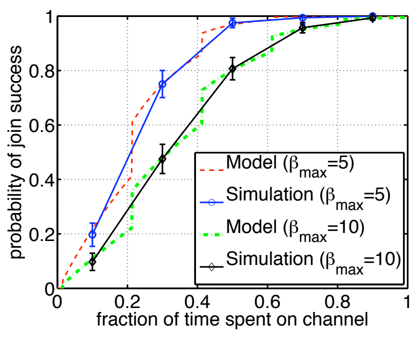


Figure 2: The probability of join success as a function of the fraction of time, f_i, spent on the AP based on the presented model (Eq. 7) and simulation. Mobile node with a scheduling period of $D = 500ms$ spends $t = 4s$ in the vicinity of the AP with $\beta_{min} = 500ms$, $\beta_{max} = 5s$ and $10s$. Switching overhead of the driver is $w = 7ms$. Join requests are sent every $c = 100ms$ and loss rate is $h = 10\%$.

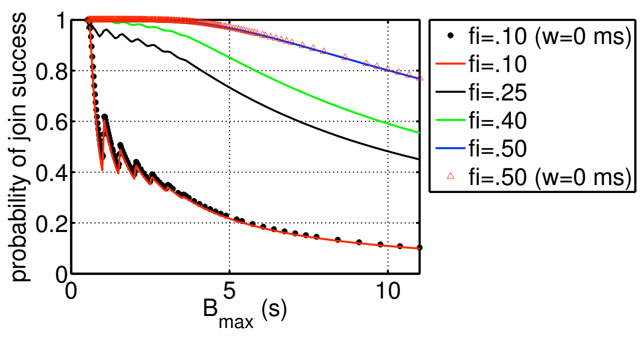


Figure 3: The probability of join success as a function of the maximum time it takes the AP to respond. When a fixed fraction of time is spent on the channel, shorter maximum join times lead to higher chances of join success. Mobile node with a scheduling period of $D = 500ms$ spends $t = 4s$ in the vicinity of the AP with $\beta_{min} = 500ms$. Switching overhead of the driver is $w = 7ms$ except mentioned otherwise (and has a negligible affect on performance). Join requests are sent every $c = 100ms$ and loss rate is $h = 10\%$.

Corroboration of Derivation. We confirm the above derivation by using a simulation with the same assumptions. Fig. 2 shows the probability of successfully joining an access point as a function of the fraction of time spent on the channel for both the model (Eq. 7) and the simulation. Each point on the simulation line represents an average of hundred runs, and error bars represent one standard deviation. Each run uses a different seed and represents 100 trials where a join is attempted and the value of $\beta_{m,k}$ is sampled from the given distribution. We set other input parameters to typical values we observed in practice (see Section 2.2). The simulation results are statistically equivalent to the model and hence, internally confirm its derivation.

The discontinuities at points where f_i is 0.2, 0.4, 0.6 or 0.8 are a result of the ceiling function in the calculation of the maximum number of join requests sent per round, $\lceil \frac{Df_i}{c} \rceil$.

2.1.2 Discussion

Eq. 7 represents a non-linear relationship between the fraction of time spent on the channel, f_i, and the probability of a successful join $p(f_i, t)$. For instance, in Fig. 3, the probability of getting a lease during the first $t = 4$ seconds falls from 75% to 20% when the percentage of time devoted to the AP reduces from 30% to 10%. Further, the node should spend nearly 100% of its time on the channel for an *assured* successful join.

These results motivate a *channel-centric* approach to scheduling concurrent Wi-Fi connections in mobile scenarios as opposed to the *AP* switching approach used in static cases [12]. Spending all its time on one channel, a mobile node can aggregate bandwidth from several APs on that channel without affecting the probability of join success.

Another implication of our analysis is the relationship between the maximum join time, β_{max}, and the probability of a successful join $p(f_i, t)$. Fig. 3 graphs the value of $p(f_i, 4)$ with β_{max} as the independent variable for four different values of f_i, using Eq. 7 . Note that β_{max} represents the maximum amount of time required to associate with an AP in the non-virtualized scenario with no losses. When a fixed fraction of time is spent on the channel, shorter maximum join times lead to higher chances of a successful join. While this is a known fact for single-AP scenarios, our results demonstrate that techniques such as caching `dhcp` leases, maintaining a history of APs with short join times, and decreasing link layer timeouts that reduce β_{max} are essential for multi-AP systems. In other words, when these techniques are not available—when mobile nodes travel in unfamiliar areas and AP responses are slow—multi-AP systems will see significant drops in performance. Furthermore, even when there is no switching delay ($w = 0$), chances of joining are not notably increased, pointing to channel schedule and `dhcp` response times as major contributors to join success rate.

2.1.3 Throughput Maximization

A major objective of maintaining concurrent connections to multiple APs is bandwidth aggregation. To understand how channel schedules and node speed affect bandwidth aggregation, we formulate an optimization framework for throughput maximization.

The objective function and the associated constraints are shown in Eq. 8. To construct it, we assume that the mobile node is in range of APs for T seconds. We let $g_T(f_i)$ denote the expected amount of time it takes to successfully obtain a lease as a function of the fraction of time spent on the channel and the total time (T) spent in the vicinity of the APs, calculated based on Eq. 7. We let B_w be the maximum bandwidth

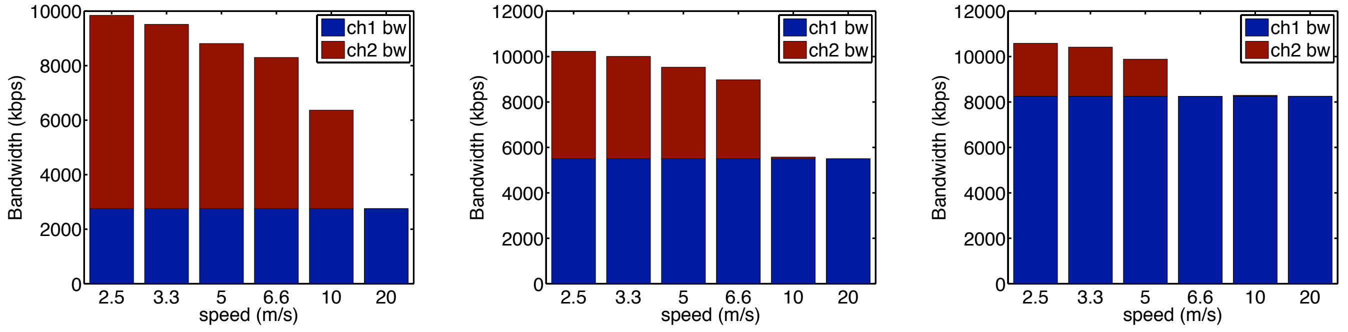


Figure 4: Maximum aggregated bandwidth for different speeds when only two channels are used. The offered bandwidth on the first and second channels are (25%,75%), (50%,50%), and (75%,25%) of the wireless bandwidth (B_w =11Mbps) respectively in each figure from left to right for $\beta_{max} = 10s$ and $\beta_{min} = 500ms$. Wi-Fi range is assumed to be 100m.

of each channel. We distinguish current and offered bandwidth: let B_j^i be the total end-to-end bandwidth from APs on channel i that the node has already joined to; let B_a^i be the end-to-end bandwidth available from APs that the node is attempting to join to and would have during the duration $T - g_T(f_i)$. We let k be the number of available channels and we state the objective function as follows.

$$\max_{f_i} \left\{ T \sum_{i=1}^{i=k} (f_i B_w) \right\} \tag{8}$$

$$\text{s.t.} \qquad 0 \le f_i \le \frac{B_j^i + (1 - g_T(f_i)/T) B_a^i}{B_w}, \ \forall i \tag{9}$$

$$\sum_{i=1}^{i=k} (f_i D + \lceil f_i \rceil w) \le D \tag{10}$$

The formulation is similar to the FatVap optimization problem [12] except for the constraint on f_i, which considers the additional bandwidth gained from APs that the node is associating with. For simplicity, we have assumed all APs are in range for a duration T and thus have the same $g_T(f_i)$.

We numerically solve the above optimization to determine the optimal schedule as a function of the speed of the node. Specifically, we evaluate three scenarios for a two-channel case:

1. $B_j^1 = 0.75 B_w$ and $B_a^2 = 0.25 B_w$
2. $B_j^1 = 0.25 B_w$ and $B_a^2 = 0.75 B_w$
3. $B_j^1 = 0.50 B_w$ and $B_a^1 = 0.50 B_w$

Our goal is to determine the speeds at which it is optimal to switch channels, given a practical Wi-Fi range of 100 meters. Fig. 4 shows the results for the three scenarios in terms of the optimal bandwidth that can be extracted from each channel. For every scenario, there is a dividing speed: when moving slower than this speed, the mobile node should switch channels to maximize bandwidth. Quantitatively, this speed is less than $10m/s$ for most scenarios. This result implies that for highly mobile networks (where the average node speed is greater than $10m/s$), the best policy to maximize bandwidth is to stay on a *single* channel. While we examined only three scenarios to produce Figure 4, we note that our model and optimization framework solve all combinations of inputs. We empirically validate and further explore this result in Section 4 using a mobile testbed.

2.2 Experimental Analysis

Our model shows that scheduling fractional time on several channels has an adverse effect on the probability of successfully joining APs (see Fig. 3). Additionally, for high vehicular speeds, maintaining concurrent connections to access points while staying on *one* channel leads to optimal bandwidth aggregation. However, our model makes two simplifying assumptions. First, it assumes that joining is a one-shot process, while in practice Wi-Fi joins involve a multi-phase bidirectional handshake involving both association and `dhcp`. Second, it assumes that the schedule is short enough so that TCP timeouts are avoided. The assumptions cause the model to be optimistic: multi-channel switching performs better in the model than can be expected in a real scenario.

To quantify the effect of switching on link-layer associations, `dhcp` and TCP, we performed two sets of experiments, one on our outdoor vehicular testbed and the other on an indoor wireless testbed. Our results quantify the relationship between channel schedule, join success, and TCP throughput. In sum, successful link-layer association and `dhcp` joins have some tolerance for switching, while TCP is more sensitive.

2.2.1 Association and `dhcp`

In our first set of experiments, we use a testbed of vehicular nodes and a Wi-Fi driver that can simultaneously associate with APs on different channels, just as we modeled above — in fact, the driver is the core of our full system, and we provide details of its operation in our technical report [23]. The goal of our outdoor experiments is to quantify the effect of varying the fraction of time spent on each channel on the fail-

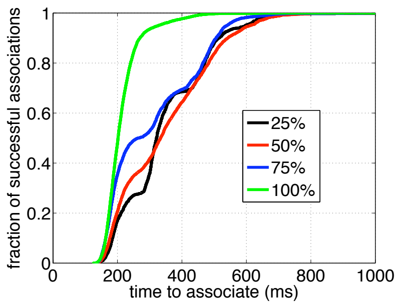


Figure 5: The rate of successful **link-layer associations** on a channel as a function of the amount of time the Wi-Fi driver spends on a single channel (out of 400ms for all channels).

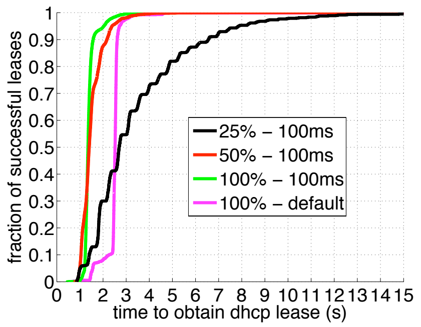


Figure 6: The rate of successful **dhcp lease acquisition** (association and dhcp) on a channel as a function of the amount of time the Wi-Fi driver spends on that channel and the dhcp timeout of 100ms or the default of 1s.

ure rate of associating and obtaining a dhcp lease. Moreover, these experiments allow us to quantify the surprisingly detrimental effect of decreased link-layer response timeouts on a multi-AP driver, an approach shown by Eriksson et al. [8] to improve single-AP/single-channel drivers.

To evaluate the effect of the driver's schedule on the join success rate, we performed several experiments, each lasting six hours on five vehicles moving around Amherst, MA, representing hundreds of trials. Each mobile node spends a fraction $f_6 = x$ of $D = 400ms$ on channel 6, and a fraction $f_1 = f_{11} = (1 - x)/2$ on channels 1 and 11, applying terminology from our model and where $0 \leq x \leq 1$. Our results show the association delays on channel 6 and consider it as our *primary* channel. Since join delays are affected by link-layer timeouts, we reduced them from a standard of $1s$ to $100ms$ in these experiments[1].

Figs. 5 and 6 plot the empirical cumulative distribution functions for durations of association and dhcp respectively for these experiments. Separate conclusions can be drawn for association and dhcp.

Our analytical framework, which only evaluates a simple join-request scenario, predicts that join success is dependent on the channel schedule. However, these real experiments demonstrate that the performance of actual link-layer joins is more complicated given the interactions of scanning and the four-way handshake used. As shown in Fig. 5, when the driver spends all its time on one channel ($f_6 = 1$), the median association time is 200ms, and all associations complete within 400ms. However, when this fraction drops to $f_6 = 0.75$, the median association time increases to 300ms and only 75% of associations are successful within 400ms. Interestingly, this performance does not degrade significantly as f_6 decreases to 0.50 and 0.25 suggesting that link layer association is in some ways robust to switching.

dhcp is a more complicated protocol. It relies on successful association and involves at least four more frames between client and AP. Moreover, in default implementations, the client attempts to acquire a lease for 3 seconds, and it is idle for 60 seconds if it fails. The performance of this default scheme dedicated to a single channel is shown in Fig. 6 (as "100% default") with a median join time of $2.5s$. The figure shows that reducing both timeouts above to 100ms [8] has a significant effect on performance. For the same schedule of $f_6 = 100\%$, the median join time reduces to $1.3s$ when a $100ms$ dhcp timeout is used. Once the schedule is set to $f_6 = 25\%$ and $D = 400ms$, equal to the timeout, repeated failures cause the accumulated time to degrade performance once again. Hence, while reconfigured dhcp timers are a boost to performance, we could not make dhcp robust to low fractions of scheduled time. *This result suggests that the driver's time cannot be divided among more than two channels at 50% each in a mobile setting where the duration of time in range of an AP is limited.*

2.2.2 TCP performance

Our throughput maximization framework assumes that the schedule does not lead to TCP timeouts. However, in a practical setting, if the channel schedule is skewed towards spending a large fraction of the time on a single channel, TCP connections on an orthogonal channel can timeout, potentially strangling performance. There is an inherent tension between the probability of successfully associating with APs on one channel and sustaining TCP connections on another channel.

To quantify this trade-off, we performed a set of controlled experiments in an indoor setting. We configured an AP on one channel (the primary channel) and varied two parameters, the fraction of time spent on the primary channel for a fixed scheduling period of $D = 400ms$ and the *total scheduling time* equally distributed across channels 1, 6, and 11 ($f_1 = f_6 = f_{11} = 1/3$). When the fraction of time spent on the primary channel is varied for a 400 ms schedule (equal to two typical RTTs), the TCP throughput increases monotonically, as shown in Fig. 7. However, when the total scheduling time is varied instead, the throughput increase is non-monotonic, as shown in Fig. 8. This behavior is a result of increasing the total schedule which increases the amount of time *spent away* from the channel which can lead to TCP timeouts.

[1]The link-layer timeout reflects a timer for each message in a multi-step protocol & not a timeout for the entire request-response process.

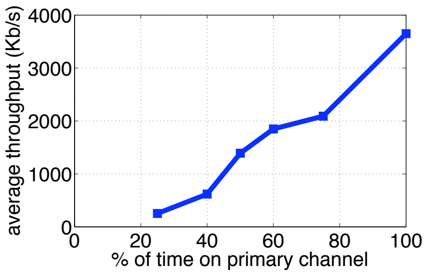


Figure 7: Average TCP throughput as a function of the **percentage of time** spent by the Wi-Fi driver on the primary channel. Since the cumulative time spent on all the channels is 400 ms (which is less than two RTTs) the throughput is proportional to the percentage of time spent on the primary channel.

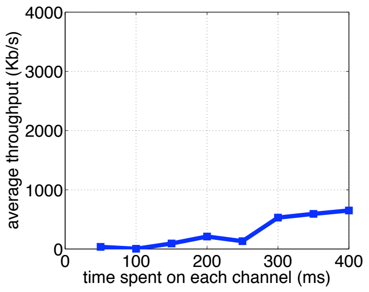


Figure 8: Average TCP throughput as a function of the **absolute time** spent on each channel. For time x spent on the channel, time $2x$ is spent away from it. The throughput is very sensitive to the amount of time spent by the driver on each channel due to TCP timeouts and TCP slow start.

2.3 Main Result of Analysis

Our analytical model, throughput maximization framework, and experimental analysis point to the following conclusion: *At vehicular speeds, the best channel scheduling policy for throughput maximization is to spend all of the time on a single channel that provides maximal bandwidth.*

Several observations from our analytical and experimental framework substantiate this result. First, as shown in Figs. 2, 5, and 6, our model predicts that the probability of successfully joining to APs within a short time is high only when the node spends close to 100% of the time on the channel. Note that continuously associating with APs is mandatory in mobile scenarios to sustain connectivity, given short encounters at vehicular speeds (median of 8 s and average of 22 s in our town). Second, link-layer association, `dhcp` joins, and TCP throughput (shown in Figs. 5, 6, and 8) are all adversely affected if the radio spends a large amount of time away from the channel. Therefore, when the mobile node spends a small amount of time in vicinity of APs, it should aggregate bandwidth from one channel while simultaneously associating with APs on that channel.

3. SPIDER

Based on the results of our analytical framework and experimental analysis in Section 2, we have developed Spider, a system that leverages concurrent 802.11 connections to improve performance in highly mobile networks.

In contrast to previous work that slices time across *individual APs* [5, 12, 19], Spider schedules a physical Wi-Fi card among 802.11 *channels*. Unlike static multi-AP solutions, Spider maintains *one* packet queue per channel that is swapped in and out of the driver; allowing it to communicate with *all* APs on the same channel simultaneously as motivated by our analysis in Section 2. Additionally, it incurs no switching overhead for interfaces on the same channel.

Unlike previous work where the set of APs in the vicinity of the client is fixed, Spider has to dynamically select the set of APs to connect to. As we prove in Appendix A of our technical report [23], selecting multiple APs while maximizing a given system utility function is NP-hard. Consequently, Spider uses a simple heuristic to select APs. The heuristic is driven by our observations in Section 2 that join times with APs is the critical factor for performance in highly mobile scenarios. Therefore, instead of choosing APs with maximum end-to-end bandwidth, we select APs that have the best history of successful joins. We have optimized our machinery for quick sampling, however, if the mobile node is moving too fast to sample, it is unlikely to get high throughput either.

The design and implementation details of Spider are presented in our technical report [23].

4. SYSTEM EVALUATION

Spider aims to improve throughput and connectivity for mobile clients using open Wi-Fi access through intelligent multi-AP selection, channel switching, opportunistic scanning, and parallel per-channel association. Here, we evaluate its performance by focusing on the following key questions: (1) What improvement in throughput and connectivity does Spider provide over stock Wi-Fi configurations? (2) What is the effect of AP density on Spider's performance? (3) Does Spider meet connectivity needs of common wireless users?

While answering these questions, we also present several micro-benchmarks and study the effect of `dhcp` and link-layer timeouts. We compare Spider against its variations (multiple or single channels; multiple or single APs) and stock Wi-Fi. Unfortunately, it is not possible to compare Spider against FatVAP or Juggler since they were built for static wireless scenarios.

In sum, Spider's best configuration (single channel multiple APs) increases throughput by 430% and connectivity by 150% over single-channel, single AP Wi-Fi. As expected, Spider's gains are the result of both connecting to multiple APs when possible and not degrading throughput while attempting to connect to APs on other channels. Spider performs well even in a low density environment. Moreover, Spider can meet the connectivity needs of common wireless users in terms of inter-connection and connection times.

	Num. of connected interfaces				
	0	1	2	3	4
Mean	4.942	4.952	5.266	5.546	5.945
Std Dev	0.009	0.009	1.236	0.823	1.121

Table 1: Channel switching latency (ms) of the Spider driver. When none of the interfaces are connected, the delay corresponds only to a hardware reset.

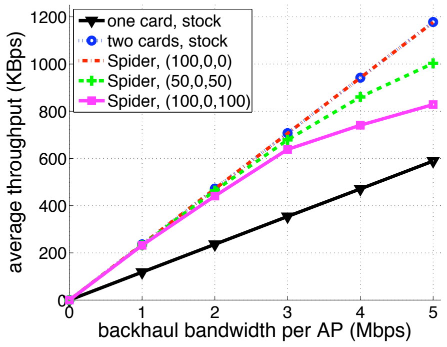


Figure 9: Throughput micro-benchmark. Spider's throughput when dedicated to two APs on a single channel is equivalent to two cards running stock drivers.

4.1 Experimental Setup

We evaluated Spider on a vehicular platform in two different cities: Amherst, MA and Boston, MA.

During experiments in Amherst, almost all APs were on channels 1 (28%), 6 (33%), or 11(34%). Cabernet [8] reported comparable numbers for the Boston area with 83% of the APs on either of the three channels and 39% on channel 6. Since majority of APs dwell on these three channels, we configured Spider to schedule among these three channels.

We test four configurations of Spider. (1) *Single-channel, Single-AP:* Spider mimics off-the-shelf Wi-Fi on a single channel. (2) *Single-channel, Multiple-AP:* Spider stays on one channel (channel 1, 6, or 11) and joins to as many APs on the channel as possible. (3) *Multiple-channel, Multiple-AP:* Spider switches between the three orthogonal channels using static schedules. (4) *Multiple-channel, Single-AP:* Spider switches channels but is associated with one AP at a time. We also tested the unmodified MadWiFi driver as a point of comparison to configuration 1.

Our mobile node comprised of a 1GHz Intel Celeron M system running Ubuntu Linux 2.6.18 and Atheros 802.11abg MiniPCI card. Duration of each experiment was 30–60 minutes with the node repeatedly following the same route.

4.2 Driver Micro-benchmarks

We ran two micro-benchmarks (in a static laboratory environment) designed to measure (1) the latency overhead incurred when switching channels, and (2) the ability of our driver to aggregate bandwidth across connections through multiple APs.

Table 1 shows the mean latency and the standard deviation of a channel switch operation. The channel switching latency is the time required to send a PSM frame to each associated AP on the old channel, perform a hardware reset to apply the channel change, and then send a PSM poll frame to each associated AP on the new channel. The latency is typically in the range of 5–6ms, increasing proportionally to the number of APs, because a separate PSM frame must be sent to each AP. Different chipsets have different delays for a hardware reset and while Spider can incorporate any hardware optimization which reduces that delay, its performance would not be dramatically affected[2].

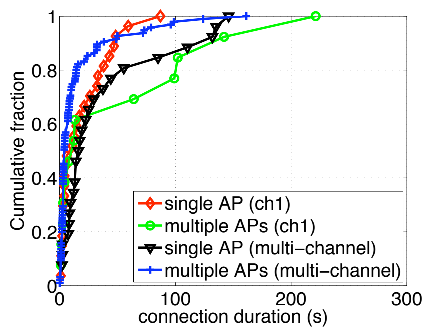
Fig. 9 shows the ability of the driver to utilize the bandwidth offered by multiple APs. We measured mean aggregate throughput achieved while downloading large files over HTTP for a number of configurations: a host with a single card running stock MadWiFi, for comparison; a host with two physical cards running stock drivers; Spider connected to two APs on the same channel; and Spider associated with one AP on channel 1 and one on channel 11, with a schedule of 50ms on each channel; and the previous configuration while spending 100ms on each channel. The APs and servers were connected via LANs in our lab, and a traffic shaper was used to adjust the backhaul bandwidth available through each AP.

The host with two physical interfaces and the host with Spider running on a single channel (connected to two APs) both achieved an aggregate throughput equal to twice that achieved by the host with a single card and stock driver—this is expected, as Spider incurs no channel-switching overheads in this configuration and does not run the risk of causing TCP timeouts when using one channel. The results for the multi-channel Spider configurations show the trade-off between exploiting new connectivity opportunities and extracting throughput from connected APs. When high-bandwidth links are available, a schedule which switches more rapidly between channels is able to achieve greater throughput by reducing the risk of TCP timeouts.

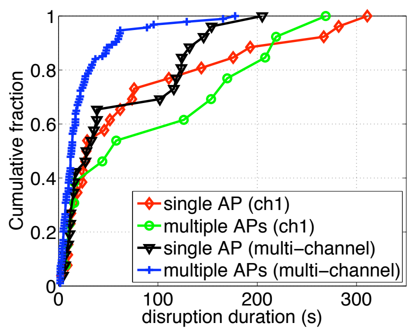
4.3 Connectivity and Throughput

We analyze throughput and connectivity of Spider using four key metrics. (1) *Average throughput*: the amount of data transferred to a sink per unit time during an experiment. (2) *Average connectivity*: the percentage of time that a non-zero amount of data was transferred to a sink. The average throughput and connectivity are bounds on open Wi-Fi performance using multi-AP solutions. (3) *Disruption length*: the contiguous period of time when there is no connectivity. This distribution indicates whether interactive applications such as VoIP or web search can be supported. (4) *Instantaneous bandwidth*: the amount of data per second transferred by a Spider node when there is connectivity. This metric indicates whether multi-AP solutions can support applications that require bursts of high throughput connectivity.

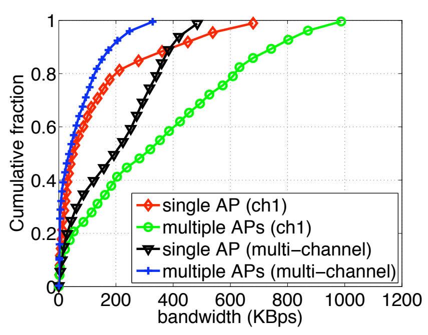
[2]Recall that `dhcp` delays, and not the switching delay, are the major determinants of performance in mobile multi-AP solutions.



(a) CDF of the connectivity duration for Spider configurations.



(b) CDF of connectivity disruptions for Spider configurations.



(c) CDF of KB/s transferred (during connectivity) for Spider configurations.

Figure 10: Single-channel configurations provide the best instantaneous throughput. Multi-AP, multi-channel reduce throughput due to the overhead of joining.

(Config) Parameters	Throughput	Connectivity
(1) Channel 1, Multi-AP	121.5 KB/s	35.5%
(2) Channel 1, Single-AP	28.0 KB/s	22.3%
(3) 3 channels, Multi-AP	28.8 KB/s	44.6%
(4) 3 channels, Single-AP	77.9 KB/s	40.2%
(2) Channel 6, single-AP*	90.7 KB/s	36.4%
MadWiFi driver *	35.9 KB/s	18.0%

Table 2: Avg. throughput and connectivity for Spider configurations. Staying on a single channel and leveraging multiple AP connections provides best avg. throughput. The multi-channel, multi-AP approach provides the best connectivity. (Multi-channel scenarios use a static schedule of 200 ms on ch. 1, 6, and 11. * denotes experiments performed in the Boston area, where Channel 6 was the best.)

We present the average throughput and average connectivity for a Spider node in its four configurations in Table 2. These experiments were performed using a car in downtown Amherst area. A static schedule of $D = 600ms$ and $f_1 = f_6 = f_{11} = 1/3$ was used in multi-channel scenarios.

Two conclusions can be drawn from the results. First, the single-channel multi-AP configuration performs best in terms of throughput. It has an average throughput of more than 4 times that of the single-AP counterpart. The use of multiple channels incurs an additional overhead associating on orthogonal channels, strangling throughput. This is in agreement with the observation made in our analytical framework where we demonstrated that at vehicular speeds, aggregating bandwidth from one channel leads to maximal performance. Second, the multi-channel multi-AP solution has the best performance in terms of connectivity. Although the average throughput is lower, multiple channels host a larger pool of APs for Spider to choose from.

Figs. 10a and 10b are the CDFs of the disruption and connection lengths for different configurations of Spider respectively. The results demonstrate several trade-offs. The longest periods of Internet connectivity are obtained by staying on one channel and maintaining concurrent connections to several APs. However, that strategy also experiences the longest disruptions due to areas where there is no Wi-Fi coverage on the chosen channel. In contrast, the multi-channel multi-AP solution experiences the shortest connections due to disruptions caused by joins on orthogonal channels. However, such an approach has the shortest disruptions due to larger set of possible APs that the node can transfer data with. The single-AP configurations provide trade-offs between the two extremes. We compare these results with usability needs of wireless users in subsection 4.7.

Fig. 10c shows the instantaneous bandwidth that Spider provides when actively transferring data. The single-channel, multi-AP configuration performs best in terms of per-connection throughput. The 60^{th} percentile is around 300 KBps and the 90^{th} percentile is around 1000 KBps — comparable to the throughput provided by FatVAP in a static environment (see Figure 13 in [12]). Spider's multi-channel, multi-AP solution performs poorly in terms of instantaneous bandwidth due to the overhead of association and `dhcp` on separate channels, clearly illustrating (as in our analytical framework) the importance of staying on a single channel if high throughput is the design goal. The higher throughput achieved in the single-channel multi-AP scheme also demonstrates the existence of multiple APs on the same channel within Wi-Fi range of the mobile node. Moreover, the throughput gains of Spider also illustrate that in urban regions the backhaul bandwidth is rarely greater than the wireless bandwidth.

4.4 Effect of AP Density

We evaluated the effect of AP density on the performance of Spider using the same set of experiments listed in Table 2. Here, we compare the configuration where Spider is allowed to associate with one AP with the case where it maintains concurrent connections with multiple APs. During our experiments, Spider associated with a maximum of three APs 5%

Parameters	Failed dhcp	
Chan 1, linklayer: 100ms, dhcp: **600ms**, 7 interfaces	23.0%	±6.4%
Channel 1, linklayer: 100ms, dhcp: **400ms**, 7 interfaces	27.1%	±5.4%
Chan 1, linklayer: 100ms, dhcp: **200ms**, 7 interfaces	28.2%	±4.0%
3 Chans, static 1/3 schedule, linklayer: 100ms, dhcp: 200ms, 7 interfaces	23.6%	±10.7%
Chan 1, **default timer**, 7 interfaces	13.5%	±6.3%
3 Chans, static 1/3 schedule, **default timer**, 7 interfaces	21.8 %	±6.9%

Table 3: dhcp failure probabilities for different timeout configurations for Spider. Primary differences are bolded.

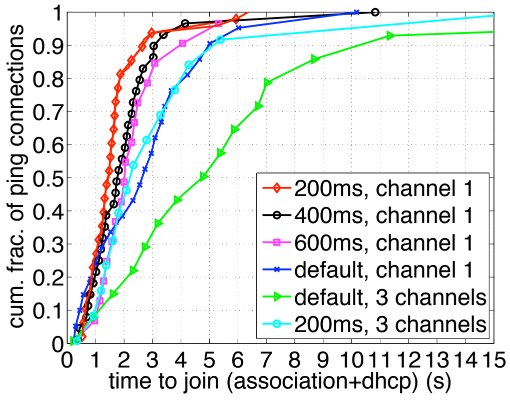


Figure 11: The rate of successful joins as a function of dhcp timeout. The cost of switching among channels overshadows the benefit of quickly establishing connections when timeouts are reduced.

of the time, 2 APs 10% of the time, and is associated with one AP around 85% of the time. Even with such a meager open Wi-Fi density in Amherst, multi-AP Spider has an average throughput that is four times that of a single AP case.

For external validation, we ran a similar set of experiments in the Boston area, an environment with a different mobility pattern and AP density from Amherst. The last two entries in Table 2 are the corresponding results. Interestingly, on Channel 6, Spider has an average throughput that is 800% more than the throughput reported by Cabernet for the same city (a throughput of 10.75 KBps [8])[3]. Additionally, when comparing the results with the stock MadWiFi driver, we find that Spider provides $2.5x$ improvement in throughput and $2x$ improvement in connectivity.

4.5 Effect of Join Timeouts

As discussed in Section 2, one of the primary challenges in designing multi-AP solutions for mobile Wi-Fi access is the overhead associated with dhcp and association. An accepted technique to minimize this overhead is to reduce

[3]It is of course impossible to set up the exact conditions in which Cabernet was tested: 802.11G is now widely available and it is not possible to determine if more or less open APs are available.

Parameters	Throughput	Connectivity
1 channel	121.5 KB/s	35.5%
2 channels (equal schedule)	25.1 KB/s	35.8%
3 channels (equal schedule)	28.8 KB/s	44.7%

Table 4: Average throughput and average connectivity seen when applying different static schedules for multi-channel configuration for Spider.

dhcp and link layer timeouts. Table 3 and Fig. 11 show the effect of reducing these timeouts on the performance of Spider. Table 3 presents the increase in failure rate of dhcp requests with reduced timeouts while maintaining concurrent connections on multiple channels. Compared to the default timers, reducing timeouts can lead to a two-fold increase in dhcp failure rates. Similarly, switching among multiple channels while trying to associate with multiple APs leads to high probability of failure (as high as 30–35%).

Although the number of failed dhcp attempts increases with timeout reduction, in Fig. 11 we find that the median time to successfully obtain a lease improves—similar to the observation made in Cabernet [8]. However, the absolute median time to join is still 2–3 seconds (equivalent to 10-15 TCP timeouts), which increases by 2x when using multiple channels. Hence, it is best to stay on one channel to maximally aggregate throughput.

4.6 Effect of Number of Channels

To understand the effect of number of channels on throughput, connectivity, and join overhead, we present Fig. 12 and Table 4. We tested three scenarios (1) three channels with an equal schedule (200 ms), (2) two channels with an equal schedule (200 ms), and (3) a single channel. Fig. 12 shows that the single channel mode with reduced timeouts performs best in terms of join time–however, the reduced timeouts lead to a large number of dhcp failures. Moreover, the three channel schedule performs worse than the dual channel scheduling. Hence, switching between channels during association is a primary source of overhead in multi-AP solutions. Table 4 presents throughput and connectivity results for the different configurations—as expected throughput is maximized when Spider uses a single channel and connectivity is maximized when it uses an equal schedule on three channels.

4.7 Matching Usability Needs

An open question that is widely asked is *whether open Wi-Fi access can cater to connectivity needs of mobile users?* To answer this question, we performed a study using data from a permanent Wi-Fi mesh we deployed in our downtown. The mesh consists of 25 nodes and covers an area of about 0.50 km^2. We collected performance data on all TCP flows from 161 wireless users for an entire day. Although, all users might not be mobile, the data provides us with a plausible baseline. Overall, there were 128,587 completed TCP connections in the collected data and a total number of 13,645,161 packets

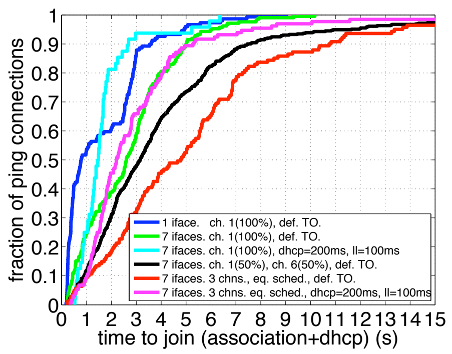


Figure 12: Delay in obtaining a dhcp lease and link layer association for different scheduling policies in Spider. The figure also considers reduced timeouts.

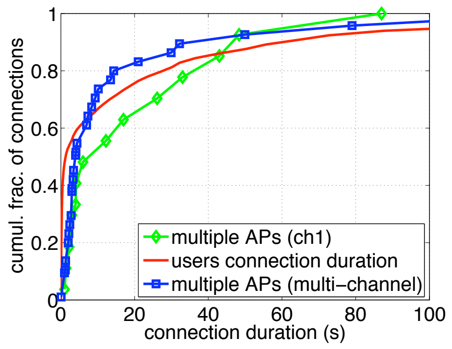


Figure 13: Comparison of connection lengths for wireless users and Spider.

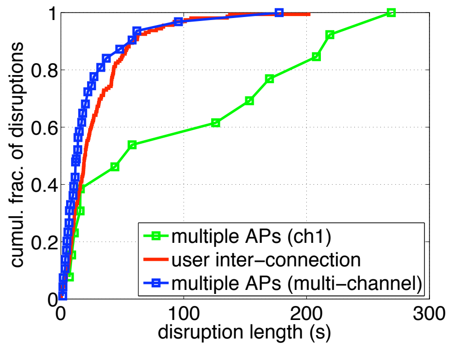


Figure 14: Comparison of disruption lengths for wireless users and Spider.

(1.7 GB) were sent by the users. Of these, 86,838 connections were made to the `http` port (68% of the connections). We compare the traffic needs of wireless users with those provided by Spider based on two key metrics: (1) distribution of the duration of TCP connections, and (2) distribution of inter-connection time.

Fig. 13 compares the TCP flow lengths gathered from actual users using our mesh network and Spider in its multi-channel and single-channel modes. The figure shows that Spider can support all the TCP flows that users need. Additionally, in Fig. 14 we compare the time between two connections for the mesh users and disruption time for Spider. When Spider uses multiple channels and multiple APs, it experiences disruptions comparable to what users can sustain.

These results present Spider as a plausible complement to cellular data services. However, more data on mobile user's connectivity needs and network usage pattern is required in order to find out the degree to which Spider can *align* itself with the needs of each individual user.

4.8 Limitations

There are few limitations in our presented study. First, while we have demonstrated that mobile users at vehicular speeds see better performance if they connect to multiple APs on a single channel, Spider does not dynamically determine the best channel to dwell on. Exploring optimal channel selection schemes that use AP density and offered bandwidth on orthogonal channels at different locations requires future work. Second, unlike FatVAP [12], Spider does not explicitly load-balance across APs. However, a simple optimization where Spider assigns traffic to APs proportional to the available end-to-end bandwidth would assure that the system is fair across APs. Spider could further be augmented by the addition of more sophisticated criteria for AP selection. In addition, investigating the effect of multi-AP systems on energy consumption of constrained devices as well as exploring potential problems raised by interference as more users adopt concurrent Wi-Fi schemes require future work.

5. RELATED WORK

Spider builds on previous work on mobile Wi-Fi, fast handoffs, and aggregating throughput from multiple APs.

Wi-Fi access from moving vehicles: Several challenges such as lossy wireless mediums [4, 8, 10, 14], tuning TCP performance for mobility [1], and AP selection [18] are well studied. Caching history has been used to reduce association and `dhcp` overheads [6]. Directional antennas have also been used to improve throughput [16]. However, this body of work assumes a stock Wi-Fi configuration, connecting to one AP at a time. We find that connecting to multiple APs can improve aggregate throughput and connectivity for mobile nodes.

Performance through diversity: Technological and spatial diversity have also been leveraged to improve performance. This class of work can be broadly classified according to whether infrastructure or clients are modified. Infrastructure modifications include coordination or selection among multiple open APs [13–15, 25]. In contrast, Spider is a purely client-side solution that aims at improving performance for mobile users in *organic* Wi-Fi settings. *Client-side* diversity-based solutions rely on aggregating bandwidth across multiple APs [5, 9, 12, 19]. However, previous solutions are tuned to work efficiently in *static* settings, with stationary clients.

An orthogonal approach to connecting to multiple APs with a single Wi-Fi card is using additional hardware—it is assumed that each client has more than one card and that data can be striped across concurrent connections [20, 22, 24]. Most of these approaches could be added to Spider to improve performance. In addition, resource-constrained devices which cannot feasibly be equipped with additional hardware can still benefit from virtualized solutions like Spider.

Soft hand-off and AP selection: Spider also builds on related work on fast cellular hand-offs and AP selection. Soft hand-offs mitigate the adverse effects of disruptions in cellular networks [7]. While this is feasible in cellular networks where towers are under the control of a central authority, the technique is not applicable in organic Wi-Fi settings with APs administered by third-party users. The only practical client-based soft hand-off approach is the one employed by Spider: virtualize the card and maintain concurrent connections.

Proposed solutions to AP selection for mobile nodes utilize RSSI [11] and history [17]. However, the problem of multi-AP selection that Spider addresses is more difficult (as we demonstrate in our technical report [23]) since it involves selecting a *set* of APs.

6. CONCLUSION

We presented the first in-depth analysis of the performance of attempting concurrent AP connections from highly mobile clients. Through an analytical model, optimization framework, and numerous indoor and outdoor experiments, we isolated several factors that affect the poor performance of multi-channel networking in contrast to single-channel, multi-AP networking and conventional approaches. We found that connection duration, AP response times, channel scheduling, and attained and offered bandwidth all affect performance. Moreover, link association, `dhcp` joins, and TCP are all negatively affected by fractional channel scheduling.

Building off these results, we designed and implemented a novel multi-AP driver. Spider uses utility-based multi-AP selection, channel-based scheduling, and opportunistic scanning to maximize throughput while mitigating the overhead of association and `dhcp`. While Spider manages multiple channels if desired, we show empirically that using multiple APs on a single channel achieves higher throughput than scheduling on multiple channels, as predicted. Our evaluation shows that Spider provides a 400% improvement in throughput and 54% improvement in connectivity over stock Wi-Fi implementations, making it an effective supplement to cellular data services for highly mobile nodes.

7. ACKNOWLEDGMENTS

This work is supported in part by NSF awards CNS-0519881, CNS-0447877, CNS-0720717, CNS-1018112, CNS-1055061, and CNS-1115728. Any opinions, findings, and conclusions expressed in this work are those of the authors and do not necessarily reflect the views of the NSF.

We thank our shepherd, Paolo Giaccone, for his helpful feedback.

8. REFERENCES

[1] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *ACM SIGCOMM*, pages 256–269, 1996.

[2] A. Balasubramanian, R. Mahajan, and A. Venkataramani. Augmenting Mobile 3G Using WiFi: Measurement, Design, and Implementation. In *Proc. ACM Mobisys*, 2010.

[3] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive WiFi Connectivity for Moving Vehicles. In *ACM SIGCOMM*, August 2008.

[4] V. Bychkovsky, B. Hull, A. K. Miu, H. Balakrishnan, and S. Madden. A Measurement Study of Vehicular Internet Access Using in situ 802.11 Networks. In *Proc. ACM MOBICOM*, pages 50–61, Sept 2006.

[5] R. Chandra, P. Bahl, and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *Proc. IEEE INFOCOM*, March 2004.

[6] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive Methods for Improved Vehicular WiFi Access. In *Proc. ACM Mobisys*, 2009.

[7] N. Ekiz, T. Salih, S. Kucukoner, and K. Fidanboylu. An overview of handoff techniques in cellular networks. In *Intl. Journal of Information Technology*, 2006.

[8] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *Proc. ACM MobiCom*, Sept. 2008.

[9] D. Giustiniano, E. Goma, A. Lopez, and P. Rodriguez. Wiswitcher: an efficient client for managing multiple aps. In *ACM SIGCOMM workshop on Programmable routers for extensible services of tomorrow (PRESTO)*, pages 43–48, 2009.

[10] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. CarTel: A Distributed Mobile Sensor Computing System. In *ACM SenSys*, October 2006.

[11] G. Judd and P. Steenkiste. Fixing 802.11 access point selection. In *ACM CCR*, 2002.

[12] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: aggregating AP backhaul capacity to maximize throughput. In *Proc NSDI*, pages 89–104, 2008.

[13] V. Leung and A. Au. A wireless local area network employing distributed radio bridges. *Wirel. Netw.*, 2(2):97–107, 1996.

[14] A. Miu, A. Miu, H. Balakrishnan, C. Emre, K. Mit, and C. Science. Improving Loss Resilience with Multi-Radio Diversity in Wireless Networks. *Proc. ACM Mobicom*, pages 16–30, 2005.

[15] A. Miu, G. Tan, H. Balakrishnan, and J. Apostolopoulos. Divert: fine-grained path selection for wireless LANs. In *Proc. ACM MobiSys*, pages 203–216, 2004.

[16] V. Navda, P. Subramanian, K. Dhanasekaran, A. Timm-giel, and S. Das. Mobisteer: Using steerable beam directional antenna for vehicular network access. In *Proc. ACM Mobisys*, 2007.

[17] A. Nicholson, Y. Chawathe, M. Chen, B. Noble, and D. Wetherall. Improved access point selection. In *Proc. MobiSys*, 2006.

[18] A. Nicholson and B. Noble. BreadCrumbs: forecasting mobile connectivity. In *Proc. ACM Mobicom*, 2008.

[19] A. Nicholson, S. Wolchok, and B. Noble. Juggler: Virtual Networks for Fun and Profit. In *IEEE Trans. Mobile Computing*, 2009.

[20] A. Qureshi and J. Guttag. Horde: separating network striping policy from mechanism. In *Proc. ACM MobiSys*, pages 121–134, 2005.

[21] A. P. Release. AT&T Launches Major Wi-Fi Initiative to Deploy More Hotzones in Key Markets, December 2010.

[22] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. Mar: a commuter router infrastructure for the mobile internet. In *Proc. ACM MobiSys*, pages 217–230, 2004.

[23] H. Soroush, P. Gilbert, N. Banerjee, B. N. Levine, M. D. Corner, and L. Cox. Spider: Improving mobile networking with concurrent wi-fi connections. *UMass UM-CS-2011-016 Technical Report*, 2011.

[24] N. Thompson, G. He, and H. Luo. Flow Scheduling for End-host Multihoming. In *Proc. IEEE INFOCOM*, 2006.

[25] A. Viterbi, A. Viterbi, K. Gilhousen, and E. Zehavi. Soft Handoff Extends CDMA Cell Coverage and Increase Reverse Link Capacity. In *Proc. Intl. Zurich Seminar on Digital Communications*, pages 541–551, 1994.