“Extra-Sensory Perception” for Wireless Networks

Lenin Ravindranath, Calvin Newport, Hari Balakrishnan, and Samuel Madden
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, MA, USA
{lenin,cnewport,hari,madden}@csail.mit.edu

ABSTRACT

Commodity smartphones and tablet devices now come equipped with a variety of sensors, including accelerometers, multiple positioning sensors, magnetic compasses, and inertial sensors (gyros). In this paper, we posit that these sensors can be profitably used to improve the performance of wireless network protocols running on these mobile devices, and introduce the idea of using *external sensor hints* for this purpose. We focus on mobility hints, including the device’s state of motion, speed, direction of movement, and position. We outline how these hints can be used to: increase throughput by adapting bit rate selection to the state of movement; reduce the bandwidth required for estimating link delivery probabilities; improve the connectivity of routes in vehicular mesh networks using directionality hints; and enable access points to tailor the management of clients to their mobility.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Network Architecture and Design — Wireless Communication

General Terms

Design, Experimentation, Measurement

1. INTRODUCTION

Mobility introduces difficult problems that wireless network protocols must solve to achieve good performance. When nodes move, the vagaries of wireless communication become more pronounced: channel quality varies rapidly, losses become bursty, and assessments of channel behavior are quickly outdated. These effects degrade many wireless protocols: for example, they impair many bit rate adaptation schemes, expand the overhead of topology and neighbor maintenance, and increase the amount of work required to correctly estimate routing metrics.

To compound the problem, strategies that compensate for the difficulties of mobility are unlikely to be optimal for static scenarios. When nodes are static, they can average estimates of channel quality, observe their neighbors, and compute routes over relatively long time scales (many seconds), obtaining and updating observations from many packets. In so doing, they can correctly avoid reacting to the inevitable short-term variations due to small-scale fading that even static wireless networks encounter. In contrast, when nodes move, they should not maintain long histories because the rapidly changing channel conditions and network topology quickly render old information invalid.

With the proliferation of “truly mobile” devices such as smartphones and light tablet computers, it is increasingly common for protocols to encounter *both* static and mobile modes in a short time period, motivating the need for protocols that can adapt to the demands of both settings.

The problem: Protocols optimized for static scenarios fall short when nodes are in motion, but protocols that compensate for the complex characteristics of mobility fall short when nodes are static (this point has been noted in previous work, for instance on bit rate adaptation [4, 11], and we provide some further evidence in Section 2.1). The fundamental differences between static and mobile channels mean not only that protocols need to adapt within each mode (static or mobile) to get good performance, but that the adaptation will likely be quite different in each case. Previous work on wireless protocols has generally not differentiated between these modes.

Our approach: Our position is that adapting using *explicit* knowledge of the operating mode is superior to schemes that adapt using only packet loss, bit error, or signal strength information obtained from network packets. The key insight in our work is that nodes can use *external sensor hints* to determine the mobility mode and adapt accordingly. By “mobility mode,” we mean attributes such as whether the device has started moving or is static, its speed of motion, the heading (direction) of motion, and the location of the device (indoors or outdoors)—all factors that affect the performance of wireless network protocols. The hints are external to the wireless network, but the sensors required for these hints are available on commodity mobile devices today, which come equipped with a wide array of position sensors (GPS, WiFi, cellular radios), accelerometers, compasses, gyros, and so on. To our knowledge, external sensors have not been previously used to augment network protocols.

Sensor hints may be used in different ways in different protocols. When a node generates a hint locally or receives a hint from a neighbor, it may adapt in response to it. The adaptation might be continuous in nature (e.g., updating protocol parameters) or hybrid (e.g., switching from a static-optimized to a mobility-optimized protocol). When hints need to be sent between nodes, they may be piggy-backed on the headers of link-layer frames, or a separate hints protocol may be used.

The rest of this paper highlights the benefits of using external hints to improve wireless protocols. We show how hints from commodity accelerometers, positioning sensors, and compasses can be used to:

1. Increase throughput with better bit rate adaptation, by 30% to 50% on average over frame-based and SNR-based protocols in our experiments.
2. Reduce the bandwidth consumed by probing protocols to accurately estimate link delivery probabilities—by a factor of 20 in our experiments.
3. Select well-connected paths in vehicular mesh networks, increasing route stability by a factor of 4 to 5 compared to a hint-free approach in our simulations.
4. Improve how access points manage association, scheduling, and pruning of clients.

2. HINT-AWARE PROTOCOLS

The ideas described in this paper use hints for *movement*, *speed*, *position*, and *heading*. Movement is a boolean hint that is true if, and only if, a device is moving. Specifically, it is true if either the device's acceleration or its speed is non-zero. On a commodity device, we obtain this information from the acceleration sensor indoors, and from the combination of GPS and the acceleration sensor outdoors. Note that it is important to capture the situation when a device has just started moving from being at rest, and vice versa, so measuring the acceleration is important. By looking for spikes in the magnitude of the 3-axis accelerometer signal over a recent window of samples, we have found that we can reliably detect movement within 100 ms on off-the-shelf smartphones.

To determine the speed and position outdoors, we use GPS. Indoors, we approximate the speed by integrating the time-series of values reported by the accelerometer (the results are more approximate than outdoors, but the range of speeds is a lot smaller). We can use a WiFi localization for indoor positioning if required. Heading can be determined directly from digital compasses (magnetometers) that are available on several devices.

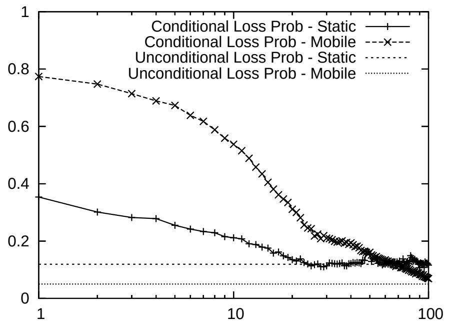

Figure 1: Given a packet lost (at 54 Mbps), this graph shows the conditional probability of losing a subsequent packet after the lag k specified by the x-axis value. Results are shown for both the stationary and mobile case. The unconditional packet loss probabilities in both cases are also shown.

2.1 Bit Rate Adaptation

Node mobility affects the performance of bit rate adaptation protocols significantly by destabilizing wireless channel conditions and causing large and bursty changes over short intervals of time. When a node moves, bit errors and packet losses exhibit a higher degree of statistical correlation with past behavior compared to the static case.

We demonstrate this effect in Figure 1, which plots the *conditional probability* of losing packet number $i+k$ at a given bit rate, *given that* packet number i was lost, for different values of k (the "lag"). In this indoor experiment, we sent back-to-back 1000-byte packets at 54 Mbps from a stationary laptop to another stationary laptop in the static case, and to a laptop carried by a human walking in the mobile case. Small values of k show a significantly higher conditional loss probability in the mobile case, demonstrating a larger degree of short-range dependence compared to the static case. In this scenario, for the mobile case, the 10 packets following a lost packet are significantly more likely to be lost than in the static case, and also compared to larger values of k. The probability does not return to the base-line loss rate until approximately $k = 50$ packets. Given that we send about 5000 back-to-back packets every second at 54 Mbps, the data suggests a channel coherence time of roughly 8 to 10 ms. These observations indicate that when a node is mobile (human walking), the channel changes approximately every 10 ms. These results also suggest that the optimal strategy for bit rate adaptation is likely to be different when nodes move than when they are static.

In the static case, where the channel remains relatively stable, it makes sense to maintain a longer history of performance at different bit rates to smooth over periods of short-term fading or contention. Such a long-history approach falters when the device is mobile. In the mobile case, it makes more sense to keep only a short history, react quickly to losses, and perhaps sample other rates with an equal aggressiveness to track the faster changes typical of a mobile channel. This observation motivates a *hint-aware bit rate adaptation scheme*: a strategy that changes adaptation protocols depending on whether or not the nodes are moving.

Although many bit rate adaptation protocols have been developed, we find that none of them work well when nodes exhibit a combination of mobile and static modes. By using external sensor hints rather than making decisions solely based on network information (the fate of packets and bits, and SNR), our approach is able to combine schemes tuned separately for the static and mobile cases. It requires no training to achieve good performance.

With these remarks in mind, we introduce *RapidSample*, a new frame-based rate adaptation protocol designed for a channel undergoing rapid changes due to movement. In the static case, we use SampleRate [3], relying on movement hints to switch between the two.

The *RapidSample* Protocol. *RapidSample* is a simple protocol that starts with the fastest bit rate. If a packet fails to get a link layer ACK, the protocol reduces to the next lowest rate and records the time of the failure. After success at a particular bit rate for more than $\delta_{success}$ milliseconds (5 in our experiments), the sender attempts to sample a higher bit rate. Specifically, it chooses the fastest bit rate such that: a) the rate has not failed in the last δ_{fail} milliseconds (10 in our experiments) and b) there are no slower bit rate that has failed within this interval. If the faster rate fails, it returns to the original rate from before the sample. If it succeeds, the protocol adopts this new faster rate.

There are four ideas motivating *RapidSample*. First, we observed in several experiments, when a packet fails, the probability the next few packets at this bit rate will fail is high (see Figure 1). Therefore, the protocol immediately reduces the bit rate to prevent oversampling the same bit rate and losing packets. Second, as we showed in our discussion of Figure 1, the coherence time of the channel during movement was small (around 8 to 10 ms). We use this value for δ_{fail} as the minimum time to wait before sampling a previously failed rate, and any rate higher than the failed rate. Sampling faster would have a high probability of failing. Third, we attempt to sample higher rates after only a small number of successes at the current rate. We set $\delta_{success}$ to be less than δ_{fail}. It is difficult to tell if the channel conditions are improving or degrading. With *RapidSample*, however, we expect that if the conditions are degrading, we would be decreasing our rate. A small number of successes at the current rate provide enough confidence to begin sampling the higher rates that have not recently failed. Fourth, if we are wrong about the channel improving, and a sample of a higher rate fails, we revert to the original rate from before the sample. This approach

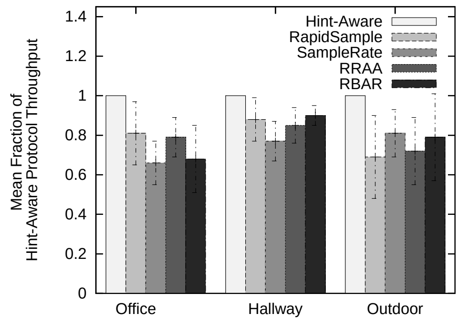

Figure 2: The hint-aware protocol that switches between RapidSample and SampleRate based on movement hint performs significantly better than other schemes in every environment.

allows for opportunistic jumps (as opposed to the common strategy of stepping by one rate).

Preliminary Experiments. Because it is difficult to replicate mobility between different experiments, we used trace-driven simulation—feeding real-world experimental data to a wireless simulator, allowing for both reproducibility and realism. We used the same experimental architecture as [11], which modified the ns-3 network simulator (*v*3.2) to read in experimental traces describing, for each 5 ms timeslot, the fate of each packet sent at each bit rate during that time slot. This setup bypasses the physical layer's propagation model, instead referencing the trace file to determine if a packet should be received successfully. To collect the traces we configured a laptop (using the Click Router, MadWifi driver, and an Atheros 802.11 chipset) to send a constant stream of 1000 byte packets, cycling through the 802.11a OFDM bit rates (6 to 54 Mbps). Each cycle through the 8 bit rates took approximately 5 ms. A second laptop logged every received packet. This laptop was equipped with an accelerometer that provides movement hints.

We collected several traces from three different environments for static and mobile scenarios, including: 1) an *office setting* with no line-of-sight between the sender and receiver, 2) a *long hallway* with line-of-sight between the nodes, and 3) an *outdoor setting* with a lightly crowded outdoor pavement area.

We evaluated the following frame-based bit rate adaptation protocols: *RapidSample*, SampleRate [3], RRAA [12], and our *hint-aware* method that switches between *RapidSample* and SampleRate, depending on the sensor hint. We also evaluated two SNR-based rate adaptation protocols: RBAR [5] and CHARM [6]. For both these schemes, we trained the protocol for the operating environment. We also assumed that the sender has up-to-date knowledge about the receiver SNR. We report only the results for RBAR, as both schemes performed almost identically. Our focus here is on frame-based and SNR-based rate adaptation schemes that can be implemented today without modifying current physical layers, so we don't consider schemes such as SoftRate [11] and AccuRate [8].

Results. Figure 2 shows the results for the three environments. For each environment, we collected 10-20 traces. Each trace is 20 second long with 50% static and mobile time. The traffic workload was TCP. The graph shows the average throughput of all the schemes as a fraction of hint-aware protocol throughput. The error bars show the 95% confidence interval. In every environment, the hint-aware protocol obtained significant performance gains. It improved over SampleRate by up to 50% and over the other schemes by up to 30% in throughput on average.

Also *RapidSample* performs best in the mobile case, and it performs worst in the static case (we do not show the separate results here due to lack of space). In most environments *RapidSample* performs up to 70% better than SampleRate in the mobile case and up to 30% better than other protocols. At the same time, its performance is nearly 30% *worse* than all other protocols when the nodes are stationary. The poor performance is because RapidSample aggressively reduces the rate even with a single loss and frequently tries to sample higher rates even when the channel conditions are almost stable. These results confirm our intuition that the optimal strategy for static and mobile modes differ, and shows the potential of our hint-aware strategy.

2.2 Probing Protocols

Many wireless network protocols maintain per-neighbor information, which the nodes obtain by periodically sending and processing *probe packets* (usually broadcast, and often at multiple bit rates). For example, mesh routing protocols maintain routing metrics for each link in the topology and distribute information about paths. A key parameter in such probing protocols is the *probing rate*.

In determining the frequency of these probes, two opposing considerations must be reconciled. On the one hand, sending frequent probes allows the nodes to maintain an accurate estimate of link qualities and identify topological changes. Maintaining accurate estimates avoids packet losses and enables the best bit rate to be used. On the other hand, frequent probe packets use large amounts of the bandwidth and increase network contention. This trade-off becomes even more acute in mobile settings, where link quality changes rapidly.

Hints from acceleration sensors can improve the performance of probing protocols. The idea is simple: because channel conditions vary much more in the presence of movement, probe more frequently when a node receives movement hints from its neighbor or itself, and probe less often when the nodes are static.

To evaluate the potential gains of this approach, we gathered data from various movement scenarios. Our experimental setup has the sender sending a probe at a high rate of 200 probes per second. We calculate the actual delivery probability over a sliding window of 10 packets, sub-sampling the outcome of these probes to determine the delivery probability at different probing rates. Each such aggregation produces one delivery probability sample.

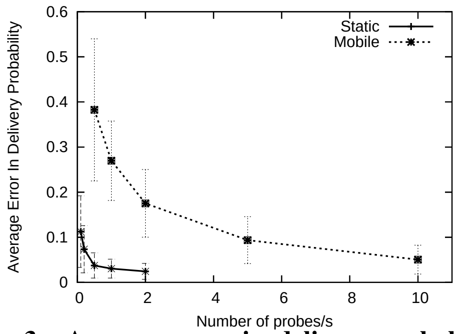

Figure 3: Average error in delivery probability by probing rate for static case and mobile case.

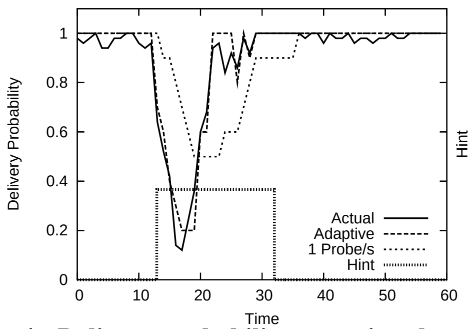

Figure 4: Delivery probability over time by probing strategy for static and mobile combined trace. A raised value of the movement hint indicates the receiver is moving.

We collected two different sets of data: the first consisting of 20 different static traces and the second consisting of 20 different pedestrian mobile traces. Each trace was 180 seconds long. For each data set, we calculate the **error in the delivery probability estimate** as a function of the probing rate, where the error is measured as the magnitude of the difference in the estimated and actual delivery rates. Figure 3 shows the *average* error in delivery probability calculated from all the error samples for the static and mobile data sets as a function of the probing rate, with the error bars showing the standard deviations. When the node is static, even low probe rates result in small error rates, whereas in the mobile case, a 20$\times$ higher probe rate is required to achieve the same error rates. For example, to achieve $< 5\%$ error rate requires .5 probes/sec for the static case and 10 probes/sec for the mobile case.

We implemented a simple hint-driven topology maintenance protocol using rates of 1 and 10 probes per second for the static and mobile cases, respectively. Figure 4 shows a representative 25-second trace that compares the performance of our protocol to the standard 1 probe per second protocol. We also plot the movement hint, with a raised value indicating movement. Notice that our adaptive protocol maintains an accurate assessment of the actual delivery probability throughout the experiment, while the non-adaptive strategy lags by multiple seconds.

2.3 Vehicular Network Route Selection

Vehicular mesh networks can be used to augment cellular WAN connections with routes to roadside infrastructure (e.g., 802.11 access points), when available. Such a system can increase throughput and reduce the load on the more expensive cellular links. Vehicular mesh networking strategies are complicated by dynamic neighbor tables. When a route breaks due to node movement, packets are lost in the buffers of the intermediate nodes on the route, and will have to be retried after discovering a new route, increasing overhead and latency. We hypothesize that selecting routes with longest expected connection time is a good idea in these highly dynamic networks. In this section we propose a hint-based metric to aid these decisions, and perform a preliminary simulation-driven analysis to support its effectiveness. We do not exhaustively compare against other possible routing metrics here.

Connection Time Estimate Metric. We would like each node to prefer neighbors who it is likely to remain connected to for longer periods of time, whenever possible. To do that, each node appends a *heading* hint to its neighbor probes. A pair of neighboring nodes can estimate their connection time using the difference in degrees between their headings. Given an underlying mobility model that assumes movement is constrained onto a common set of one-dimensional segments—as is the case for roads—it follows that smaller differences in headings should predict a longer duration as neighbors. Hence, we propose a metric called the connection time estimate (CTE), which is the inverse of the difference in heading between the two nodes sharing a link, where difference in heading is a value between 0 and 180 degrees. The CTE value for a multi-hop route may be estimated as the minimum CTE value over all hops. Though more complex estimates are possible—for example, combining position, direction, and speed with detailed road geometry—the heading-based approach is simple.

Results. To evaluate CTE, we used a collection of vehicular mobility traces generated from raw position samples gathered from taxis in an urban setting, map-matched to an underlying road network. We combine a collection of traces into a *network*, and then simulate, for each second, the position of every vehicle in the network. We consider two vehicles to have a *link* at a given time if and only if they are within 100 meters at that time in their traces.[1]

We measured the relationship between heading difference and link duration. Specifically, we studied 15 networks consisting of 100 vehicles each, representing a variety of day-time traffic conditions. For each of the $16,523$ links observed in these simulations, we calculated the difference in headings between the two vehicles when the link begins (in degrees), and the total duration of the link

[1]For simplicity, we use geographic proximity as a crude surrogate for a connection.

[0,9)	[10,19)	[20,29)	[30,180]	All Links
66	32	15	9	16

Table 1: The median link duration in seconds for different intervals of heading differences in degrees (180^o indicates nodes headed in opposite directions). Links with similar headings have a median duration 4 **to** 5 **times longer than the median duration over all links.**

(in seconds). We divided the links into four buckets based on the difference in initial headings. Table 1 reports the median link duration in seconds for each of these heading difference buckets. We find that the difference in heading is a strong predictor of link duration. For vehicles with headings within 10 degrees, the median link duration is 66 seconds. This value roughly halves with each successive increase of 10 degrees, falling to a median of 9 seconds by the time the headings are 30 degrees apart.

2.4 Access Point Policies

In this section we introduce potential improvements to three basic WiFi access point (AP) functions using hints: association, packet scheduling, and disassociation. We have not implemented or evaluated these ideas.

Adaptive association. Most clients today associate with the AP that has the strongest signal. When a client node is moving, however, other factors such as the node's heading might provide an important clue about the best AP to associate with. For example, if a node is moving towards an AP, then it is likely to remain associated longer than if it is moving away. We suggest that clients include mobility hints—whether they are moving, their position, and their heading—in their probe requests. APs that receive the request can respond with a *score* indicating the predicted association lifetime, taking these hints as well as signal strength information into account. Alternatively, clients can query a local or network database to determine the score, allowing the APs to remain unmodified. Clients select the AP with the highest score.

Adaptive packet scheduling. The standard approach to AP packet scheduling is to divide transmissions evenly among the clients with pending packets. When mobility is introduced, however, this approach may not optimize throughput. Consider a static client, S, associated with an access point A, and a mobile client, M, that associates with A for a brief period before disassociating. Suppose A dedicates more time to M than S during the interval when M is associated: although this approach temporarily increases the latency for S, it does not decrease its overall throughput, assuming that the batch of packets to be sent to S is finite. This strategy, however, does increase the total number of packets received by M, assuming that there are sufficient packets for M in A's queue. Thus, aggregate throughput will increase.

Adaptive disassociation. Another mobility-induced issue in infrastructure networks is pruning AP state when a client moves out of range. We performed an experiment with a commercial AP and two associated client nodes—one static and one that begins static and then moves out of range. We calculated the received TCP throughput at each client. Initially, both clients roughly share the available bandwidth. Once the mobile client moves out of range, however, the throughput to the *remaining* client drops significantly and remains low for about 10 seconds, before recovering to use the entire available bandwidth. The throughput drops because the AP is unaware of the movement of the first client, and continues to send it packets. None of these frames generates a link-layer ACK, so the AP retries each at the slowest bit rate. Meanwhile, the bit rate to the second client remains high, but because the AP implements frame-level fairness—attempting to send roughly an equal number of packets to each client—the result is a drop in throughput at the second client.

To circumvent this problem we suggest a better disassociation protocol. When a client detects movement, it informs the AP. The AP handles failed ACKs from a mobile client more conservatively—only occasionally probing to determine the nodes presence and not retransmitting unacknowledged packets. Once the AP receives a packet from the mobile client again, it can return to its default aggressive transmission behavior. If the AP fails to hear after a sufficient period, it prunes the association. This scheme avoids the significant degradation observed in our experiment, and incurs only a small overhead.

3. DISCUSSION

Related work. We believe that this paper is the first to advocate the use of sensor hints to improve wireless network protocols. There has been some work that has looked at the use of additional sensors (e.g., low-power radios) as a hint for when a device should power on [10, 1, 2]. BlueFi [2] uses GSM towers and nearby Bluetooth devices to predict if WiFi connectivity is available. In the context of vehicular networking, there has been some work on using GPS position to choose which AP to associate with [7]. CARS [9] is an inter-vehicle bit rate adaptation protocol that uses knowledge of the speed and distance between communicating cars to pick a bit rate. Their method is to collect a large amount of training data for an environment to determine the best bit rate to use at different speeds and distances; in contrast our hint-aware bit rate adaptation method does not require any such training.

We conclude by outlining some additional promising uses of sensor hints in wireless networks.

Changing physical layer parameters. If we are able to modify the physical layer, then changing OFDM parameters based on whether or not the device is outdoors may be beneficial. 802.11a/g is known to work poorly in outdoor environments because of the longer and more varied multipath effects outdoors, which induce a longer delay spread and increase inter-symbol interference. A node that knows it is outdoors can adjust the length of the cyclic prefix parameter to adjust the delay spread to be more tolerant to longer delays.[2] A simple way to determine if a node is outdoors is to see if it acquired a GPS lock.

The coherence time of a channel depends on the node's speed. At vehicular speeds, the coherence time can drop to less than the duration of a single packet [4, 11]. Hence, the channel estimation from the packet preamble might not hold for all symbols in the packet. Using a speed hint from the GPS, the sender can perform channel estimation mid-packet, or reduce the maximum frame size it sends.

In conclusion, we think the use of external sensor hints has the potential to substantially alter how wireless networks operate.

4. ACKNOWLEDGMENTS

This work was supported by NSF grants 0931550 and 0721702, and in part by Ford Motor Company.

5. REFERENCES

[1] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *MobiSys*, 2007.

[2] G. Ananthanarayanan and I. Stoica. Blue-Fi: Enhancing Wi-Fi performance using Bluetooth signals. In *MobiSys*, 2009.

[3] J. Bicket. Bit-rate Selection in Wireless Networks. Master's thesis, MIT, February 2005.

[4] J. Camp and E. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. In *MOBICOM*, 2008.

[5] G. Holland, N. Vaidya, and P. Bahl. A Rate-adaptive MAC Protocol for Multi-Hop Wireless Networks. In *MOBICOM*, 2001.

[6] G. Judd, X. Wang, and P. Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *MobiSys*, 2008.

[7] V. Navda, A. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. MobiSteer: Using directional antenna beam steering to improve performance of vehicular Internet access. *MobiSys*, 2007.

[8] S. Sen, N. Santhapuri, R. R. Choudhury, and S. Nelakuditi. AccuRate: Constellation Based Rate Estimation in Wireless Networks. In *NSDI*, 2010.

[9] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode. CARS: Context aware rate selection for vehicular networks. In *ICNP*, 2008.

[10] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *MOBICOM*, 2002.

[11] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. In *SIGCOMM*, 2009.

[12] S. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *MOBICOM*, 2006.

[2]One might imagine simply searching for a good cyclic prefix, but our point is that the hint can make such a search efficient.