Predictive Methods for Improved Vehicular WiFi Access

Pralhad Deshpande
Computer Science Department
Stony Brook University
Stony Brook, NY 11794, USA

Anand Kashyap[*]
Symantec Corporation
Mountain View, CA 94043
USA

Chul Sung, Samir R. Das
Computer Science Department
Stony Brook University
Stony Brook, NY 11794, USA

ABSTRACT

With the proliferation of WiFi technology, many WiFi networks are accessible from vehicles on the road making vehicular WiFi access realistic. However, several challenges exist: long latency to establish connection to a WiFi access point (AP), lossy link performance, and frequent disconnections due to mobility. We argue that people drive on familiar routes frequently, and thus the mobility and connectivity related information along their drives can be predicted with good accuracy using historical information – such as GPS tracks with timestamps, RF fingerprints, and link and network-layer addresses of visible APs. We exploit such information to develop new *handoff* and *data transfer* strategies. The handoff strategy reduces the connection establishment latency and also uses pre-scripted handoffs triggered by change in vehicle location. The data transfer strategy speeds up download performance by using prefetching on the APs yet to be encountered. Experimental performance evaluation reveals that the predictability of mobility and connectivity is high enough to be useful in such protocols. In our experiments with a vehicular client accessing road-side APs, the handoff strategy improves download performance by roughly a factor of 2 relative to the state-of-the-art. The data transfer strategy further improves this performance by another factor of 2.5.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless Communications—*Vehicular Communications*; C.4 [**Performance of Systems**]: Measurement techniques.

General Terms

Performance, Design, Measurement, Experimentation.

[*]Work done when the author was in Stony Brook University.

Keywords

Vehicular Internet Access, WiFi, Fast Handoff, Prefetching.

1. INTRODUCTION

The proliferation of IEEE 802.11 (WiFi) access points (AP) in homes and businesses in recent years has tremendously increased the coverage of wireless Internet in densely populated urban areas. Dense deployment of these APs now allow a computer inside a mobile vehicle to access the Internet. Researchers have started looking at the type of performance possible for such an access, and whether or not this could be adequate for certain applications. For example, in the CarTel project in MIT [15], researchers conducted a detailed evaluation of the performance and usability of these road-side open WiFi networks for vehicular Internet access. Their measurements demonstrate that at urban vehicular speeds, a regular mobile client can gain connectivity for several seconds (median of 13 seconds per AP), and large amount of data (median of 216 KB using TCP) can be transferred. This study used stock implementations of network protocols such as for 802.11 connection establishment and TCP. Various improvements are indeed possible using custom protocols as demonstrated by several related studies [19, 22, 11].

Motivated by the feasibility of WiFi connectivity at vehicular speeds, we are specifically interested in studying a content delivery network for moving vehicles in urban areas [19, 37]. Such a network can be used for downloading (e.g., songs, movies, podcasts, maps, traffic data, etc.) as well as for uploading (e.g., sensor data, etc.). The APs in such a system may consist of subscription based APs installed by a service provider, municipal WiFi APs, or APs with an architecture like FON [1] or WiFi.com [3] built on the concept of sharing WiFi connectivity. The FON or WiFi.com models are particularly relevant, as they rely on a community model where AP owners share their Internet connection with others in return of a similar service. These models can potentially enable deployment of custom protocols on the APs where APs can 'co-operate' among themselves and with the mobile client for better performance.

1.1 Challenges

Content delivery for vehicles using urban WiFi networks faces three challenges.

i. The *connection establishment latency* is ordinarily very high due to large probing delays for AP discovery (a typical client probes on all 11 channels before connecting), high loss rates (control packets during connection establishment may get lost and require retransmissions), and delay in acquiring IP address via DHCP. Several methods have been proposed to decrease this delay, such as using selective scanning, reducing timeout periods in case of losses, and using static IPs instead of DHCP [19, 15]. A recent work also proposes improving link layer performance using macrodiversity and opportunistic reception [11].

ii. A good *handoff strategy* is critical. Several APs may be visible at any location along the drive, and it is important to choose the best AP to connect to. Regular WiFi clients initiate handoff only when disconnected and choose the AP with the strongest signal strength. While this works well in home or office scenarios where clients are rarely mobile, this is not a good option for vehicular mobility. One can lose opportunity to connect to a strong AP (both in terms of signal strength and available backhaul bandwidth). A method of active scanning while connected has been proposed to address this issue [22].

iii. Finally, a suitable *data transfer strategy* is required, as vehicular WiFi access is characterized by frequent disconnections and lossy links. All these are known to degrade TCP performance, and impact applications requiring session maintenance (e.g., FTP, HTTP, etc.). UDP-like connectionless transport protocols have been proposed to alleviate this issue [19, 37].

1.2 Using Predictive Methods

We take a unique approach to address the above challenges by using a set of predictive methods that combines various forms of caching and estimation at link, network and application layers. Many studies have shown that people often drive on familiar routes, i.e., routes they have used before and the routes are highly predictable [30, 21]. In fact, our most frequent drives are only between a few locations (e.g., home, office, school, etc.). Further, it is expected that people have pretty consistent driving habits, e.g., use of lanes and speed. We show that these considerations can greatly improve the performance of vehicular WiFi access. The WiFi connectivity along a familiar route, and the mobility pattern of the vehicle can be predicted with a reasonably high accuracy using historical information. We develop techniques based on these predictions to readdress the three challenges above. Our specific contributions are as follows.

Faster connection establishment.

Driving on familiar routes provides the opportunity to learn and cache the relevant information about APs along a route, which can aid in quick connection establishment. During periods of inactivity, the client listens for beacons and records the channel, network name (ESSID), and MAC address of the APs it can successfully associate with. This information is tagged with GPS locations. In addition, cooperation with the APs and clients and/or use of autoconfiguration [23, 16] can eliminate the need for an IP address assignment via DHCP each time the client associates with an AP. All these speed up the connection establishment process significantly.

Scripted handoffs.

In addition to recording the connectivity parameters for an AP (as above), the client also builds a *radio frequency (RF) fingerprint* for the route when inactive, by recording the signal strength from beacons and tagging them with the GPS location of the car where the beacon is heard. This data, when collected over a period of time, provides a rich estimate of the RF level connectivity of various APs along the route. This connectivity estimate combined with an estimate of the vehicle's mobility are input to an algorithm which computes the locations where the client needs to handoff and to which AP. Thus, the handoffs are *scripted*. This computation is done offline and hence no bandwidth is wasted in scanning for better APs as in other online techniques. Also, the algorithm ensures that the client is always connected to the best estimated AP. This is unlike most stock implementations, where a connection is maintained until it breaks.

Prefetching at APs.

The scripted handoffs and mobility estimates as above predict the periods of connectivity to various APs in future. This can provide further performance gains in download applications by having such AP *prefetch* part of the content to be downloaded. Essentially, the APs now collectively form a distributed cache for the mobile client, and in cooperation with the mobile client prefetch pre-determined portions of the content. This helps mask the large Internet delay on the WAN side of the AP. Inaccurate estimation of mobility causes 'cache misses,' hurting download performance. On the other hand, duplicate prefetches (i.e., more than one AP prefetching the same bytes of the content) increases load on the WAN and the content server. These must be optimized carefully.

The rest of the paper is organized as follows. We start out by presenting the related works in Section 2. We discuss the predictability of mobility and connectivity in Section 3 and the handoff strategy in Section 4. Section 5 evaluates the handoff strategy. Section 6 discusses prefetching and Section 7 evaluates prefetching. We conclude in Section 8.

2. RELATED WORK

2.1 Vehicular WiFi Access

Several experimental studies have explored the potential of using intermittently available WiFi connectivity from moving vehicles for data transfers [36, 37, 15]. In the Drive-thru Internet project [36, 37] controlled experiments are done with a single car driving past a single access point to measure range and connectivity in an intermittent network. The key contribution in this work is a session protocol that provides persistent end-to-end communication even in the presence of intermittent connectivity. More recently, the CarTel project [15] has focused on upload performance while using APs in the wild. This paper observes an upload TCP bandwidth of 240 Kbps and median transfer size per contact of 216KB. In the ViFi project [11] the link layer performance is improved by exploiting macrodiversity (using multiple APs simultaneously), and opportunistic receptions by nearby APs. In their opportunistic reception scheme when an AP overhears a packet but not its acknowledgement, the AP probabilistically relays the packet to the intended next hop in order to minimize the wasted transmissions. By the nature of their design, both CarTel and ViFi focus on upload. Downloads and intermittent connectivity have been studied in a vehicular environment in a fleet of taxis in the Cabernet project [19] with an improved handoff scheme and a new transport protocol providing several times better throughput than stock implementations. Our goal is to improve handoff performance even further by using prior RF fingerprinting and scripted handoffs.

2.2 Fast WiFi Handoffs

Several techniques have been proposed to improve the handoff mechanism of a mobile client when switching associations in a wireless network. One of the earlier work in this context was done by Shin *et al.* [40]. They proposed a technique of using neighbor graphs in a WLAN to reduce the number of channels to probe, thus reducing the probe latency during association. In [32], Mhatre *et al.* propose a scheme for improving handoff decisions based on long-term and short-term trends in signal strengths observed from beacons from nearby APs. This work however is not in a vehicular environment. More recently, in the Cabernet project [19] an optimized handoff technique is developed for vehicular WiFi access. Here, a probing sequence is determined among channels to reduce the link association delay. They also propose techniques for reducing the DHCP delay.

On the standards track, a recent development is the IEEE 802.11r [4]. Its goal is to permit continuous connectivity to wireless devices in motion, with fast and secure handoffs between APs. This amendment is specifically targeted for vehicular environments where the mobile device moves from one AP to the other in a matter of seconds. 802.11r minimizes the number of transition messages to 4 by piggybacking the security and QoS related messages with the 802.11 authentication and reassociation messages, but does not target specifically to reduce scanning/probing delays. redOur work does not incur any scanning/probing delay and handoff decisions are precomputed.

The above techniques reduce the handoff delay, but they do not directly address the problem of when to perform a handoff. Giannoulis *et al.* [22] proposed a solution for vehicular clients, where they keep scanning for better APs, even while they are associated. Ramani *et al.* [39] have developed a system called SyncScan which reduces the cost of active scanning with short periods of passive scanning. The short listening periods are synchronized with regular periodic transmissions from each access point. However, this system has not been tested in vehicular environments. In [14], two wireless cards are used so that a client can associate with more than one AP at the same time thus eliminating most of the handoff delay. Again vehicular environment has not been studied. In contrast to these works, our scheme uses historical signal strenghts information to make handoff decisions.

Applications that require maintaining a session face problems in vehicular networks. Some papers address this issue by creating a transport layer protocol that maintains sessions transparently to changing IP addresses [19, 37]. The specific prefetching-based download application we develop does not need to maintain session at the transport layer. Sessions are maintained only at the application layer.

2.3 Mobility Predictions and Prefetching

Mobility prediction is well-studied in the domain of mobile phone networks. In [7, 8, 13, 31, 38, 42, 48], a central authority tracks the movement of devices to pre-provision network resources. In [29, 47], user mobility models are built from traces of several users, while Breadcrumbs [35], advocates that the mobile devices should retain their own mobility models for reasons of security and fine grained accuracy. In [43], authors compare different Markov based and compression based prediction models. Their study show that a second-order Markov model with fallback to a first-order model when the second-order model fails is the most accurate. Incidentally, Breadcrumbs [35] also implements a second-order Markov model for predicting mobility.

In addition to predicting mobility, it is also important for our system to be able to predict future network connections. War-driving databases could be used to predict networks to be encountered in the future. A system like Virgil [34], can be used to determine the quality of these connections in terms of available bandwidth, network connectivity parameters, availability of access to certain ports, etc. In [24] authors show that it is possible to predict wireless conditions of roadside APs and that this information can be used to improve vehicular network access.

Finally, prefetching has been used in widely different contexts in computer systems to speed-up downloads by hiding latency. As expected it also has been used heavily in mobile environments to prefetch data, but often in the context of prefetching related objects or content that are expected to be used in near future. See, for example, [17, 27, 28, 35]. In the context of vehicular networks as in our work, [46] has considered a combination of infostations (with high bandwidth links) and basestations (with low bandwidth links) and developed prefetching ideas based on the mobile client's location, direction and speed. In [12, 10] aggressive prefetching is used to make the results from web search queries available to mobile clients in buses using a combination of ad hoc and infrastructure networks. In [25] a technique has been described to use the cellular network to send data prefetch requests and the data is prefetched at the prefetch agents located at hot-spotted networks. To the best of our knowledge no work prior has considered prefetching parts of the same large object at different locations to improve download performance in a highly mobile environment.

3. PREDICTING MOBILITY AND WIFI CONNECTIVITY

In our context, predicting mobility essentially means estimating, at time $t = T$, the location of the vehicle at a future time instant $t = T + \Delta T$. Predicting connectivity means predicting all physical, link and network layer information necessary or useful for establishing connectivity to each AP visible at every location. The connectivity information includes (i) physical layer information, such as the received signal to noise ratio (SNR) for each visible APs, (ii) link layer information, such as the MAC address, identifying name of the wireless network (ESSID), channel, and wireless security information (WEP, WPA, etc.) for each AP, and (iii) network layer information, such as a usable IP address, default gateway, and DNS server information. All the above information are tagged with GPS location. The GPS-tagged SNR data is also referred to as the *RF fingerprint*.

The inherent assumption that guides the mobility prediction is that people's driving habits are highly predictable. Similarly, the assumption that guides the connectivity prediction is that the set of necessary or useful physical, link and network layer information for the APs at a location are typically stable over time. These assumptions can give rise to simple-to-use predictive models based on historical data that can improve handoff performance and make prefetching possible and worthwhile.

3.1 Data Collection

To exploit predictability, we propose an architecture where each car maintains an estimate of its mobility and connectivity as defined above. This requires collecting related historical data using a methodology we will outline here. This data is later used in estimations.

The data collection is done when the mobile client is 'idle', i.e., not communicating. The idle periods are chosen simply to avoid impacting actual data communication performance, as we use a single processor and single radio interface to manage all aspects of the system. Use of additional hardware can allow for concurrency, and then data collection can be done at all times, whenever the car is switched on.

The data related to mobility is simply a sequence of timestamped GPS locations collected periodically (every second). The data related to connectivity is collected using a 'sniffing' mechanism to log all beacons heard from any AP. The 802.11 beacon header contains the information about MAC address, ESSID, channel, security info, etc., while the Prism Monitoring Header[1] contains the SNR of the received packet. This information is tagged with the GPS location and timestamp where the beacon is heard. Note this data subsumes the mobility data wherever beacons are heard (default beacon frequency in most 802.11 APs is about 100ms). A careful reader will note that this data collection is somewhat similar to traditional *war-driving* [45]. GPS-coded SNR data helps build the RF fingerprint mentioned before. The client also records whether the client has appropriative credentials to connect to the AP and whether the AP provides backhaul connectivity. Depending on the usage scenario, this may involve a table lookup with a pre-existing table provided by a provider, or an actual association attempt with the AP and ensuring that the AP indeed provides backhaul connectivity. The client IP addresses are to be randomly generated following the guidelines in the IETF Zeroconf working group [23] and RFC 3927 [16]. More is detailed in Section 5.

The key aspect of our work is to develop protocols that exploit such historical data about mobility and connectivity to design predictive schemes to improve download performance. In the following subsections, we analyze the stability aspects of such collected data. The experimental data for this analysis was collected using two of our colleagues' cars. There are two data sets from the two cars – data set 1 is for a period of about 6 months and data set 2 is for a period of about 3 months. During this period, the cars were driven normally by their owners, carrying out their normal daily routines. Both cars carried the experimental platform described below.

[1]Prism Monitoring Header is added by certain 802.11 card drivers and contains information such as rate, SNR, noise, etc. for all received frames.

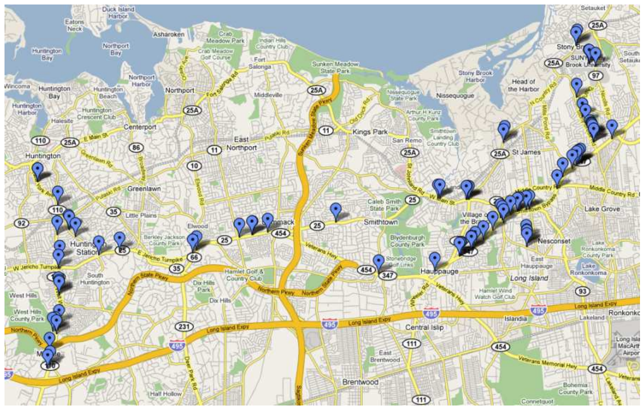


Figure 1: Map of the area driven showing the location of popular APs.

3.2 Experimental Platform

Our experimental platform is essentially a small form-factor embedded single-board computer (SBC) with a GPS unit and an 802.11 interface. The same platform is used in the protocol evaluations in the following section. Specifically, we use a Soekris 4801 [41] for the embedded SBC, with an 18dBm 802.11b miniPCI card with Atheros chipset with an external 5 dBi rubber duck antenna, and a Garmin USB GPS receiver. A 4GB flash drive is used to collect the logged data. The data is periodically manually uploaded on a server for sanity check and later analysis. The computer runs Linux kernel 2.6.19 and the latest `madwifi` driver [2] for 802.11. The computer draws power from the car battery from the regular 12V automobile socket and remains switched on whenever the car is running.

On startup, the 802.11 card is initialized in monitor mode, the computer system clock is synchronized with the GPS receiver and the computer starts running `tcpdump` to log all the packets received by the 802.11 card. The first 200 bytes of every packet are logged which contain the Prism Monitoring header and the 802.11 header. Beacons are usually small packets, and are captured in their entirety. The computer time at which the packet was received is also recorded for every packet. The card switches between the non-overlapping channels 1,6, and 11 at intervals of 0.5 seconds each. Initial tests showed that a very large fraction of APs are on these three channels [6]. Moreover, scanning just these three channels is sufficient to get a large number of packets in other overlapping channels too. Also on startup, the GPS receiver is initialized and the GPS location and timestamps are logged every second. The packet timestamps in the `tcpdump` trace are later correlated with the GPS timestamps to determine the location at which each packet was received.

3.3 Analysis

As mentioned before, data set 1 has been collected over a period of 6 months while data set 2 has been collected over a period of 3 months. A significant portion of these drives is between home and school. In data set 1, this distance is about 25 miles and consists of a combination of freeways and local roads. In data set 2, on the other hand, it is about 5 miles. It consists entirely of local roads. A few other drives do exist in the data set that are repeated several times. They are usually short and use local roads predominantly. The

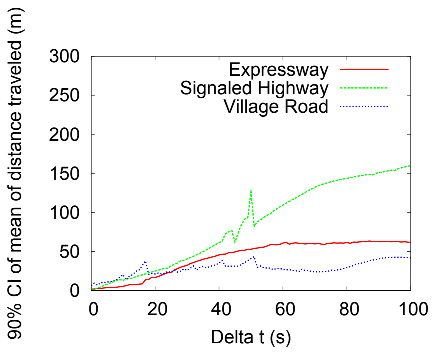
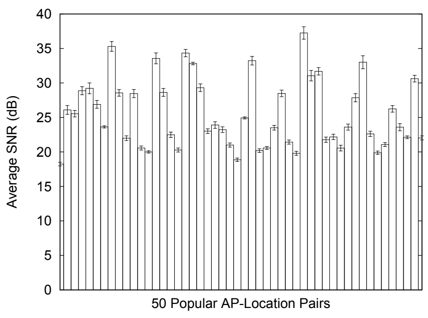
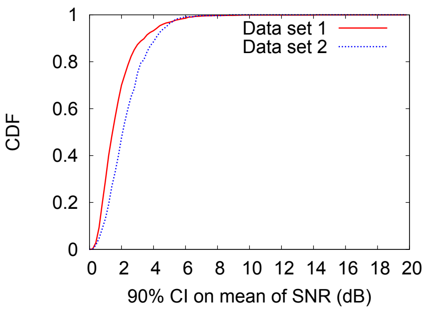


Figure 2: Mobility estimator for the data set.



(a) Average SNR for the 50 most popular AP-location pairs over the two data sets showing 90% confidence intervals.



(b) CDF on the width the 90% confidence intervals for the SNR of all APs for all locations.

Figure 3: Statistics showing the stability of RF fingerprint data.

dataset has 3618 distinct APs. See Figure 1 for a map of locations of APs which are seen with a high frequency.

Figure 2 shows the quality of mobility prediction. We determine the distances traveled from a given start point for different time intervals ΔT. The average of all these distances across all drives on the same route available in the log is used as the predictor for mobility. The quality of the estimation is modeled by the width of the 90% confidence interval. We predict mobility on three different kind of roads found in our data set – (i) an expressway, where the car typically travels at a sustained speed of about 100 kmph, (ii) a signaled highway, where there are large variations in speed due to several signals and also due to traffic variations, the maximum speed being about 90 kmph, and (iii) a village road, where the car travels at a fairly constant speed of about 50 kmph. As shown in Figure 2, the accuracy of the mobility estimates is very high for small values of ΔT, while expectedly, the accuracy decreases for large values. It is interesting to see that the accuracy is low initially for the expressway than for other types of roads. This is due to the high speed on expressways, and any variation is magnified in terms of distance traveled. Also, as expected the mobility estimates become less accurate with time on a signaled highway, but remain excellent for the expressway and the village road.

From our experience, (also in the CarTel data set [15]), the median connectivity duration with an AP is about 13 seconds at regular driving speeds in an urban area. Taking this number as a guide, observe that the uncertainty in distance is very small for this period – about 20 m, much less than a typical range of an AP (about 100 m). Also, to accumulate an uncertainty roughly equivalent to the typical range of an AP, ΔT needs to be very large – about a couple of minutes – almost an order of magnitude more than the typical visibility period of an AP. Thus, one can expect decent prefetch performance with a *prefetch lookahead* of a few APs. By 'prefetch lookahead' we mean the number of APs ahead of the current AP that are asked to prefetch data by the current AP. More will be discussed about this concept in Section 6.2.

We now evaluate the accuracy in estimating connectivity by showing that RF fingerprints remain moderately stable over time. For a particular AP, and a particular GPS location, the average SNR of beacons received over several drives from that AP in that location is used as a predictor. 90% confidence interval is again used to model the quality of prediction.

We must first discuss the practicality aspects of this evaluation before presenting results. This discussion also lays out how RF fingerprints are used in practice for use in the protocols discussed later. It may not be possible to obtain RF samples at exactly the same locations from different drives. The reason for this is that (i) the car positions could be slightly different on different drives even when driving on the same lane; (ii) the RF samples could be obtained at slightly different points; (iii) GPS is not perfect – typical GPS error is a few meters, as prior measurements with the same GPS unit have shown [44]. To address these issues, we discretize the location data in the RF fingerprint. To do this, we overlay a 10m×10m grid[2] on the geographic area spanned by the data set. All RF samples corresponding to a grid square are mapped to the center of that square. Since we are interested in variations across drives, we assume that for each drive, there is only one RF sample per AP per location (center of a grid square). If there are more than one such sample on the same drive, the SNR values are averaged to produce just one sample. Now, for each location that actually has samples from sufficient number of drives (more than 5), the *average SNR over these multiple drives* is computed.We also compute the 90% confidence interval for this statistic.

The statistics (average and the confidence intervals) of the 50 most popular AP-location pairs and CDF of confidence

[2]We also tried 20m×20m grid with no significant variation of results.

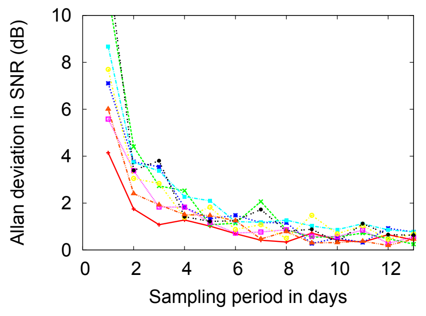


Figure 4: Allan deviation in different time scales for 8 selected AP-location pairs for which we have the most number of samples.

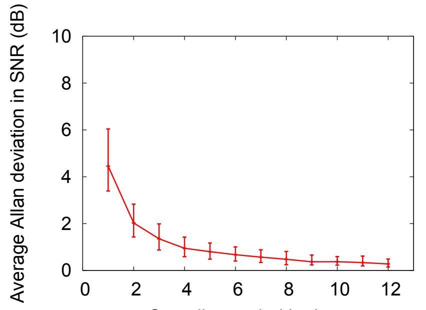


Figure 5: Summary statistics of Allan deviation on different time scales (median, top and bottom quartiles).

interval widths for all APs for all locations are shown in Figure 3. Note excellent stability of the SNR statistic in Figure 3(a). Also, note that the median width of the confidence interval is around 2 dB (in Figure 3(b)). This is small when compared with the median overall SNR values logged, which is about 22 dB. In prior work for vehicular WiFi access, 2 dB has been considered as a margin of error in estimating the mean signal strength [22]. In that work, a client scans for better quality APs even when connected, and a handoff is triggered, if an AP is discovered with an average signal strength 2 dB greater than the current AP.

It is indeed true that about 10-15% of the data shows 4 dB or larger confidence intervals that may not be acceptable to make good handoff decisions. However, data stability aspects can be taken into account in making handoff decisions. Thus, it is not a critical issue.

The above analysis uses the complete data set as actually collected. While this presents the summary statistics, it does not directly analyze concerns about temporal nature of the data set, for example, whether the SNR is relatively independent in the sequence of drives, or whether there are short term temporal dependencies. In other words, is it possible that it is best to estimate SNR using relatively fewer, but more recent samples, as opposed to long term measurements using the entire available data set? To study this, we use Allan deviation [9] as a metric. It is similar in spirit as standard deviation, but uses difference between subsequent samples rather than differences from the mean. Allan deviation is typically used to quantify the burstiness in the variation of any quantity and has been previously used to study temporal variations in link quality in 802.11 networks [5]. Allan deviation is computed at various time scales from 1 to 12 days for all APs for all locations as above. In Figure 4 we plot the Allan deviation in SNR for 8 ⟨AP, location⟩ combinations for which we had the most number of samples. Note that while there are fluctuations at different time scales, there is indeed a downward trend signifying lack of strong temporal correlation. Summary statistics are plotted in Figure 5, where we plot median, top and bottom quartiles for the Allan deviation across the data sets for different time scales. Again note the strong downward trend and stabilization beyond just 1 week.

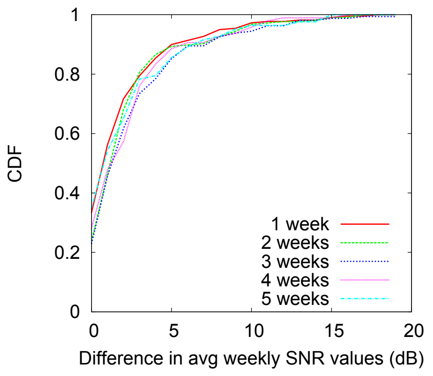


Figure 6: CDF of difference in average weekly SNR values for ⟨AP,location⟩ pairs as the separation between the weeks increases.

The above analysis confirms that average SNR over multiple drives is a good measure for estimating SNR as it shows little variation when averaged over for a week or more. Another concern could be about the staleness of the historical information collected. For example, the average of SNR values for an ⟨AP, location⟩ combination computed over a week may change when measured again after a few weeks. We evaluate the extent of staleness by comparing the difference between weekly SNR averages of various ⟨AP, location⟩ combinations, as the distance between the weeks considered increases. The CDF of various such samples is shown in Figure 6. The graph shows that the median difference between weekly SNR averages remains within 1 dB even if we consider weeks as far as 5 weeks apart. This demonstrates that the SNR data remains stable over several weeks, and data collected even 5 weeks ago should be usable for prediction.

4. CONTROLLING HANDOFFS

As mentioned before, stock 802.11 clients typically maintain association with the same AP until the connection breaks, which is determined by the absence of beacons for a certain fixed period of time. Then it searches for another AP to associate with via either active or passive scanning techniques. In active scanning, the client switches between all channels, and for each channel, it broadcasts a probe request message and waits for a finite and configurable timeout period for APs on that channel to reply with a probe response message. This takes about 250ms for 802.11b. In passive scanning, the client just switches between channels, and passively listens

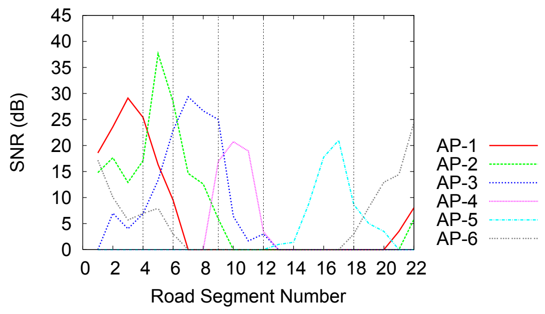


Figure 7: Expected variation of SNR (smoothed) along the drive based on historial RF fingerprint data. Dotted lines show the precomputed locations of handoffs.

for the periodic beacons that are broadcast by APs in each channel.

In our knowledge, the state-of-the-art in determining when to handoff in a mobile scenario is the mechanism proposed in [22]. There the authors propose a method of active scanning even while connected, to maintain a quality score for each AP, and switch association as soon as a better AP is discovered. To counter short-term fluctuations in signal strengths, the authors propose using an exponential moving average of the signal strength. They also propose using hysteresis to prevent handoffs from occuring very frequently. This method is similar to our strategy to connect to the strongest AP always. However, it suffers from the problem that the process of active scanning wastes significant bandwidth for the client when it is connected. This is certainly detrimental to performance when the client is actively communicating. Also, scanning needs to be done very frequently to make an optimum choice in a mobile scenario.

In the technique we propose, historical RF fingerprint data is used to estimate how signal strength is expected to vary along the drive.[3] We already established in the previous section that the data is stable enough that it is feasible to do so. See Figure 7 for an example, using our own experimental data set from the evaluations that we will describe momentarily. The road segments are simply the intersections of the driving path with the grid squares discussed in the previous section. Note that the rises and falls are expected as the car comes near to the AP and then goes away. The brief dips are often due to turns or due to structures causing radio shadows.

A straightforward technique could be as follows. When disconnected, the client establishes connection with the first AP that becomes visible. However, when the client is already connected, the handoff is performed at the intersections of the 'smoothed' SNR vs. distance curves to that AP that will provide the strongest signal for the next segment. See Figure 7. The locations at which each handoff is triggered is computed offline. This could be done sufficiently in advance to the actual handoff. This is the reason behind the name – *scripted handoff*. We present experimental results for this technique in the next section.

Depending on the handoff latency in both the link and network layers, the above simple technique may lead to frequent handoffs that could impact throughput adversely. Stock im-

[3]The route can be easily estimated using historical data again (see, e.g., [21]) is not discussed here.

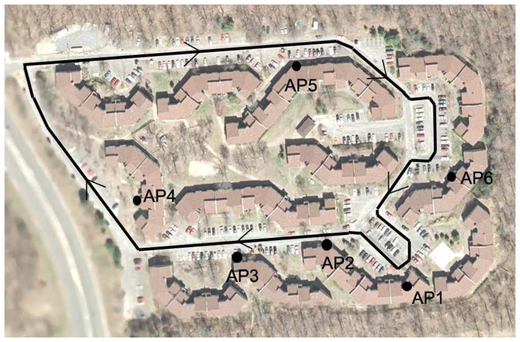


Figure 8: Map of the drive. The AP locations are noted.

plementation of connection establishment process between a client and an AP requires several message exchanges for the link layer association followed by IP address allocation via DHCP. The vehicular environment is often characterized by high loss rates as shown in [19], which leads to significantly high connection establishment time. The mean connection establishment time has been reported as 12.9 seconds in [15].

For large handoff latencies, an optimal handoff strategy thus needs to factor in the handoff latency. This can be done via *scripted handoff* as well. Here, the predicted route and speed can be used to estimate the visibility periods of individual APs along with their signal strengths. This information then can be used to drive an algorithm that precomputes the optimal handoff points and APs (i.e., <AP, location> tuples) to maximize throughputs by factoring in the handoff latency. The mobile client hands off to the precomputed AP when the car reaches the specific location.

The scripted handoff is reminiscent of location-based beam steering and AP selection method in our prior work in MobiSteer [33]. However, the focus in MobiSteer was primarily on beam steering and the handoff performance vis-a-vis other fast handoff techniques was not evaluated. In the current work the focus is primarily on AP selection (handoff). In the next section we will evaluate the handoff performance with respect a state-of-the-art method.

5. EVALUATING SCRIPTED HANDOFF

We evaluate the simple scripted handoff technique (called SH for easy reference) by comparing them with two existing methods: (i) the naive method used by stock WiFi clients (maintain until broken followed by active scanning or MUB for easy reference), and (ii) the state-of-the-art method (active scanning while connected or ASC for easy reference).

While the scripted handoff (SH) technique relies on historical information, and is thus an offline technique, the other methods, MUB and ASC, are online methods. As an offline method, the SH technique has several advantages over the other online methods. The need for scanning is eliminated as the client already contains historical information about the AP's MAC address and channel, and can thus send an association request message even without scanning.

The need for having to do DHCP for IP address assignments can be eliminated by using a randomly generated link-local IP address in the client per RFC 3927 [16] and then using a NAT at the AP. This approach has been promoted by the IETF Zeroconf working group [23]. The idea is to generate a random IP address from a pre-specified address pool (with 169.254/16 prefix per the RFC [16]). The RFC proposes to detect conflicts by a broadcast ARP probe sent prior to real communication. In our application, the AP can help detect and resolve conflict via a control message and the ARP probe is not needed. Given the large pool of possible addresses, we anticipate that conflicts will be rare. If IPv6 is used, such conflicts can be eliminated as the MAC address can be used to form the host part of the IP address. In our evaluations, however, we have used static IP addresses by means of sufficiently long DHCP leases obtained prior to measurements and then recording the IP addresses as part of the RF fingerprint.

As we want to evaluate the performance of the handoff strategy, it would be unfair to compare the online techniques with their stock implementations, as they would incur a significant handoff latency impacting their performance. We thus optimize the connection establishment process for online techniques as well. The scanning is performed only in three channels – 1, 6 and 11, as all our APs are in one of these three channels. We also optimize the DHCP client as in [19] by reducing the timeout periods for the DHCP DISCOVER and DHCP REQUEST messages. We further describe some implementation details below. We use the same hardware setup as described earlier in Section 3.

Maintain until broken (MUB) – This is the stock implementation used by most clients. Once connected, the client maintains the connection until it stops receiving beacons for 5 seconds, after which it disconnects. When disconnected, the client starts the connection establishment phase again.

Active scanning while connected (ASC) – This is based on the method described in [22]. We have described this method briefly in Section 4. We use the best-performing parameters for the scan period, hysteresis threshold, and the weight for exponential averaging from the paper. The scan, performed once every second, is done on the three channels – 1, 6 or 11 by sending probe messages and waiting for replies. The signal strength of received probe responses is recorded. For each AP, the value of its signal strength is maintained by taking a weighted average of the current SNR (with a weight of 0.35), and the previous estimate of SNR (weight of 0.65). Whenever an AP is found with the estimate of signal strength greater than a certain threshold (2dB in our case), a handoff is triggered.

5.1 Experimental Setup

Scripted handoff is most useful when there is a dense AP deployment with overlapping coverage and no coverage holes in the driving path. We were unable to find such a region with open access APs in our neighborhoods. This is because a significant fraction of APs that we analyzed in Section 3 use some form of security. We perform the experiments in a residential area with several APs that we deployed ourselves inside homes to carry out the experiments. See Figure 8. The APs are all connected to a backhaul providing Internet connectivity. We choose a circular drive of about 650 meters. The driving speed is slow – between 16 Km/h and 24 Km/h. Thus, each round of the 650 meter drive is completed in roughly 2 minutes. At least one AP can be heard from almost all locations along the drive. 10 of these rounds are used to create the RF fingerprint of the area. The RF fingerprint representation according to drive segments is shown in Figure 7. The drive is divided into 22 segments of approximately 30 meters each, and the best AP to connect to in each segment is computed in advance. The five handoffs that need to be performed during this drive are also shown in the figure.

We perform 10 rounds of drive each for the 4 handoff strategies. In each drive, we connect to the APs using the specified handoff strategy. Whenever connected, we send a stream of CBR traffic at 560 Kbps consisting of 1400 byte UDP packets (thus, 50 packets per second) to a server on the Internet, which records the timestamp at which each packet is received. The MAC address of the associated AP is included the packet; so the server can detect change in association. The server calculates the "throughput" in Kbps from the mobile client to the server. The server also computes the "duration of connectivity per AP" by subtracting the timestamps of the last and the first packet received via an AP, and "outage period", by subtracting the timestamp of the first packet received via a new AP and the timestamp of the last packet received via a previous AP. The mobile client records the latencies for the three handoff steps – "scanning latency", "link association latency", and "DHCP latency" for each successful connection establishment. It also keeps a record of the "number of attempted connection establishments", and "number of successful connection establishments."

5.2 Results

The results for the three mechanisms are summarized in Table 1. The numbers in the brackets in the table are 90% confidence intervals of the mean value presented. The results clearly show that our handoff strategy based on the offline, scripted handoff method outperforms the other "online" methods by a factor of more than 2. The SH protocol provides an average throughput of 232.8 Kbps which is more than twice the average throughput of other protocols. The average outage is less than half of the other methods. The average duration of connectivity per AP is about 20% higher for SH than other methods as the connection establishment latency is very low. The method of active scanning (AS) performs only slightly better than the maintain-until-broken method (MUB). This is because significant bandwidth is wasted in scanning once every second. SH has no scanning or DHCP latencies, that present high overheads for the other two methods.[4]

We do not show the link association latency because it is very low, and similar for all the three techniques. Due to the high connection establishment latencies for MUB and AS techniques, they have a poor percentage of successful connection establishment.

6. PREFETCHING

So far, we have used predictions to improve handoff (fast handoff and handoff to an 'appropriate' AP). The techniques have been thus limited to the link and network layers. Now we turn our attention to the application layer and show how prefetching at the APs can improve download performance. Since a vehicle's mobility and connectivity can be predicted quite well on familiar routes (Section 3), it is possible to create a protocol for prefetching portions of a large object to be downloaded on the APs on the route of a vehicle.

[4]We did not separately quantify how much improvements are obtained individually due to the absence of scanning or DHCP latencies. However, lab experiments show saturated UDP throughput drops by 37% when scanning every 2 seconds using the ASC implementation.

Metric	MUB	ASC	SH
Average throughput (Kbps)	101.6	113.52	232.8
Avg. duration of connectivity per AP (s)	11.04 (2.5)	9.28 (3.1)	12.89 (1.4)
Avg. outage period per AP (s)	36.82 (9.1)	31.2 (9.8)	13.45 (3.9)
Avg. # successful connection establishments per drive	2.5	2.7	4.5
Avg. # attempted connection establishments per drive	10.3	4.9	5
Percentage of successful connection establishments	24 %	53 %	90 %
Avg. scanning latency (s)	0.42 (0.0)	0.32 (0.0)	0 (0.0)
Avg. DHCP latency (s)	3.76 (0.7)	3.88 (0.6)	0 (0)

Table 1: Summary of results for handoff control experiments.

When the vehicle approaches and connects with an AP, the prefetched data can be directly downloaded from the AP instead of connecting to a server on the Internet.

Such prefetching has several advantages. First, the multi-hop Internet path is replaced by a one hop path when a client downloads prefetched data, and thus a higher throughput can be obtained. Recent measurement studies in [18] found that median bandwidth of Internet paths leading to residential broadband hosts is about 5 Mbps. But this paper noted that the downstream bandwidth allocated by DSL and cable modem providers varied widely. Median numbers for different providers studied varied roughly between 1-3 Mbps for DSL and 2-6 Mbps for cable modem. Contrast these numbers with a recent vehicular networking study in the Cabernet project [19] that promotes the use of 11 Mbps data rate for the 802.11 AP-to-client link. This is likely to improve with the use of 802.11g and/or beam steering technologies such as MobiSteer [33]. Also, round-trip times for multihop Internet paths are likely to be much larger than the one hop wireless link, affecting TCP throughput significantly than just what the bandwidth numbers show.

Second, the client can use a specialized transport protocol for downloading data from the AP which is not as sensitive to packet losses as TCP. This is not possible if the downloading client is only likely to run legacy protocols.

An important challenge in designing a prefetching protocol is that inaccurate predictions of mobility and/or connectivity can make make impacts in two different ways. When the estimation is 'noisy' (i.e., estimation error is relatively high), prefetching could be done 'conservatively.' This means that the ranges of data to be prefetched in subsequent APs could be allowed to overlap, likely to a significant extent. The extent of overlap depends on estimation errors. (An extreme view of this would be to prefetch all data on all APs, when no estimation is available). This, however, wastes backhaul bandwidth due to redundant prefetching. It wastes cache storage in APs as well. However, we do not anticipate storage to be a significant issue with the current advances in flash memory technology.

While lesser or zero overlaps save on backhaul bandwidth, they increase the possibility of 'cache misses,' slowing download performance. A thorough evaluation of estimation and prefetching strategies would be interesting directions of research. But, our focus in this paper is more towards systems rather than performance evaluation. We propose the protocol architecture, build a proof-of-concept system, and experimentally demonstrate the performance potential of the prefetching protocol. We start with the protocol description in the following.

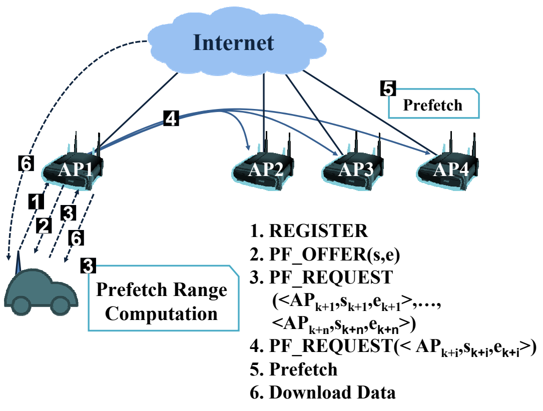

Internet
5
Prefetch
4
6
AP1
AP2
AP3
AP4
1
2
3
6
3
Prefetch Range Computation
1. REGISTER
2. PF_OFFER(s,e)
3. PF_REQUEST (<AP_{k+1},s_{k+1},e_{k+1}>,…, <AP_{k+n},s_{k+n},e_{k+n}>)
4. PF_REQUEST(< AP_{k+i},s_{k+i},e_{k+i}>)
5. Prefetch
6. Download Data

Figure 9: Architecture of prefetching protocol.

6.1 Protocol

We assume that APs are cooperative and can communicate with one another directly on the WAN side. This is possible if APs have a public IP address on the Internet, and they are part of the same peer or community group, or are managed by a service provider. These APs could also be nodes in a mesh network.

In our design we have used *HTTP range requests* (partial GETs [20]) for downloads (from client or AP to the server on the Internet). This makes our evaluation reflective of a standards-based protocol that is widely supported on HTTP servers. Similar HTTP range requests are also used for downloading prefetched content from the APs to the client as well. To facilitate this, the APs run a custom HTTP server to serve the prefetched data. The client either makes a HTTP range request to the AP, or to the original server on the Internet (or, to both sequentially). The byte ranges to be used in such requests depend on the byte ranges the AP might have prefetched and client might have downloaded previously.

We now describe the step-by-step operation of the prefetching protocol. The steps are also shown in the Figure 9. To simplify the description, the URL of the object to download is not explicitly mentioned in the below description.

1. When a client connects with an AP (say AP_k), it sends a *REGISTER* message to the AP indicating the URL of the file it wishes to download, and requesting the range of bytes of the file that has already been prefetched at the AP (if at all).

2. The AP replies with the prefetched byte range to the client using the message *PF_OFFER*(s, e), where s and e denote the start and end offset of the prefetched byte range respectively. If the AP does not have any prefetched data, it responds with an empty range, i.e., $(s, e = 0)$.

3. The client then computes the byte range that has to be prefetched at the next n APs (n is the lookahead we discussed in Section 3.3) and constructs a message of the format,

$$\textit{PF\_REQUEST} \quad (\langle AP_{k+1}, s_{k+1}, e_{k+1}\rangle, \cdots, \\ \langle AP_{k+n}, s_{k+n}, e_{k+n}\rangle),$$

   where each tuple indicates the byte range a particular AP needs to prefetch. We will explain this computation momentarily in Section 6.2. The client then sends the above constructed message to the AP it is connected to.

4. The AP on receiving the above message splits it in n *PF_REQUEST*$(\langle AP_{k+i}, s_{k+i}, e_{k+i}\rangle)$ messages, one for every tuple, and sends them to the corresponding APs. Note our assumption that the APs can communicate directly to each other (over the Internet, for example).

5. Upon receiving a *PF_REQUEST* $(\langle AP_{k+i}, s_{k+i}, e_{k+i}\rangle)$ message, the AP AP_{k+i} prefetches the bytes ranging (s_{k+i}, e_{k+i}) from the server. If part or all of this range has already been prefetched, the AP prefetches only the difference.

6. Right after step 3, the client starts the download process. By observing the range (s, e) in the *PF_OFFER* message and the byte ranges the client has already downloaded before, the client determines the byte range of the download request for this AP, which could be a subset of the range (s, e). After the download from the AP is complete (or, if the client already had downloaded the entire byte range (s, e) prior to entering the current AP), the client makes a download request to directly to the server.

The client can download broadly in two ways:

(i) *Sequential download* – the client only fetches bytes sequentially. If the next sequential bytes have not been prefetched in the current AP, the client downloads directly from the server. This technique is appropriate for downloading media that could start playing even before the entire download is complete.

(ii) *Non-sequential download* – the client downloads opportunistically. It first downloads the bytes that have been prefetched in the current AP (excluding duplicates) even if such bytes are not sequential. Then the client downloads from the server byte ranges for which (a) no prefetch instruction has been given yet, and/or (b) prefetched byte ranges in prior APs that could not be downloaded because of insufficient contact time with the APs. The latter can happen when an AP prefetches successfully, but the mobile client has insufficient contact with the AP because of fast movement or poor link quality, for example.

Evidently, non-sequential download can provide better throughput.

6.2 Prefetch Range Computation

The scripted handoff technique developed earlier already computes the handoff locations a priori. The analysis in Section 3.3 shows that given a time interval ΔT, the location of the car can be predicted quite well for small ΔT (few minutes). Thus, it is possible to estimate the time instances in the future when the client will connect with the APs ahead in the route. These estimates, together with the knowledge of the current location, the byte ranges already downloaded, and an estimate of AP-to-client bandwidth,[5] are used to compute the byte ranges of the file that should be prefetched on the APs the client is going to connect with in future. This constitutes the prefetch range computation.

As shown in Section 3.3, such estimates could be quite accurate for the near future, but losing accuracy further in the future. Thus, the challenge is determining the right 'lookahead' (n) value to use. This requires analysis with significant amount of urban traffic data. Lookahead will also depend on backhaul bandwidth as it influences the time needed to perform the prefetching. Data for backhaul bandwidth can be collected at the APs and be used for lookahead computations. Such analysis is beyond the scope of the current paper. We do note that our protocol architecture is flexible, and can accommodate a wide range of estimation methods based on available information. The protocol allows for the *PF_REQUEST* instruction be received by the same AP multiple times, possibly with more refined or corrected estimates. However, this might incur some unnecessary or redundant downloads. Also, the estimates themselves can always be made conservative. For example, APs could be instructed to prefetch overlapping byte ranges. This simply shifts the burden to the backhaul to favor better pre-fetch performance.

Finally, note that in our design the onus is on the client to do such computation. The key reason for this design choice is that data used for estimation reveals driving habits and this can stay private with client-side computation. The downside of this design choice is that the client can potentially ask for excessively redundant prefetches overwhelming the AP backhauls. It is possible to address this issue using some sort of pricing mechanism, or limiting the total amount a client can ask to prefetch etc.

7. EVALUATION OF PREFETCHING

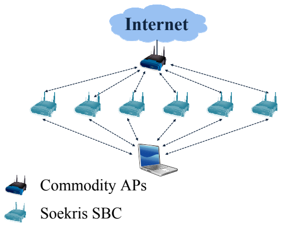
We use the metric *throughput gain* to measure the benefit of prefetching. This metric is the ratio of the throughputs with and without prefetching in an otherwise identical experimental condition.

The effectiveness of prefetching depends on several factors:

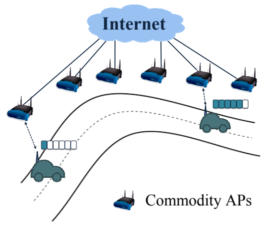
- Difference between backhaul bandwidth and client-to-AP link bandwidth: If the AP-to-client hop is a bottleneck, prefetching may not provide any tangible benefit. On the other hand, prefetch will be significantly beneficial for poor backhaul bandwidths.

- Use of overlapped ranges: As discussed before, the prefetch instructions could be conservative when estimation is poor. This allows for byte ranges prefetched on subsequent APs to overlap to different extents. In

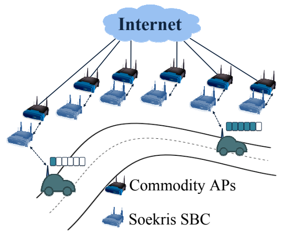
[5]This is the measured average bandwidth across various association sessions. More interesting, or online methods to estimate this are not evaluated in this paper.



(a) Scenario A (Lab).



(b) Scenario B (Drive with Uncontrolled APs).



(c) Scenario C (Drive with Controlled APs).

Figure 10: Three experimental scenarios are shown. The commodity APs are already deployed in the residences. The Soekris SBC has two cards, one connecting the client and the other connecting to the commodity APs on two orthogonal channels.

experiments we use *padding* to evaluate its impact. We assume that the byte range computation is non-overlapping, but then $x\%$ extra bytes ($x\%$ padding) are prefetched on both sides of the byte range.

- Duration of connectivity: The throughput gain when prefetching also depends on the duration for which clients associate with the APs. Higher durations of connectivity should aid prefetching as it allows the client's auto-rate control to stabilize. Also, small scale variations of the vehicle speed have now less impact on prefetching performance.
- Download method: As discussed in the previous section, non-sequential download may improve throughput by allowing for opportunistic downloads.

We use three different scenarios to evaluate the impact of the above factors on the throughput gains achieved by prefetching. We first perform a comprehensive set of evaluation experiments in a controlled lab environment. Then, we supplement this evaluation with real driving experiments.

7.1 Controlled Lab Experiments

These experiments are in a controlled lab setting to quantify the benefits of the various factors discussed above. We set up six APs in an indoor lab environment in close vicinity, and have a client which simulates a drive through these APs by switching associations between them via a script. The client is in close vicinity with all APs and thus has a good link quality. The APs connect wirelessly to a wireless gateway, which in turn connects to the Internet. We call this as Scenario A as shown in Figure 10(a).

The APs used in this experiment are Soekris embedded SBCs as used in Section 3.2, with the difference that they consist of two 802.11 cards, one to provide access to the client, and the other to connect with the gateway. The interface which runs as an AP also runs a DHCP server. We use NAT (Network Address Translation) to switch packets between the two interfaces. The prefetching protocol described in the previous section is implemented on this AP. We use these customized APs later in Scenario C as well, to be described momentarily. These APs provide the advantage that they can act as a bridge between the mobile client and an actual AP deployed by a provider where we cannot install any software.

As an indoor lab setup, this scenario differs from a real driving experiment in two aspects. First, the fluctuations in link qualities experienced while driving are not experienced here. Second, as association changes are driven via a script, the mobility estimate is almost perfect. Of course, noise in these parameters can be simulated, but we do not do it here, as we will also describe actual driving experiments using the same hardware in Scenarios B and C.

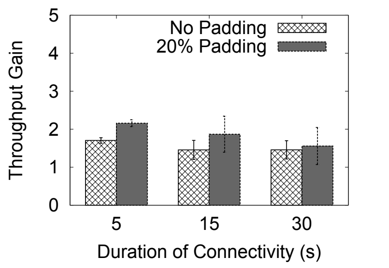
For the above described scenario, we perform 'simulated' driving experiments, where the client associates sequentially with the six APs, and downloads a very large file from an Internet server 10 hops away. To evaluate the various factors which impact prefetching, we make the following choices. The APs prefetch the estimated byte range with 0% or 20% padding. The duration of connectivity is chosen as 5, 15, or 30 seconds (same for all APs), which are within the typical range found in our driving measurements. We vary the backhaul link badwidth between 2 Mbps and 11 Mbps, while the client connects with the AP using auto-rate algorithm, which generally uses 11 Mbps due to the excellent link qualities. We also experiment with sequential and non-sequential downloads. We perform 15 runs for each experiment. The results for the average throughput gain along with the 90% confidence interval are shown in Figure 11.

Though the use of padding appears to improve throughput for sequential download, the improvement is actually in the noise. Similarly, the difference in performance due to different connectivity duration also appears to be in the noise. However, roughly 50% or more throughput gain is observed when sequential download is used and the backhaul link to the gateway is poor. As expected, the improvement is very marginal, if at all, when the backhaul link has higher capacity similar to the AP-to-client link. However, when non-sequential download is used, because of the use of opportunism the performance improvement is significant, roughly by a factor of 4, even with the higher capacity backhaul.

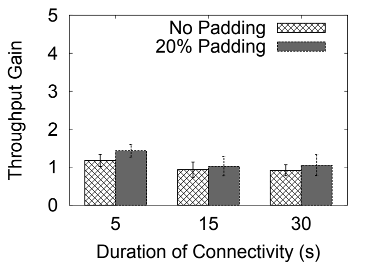
7.2 Driving Experiments with Uncontrolled APs

We now evaluate prefetching in realistic driving scenarios. Here, we cannot control the backhaul bandwidth and the duration of connectivity, as they now come from historical estimates. We show the throughput gains when using prefetching, and using different padding levels and sequential downloads.

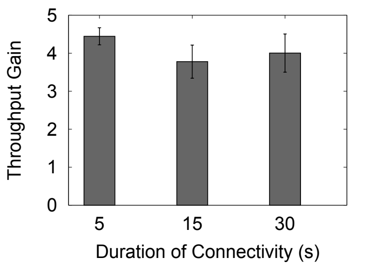
We use the same location where we evaluated handoff strategies as shown in Figure 8. This experimental setup is



(a) 2Mbps backhaul, 0.97 Mbps TCP throughput, sequential download.



(b) 11Mbps backhaul, 3.75 Mbps TCP throughput, sequential download.



(c) 11Mbps backhaul, 3.75 Mbps TCP throughput, non-sequential download, no padding.

Figure 11: Results from Scenario A (Lab) experiments.

shown as Scenario B in Figure 10(b). Since the APs at the location are not in our control[6] we cannot install any software on them to do prefetching. Here, we 'emulate' prefetching using exchange of large ping packets instead of actual file download. Prefetched data downloaded from an AP to a client is emulated as a one-hop ping on the wireless link between them. Data being downloaded from the server is emulated as an end-to-end ping between the server and the client. The client generates 1400 byte ping request packets and waits for a reply for 1 second. On receiving a reply, or after a 1 second timeout, whichever comes earlier, another ping request is generated. This simulates a simple, stop-and-go transport protocol. The choice of transport protocol is not important since we choose the same transport protocol for prefetching and for downloading from the server on the Internet. We choose to use ping because it is the only packet that commercial APs respond to. Based on the successful pings received, the client maintains an offset of the last byte downloaded. The client maintains the prefetch ranges that are computed and acts as if the ranges are always successfully prefetched at the APs. Thus, there is no messages such as *PF_OFFER* or *PF_REQUEST*, etc.

In Figure 12 we present the averaged results (along with the 90% confidence intervals) in the form of the total data in bytes that are downloaded per drive. There are 10 drives for each different prefetch strategy as outlined in the figure labels. Each drive is approximately 120 s long. For handoffs we have used the scripted handoff (SH) technique described before. Note approximately 20-60% improvement in download performance by using prefetch with various padding levels. With 50% padding, there is an 100% overhead as each byte is prefetched twice on different APs; however, the download performance is about 60% higher.

7.3 Driving Experiments with Controlled APs

These experiments are the closest to a real deployment that we have envisioned in this paper. See Figure 10(c). This is same as the Scenario B except that APs with the implementation of the prefetch protocol as used in Scenario A are now also deployed in the residential area. These APs connect to the same 6 neighborhood APs in Scenario B that now act as gateways.

For this deployment we use higher power 802.11b/g cards 25 dBm on the Soekris SBCs for better wireless link quality. Similar cards are also used on the client side. This is done

[6]In fact, they are various commodity APs that are closed boxes belonging to the residents in the community.

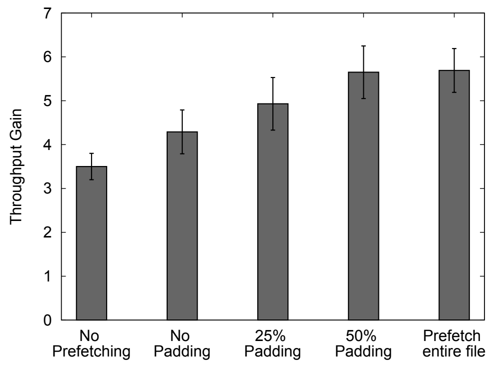


Figure 12: Results for Scenario B (driving experiments with uncontrolled APs) with sequential downloads.

to make the experiments more similar to Scenario B before. In our experience, the commodity APs in the residences are already using wireless interfaces in a similar power range. In general, however, since the scenarios A,B,C are using slightly different hardware, the data are not comparable across scenarios. However, relative comparisons within each scenario are still quite valuable.

In these experiments we perform 10 drives. Non-sequential download is used as it provides the maximum potential performance benefit. No padding is used. Note the significant performance gain over no prefetching scheme with prefetch now shown in a per-AP fashion (Figure 13). No strong correlation is observed for the connectivity duration which are all somewhat similar. The performance differences are likely due to variations on the backhaul bandwidth and the quality of the AP-to-client links. The overall aggregate throughput gain is about 2.5.

8. CONCLUSIONS AND FUTURE WORK

WiFi was not developed for outdoor use, let alone using it from moving cars. Every aspect of WiFi communication, from the physical layer performance (due to the change in the fading environment), to handoff performance (due to use of slow and suboptimal handoff strategies) suffers when used from a moving car accessing APs in the wild. While the basic physical layer performance cannot be helped without a change in the standard (these issues are already in

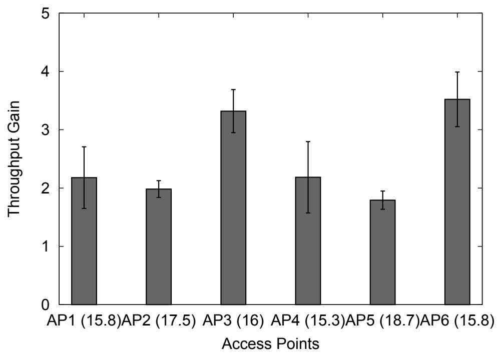


Figure 13: Results for Scenario C (driving experiments with controlled APs) with non-sequential downloads and no padding. The numbers within bracket are the connectivity durations in sec for the APs.

the works in the 802.11p working group [26]), our work demonstrates that handoffs can be implemented very efficiently by using predictive methods at the link layer and up. While these techniques depend on use of historical information, such data collection is straightforward and can be done directly on-board the client device. Use of scripted handoff based on advance RF fingerprinting reduces failed handoffs, improves outages and improves overall throughput by a factor of 2, as our experiments demonstrate. Further, use of prefetching improves the performance by a factor of another 2.5, if non-sequential download is chosen. Overall, our experiments show that more than 200 Kbps application layer throughput is possible with scripted handoff and over 500 Kbps with the use of pre-fetching, rivalling cellular data speeds on many available technologies. Note also that the prefetching technique is link-layer independent and should be useful in cellular data networks as well.

There is still a significant amount of work that needs to be done before the ideas in the paper can be gainfully used. First, the current work lays out the prefetching architecture and demonstrates that prefetching can significantly accelerate downloads. But it does not investigate estimation mechanisms for the prefetch range computation (Section 6.2). More data collection and analysis from realistic drive traces are needed to develop appropriate estimation mechanisms. Second, sharing of historical data by clients will enable better scripted handoff and prefetching. However, sharing of such data in a privacy-preserving fashion must be devised. Sharing RF fingerprints also gives rise to other issues. For example, different clients are likely to use different wireless cards. Thus, RF fingerprints from different cards may not be comparable. But the handoff locations may still be independent of the card used. More study is needed to ascertain this aspect. To enable sharing, an appropriate architecture needs to be designed as well. Third, we have not yet explored how the system could scale to a large number of vehicles. Our focus in the current work is not on capacity constrained networks. We have studied only connectivity aspects. Large number of vehicles using the proposed techniques in the same area will raise several additional technical challenges. For example, scripted handoff may result in all cars in the same location trying to handoff to the same APs. Some amount of diversification is thus needed. Also, interference and bandwidth sharing between different cars connected to the same AP will be important for performance. Storage limitations on the APs for prefetching can be an issue as well.

ACKNOWLEDGMENTS

This work was partially supported by NSF grants CNS-0751121, CNS-0721665, CNS-0721455, IIS-0713186, CNS-0519734 and CNS-0423460; a grant from CONACYT, Mexico; and equipment support from the NY Center of Excellence in Wireless and Information Technology (CEWIT). We thank the anonymous reviewers and our shepherd Professor Marco Gruteser for their insightful comments that helped improve the paper.

9. REFERENCES

[1] Fon. `http://www.fon.com/`.
[2] Multiband Atheros driver for WiFi (MADWIFI). `http://sourceforge.net/projects/madwifi/`.
[3] Wifi.com. `http://www.wifi.com/`.
[4] IEEE Standard for Information Technology – Telecommunications and information exchange between systems – Local and metropolitan area networks specific requirements 802.11r Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Fast Basic Service Set (BSS) Transition , July, 2008.
[5] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level measurements from an 802.11b mesh network. *ACM SIGCOMM Comput. Commun. Rev.*, 34(4), 2004.
[6] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proc. ACM MobiCom*, 2005.
[7] I. F. Akyildiz and W. Wang. The predictive user mobility profile framework for wireless multimedia networks. *IEEE/ACM Trans. Netw*, 12:1021–1035, 2004.
[8] A. Aljadhai and T. F. Znati. Predictive mobility support for QoS provisioning in mobile wireless environments. *IEEE Journal on Sel. Areas in Comm. (JSAC),*, 19(10):1915–1930, 2001.
[9] D. W. Allan. Time and frequency (Time-Domain) characterization, estimation and prediction of precision clocks and oscillators. *IEEE Transactions UFFC*, 34(6), Nov 1987.
[10] A. Balasubramanian, B. Levine, and A. Venkataramani. Enhancing interactive web applications in hybrid networks. In *Proc. ACM MobiCom*, pages 70–80, 2008.
[11] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan. Interactive WiFi connectivity for moving vehicles. *ACM SIGCOMM Comput. Commun. Rev.*, 38(4):427–438, 2008.
[12] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani. Web search from a bus. In *Proc. ACM CHANTS Workshop*, 2007.
[13] A. Bhattacharya and S. K. Das. LeZi-Update: an information-theoretic framework for personal mobility

tracking in PCS networks. *Wireless Networks*, V8(2):121–135, March 2002.

[14] V. Brik, A. Mishra, and S. Banerjee. Eliminating handoff latencies in 802.11 WLANs using multiple radios: Applications, experience, and evaluation. In *Proc. Internet Measurement Conference (IMC)*, 2005.

[15] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden. A measurement study of vehicular Internet access using in situ Wi-Fi networks. In *Proc. ACM MobiCom Conference*, 2006.

[16] S. Cheshire, B. Aboba, and E. Guttman. Dynamic configuration of IPv4 link-local addresses, May 2005. RFC 3927.

[17] G. Cho. Using predictive prefetching to improve location awareness of mobile information service. *Springer Lecture Notes in Computer Science*, pages 1128–1136, 2002.

[18] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing residential broadband networks. In *Proc. Internet Measurement Conference (IMC)*, pages 43–56, 2007.

[19] J. Eriksson, H. Balakrishnan, and S. Madden. Cabernet: A WiFi-Based Vehicular Content Delivery Network. In *Proc. ACM MobiCom Conference*, 2008.

[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1, RFC 2616.

[21] J. Froehlich and J. Krumm. Route prediction from trip observations. In *Proc. Society of Automotive Engineers (SAE) World Congress*, April 2008. Paper 2008-01-0201.

[22] A. Giannoulis, M. Fiore, and E. W. Knightly. Supporting vehicular mobility in urban multi-hop wireless networks. In *Proc. ACM MobiSys Conference*, 2008.

[23] E. Guttman. Autoconfiguration for IP networking: Enabling local communication. *IEEE Internet Computing*, 5(3):81–86, 2001.

[24] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular opportunistic communication under the microscope. In *Proc. ACM MobiSys*, 2007.

[25] N. Imai, H. Morikawa, and T. Aoyama. Prefetching architecture for hot-spotted networks. In *IEEE International Conference on Communications (ICC), vol.7*, 2001.

[26] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *Proc. IEEE Vehicular Technology Conference (VTC) Spring 2008*, pages 2036–2040, 2008.

[27] Z. Jiang and L. Kleinrock. Web prefetching in a mobile environment. *IEEE Personal Communications*, 5(5):25–34, 1998.

[28] A. Joseph, J. Tauber, and M. Kaashoek. Mobile computing with the Rover toolkit. *IEEE Transactions on Computers*, 46(3):337–352, 1997.

[29] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *Proc. IEEE Infocom*, pages 1–13, 2006.

[30] J. Krumm. A Markov model for driver turn prediction. In *Proc. Society of Automotive Engineers (SAE) World Congress*, April 2008. Paper 2008-01-0195.

[31] B. Liang and Z. J. Haas. Predictive distance-based mobility management for multidimensional PCS networks. *IEEE/ACM Trans. Netw.*, 11(5):718–732, October 2003.

[32] V. Mhatre and K. Papagiannaki. Using smart triggers for improved user performance in 802.11 wireless networks. In *Proc. ACM MobiSys Conference*, pages 246–259, 2006.

[33] V. Navda, A. 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 Conference*, 2007.

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

[35] A. J. Nicholson and B. D. Noble. Breadcrumbs: forecasting mobile connectivity. In *Proc. ACM MobiCom Conference*, pages 46–57, 2008.

[36] J. Ott and D. Kutscher. Drive-thru Internet: IEEE 802.11b for automobile users. In *Proc. IEEE Infocom*, 2004.

[37] J. Ott and D. Kutscher. A disconnection-tolerant transport for drive-thru Internet environments. In *Proc. IEEE Infocom Conference*, 2005.

[38] P. N. Pathirana, A. V. Savkin, and S. Jha. Mobility modelling and trajectory prediction for cellular networks with mobile base stations. In *Proc. ACM MobiHoc*, pages 213–221, 2003.

[39] I. Ramani and S. Savage. SyncScan: practical fast handoff for 802.11 infrastructure networks. In *Proc. IEEE Infocom 2005*, pages 675–684 vol. 1, 2005.

[40] M. Shin, A. Mishra, and W. A. Arbaugh. Improving the latency of 802.11 hand-offs using neighbor graphs. In *Proc. ACM MobiSys Conference*, 2004.

[41] Soekris Engineering. `http://www.soekris.com`.

[42] L. Song, U. Deshpande, U. C. Kozat, D. Kotz, and R. Jain. Predictability of wlan mobility and its effects on bandwidth provisioning. In *Proc. IEEE Infocom*, pages 1–13, 2006.

[43] L. Song, D. Kotz, R. Jain, X. He, D. Coll, and N. Hanover. Evaluating location predictors with extensive Wi-Fi mobility data. In *Proc. IEEE Infocom*, volume 2, 2004.

[44] A. Subramanian, P. Deshpande, J. Gao, and S. R. Das. Drive-by localization of roadside WiFi networks. In *Proc. IEEE Infocom Conference*, pages 718–725, 2008.

[45] War-driving. `http://www.wardriving.com`.

[46] T. Ye, H. Jacobsen, and R. Katz. Mobile awareness in a wide area wireless network of info-stations. In *Proc. ACM MobiCom Conference*, pages 109–120, 1998.

[47] J. Yoon, B. D. Noble, M. Liu, and M. Kim. Building realistic mobility models from coarse-grained traces. In *Proc. ACM MobiSys Conference*, pages 177–190, 2006.

[48] F. Yu and V. Leung. Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Comput. Netw.*, 38(5):577–589, April 2002.