Improving Mobile Networking with Concurrent Wi-Fi Connections

Hamed Soroush* Peter Gilbert† Nilanjan Banerjee△
Brian Neil Levine* Mark Corner* Landon Cox †
*Dept. of Computer Science, Univ. of Massachusetts Amherst, {*hamed,brian,mcorner*}*@cs.umass.edu*
† Dept. of Computer Science, Duke Univ., {*gilbert,lpcox*}*@cs.duke.edu*
△ University of Arkansas, Fayetteville *nilanb@uark.edu*

ABSTRACT

Achieving ubiquitous connectivity or high aggregate throughput using a series of Wi-Fi access points can cause serious performance problems for highly mobile clients. While data transfer using a single access point at a time is insufficient for supporting delay- and throughput-sensitive applications, leveraging concurrent data transfers through multiple access points offers a plausible alternative. To this end, we design, implement, and evaluate a system that establishes and maintains concurrent connections to 802.11 APs in a mobile environment. We demonstrate that existing multi-AP solutions do not perform well in mobile settings due to limitations imposed by the association and `dhcp` *processes and propose an alternative design. We have implemented and evaluated our system on a vehicular testbed. Our results show that our system provides manifold improvements in throughput and connectivity over stock WiFi implementations. We compare our results against measurements of real user needs in mobile settings in terms of application use and desired session length.*

1. INTRODUCTION

Cellular infrastructure provides the most common method of mobile access today because it offers high bandwidth and long-range coverage. Unfortunately, in areas where cellular demand is high, the deployed capacity cannot keep up. Carriers are now imposing caps on bandwidth and differentiated plans. In response, users and networks wish to supplement their connections with Wi-Fi networks. And recent measurement work [2] has demonstrated 3G data can be offloaded successfully.

Unfortunately, these same measurements show Wi-Fi bandwidth is often a fraction of 3G. Wi-Fi has other disadvantages compared to 3G infrastructure. Wi-Fi is short range. It is in an unlicensed band that is often crowded. It can be deployed as a mesh, but to do so over large areas is expensive. Moreover, its deployment is often ad-hoc, offered by independent homes and apartments. On the other hand, Wi-Fi is a cheap and common interface in mobile devices, and when it is available, it is often free and available on several different channels at once [21]. Additionally, many people choose to use Wi-Fi intermittently rather than pay 3G fees. Whether used as the best choice for an auxiliary channel for 3G, or used as a stand-alone option for access, a critical question is how can we improve Wi-Fi throughput for mobile clients?

An effective approach for improving Wi-Fi performance is to split a single physical wireless interface into several virtual interfaces that each connect to a different AP. This idea was first proposed by Chandra et al. [19]. Kandula et al. [10] proposed *FatVAP*, which includes a scheduler that distributes the client's time across APs so as to maximize throughput; they improve throughput by $3x$. Similarly, Nicholson et al. [17] demonstrated significant bandwidth gains with *Juggler* for mobile Wi-Fi clients. These gains are possible because these systems rely on a driver that can switch between APs by telling one AP that it is in power save mode and then moonlighting on another AP. These drivers attempt to keep the switching delay small, making sure the deadline for retrieving stored packets at the first AP is met so that TCP flows are not disrupted.

It seems straightforward to directly apply virtualized Wi-Fi drivers to mobile 3G and Wi-Fi users. Unfortunately, these past works have based their designs and evaluations on *stationary* wireless clients. In this paper, we evaluate virtualized Wi-Fi for *highly mobile* clients, and we demonstrate that it is not a straightforward win. When Wi-Fi is aggregated across multiple channels, scheduling is an NP-hard problem for mobile clients, which must consider association joins and `dhcp` requests that cannot be delayed by the power-save maneuver. As we show, the best application of virtualized Wi-Fi is in areas where the density of APs is so high that all the benefits of bandwidth aggregation can be had from APs on a single Wi-Fi channel. Our measurements show that, otherwise, the benefits of standard drivers that have agile serial switching between APs (and are simpler to optimize from a coding standpoint) can be the better choice.

Specifically, we offer the following contributions in this paper:

- We present the design and implementation of *Spider*, a system that aims at improving the performance of highly mobile networks by providing concurrent 802.11 connections to open access points. Spider achieves this goal by virtualizing a wireless card and efficiently managing connections to APs through each virtual interface. In contrast to past works which use a per-connected-AP granularity to schedule concurrent connections, Spider schedules 802.11 channels. We argue that this design is more suitable for highly mobile scenarios where it is critical to quickly join to nearby access points.
- We quantify the costs of link-layer association and `dhcp` on virtualized Wi-Fi, demonstrating that scheduling on multiple channels can thwart successful joins to APs and limit TCP performance. We also demonstrate that reducing timeouts [7] lowers join success rates when virtualized Wi-Fi is used.
- We experiment with numerous link management policies in Spider using vehicles driven in two cities that offer intermittent Wi-Fi access. The results show that mobile virtualized Wi-Fi does not maximize both throughput and connectivity simultaneously. We show that if connectivity is a priority, then joining to multiple APs on multiple channels is best: 44% connectivity, 28KBps. However, if throughput is a priority, then joining to multiple APs on only one channel is best at the cost of connectivity: 35% connectivity, 122KBps. In the middle is a non-virtualized policy of one AP at a time from any channel available: 40% connectivity, 77KBps.

We begin with experiments that demonstrate the effects of virtualized Wi-Fi scheduling on link-layer association and `dhcp`, motivating the problems we address in this paper. We then provide the details of Spider's design and implementation. Next, we present the results of extensive experimentation with several virtualization policies. Finally, we offer a review of related work and our conclusions.

2. MULTI-AP CHALLENGES FOR MOBILE CLIENTS

Concurrent connections between a client and multiple APs [10, 17, 19] are possible for Wi-Fi because the APs can be instructed by the client to buffer packets. The client falsely claims it is entering the power-save mode (PSM) and then communicates with another AP. What past works overlook is that clients in a mobile Wi-Fi environment must continuously associate and obtain `dhcp` leases from APs as they become available. These

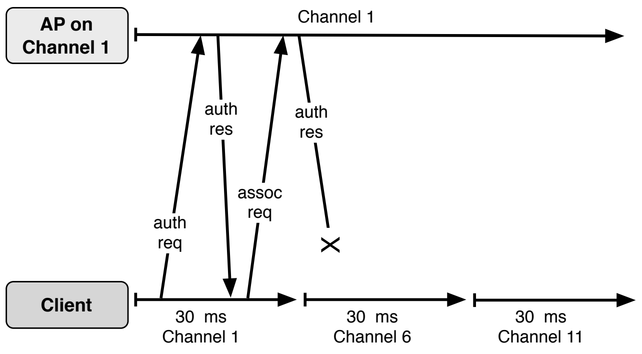

Figure 1: The Wi-Fi card spends 30ms each on Channels 1, 6, and 11. During an association phase the card loses "association response" from the AP due to channel switching.

packets cannot be buffered by the PSM request. As we show, a schedule that maximizes aggregate bandwidth for already-joined APs can prevent successful association and `dhcp` requests, which are critical components of these AP hand-offs. Accordingly, past works on virtualized Wi-Fi all experiment with their mechanisms in stationary contexts. In particular, these past experiments consider throughput and other characteristics only after permanent connections to APs have been formed.

In this section, we quantify the performance penalties virtualized Wi-Fi imposes on association and `dhcp` requests in an outdoor mobile setting. We also experiment with reduced link layer and `dhcp` timeout values [7], showing that they increase the number of failed connection attempts.

2.1 Association and DHCP

Figure 1 illustrates one of the major problems that makes virtualized Wi-Fi difficult in a mobile scenario. Suppose that according to the driver's schedule, the Wi-Fi card divides a fixed interval among three APs each on one of the three non-overlapping Wi-Fi channels (1, 6, and 11). The association process involves a four-way handshake: authentication request, authentication response, association request, and association response. A failure will occur if the client switches channels before receiving the final response.

After a client associates, the AP can be asked to buffer non-broadcast packets. Unfortunately, the four messages to complete a `dhcp` request (discover, offer, request, and ack) are broadcast messages, and delays in the AP's response can be relatively long and difficult to predict. Hence, if a client switches channels before obtaining a `dhcp` lease, the process must restart. A timeout during association or `dhcp` triggers a seconds-long back-off, which is catastrophic given the typical duration of mobile Wi-Fi access (typically 10–20 seconds

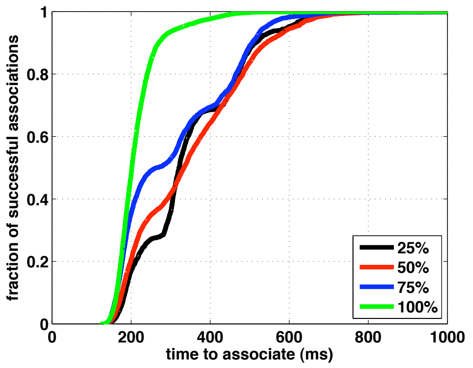

Figure 2: The rate of successful link-layer associations on a channel as a function of the amount of time the Wi-Fi driver spends on a single channel (out of 400ms for all channels).

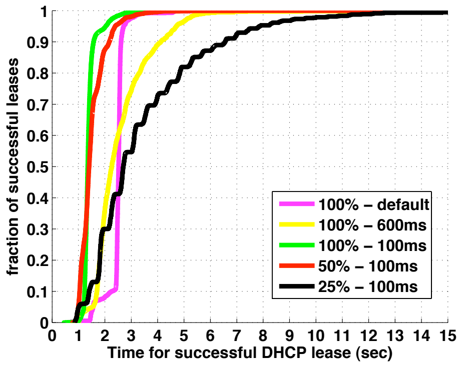

Figure 3: The rate of successful dhcp requests on a channel as a function of the amount of time the Wi-Fi driver spends on that channel as well as the dhcp timeout.

for a vehicle). If there were no delays for association or dhcp, then virtualized Wi-Fi would be an obvious win for any number of channels. Erikson et al. [7] have shown that these delays can be minimized in non-virtualized Wi-Fi by reducing the timeouts used in these protocols. We evaluate these reduced timer settings below; we find that reduced timeouts can increase dhcp failure rates and overall time to get a lease when the driver spends a portion of its time on other channels.

We performed a series of experiments to quantify how channel scheduling settings can severely degrade the probability of successfully associating and obtaining a dhcp lease. Each experiment was performed on five different vehicles moving around Amherst, MA for 6 hours each. The wireless card of each mobile node was set to spend a fraction x of a 400ms round-robin scheduling interval on channel 6, and $(1-x)/2$ of the interval on each of channels 1 and 11. Also, we changed the link layer association timeout value from a default of 5 seconds to 100ms for these experiments.

Whether evaluating association or dhcp, we find that the success is strongly positively correlated with the amount of time scheduled on the primary channel. Figure 2 shows empirical CDFs of these experiments. When a single channel receives 100% of the interval, the median association time is 200ms, and all associations complete within 400ms. With a small drop to 75% of the 400ms schedule, we see a sharp decrease in performance: the median association time drops to 300ms and only 75% of associations completing within 400ms. Note that in all cases, it is possible to connect in under 200ms, but the success rate is below 20%.

dhcp performance is more sensitive to the schedule, as shown in Figure 3. Of course, dhcp requests are dependent on successful association during the experiment, and we call a *join* success in both cases. When 100% of the schedule is spent on a single channel, the median join time decreases from 2.5s for the default dhcp packet timeout of 1 sec, to a median join time of 1.3s for 100ms dhcp timeouts. Unfortunately, the situation is more complicated for values above the median. At the 90^{th} percentile, the default dhcp packet timeout achieves 2.6s joins; but the 90^{th} percentile is worse at 4.1s for 600ms time outs, and then shorter again at 1.6s for 100ms timeouts. These swings can be explained by a sharp increase in completely failed dhcp requests. Moreover, the same graph shows the effects of using a reduced schedule on the channel. With only 50% of the schedule, the 90^{th} perc. join times drop to 2.2s, and then down to 6.6s at 25%. Hence, reduced timeouts *increase* failure rates and overall join time when the driver spends a portion of its time on other channels.

We further evaluate the effects of reduced timeouts on the performance of our driver in conjunction with the overheads of managing multiple virtual interfaces in Section 4.

2.2 TCP and UDP performance

In a multi-AP setting, the throughput of a UDP stream is proportional to the amount of time spent by the card communicating with an AP. For TCP streams, this relationship is not straightforward due to non-deterministic effects of TCP timeouts and slow start. The experiments in this subsection took place on a desktop to prevent mobile factors from affecting performance; in Section 4 we report TCP throughput for vehicular experiments. Note that previous works on virtualized Wi-Fi do not quantify how scheduling affects transport performance since they do not consider schedules that might be sufficient to allow joining new APs on other channels.

Figure 4 shows the effects of channel scheduling on TCP. We vary the *percentage of time* out of 400ms that the card spends on a primary channel just as in Section 2.1. In other words, the card spends between 0 and 400ms on the primary channel. As expected, varying

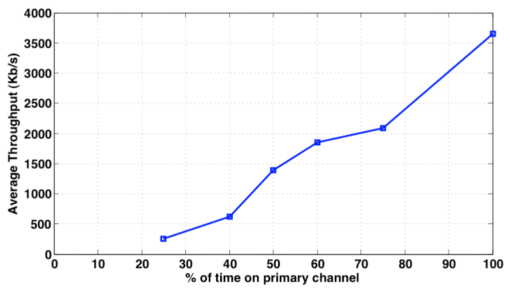

Figure 4: Average TCP throughput as a function of the *percentage of time* spent by the Wi-Fi driver on the primary channel. Since the cumulative time spent on all the channels is 400 ms (which is less than two RTTs) the throughput is proportional to the percentage of time spent on the primary channel.

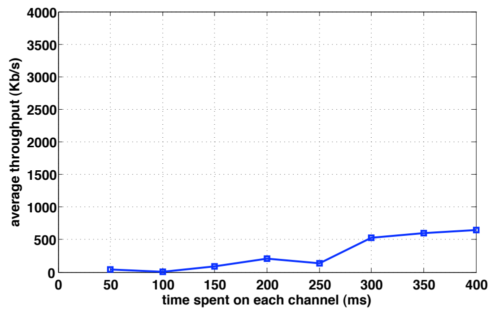

Figure 5: Average TCP throughput as a function of the *absolute time* spent on each channel. The throughput is very sensitive to the amount of time spent by the driver on each channel due to TCP timeouts and TCP slow start.

the percentage of time spent on the primary channel leads to a monotonic improvement in throughput. Since the cumulative time spent on all the channels is 400ms the throughput is proportional to the percentage of time spent on the primary channel; 400ms is less time than two RTTs (in this experiment), which would have triggered a TCP timeout and slow start.

In a second experiment, we vary the *absolute amount of time* the card spends on a primary channel, shown in Figure 5. In other words, a point on the x-axis corresponds to a schedule where the card spends x ms on each of channels 1, 6, and 11; hence the total time spent away from the TCP channel is $2x$, ranging from 0 to 800ms.

As we vary the amount of time the Wi-Fi card spends on each channel, the throughput varies more wildly. Due to the interaction between the RTT of TCP packets and the card schedule, packets from the AP to the client can get lost. Hence, TCP times out or gets stuck in slow start, strangling throughput.

TCP/UDP sensitivity to the percentage of time spent on a channel presents a tension between throughput and connectivity. For example, suppose the Wi-Fi card is transferring data with two AP on channel 1 and starts to associate with two other APs on channel 6. For the associations to be successful, the card must spend more than 80% of the time on channel 6. However, this implies that TCP/UDP throughput on channel 1 would degrade substantially. Addressing this tension between throughput of ongoing TCP/UDP streams with successful associations and `dhcp` attempts is the fundamental problem of applying multi-AP solutions to the mobile environment.

3. Spider

Spider is designed to leverage concurrent 802.11 connections to improve performance in highly mobile networks. To this end, we have made several design choices that distinguish Spider from solutions that either target stationary wireless settings [10, 17, 19] or enhance stock Wi-Fi networks [7].

Design Choice 1: In contrast to previous work that slices time across *individual APs* [10,17,19], Spider schedules a physical Wi-Fi card among 802.11 *channels*. Per-channel queues incur no switching overhead for interfaces on the same channel. Staying on one channel also mitigates the problem of failed associations discussed in the previous section since broadcast packets (which are not buffered in power-save mode) can still be received if the virtual interface is on the right channel. Moreover, it allows Spider to *opportunistically scan* for new APs without losing connectivity to the old ones. Communicating with multiple APs on the same channel can amplify the hidden terminal problem, but as we show in Section 4 aggregate throughput still increases.

Design Choice 2: Spider addresses both *selection of multiple APs* and *Wi-Fi card scheduling*. Both problems are NP-Hard (see Appendix A and [10]). It is possible to design optimal dynamic programming solutions for them. However, we found that using heuristics is more effective in practice given that short encounters makes it difficult to solve the related equations in real-time. Spider can use pre-defined static channel schedules or dynamic schedules that adapt based on ongoing associations and data transfers. The AP selection heuristic of Spider is discussed later in this section.

Design Choice 3: Unlike Juggler [17], Spider does not hide the existence of multiple virtual interfaces from applications. Spider exposes a separate Linux network device interface for each connection, allowing maximal flexibility in the way that applications may use concurrent Wi-Fi connections. Spider does not require a customized kernel or modification to the kernel socket data structure [17]. Finally, Spider is a completely standalone loadable module that is compatible with off-the-shelf wireless utilities such as `iwconfig, iwlist, iwspy` and `iptables`, making link management and network debug-

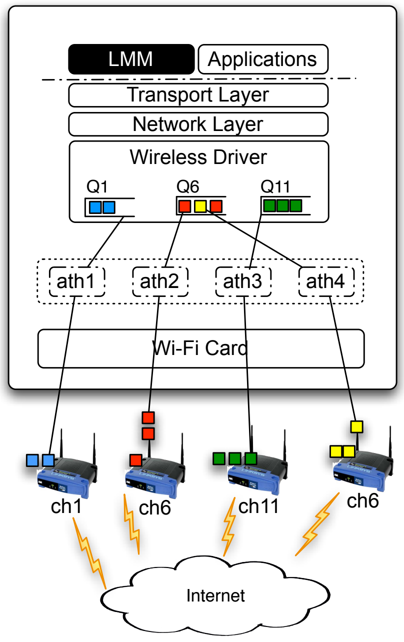

Figure 6: An illustration of Spider's components.

ging easier.

Spider works in both ad-hoc and infrastructure modes, though our focus is the latter. The overall design of Spider is illustrated in Figure 6. Spider has two major components: a *customized wireless device driver* and the *link management module* (LMM).

3.1 Wireless Driver

Like FatVAP [10], Spider customizes the popular MadWifi driver to present multiple "virtual" network interfaces. Similar to Juggler [17], Spider's driver exposes a `/proc` file-system interface through which parameters such as the channel-switching schedule and various timeout values can be configured by a link management module running in user space. The key differences between Spider's driver are 1) Spider's capability to schedule multiple APs on a channel concurrently rather than individual APs, and 2) Spider's support for opportunistic scanning, which allows new APs to be discovered without sacrificing data-transfer opportunities.

3.1.1 Switching Among Channels

Spider's design is intended to maximize connectivity opportunities in mobile environments without sacrificing throughput: Spider can attempt to associate and obtain `dhcp` leases from many APs on a channel in parallel, without dedicating the wireless card and giving up opportunities to transfer data through other APs on that channel. In this way, Spider can overlap the long and unpredictable delays experienced while waiting for responses from the AP during the association and `dhcp` phases. Accordingly, the procedure for switching channels is slightly different than in other multi-AP solutions.

- First, the client starts buffering outgoing packets on the virtual interfaces communicating on the current channel.
- Next, the client sends to each AP that it is associated with on the current channel an 802.11 management frame with the PSM bit set, indicating that it is entering power-save mode. This causes the APs to buffer all frames destined for the client until it returns to the channel.
- After deactivating the virtual interfaces on the previous channel, the client changes the state of the wireless card to the new channel. A hardware reset is required to apply the change.
- Finally, the set of interfaces which are associated or attempting to associate with APs on the new channel are activated. A management frame with the PSM bit cleared is sent out over each virtual interface associated on the new channel, and the outgoing packets stored in the PSM queues of the newly activated virtual interfaces are submitted to the card's transmit queue.

3.1.2 Opportunistic Scanning

To maintain connectivity, mobile clients must be able to discover new APs as they move. Typically, clients scan for new APs continuously when not associated by cycling through the available channels, transmitting probe requests and listening for responses. Even after successfully associating with an AP, a traditional wireless driver may periodically switch away from the AP for tens or hundreds of ms to scan on other channels. The multi-AP solution Juggler [17] allows "scanning slots" to be added to the switching schedule, during which the card rotates among channels 1, 6, and 11 and listens for beacons. Unfortunately, these approaches sacrifice time which could be spent transferring data or associating with an AP.

To maximize the time available for useful work, Spider scans opportunistically in the background without disrupting ongoing connections. While associated with an AP, a client often receives beacons and probe responses transmitted by other APs on the same channel—wireless drivers typically drop these frames early in the receive process to avoid additional processing overhead. Spider instead accepts these frames and maintains a list of APs which it has heard from recently. To maximize the probability that we receive a response from each AP, Spider can be configured to periodically broadcast probe requests.

3.2 Link Management Module

Spider's link management module implements connection establishment and management policies. This module is responsible for applying AP selection policies, managing concurrent connections, detecting lost connections and establishing new ones, as well as notifying applications of the availability of a link.

The link management module creates a configurable number of virtual interfaces on boot-up and sets an appropriate channel schedule given a specific *operation mode*. Each operation mode is defined by the total amount of time to be scheduled among channels, as well as the fraction of time which is to be spent on each channel. The link management module provides support for dynamically changing the schedule; however, static schedules configured before boot-up are used as default. We present results from experiments using different operation modes in Sections 2 and Section 4.

After the initial setup phase, a thread is created for each interface to perform link discovery and management operations for that interface. Software locks are used to ensure that no two threads select and attempt to connect to the same AP. Furthermore, threads safely share their connection history and estimated rank of the APs. As soon as an interface joins a network and obtains an IP address, corresponding `iptables` rules are set to allow routing traffic to and from that specific interface. If the same IP address is assigned to different virtual interfaces by different APs, we only use the most recently assigned interface with that IP address, bring down those other interfaces, and then attempt to connect them to different APs. However, we observed that such events were rare.

Similar to Cartel [8] and Cabernet [7], we have incorporated specific optimizations in our implementation of the link management module such as caching `dhcp` leases and configuring link-layer and `dhcp` timeouts. Upon a successful link-layer association, the corresponding thread first consults its per-BSSID cache of `dhcp` leases before issuing `dhcp` requests. Furthermore, after a connection is established, the thread continuously pings a known host to monitor the availability of the connection. In case an AP does not allow `icmp` pings to propagate pinging the gateway is used to establish connectivity. If 30 ping attempts fail (sent at a rate of 10 pings per second), Spider assumes that the connection is dropped, notifies the application, and tries associating with another AP. Spider notifies applications of connectivity loss using a shared flag resident on the system's RAM disk.

3.2.1 Access Point Selection & Scheduling

Spider's link management module deals with selecting a set of available APs and scheduling time of the Wi-Fi card. In Appendix A, we show that the multi-AP selection problem while maximizing a given system utility function is NP-Hard. Since Spider selects from the power-set of available APs, the complexity of an optimal dynamic programming solution grows exponentially.

Therefore, to make the problem tractable, Spider uses the following heuristic to select APs: Each AP, i, in a Wi-Fi, is assigned a utility U_i. U_i is a function of the number of successful *join* attempts the client has made with the AP previously. A successful *join* consists of three phases: *i*) link-layer *association*, *ii*) *dhcp* lease acquisition, and *iii*) end-to-end *connectivity* test as described in Section 3.2. Spider assigns fixed values v_a, v_b, or v_c ($v_a < v_b < v_c$) to the each specific join attempt depending on how far it succeeds in the association process. Failed associations are assigned a zero value. Utility U_i of each AP i are the weighted average of previous and recent association attempts made with the AP—the recent association are given larger weights. Each management thread of the link layer module selects the AP with highest utility which has not already been selected by other threads. It uses signal strength to break ties when APs have the same utility.

The link management module can be configured to use other criteria such as the amount of time the AP is in data-transfer range with the client or the amount of data transferred during a connection as the utility function. However, by default it uses *join success* as discussed above since it is a primary determinant of AP usefulness in highly mobile scenarios and is stable compared to bandwidth measures that can vary with distance, mobility, and environmental factors. The link management module can be configured to dynamically change the channel switching schedule by estimating the amount of time required by each ongoing connection depending on its state (association, dhcp, data-transfer). However, pre-defined static channel schedules proved to be more stable and therefore, are used in Spider's evaluation.

4. EVALUATION

We envision Spider as a solution that complements cellular data services by providing for an effective use of concurrent 802.11 connections to open APs in highly mobile scenarios. Such a solution targets applications that do not justify the per-month per-device cost of subscribing to cellular data plans. Furthermore, even in cases that cost is not an issue, open Wi-Fi solutions like Spider are a natural supplement to cellular networks due to their higher *fundamental capacity*. Their drawback, however, is that they do not provide continuous connectivity.

Spider aims at improving throughput and connectivity for mobile clients using open Wi-Fi access by synergistically using multi-AP selection, channel switching, opportunistic scanning, and parallel per-channel association. Here, we evaluate the performance of Spider by focusing

# interfaces	0	1	2	3	4
Mean	4.942	4.952	5.266	5.546	5.945
STD	0.009	0.009	1.236	0.823	1.121

Table 1: Channel switching latency (ms)

on the following key questions:

- What improvement in throughput and connectivity does Spider provide over stock Wi-Fi access?
- What is the effect of AP density on Spider's performance?
- Does Spider meet connectivity needs of common wireless users?

While answering these questions, we also present several micro-benchmarks and explore trade-offs available when setting different `dhcp` and link-layer timeouts.

4.1 Experimental Setup

We have evaluated Spider on UMass DOME [21], an outdoor vehicular testbed designed for mobile experimentation. Most of our experiments take place on DOME public transit buses.We also performed experiments on a passenger car using the same hardware equipments as DOME. This allowed us to focus experiments in downtown area of Amherst, MA which has Wi-Fi coverage that is more typical of a city. We also conducted experiments using the passenger car in Cambridge, MA area, as we detail below. In all cases, the equipment consisted of a Hacom OpenBrick 1GHz Intel Celeron M system running Ubuntu Linux 2.6.18 and Atheros 802.11abg MiniPCI Wireless Card.

The analysis of traces collected from performing initial experiments verified our conjecture that the majority of the APs are on either of the three "orthogonal" channels 1 (28%), 6 (33%), or 11(34%). Cabernet [7] reports comparable numbers for the Boston area with 83% of the APs on either of the three channels and 38.5% using channel 6. Due to this non-uniformity in the distribution of the APs over channels, we have configured Spider to only schedule among these three channels in the following experiments.

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

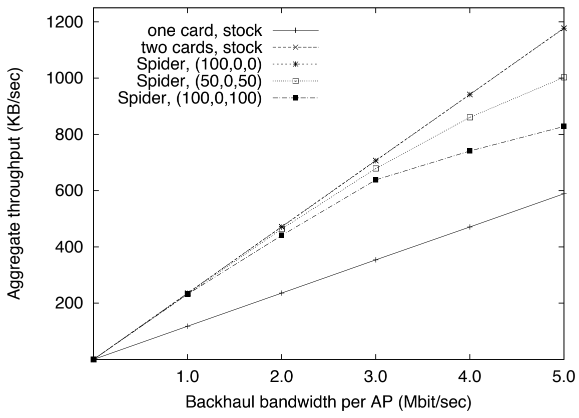

Figure 7: Throughput micro-benchmark

4.2 Driver Micro-benchmarks

We present the results of two micro-benchmarks 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 of a channel switch operation and the standard deviation. 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 proportional to the number of APs,because a separate PSM frame must be sent to each AP. The largest contributor to the latency is the hardware reset step, which can vary depending on the model of card. The latency is within a few ms of that achieved by other multi-AP drivers [10, 17].

Fig. 7 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 connected to one AP on channel 1 and one on channel 11, with a schedule of 50ms on each channel; and the previous configuration 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 both achieved 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 and does not run the risk of causing TCP timeouts by

(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) Multi-channel, Multi-AP	28.8 KB/s	44.6%
(4) Multi-channel, 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%

Figure 8: 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 Cambridge, where channel 6 was the best.)

going off-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*: amount of data transferred to a sink per unit time over an experiment. (2) *Average connectivity*: 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*: contiguous period of time when there is no connectivity. This distribution indicates whether Wi-Fi can support interactive applications such as VoIP or web search. (4) *Instantaneous bandwidth*: amount of data per second transferred by a Spider node when there is connectivity. The instantaneous bandwidth indicates whether multi-AP solutions can support applications that require bursts of high throughput connectivity.

We present the average throughput and average connectivity for a Spider node in its four configurations in Fig. 8. These experiments were performed using a passenger car around downtown Amherst. From the table we draw two conclusions. 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 of switching and associating on orthogonal channels, strangling throughput. Second, the single-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. 10 and 9 are the CDFs of the disruption and

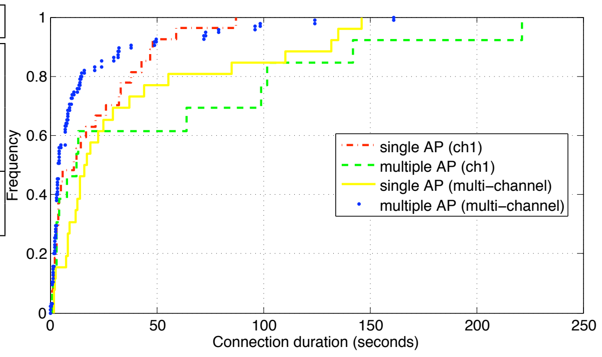

Figure 9: CDF of the Internet connectivity duration for Spider configurations.

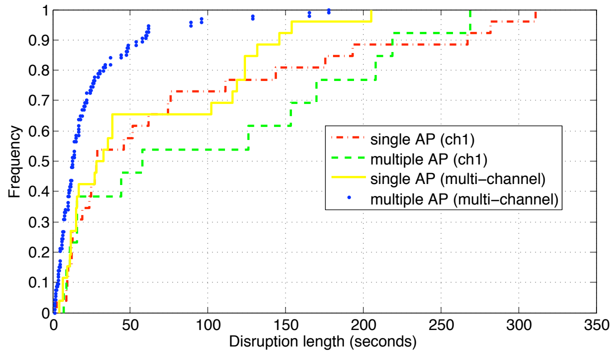

Figure 10: CDF of disconnectivity for Spider configurations.

connection lengths for different configurations of Spider respectively. The results demonstrate several interesting 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 multiple-channel multi-AP solution experiences shortest connections due to disruptions caused by the time it takes to join APs on separate channels. The single-AP configurations provide trade-offs between the two extremes. We compare these results with the needs of wireless users in Section 4.7.

Fig. 11 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 [10]). 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 the importance of staying on a single channel if high throughput is the design goal.

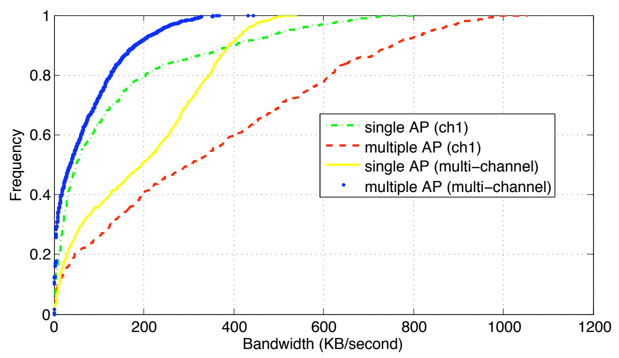

Figure 11: CDF of KB/s transferred (during connectivity) for Spider configurations. Single-channel configs. provide the best instantaneous throughput. Multi-AP, multi-channel reduce throughput due to the overhead of joining.

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%

Figure 12: `dhcp` failure probabilities for different timeout configurations for Spider. Primary differences are bolded.

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 8. Here, we compare the case in which Spider is allowed to associate with one AP with the case where it maintains connection with multiple APs. During our experiments, Spider associated with a maximum of three APs 5% of the time, 2 APs 10% of the time, and is associated with one AP around 85% of the time. It is notable that, even with such a meager open Wi-Fi density in Amherst, multi-AP Spider has a average throughput that is four times that of a single AP case.

We also used Spider in a set of experiments performed in Cambridge with a twofold goal: First, for external validity, we wanted to test Spider in an environment with a different mobility pattern and AP density. Second, we intended to compare our results with those obtained by Cabernet in the same area. The last two entries in Table 8 are results for experiments conducted in Cambridge. It is of course impossible to set up the exact conditions in which Cabernet was tested: 802.11G is now widely available and it's not possible to determine if more or less open APs are available. However, it is striking that on Channel 6, Spider has an average throughput that is 800% greater than the results reported by Cabernet (a throughput of 10.75 KBps [7]). 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

One of the primary challenges in designing multi-AP solutions for mobile Wi-Fi access is the overhead associated with `dhcp` and association. A way to minimize this overhead is to reduce the timeouts associated with `dhcp` and link layer retries. Fig. 12 and Fig. 13 show

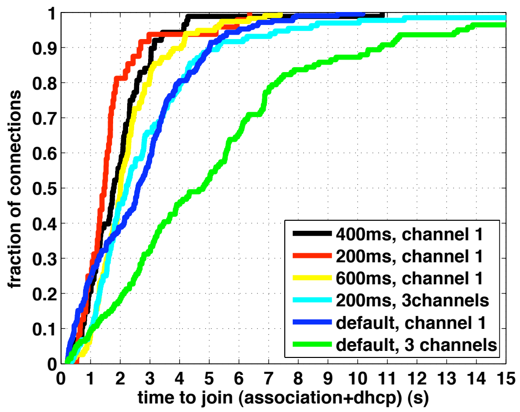

Figure 13: The rate of successful joins as a function of `dhcp` timeout. The cost of switching among channels overshadows the benefit of quickly establishing connections when timeouts are reduced.

the effect of reducing these timeouts on Spider. Fig. 12 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. 13 we find that the median time to a successful association improves—similar to the observation made in Cabernet [7]. However, the absolute median time to associate is 2–3 seconds, which increases by 2x when using multiple channels. This suggests the either a mobile multi-AP solution should stay on one channel or have a scheduling algorithm which

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

Figure 14: Average throughput and average connectivity seen when applying different static schedules for multi-channel configuration for Spider.

accounts for high association overheads.

4.6 Effect of Channel Schedule

To understand the effect of different channel schedules on throughput, connectivity, and association overhead, we present Fig. 15 and Table 14. The association experiments were performed on transit buses that are part of the DOME testbed while the throughput and connectivity experiments were performed in a car with the same hardware and software. We tested two schedules (1) equal schedule (200 ms each) on the three channels and (2) equal schedule (200 ms each) on two representative channels. Our experiments were performed in areas where there are sufficient APs on all the three channels. While more complicated dynamic schedules are possible, we argue that it is difficult, if not infeasible, to develop an efficient schedule that considers APs connections that are in data transfer and association phases simultaneously.

Fig. 15 shows that the single channel mode with reduced timeouts performs best in terms of association time–however, the reduced timeouts lead to a large number of `dhcp` failures, which is a huge deterrent. 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 14 presents throughput and connectivity results for the different schedules—throughput is maximized when Spider uses a single channel and connectivity is maximized when it uses a equal schedule on three channels.

4.7 Matching Usability Needs

A central question that has intrigued researchers is whether open Wi-Fi access can provide the connectivity needs of people when in transit. Unfortunately, we are not aware of the availability of any data on the actual connectivity requirements of mobile wireless users. To this end, we performed a study using data from a permanent Wi-Fi mesh we deployed in downtown Amherst. 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 a plausible baseline. Overall in

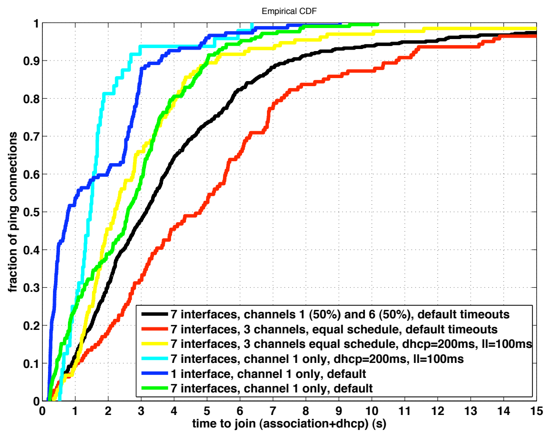

Figure 15: Delay in obtaining a dhcp lease and link layer association for different scheduling policies in Spider. The figure also considers reduced timeouts.

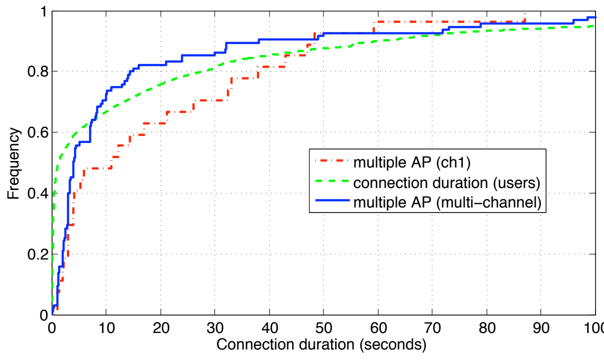

Figure 16: Comparison of connection lengths for wireless users and Spider.

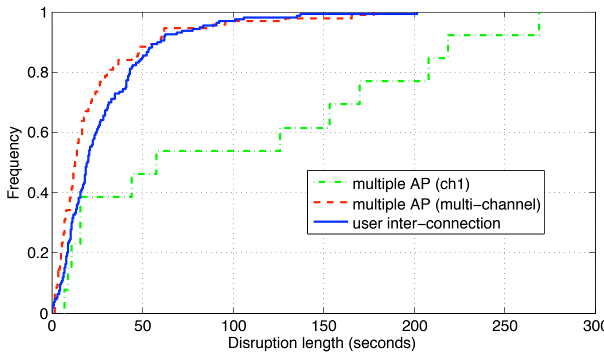

Figure 17: Comparison of disruption lengths for wireless users and Spider.

the data collected, there were 128,587 completed TCP connections and 13,645,161 packets were sent (1.7 GB). 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. 16 compares the TCP flow lengths seen from actual users using our mesh network and Spider in its multi-channel and single-channel modes. It is clear from the figure that Spider can support all the TCP connection flows that users require. Additionally, in Fig. 17 we compare the time between two connection for the mesh users and disruption time for Spider. When Spider uses multiple channels and multiple APs, it experiences disruptions comparable to what real users can sustain. While the single channel multiple AP sees large disruption, we believe that with a higher density of APs on one channel, this gap can be bridged.

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. Conducting this study forms part of our future work with Spider.

5. RELATED WORK

Spider builds on previous work on Wi-Fi access from mobile nodes, fast Wi-Fi and cellular hand-offs, and using multiple APs for throughput aggregation. Here, we compare and contrast Spider with the most relevant literature.

Wi-Fi Access from Moving Vehicles: Several challenging problems such as lossy wireless mediums [4, 7, 8, 12], tuning TCP performance for mobility [1], and AP selection [16] are well studied. The feasibility of using cached history to reduce association and dhcp overheads [5] has been demonstrated. Directional antennas have also been used to improve throughput [14]. However, this body of work concentrates on using a stock Wi-Fi model—association and data transfer with one AP at a time. However, as we have demonstrated, using a aggregation of APs can provide high aggregate throughput and better connectivity in a mobile environment.

Performance through diversity: Using technological and spatial diversity to improve Wi-Fi connectivity has also been studied in the past. This class of related work can be broadly classified by either infrastructure-end or client-end modifications. Infrastructure-end modifications includes coordination or selection amongst multiple open APs [3, 11–13, 13, 23]. In contrast to these approaches, Spider is a purely client-side solution that aims at improving a mobile user's performance in an *organic* Wi-Fi setting. *Client-side* diversity-based solutions rely on aggregating bandwidth across multiple APs [10, 17, 19]. However, these solution are tuned to work efficiently only in a *static*, stationary wireless environment.

An orthogonal approach to aggregating bandwidth in mobile nodes is using additional hardware—the assumption is that each client has more than one Wi-Fi card and data can be striped across concurrent connections to APs. PERM [22] is a multi-homed solution that aggregates throughput across multiple residential ISPs, profiles on-going connections, and assigns flows to interfaces to minimize delay. MAR [20] exploits heterogeneity of existing wide-area wireless networks by using a router architecture that aggregates independent cellular links into one fat reliable virtual data transfer pipe. Horde [18] is a middle-ware solution on mobile nodes that performs network striping over diverse cellular links tuned to application needs. Most of these data striping approaches can be built into Spider to enhance mobile user performance.

Soft hand-off and AP selection: Spider also builds on related work on fast cellular hand-offs and AP selection in mobile Wi-Fi networks. Fast hand-off is used to mitigate the adverse effects of disruptions in cellular networks [6]. While fast soft hand-off is plausible in a cellular network where the cell towers are under the control of one central authority, it is not feasible in Wi-Fi networks laid down by third-party users. The only practical soft hand-off solution using client side modifications is Spider that virtualizes the Wi-Fi card and maintains concurrent AP connections. Access point selection has also been an active area of research in Wi-Fi mobile networks. Several techniques including RSSI [9] and history-based techniques [15] have been proposed. However, multi-AP selection, solved by Spider is a harder problem (as we demonstrate in the Appendix) since it involves selection of a *set* of APs.

6. CONCLUSION

We presented the design, implementation, and evaluation of Spider, a system that can maintain concurrent connections to multiple WiFi access points in highly mobile scenarios. Spider concomitantly uses utility-based multi-AP selection, channel-based scheduling, and opportunistic scanning to maximize throughput and connectivity while mitigating the overheads of association and obtaining `dhcp` leases. Our evaluation of Spider on a vehicular testbed shows that it can provide manifold improvement in throughput and connectivity; allowing it to be a plausible supplement to cellular data services.

7. REFERENCES

[1] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *SIGCOMM '96: Conference proceedings on Applications, technologies, architectures, and protocols for computer communications*, 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. *Proc. ACM SIGCOMM*, pages 427–438, 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] P. Deshpande, A. Kashyap, C. Sung, and S. Das. Predictive Methods for Improved Vehicular WiFi Access. In *Proc. ACM Mobisys*, 2009.

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

[7] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: Vehicular Content Delivery Using WiFi. In *Proceedings of ACM MobiCom*, San Francisco, CA, September 2008.

[8] 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*, pages 125–138, October 2006.

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

[10] 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.

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

[12] 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.

[13] 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.

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

[15] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall. Improved access point selection. In *Proc. MobiSys*, pages 233–245, 2006.

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

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

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

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

[20] 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.

[21] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn. DOME: A Diverse Outdoor Mobile Testbed. In *Proc. ACM Intl. Workshop on Hot Topics of Planet-Scale Mobility Measurements (HotPlanet)*, June 2009.

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

[23] A. J. Viterbi, A. M. Viterbi, K. S. 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. Springer-Verlag, 1994.

APPENDIX

A. PROOF OF NP-HARDNESS

We assume that the mobile node spends T seconds on a road segment that has n open WiFi access points. We also assume that the driver has n virtual interfaces, thus, it is possible to establish concurrent connections to the n APs. Let S_i be the i'th subset of the power-set of n APs. We define a *value* V_i for each subset S_i. V_i is a function of the access points in the subset and quantifies connectivity or throughput. For example, if cumulative throughput is our desired metric, and the wireless bandwidth that S_i can provide is W_i then $V_i = T_i \times W_i$, where T_i is the time spent by a Spider node within the range of the APs in S_i. We also define a cost C_i associated with S_i. C_i is the sum of the time that Spider spends within range of the APs in S_i, the association time, and the switching overhead among channels and processing per channel queues. If D_i is this overhead, $C_i = T_i + \lceil T_i/T \rceil \times D_i$.

With these parameters as input, the goal of the multi-AP optimization problem is to select a set of subsets S_i, such that the sum of their values is maximized subject to the following constraints: (1) the total cost should not exceed the total time T and, (2) each T_i must be positive and less than T. Formally,

$$\max \sum_i T_i.W_i$$

such that

$$\sum_i (T_i + \lceil T_i/T \rceil .D_i) \leq T$$
$$\forall i, 0 \leq T_i \leq T$$

The above problem is equivalent to the 0-1 knapsack problem, which is known to be NP-hard.