Improving Wireless Network Performance Using Sensor Hints

by

Lenin Ravindranath Sivalingam

B.E. Computer Science
Anna Univeristy, 2006

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2010

Author ....
Department of Electrical Engineering and Computer Science
August 20, 2010

Certified by ....
Hari Balakrishnan
Professor, MIT EECS
Thesis Supervisor

Accepted by ....
Professor Terry P. Orlando
Chairman, Department Committee on Graduate Students

Improving Wireless Network Performance Using Sensor Hints

by

Lenin Ravindranath Sivalingam

Submitted to the Department of Electrical Engineering and Computer Science
on August 20, 2010, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

Users of wireless devices often switch between being stationary and in motion while transferring data. Protocols that perform well in the static setting (where the channel conditions are relatively stable), however, tend to perform poorly when in motion (where channel conditions change rapidly), and vice versa. To circumvent this problem, we note that commodity smartphones and tablet devices come equipped with a variety of sensors, including accelerometers, multiple positioning sensors, magnetic compasses, and inertial sensors (gyros) that can provide hints about the device's mobility. In this thesis, we posit that these sensors can be profitably used to improve the performance of wireless network protocols running on these mobile devices and introduce an architecture for using *external sensor hints* for this purpose.

We validate this idea with many different wireless protocols. First, we show how access points can perform better rate adaptation by changing strategies when they receive hints about a client's mobility. Second, we show how probing protocols for topology maintenance in mesh networks can increase both their efficiency and accuracy by adaptively probing based on movement. Third, we show how vehicular mesh networks can use directionality hints to improve the connectivity of routes. Finally, we outline several other novel applications of *external sensor hints* for improving wireless network performance.

We evaluate our protocols using trace-driven simulation and real-world experiments. We show that our hint-aware rate adaptation protocol increases throughput by 30% to 50% on average over frame-based and SNR-based protocols. Our hint-aware probing protocol reduces the bandwidth consumed by probing to accurately estimate link delivery probabilities—by a factor of 20 in our experiments. And, our hint-aware route selection in vehicular mesh networks increases route stability by a factor of 4 to 5 compared to a hint-free approach in our simulations.

Thesis Supervisor: Hari Balakrishnan
Title: Professor, MIT EECS

Acknowledgments

Foremost, I would like to thank my advisor, Hari Balakrishnan, for his constant support and encouragement. Over the past two years, Hari has been a friend, a guide, and most of all, a great source of inspiration. I would also like to express my gratitude to Sam Madden, who has been like a co-advisor to me. After Hari, Sam has been my next stop for advice on research and things beyond research.

I am grateful to my mentors and colleagues at Microsoft Research for introducing me to research and for motivating me to pursue graduate school.

I thank my collaborator, Calvin Newport, for his contributions to this thesis. I thank Bernhard Haeupler and Michal Rabani for the discussions during early stages of this project. I thank my officemates, Arvind, Katrina, and Mythili for their patience and help on various things, from linux to latex.

I thank all my friends at school, college, and elsewhere, for their advice and support. I especially cannot express enough thanks to Thangam for always being next to me.

Finally, I am extremely grateful to my family: my parents and my sister, for their love, care, and motivation all through these years.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been a phenomenal increase in the usage of handheld wireless devices such as smartphones and tablets. With over 172 million devices sold in 2009, smartphones represent one of the fastest growing markets worldwide [26]. Today, smartphones account for 25% of the US mobile phone market [21] and is expected to double within the next two years [21]. It is also predicted that smartphones and tablets will surpass worldwide PC sales by the end of 2011 [20]. With such tremendous growth, these Wi-Fi devices are increasingly preferred by users for Internet and data access. In fact, in 2009, smartphones generated 48% of the mobile web and application traffic [3] and is expected to be the dominant mode of Internet access in the future [11, 19].

In addition to computing and connectivity, today almost every smartphone and tablet comes equipped with a wide array of sensors like GPS, accelerometer, compass, and so on[1]. The primary application of the sensors on these devices include navigation, gaming, and providing better user interfaces. And recently, these sensors have been extensively used by several crowd-sourcing and sensing applications [7, 14, 23]. Despite these *application layer* uses, these sensors are completely ignored by the lower layers such as the wireless networking stack and the protocols that drive them. Moving away from the norm, in this work, we ask the following question: *Can information from the sensors be profitably used by the networking stack to improve the performance of wireless network?*

[1]Most laptops today also have built-in sensors such as accelerometers [1] that are accessible by applications [2]

With the proliferation of "truly mobile" connected devices such as smartphones and tablets, there is a greater demand for wireless network protocols to adapt to different settings. It is increasingly common for smartphone and tablet users to be mobile while accessing data over the wireless. And, unlike laptop users, they switch between stationary and mobile modes more often and wireless protocols should deal with *both* static and mobile setting in a short time period. (Consider, for example, the smartphone user at the supermarket who alternates between standing still in front of product displays and moving between aisles, all the while streaming through the in-store network.).

Wi-Fi protocols today are typically optimized for static settings and they perform poorly when users are mobile. And even those protocols that try to adapt to the complex characteristics of mobility fall short when nodes are static. This is due to the fundamental difference between the wireless channel and the protocol strategies when nodes are static and when they are mobile. This point has been noted in previous work as well, for instance on bit rate adaptation [6, 24], and we provide some further evidence in Chapter 3 and Chapter 4.

When nodes are static, channel conditions are relatively stable, so protocols can average estimates of channel quality and update observations from many packets. In so doing, they can correctly avoid reacting to the inevitable short-term variations that even static wireless networks encounter (for example, due to short-term fading). Also, with static nodes, the network topology is hardly changing, so protocols can probe for neighbors less frequently, and compute routes over long time scales (many seconds) reducing the overhead of link maintenance.

When nodes move, the vagaries of wireless communication become more pronounced: channel quality changes rapidly, losses are bursty and assessments of channel behavior are quickly outdated. So protocols should not maintain long histories because the rapidly changing channel conditions would quickly render them invalid. And, due to node mobility, network topology changes rapidly, so protocols should probe for neighbors more often and compute routes over much shorter time scales.

These differences between static and mobile settings 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 has generally not made the distinction between these modes due to lack of explicit knowledge about a devices state (whether it is static or mobile). Hence, they attempt to adapt seamlessly across extremely different network conditions leading to sub-optimal performance.

We argue that, the demand for Wi-Fi protocols to adapt to mixed mobility settings created by the proliferation of smartphones and tablets can in fact be met by leveraging additional information that are available from the sensors in these devices. Sensors such as GPS, accelerometer and compass can readily give us *hints* about the mobility mode of the device. By mobility mode, we mean attributes such as whether the device has started moving or is static, its speed of motion and the heading (direction) of motion—all factors that affect the performance of wireless network protocols. Using explicit knowledge of the mobility mode from *sensor hints*, wireless protocols can deal with different modes separately. These *hint-aware protocols* are superior to protocols that try to meet the demand of different settings using only packet loss, bit error, or signal strength information obtained from network packets.

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). We introduce a novel sensor-augmented wireless architecture and a hint protocol that allows devices to communicate hints and adapt based on them. To the best of our knowledge, ours is the first work to make use of *external sensor hints* to augment network protocols and improve wireless network performance.

We validate this sensor-augmented wireless architecture by implementing and evaluating three hint-based protocols: 1. for bit rate adaptation 2. for neighbor/topology maintenance 3. routing in vehicular mesh networks. These protocols make use of *mobility hints* generated from the sensors, and we show through trace-based simulation and real world experiments that switching strategies based on these hints yields significant performance increases. We also outline several other novel applications of using *external sensor hints*. We describe how clients can perform better access point association, how access points can better schedule their transmissions between static and mobile clients and how access points can better prune mobile clients. We also describe how sensor hints can be used for power saving and changing wireless physical layer parameters.

Though the protocols in this paper focuses on "mobility hints" and on improving WiFi performance, we argue that the general impact of sensor hints on network performance can be much broader.

1.1 Contributions

In this thesis we offer the following main contributions:

- We present a novel wireless architecture that uses sensor hints to augment network protocols. We generate mobility hints from the sensors found on commodity mobile devices, to circumvent the mobile/static problem fundamental to any device that communicates wirelessly while in motion. We then describe a *hint protocol* that can be used to communicate a broad class of sensor hints between devices.

- We implement and evaluate a novel frame-based bit rate adaptation protocol, *Rapid-Sample*, and show through trace-based simulation that it obtains upto 70% better throughput than existing frame-based and SNR-based rate adaptation protocol when a node is in motion.

- We implement a hint-aware bit rate adaptation protocol that switches strategies based on mobility hints and show through exhaustive trace-based evaluation that it obtains between 30% to 50% better throughput than existing frame-based and SNR-based rate adaptation protocol in various environments.

- We show through real-world experimentation that maintaining acceptable error rates for topology maintenance while mobile requires over 20 times the amount of traffic as compared to the stationary case. We implement a hint-aware protocol that switches to this expensive probing only when in motion—providing significant bandwidth savings during mixed mobility scenarios.

- We implement a hint-aware route selection for vehicular networks and show using simulations that our protocol increases route stability by a factor of 4 to 5 compared to a hint-free approach.

- We describe how clients and access points can apply hints to improve association, scheduling, and pruning of clients.

The rest of the thesis is organized as follows. In Chapter 2, we describe a novel hint-aware wireless architecture and a framework to capture information from external sensors and incorporate them in network protocols. We propose and implement techniques to extract mobility hints from different sensors in the phone. In Chapter 3, we implement and evaluate a hint-aware bit rate adaptation protocol. In Chapter 4, we implement and evaluate a hint-aware probing protocol for topology maintanence. In Chapter 5 we evaluate the benefit of using direction hints for finding routes in vehicular mesh networks and describe other novel applications of sensor hints. We discuss related work in Chapter 6 and conclude in Chapter 7.

Chapter 2

Design

In this chapter, we first present a wireless architecture that allows protocols to leverage *hints* from sensors found in commodity mobile phones to augment and improve network protocols. Next, we describe simple and accurate techniques for extracting mobility hints from sensors such as GPS, Accelerometer and Compass. We then describe our *hint protocol*—an extension of 802.11 allows communicating devices to share hints. The applications proposed in the rest of the thesis leverage this new wireless architecture and the hint protocol to improve performance.

2.1 Hint-Aware Wireless Architecture

Current wireless protocols use only "in-network" information such as received packet, packet loss, bit error or signal strength information to make decisions or to adapt itself. For example, current rate adaptation protocols use only frame loss, bit error, SNR or PHY information as a cue to adapt bit rate; current routing protocols find routes based on received probes that are periodically sent; and today's access point policies on association, scheduling and pruning clients are purely based on received packet information. Moving away from this norm, we present a novel architecture in which wireless protocols can leverage *external hints* from sensors found on commodity smartphones and tablets in addition to "in-network" information to improve their performance.

Figure 2-1 shows our hint-aware wireless architecture. GPS, accelerometer, compass,

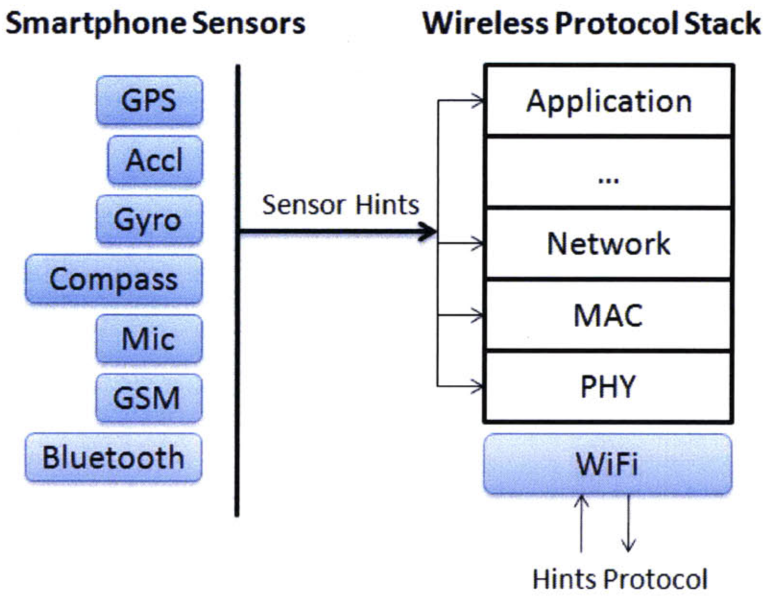

Figure 2-1: The figure shows our hint-aware wireless architecture. *Hints* from sensors on commodity wireless devices feed directly into the wireless networking stack and can be beneficially used by protocols in every layer. In addition to using local hints, nodes communicate hints to other nodes using the *Hint Protocol*.

gyroscope, microphone etc can provide rich contextual information (say, about a device's mobility state or the surrounding environment) which cannot otherwise be inferred from just packet information. In our architecture, these hints can be directly used by protocols at any layer in the networking stack. In addition to using local hints, a protocol can adapt based on hints communicated from other nodes. For instance, a sender can adapt its bit rate based on the mobility state of the receiver (whether the receiver is mobile or static). In this case, the receiver communicates its mobility hint to the sender. This communication is achieved using the *Hint Protocol* we describe in Section 2.3.

Throughout this thesis, we describe several applications to this hint-aware architecture and show how wireless protocols at different layers in the networking stack can improve performance.

2.2 Extracting Mobility Hints

The protocols described in this thesis uses mobility hints. Hints about mobility include *movement*, *heading*, *speed* and *position*. We describe techniques to extract each of these hints in the sections below.

2.2.1 Movement Hint

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.

We now show in detail how to generate these hints using accelerometer. The accelerometer reports force values for its x, y, and z axes, once every 2 ms. These values are reported in custom units.[1] Let $F_t = \langle x_t, y_t, z_t \rangle$ be the force vector reported by the accelerometer, where t is a *report number* that increments with each report. For each report number t we do the following:

1. Calculate the average force values from the recent history:

$$x = \frac{x_t + .. + x_{t-4}}{5}$$

$$y = \frac{y_t + .. + y_{t-4}}{5}$$

$$z = \frac{z_t + .. + z_{t-4}}{5}$$

$$x' = \frac{x_{t-5} + .. + x_{t-9}}{5}$$

$$y' = \frac{y_{t-5} + .. + y_{t-9}}{5}$$

[1] An advantage of our hint algorithm is that we do not need to convert these units into standard force values and/or calibrate with each use of the device to account for the force of gravity, etc.

$$z' = \frac{z_{t-5} + .. + z_{t-9}}{5}$$

2. Calculate the value J_t as:

$$J_t = \sqrt{(x-x')^2 + (y-y')^2 + (z-z')^2}$$

   We call J_t the *jerk* at report time t, as it captures, roughly, the recent change in force on the accelerometer (a big value corresponds to a forceful jerk of the device).

3. Calculate the movement hint H_t as:

$$H_t = \begin{cases} 1 & H_{t-1} = 0 \text{ and } J_t > 3, \\ 1 & H_{t-1} = 1, \exists t' \in \{t-50, ..., t\} : J_{t'} > 3, \\ 0 & H_{t-1} = 1, \forall t' \in \{t-50, ..., t\} : J_{t'} \leq 3, \\ 0 & H_{t-1} = 0, \text{ and } J_t \leq 3. \end{cases}$$

   Note, we initialize $H_0 = 0$.

When queried, the movement hint service returns the most recently calculated hint value. We are able to detect changes in movement status in under 100 ms. Figure 2-2 shows the jerk values over time for an experiment in which a laptop started stationary, was moved, then return to a stationary position. Notice that the value never exceeds 3 when the device was stationary, and that it frequently exceeds 3 (by a significant amount) during the interval of movement. This same result was replicated over many different accelerometers of this type, environments, and movement scenarios (the latter including: carried by a human, rolled on a wheeled chair, and driven in a car). The thresholds of 10 and 50 were also empirically determined, but are more flexible: averaging of the force values when calculating jerk, and considering a small window when determining the hint, compensate for the natural fluctuation in values returned by the accelerometer.

Notice that the calibration of this algorithm is performed only once for this *type* of serial accelerometer. This is more practical than schemes that require calibration for each use of the device.

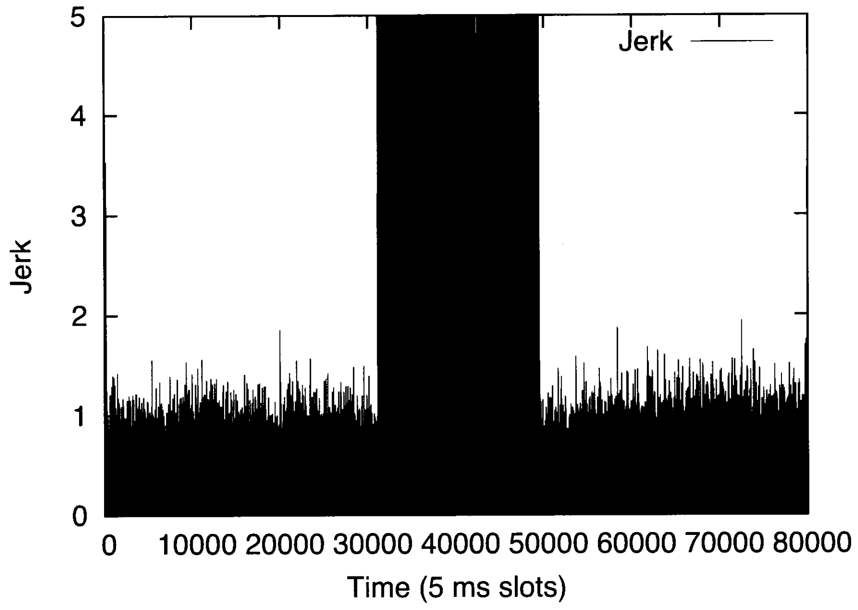

Figure 2-2: The *jerk* value plotted over time. In this experiment, the device started stationary, was moved, and then returned to a stationary position. Notice that that jerk values clearly identify the interval of movement.

This simple boolean hint on movement is used by the protocols described in chapter 3 and chapter 4 to improve bit rate adaptation and topology maintenance respectively.

2.2.2 Heading Hint

Heading can be determined directly from digital compasses (magnetometers) that are available on several devices. GPS also provides us with heading when a device is moving outdoors. These sensors produce a heading in degrees relative to the earth's magnetic north pole.

In addition to compass and GPS, gyroscope sensors found of these devices can be used to track the heading of the device accurately with respect to a reference heading. In practice, the accuracy of the readings from digital compasses found in smartphones depends on the magnetic influence in the environment and can become extremely noisy in some indoor environments. In such scenarios, we propose to use the gyroscope in conjunction with the compass to produce accurate headings.

These heading hints are used by protocols described in chapter 5 to improve vehicular route selection and access point selection respectively.

2.2.3 Speed and Position Hints

To determine the speed and position outdoors, we can directly use GPS. Indoors, we can approximate the speed by integrating the time-series of values reported by the accelerometer (the results will be more approximate than outdoors, but the range of speeds is a lot smaller). For indoor positioning, we can use WiFi localization. We do not use speed and position hints in protocols implemented in this thesis, hence we do not evaluate the techniques for obtaining such hints.

2.3 The Hint Protocol

Our goal is a protocol that allows nodes share their hint values with nearby nodes and access points. Specifically, when a node A is sending packets to a node B, A should learn B's hint(s), allowing it to adapt its strategy accordingly.

For the 802.11 protocol, we stuff these hints is in the link-layer frames. For a simple binary hint, such as the movement hint described in Section 2.2.1, the protocol can use one of the unused bits in the standard 802.11 ACK frame or probe request frame or any other frame, preventing the need to expand the standard. This allows nodes to opportunistically package hints without worrying about whether its neighbors are also running the hint protocol.

For a more general class of hint, the link-layer frame format can be expanded to include an additional two-byte field, sufficient to contain the pair: $\langle hintType, hintVal \rangle$, where the first field describes the type of hint being reported and the second its value.

Nodes can also piggy-back hints at the end of the data frames. If a node has no frames to send, hints can be communicated using a short frame that is recognized only by nodes running the hint protocol. This technique can be easily implemented today without the need to modify the wireless driver. And nodes running the hint protocol can coexist with legacy nodes that are hint-oblivious.

Chapter 3

Bit Rate Adaptation

In this chapter we describe a bit rate adaptation strategy that uses our method to detect movement and the hint protocol to improve performance.

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 3-1, 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 packets at 54 Mbits/s 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 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 Mbits/s, 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 coherence time of the channel is approximately 10 ms. The coherence time of the channel is the time period for which the channel remains the same. These results

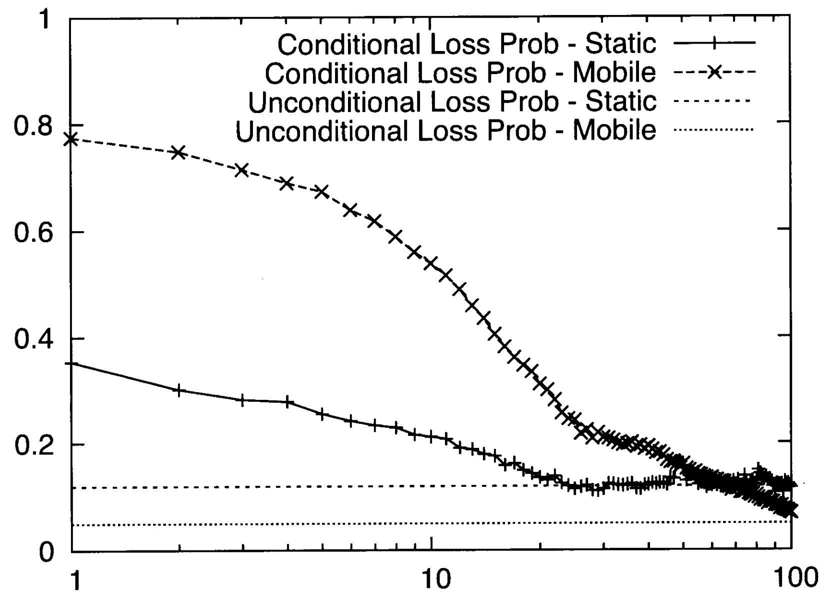

Figure 3-1: Given a packet lost (at 54 Mbits/s), this graph shows the conditional probability of losing a subsequent packet after the lag k specified by the x-axis value. Results are shown for both the stationary and mobile case. The conditional probability of packet loss is much higher in the mobile case than in the static case for $k \leq 10$ packets. The unconditional packet loss probabilities in both cases are also shown.

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 errors, 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 make its decisions based solely 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.

3.1 The *RapidSample* Protocol

The *RapidSample* bit rate adaptation algorithm is shown in Figure 3-2. *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.

```
$\textbf{RapidSample}(lastbr, gotack)$ :
  if $(!gotack)$ then
    $failedTime[lastbr] \leftarrow \textbf{CurrTime}()$
    if $(sample)$ then
      $br \leftarrow oldbr$
    else
      $br \leftarrow max\{0, lastbr - 1\}$
      $sample \leftarrow 0$
  else
    $sample \leftarrow 0$
    if $(\textbf{CurrTime}()$ - $pickedTime[lastbr] > \delta_{success})$ then
      $br \leftarrow max\{i \mid \forall j < i :$
              $failedTime[j] - \textbf{CurrTime}() > \delta_{fail}\}$
      $sample \leftarrow 1$
      $oldbr \leftarrow br$
    else $br \leftarrow lastbr$
  if $br != lastbr$
    $pickedTime[br] \leftarrow \textbf{CurrTime}()$
  return $br$
```

Figure 3-2: The RapidSample bit rate adaptation algorithm. It is called for each packet with *lastbr* describing the bit rate index and *gotack* describing whether an ack was received for the previous packet. Time is reported in elapsed milliseconds.

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 3-1). 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 3-1, 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 allows for opportunistic jumps (as opposed to the common strategy of stepping by one rate). We experimented with different values of $\delta_{success}$ across a range of experiments, and found little difference.

3.2 Hint-Aware Bit Rate Adaptation Protocol

The Hint-Aware Rate Adaptation Protocol implemented at the sender uses *RapidSample* when a node is mobile and uses SampleRate [5] when a node is static. It relies on movement hints from the receiver to switch between the two. We chose to use SampleRate for the static case as it performed better compared to other frame-based and SNR-based protocols in various environments (see 3.5).

3.3 Experimental Setup

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

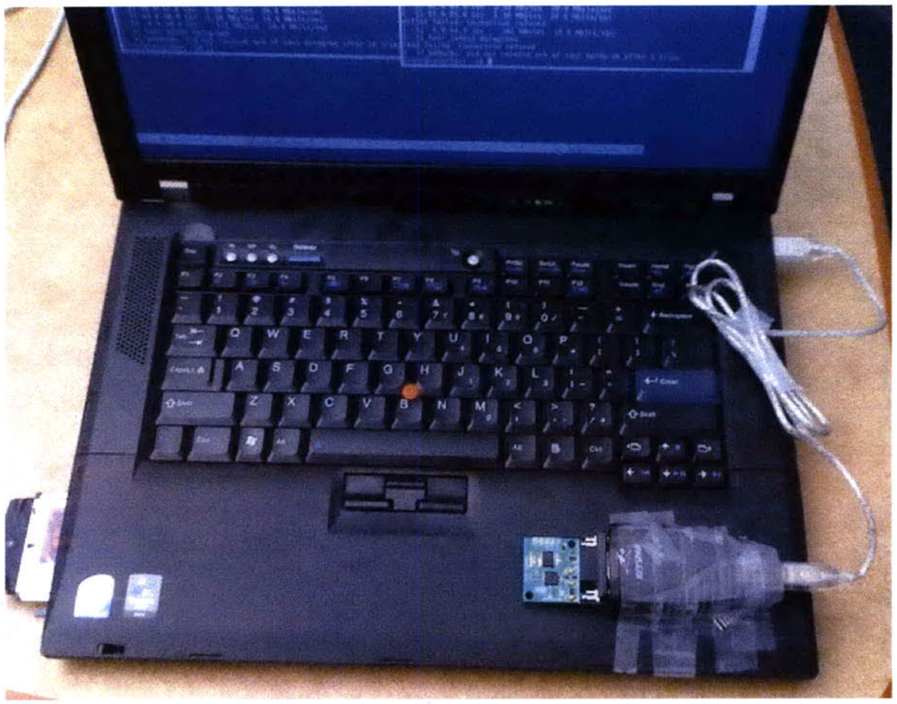

Figure 3-3: Laptop equipped with an accelerometer—the receiver in our measurements.

for both reproducibility and realism. We used the same experimental architecture as [24], which modified the ns-3 network simulator ($v3.2$) 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 Linux laptop as a sender. It ran the Click router using the MadWiFi 802.11 driver, which in turn used an Atheros 802.11 chipset. The laptop sent a constant stream of 1000 byte packets, cycling through the 802.11a OFDM bit rates 6, 9, 12, 18, 24, 36, 48, 54, in round-robin order. Each cycle through all 8 bit rates took approximately 5 ms. We used the 802.11a band to minimize interface with local infrastructure network. We configured a second laptop, also configured Click, MadWiFi, and Atheros, to act as a receiver—logging every received packet. This laptop was additionally equipped with an Sparkfun serial accelerometer that reports force on its three axes once every 2 ms. In Section 2 we describe how to use the accelerometer reports to generate an accurate movement hint. Figure 3-3 shows our experimental setup.

Experiment Type	Method
Stationary	The sender was on a desk and the receiver was positioned at a fixed distance on a wheeled chair. The experiment took place in an open area in one floor of our office building.
Human/Mobile	The sender was on a desk and the receiver was moved at standard indoor walking speed on a wheeled chair. The experiment took place in the same setting as the stationary experiments.
Vehicle/Mobile	The sender was on a picnic table abutting the sidewalk on a long stretch of straight road. There was a light density of parked cars on the sender's side of the road. The receiver was placed on the passenger seat of a car which drove back and forth path the sender at varying speeds between 8 and 72 kilometers per hour (5 and 45 miles per hour).

Figure 3-4: A summary of the experiment types used to evaluate channel conditions under mobility and our protocols from Sections 3 and 4.

Figure 3-4 summarizes the three categories of experiments we conducted: Stationary, Human/Mobile, and Vehicle/Mobile.

We collected several traces from 4 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 3) an *outdoor setting* with a lightly crowded outdoor pavement area, and 4) a *vehicular setting* where the sender is stationary on the roadside and the receiver is in a moving car.

3.4 Evaluation

We evaluated the following frame-based bit rate adaptation protocols: *RapidSample*, SampleRate [5], RRAA [25], and our *hint-aware* method that switches between *RapidSample* and SampleRate, depending on the sensor hint. In our experiments, when we compare against SampleRate, we post-process the trace to determine the best SampleRate parameter to use in each case; this biases our experiments in favor of SampleRate since in reality it would have to select a parameter value before experiments are run. as such, it is more optimistic than what SampleRate with a fixed parameter achieves, but we used this approach to show that the hint-aware scheme can do better than even the best post-facto setting of the

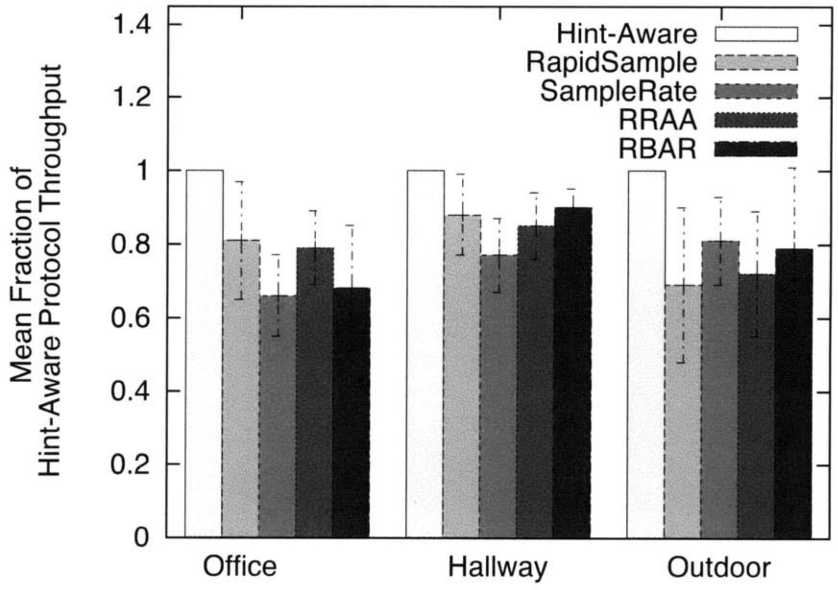

Figure 3-5: The hint-aware protocol that switches between RapidSample and SampleRate based on movement hint performs significantly better than other schemes during mixed mobility scenarios in every environment.

static parameter in SampleRate.

We also evaluated two SNR-based rate adaptation protocols: RBAR [10] and CHARM [12]. 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 [24] and AccuRate [16].

3.5 Results

Figure 3-5 shows the performance of the hint-aware protocol compared to other rate adaptation protocols for three environments. For each environment, we collected 10-20 traces. Each trace is 20 second long with 50% static and mobile periods. The receiver was static for the first 10 seconds and mobile for the next 10 seconds or the vice versa. The traffic

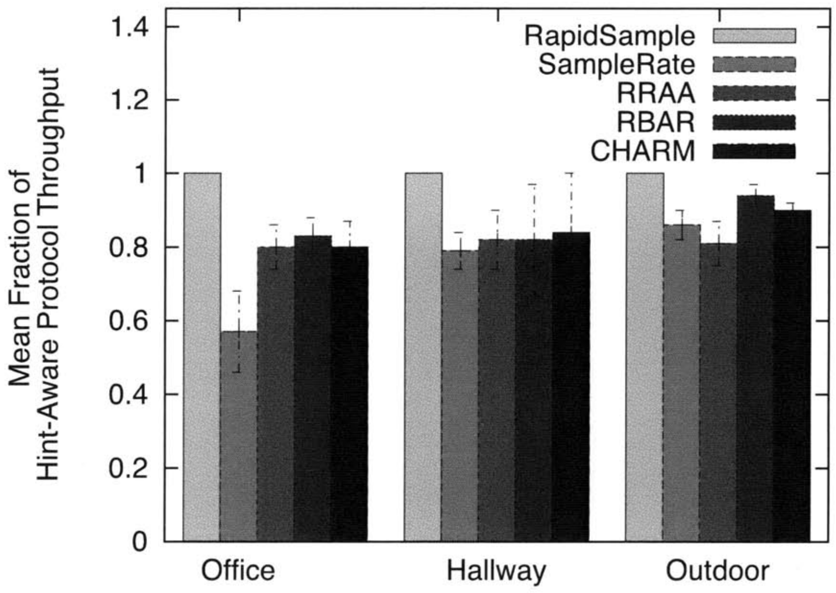

Figure 3-6: In mobile scenarios, *RapidSample* performs significantly better than other protocols.

workload we used to evaluate was TCP. The graph shows the average throughput of all the schemes as a fraction of the throughput obtained by the hint-aware protocol. The error bars show the 95% confidence interval. In every environment, the hint-aware protocol obtained significant performance gains. It improved over SampleRate by 23% to 52% in throughput on average, over RRAA by 17% to 39% and over RBAR by up to 47%. We do not show the numbers for CHARM as the performance of RBAR and CHARM was almost similar in all the cases with RBAR performing slightly better.

In Figure 3-6 and in Figure 3-7 we dive deeper into the evaluation of each of the protocol in mobile and static cases separately. We collected 10 traces that are 20 second long for both the scenarios in each of the environment shown in the result above. Figure 3-6 shows the result for the mobile case. The graph shows the average throughput of all the schemes as a fraction of the throughput obtained by *RapidSample*. It can be seen that *RapidSample* performs significantly better than other schemes in every environment. *RapidSample* obtained up to 75% better throughput on average than SampleRate in the mobile case and up to 25% better than other protocols. The performance gains come from *RapidSample*'s ability to cope up with the rapid fluctuations in the channel conditions when a node is mobile. Figure 3-7 shows the results for the static case. Although, *RapidSample* performs best in the mobile case, it performs worst in the static case. Its achieved 12% to 28% *less* throughput on average compared to SampleRate and upto 18% *less* throughput compared to RRAA 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.

As seen in Figure 3-7, SampleRate consistently achieves higher throughput than other protocols when nodes are static. Hence, we decided to use SampleRate for the static case in our hint-aware rate adaptation protocol.

From Figure 3-6 and in Figure 3-7, it can also be noted RBAR performs slightly better than CHARM in the mobile case and CHARM performs slightly better than RBAR in the static case. RBAR and CHARM are both SNR-based rate adaptation protocols. While CHARM maintains a history of SNR values of recent packets and uses the average SNR, RBAR uses the SNR of the most recently received packet alone to compute the optimal bit rate. In the static case, CHARM performs better than RBAR as it is robust to short-term fluctuations in the SNR. But in the mobile case, CHARM performs slightly worse than RBAR (this results has also been noted in [24]) as the SNR values of received packets become quickly obsolete due to rapidly changing channel conditions. Hence, instantaneous SNR values reflect the channel conditions more accurately than average SNR. These results confirm our intuition that the optimal strategy for static and mobile modes differ, and shows the potential of our hint-aware strategy.

Finally, we evaluate the performance of *RapidSample* in a *vehicular setting*. In this setting, the sender is stationary on the roadside and the receiver is placed in a moving car. We collected 10 traces each 10 second long. The traffic workload in this experiment is UDP, as TCP times out when faced with the high loss rate of the mobile case (a phenomenon also observed in previous work [9]). Figure 3-8 shows the average UDP throughput of all the schemes as a fraction of the throughput obtained by *RapidSample*. Similar to other environments, *RapidSample* outperforms other schemes. It achieved about 28% more throughput than SampleRate, 36% more throughput than RRAA and nearly 2x more throughput than SNR based protocols.

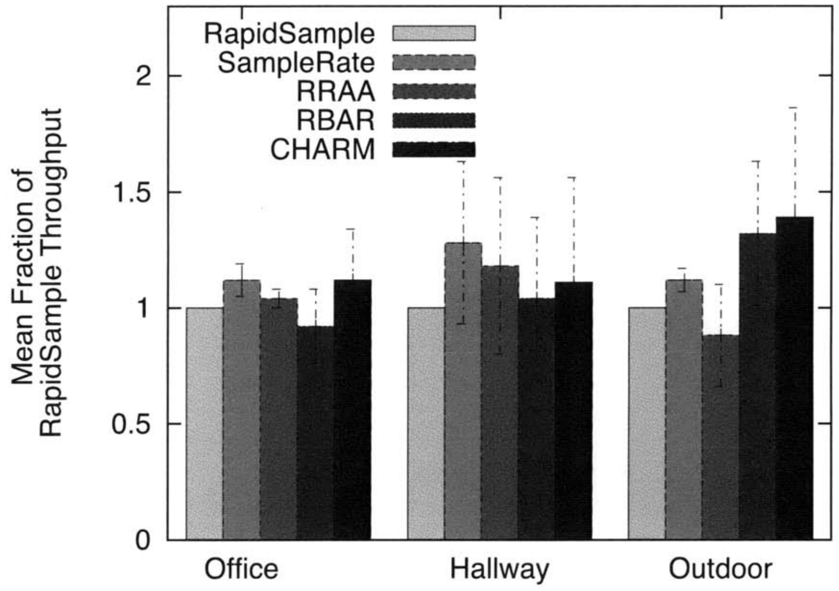

Figure 3-7: In the static case, *RapidSample* performs poorly compared to the other schemes. SampleRate achieves higher throuput in all the environments.

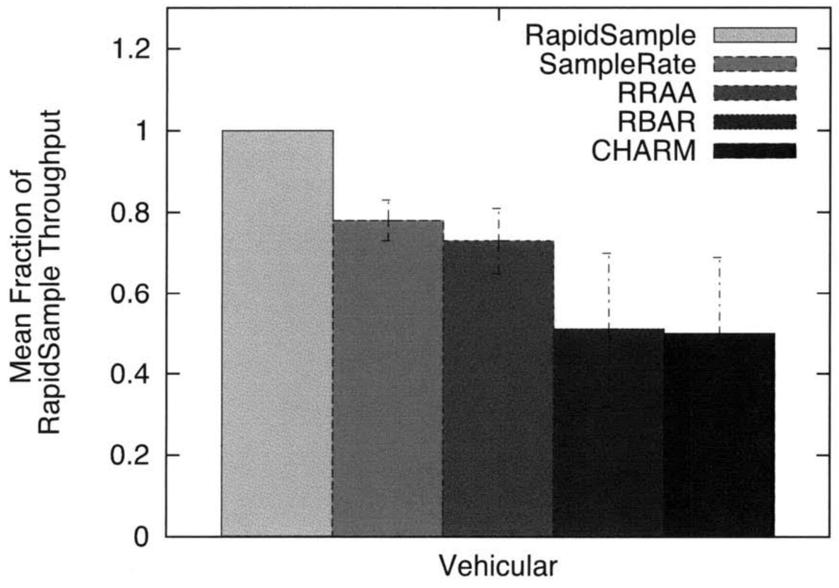

Figure 3-8: In vehicular environments, *RapidSample* achieves significantly higher throughput compared to other schemes.

Chapter 4

Topology Maintenance

Many wireless network protocols maintain a list of neighbor nodes and information of the quality of connectivity to each neighbor. For example, in mesh networks and sensor networks, each node maintains an estimate of the link quality to each neighbor for use in the routing protocol. Another example is in infrastructure networks where each access point (AP) maintains information about currently connected clients.

The standard method to maintain neighbors and link quality information is for a node to send periodic probe packets. Each recipient maintains the packet loss rate of packets from its neighbor, and may exchange this information in the routing protocol's messages. A key parameter is the *probing rate*: how often should these periodic messages be sent? In practice, a node may send these messages at more than one bit rate to produce link quality information at different bit rates.

In determining the frequency of these probes, two opposite considerations must be reconciled. On the one hand, sending frequent probes allows the nodes to maintain an accurate estimate of link qualities and identify changing topology. Maintaining accurate values for this metric is important to avoid packet losses, which can increase the number of retries and also slow down the bit rate, wasting channel bandwidth and increasing packet latency. On the other hand, frequent probe packets themselves use large chunks of the bandwidth and increase network contention. This tradeoff becomes even more acute in mobile settings, where link quality changes rapidly. For instance, Figure 4-1 captures the channel behavior that we observed consistently in all our experiments. This plot shows the packet delivery

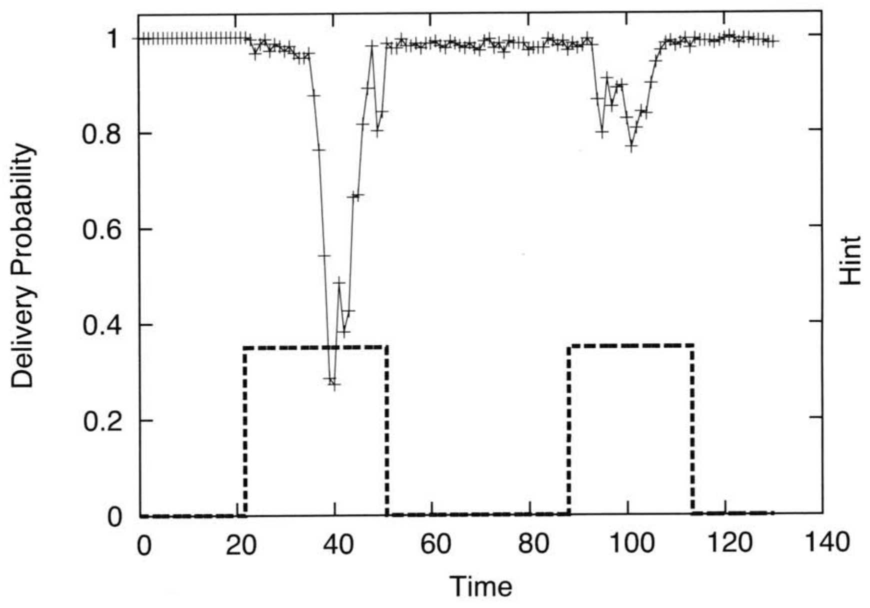

Figure 4-1: Packet delivery rate for 6 Mbps packets over time and movement. The raised hint line indicates the device is moving.

ratio and movement hint over time for one specific trace. To calculate the delivery ratio, we bucketed time into intervals of 1 second, a sufficient duration for the sender to transmit approximately 200 packets at each bit rate. A non-zero value for the movement hint indicates the device is moving; a value of 0 indicates it is stationary. The key observation is that motion causes the packet delivery ratio to fluctuate from second to second, with many of the jumps in the delivery ratio exceeding 20%.

In this chapter, we investigate the extent to which mobility hints can help in adapting the frequency of probing. The idea is simple: because channel conditions vary much more in the presence of movement, probe frequently when a node receives movement hints from its neighbor or itself moves, and probe much less often when the nodes are static.

4.1 Measurement

To evaluate the potential gains of this approach, we gathered data from the stationary and human/mobile scenarios the same way as described in Chapter 3. Our experimental setup

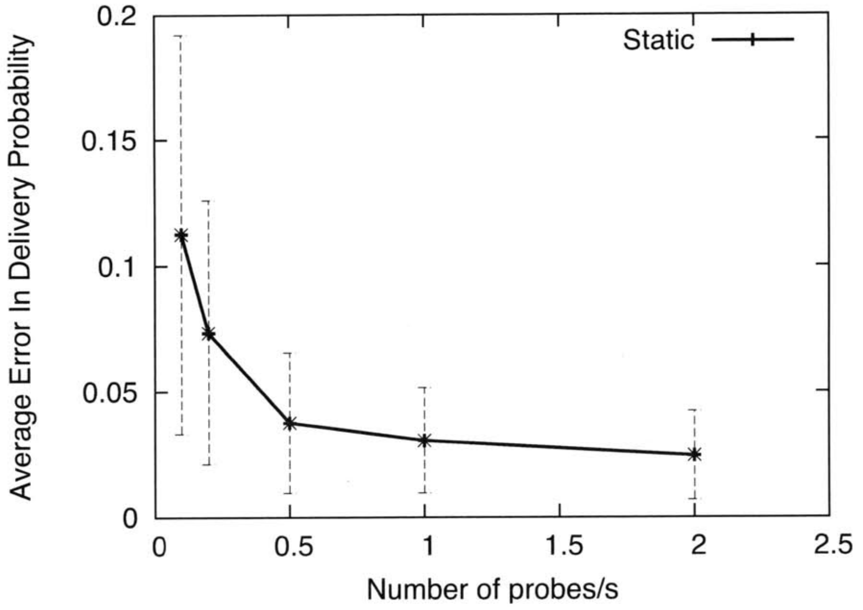

Figure 4-2: Average error in delivery probability versus probing rate for static case.

has the sender sending a probe at an aggressive (essentially continuous) rate of 200 probes per second. We calculate the actual delivery probability over a sliding window 10 packets from these rapidly sent probes, sub-sampling the outcome of these probes to determine the delivery probability at different probing rates. That is, to compute the loss rate at a probing rate of 10 packets per second, we sub-sample the original 200 packets per second stream at the lower rate, and then aggregate the packet delivery probability over 10 sub-sampled packets. Each such aggregation produces one delivery probability sample.

We collected 20 different stationary and 20 different human/mobile traces, each about 180 seconds long. We aggregate the results of the static cases together into one set, and the mobile cases together into another set. For these two cases, we calculate the **error in the delivery probability estimate** as a function of the probing rate. We define the error as follows:

$$\text{Error} = |\text{Observed probability} - \text{Actual probability}|$$

Figure 4-2 shows the *average* error in delivery probability calculated from all the error samples for the static case as a function of the probing rate; the error bars show the standard deviations. Even a low probing rate of 1 packet every 10 seconds has an error of only

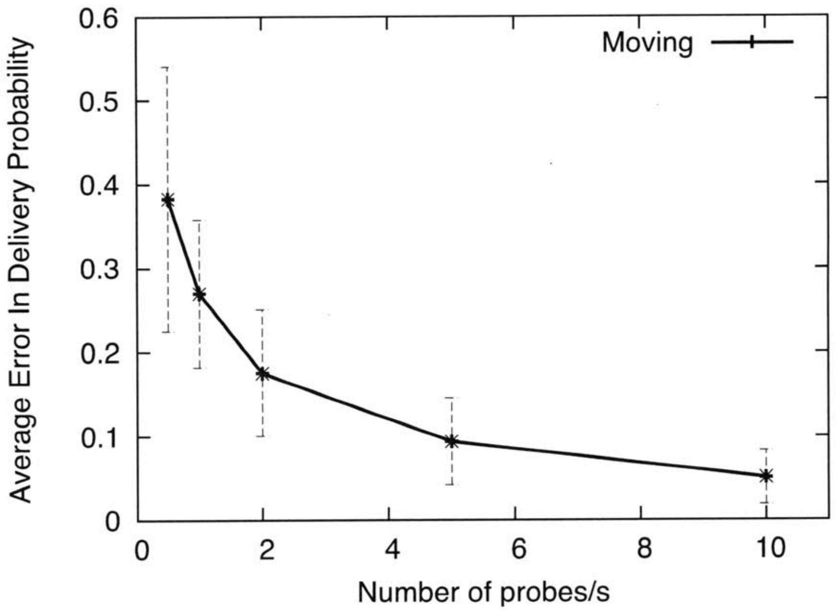

Figure 4-3: Average error in delivery probability versus probing rate for mobile case.

11%, suggesting that at least in our experiments, the default probing rate of many wireless networks of 1 probe per second may actually be too high. In contrast, as shown in Figure 4-3, the error in delivery probability is much higher in the mobile case, going up to over 35% even at a probing rate of 1 packet every 2 seconds. To achieve an error of about 10%, the mobile case requires a probing rate of 5 probes per second, which is more than 25× higher than for the static case at the same error rate. If we want the error to be just 5%, then the mobile case needs 10 probes per second, while the corresponding rate for the static case is 0.5 probes per second, a 20× difference.

Our main finding is that there is a significant (factor-of-20) difference in the probing rates required between the static and moving cases, in order to maintain link quality information to within 5%-10% of the correct value.

The reason for this difference may become apparent from looking at one representative 25-second trace for the static and mobile case, which we show in Figures 4-4 and 4-5 respectively. In the static case, the delivery probability tracks the actual one relatively closely at the three different probing rates. In contrast, in the mobile case,only the high probing rates do; at 1 probe per second (currently the default in many wireless networks),

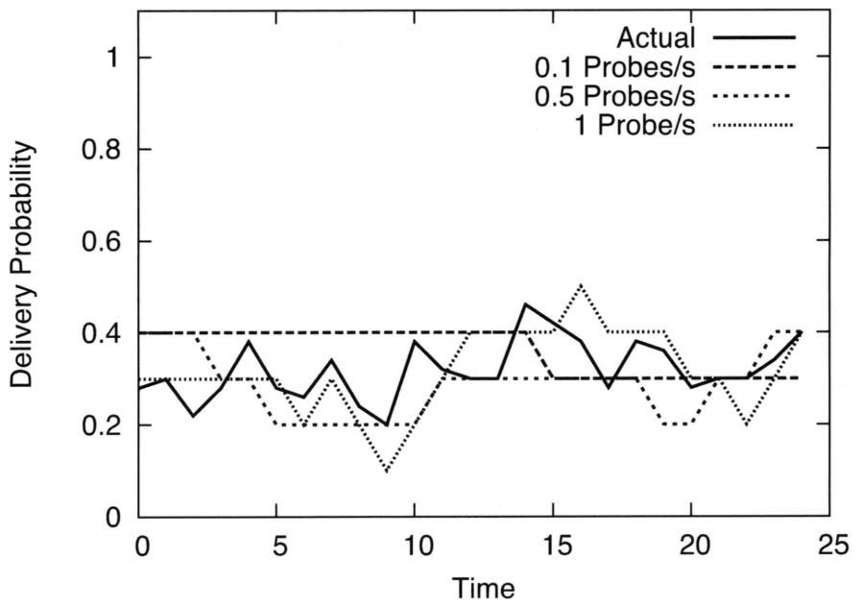

Figure 4-4: Delivery probability by probing rate over time for stationary trace.

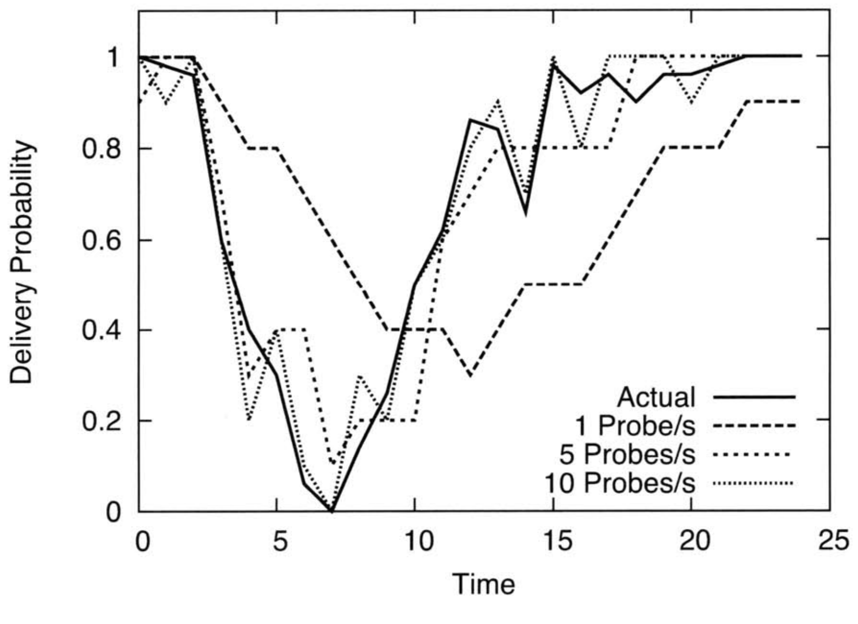

Figure 4-5: Delivery probability by probing rate over time for mobile trace.

the difference from the actual delivery probability is substantial, erring in both directions (i.e., both over- and under-estimating the true value).

4.2 Hint-Aware Topology Maintenance Protocol

Mobility hints allow us to design a protocol that can realize the potential benefits suggested by the measurements described above. The protocol itself is simple: when the hint protocol indicates neighbor movement, or when the node itself moves, increase the probing rate to a value that will provide an acceptable bound on the error in delivery probability. This value has to be trained in our current protocol, but given the large difference in error in the static and mobile cases, we believe that static training is enough. That is, if we probe at 0.5 probes per second in the static case, a movement hint would cause the probing rate to increase by a factor of 10 or 20, to about 10 probes per second for the duration of movement. Of course, one may trade-off between error in delivery probability and probing frequency dynamically as well. If the nodes in the network alternate between moving and not moving, as might be the case for users with laptops and phones, the resulting savings in bandwidth will be substantial, or the accuracy in estimating link quality will be much better (the specific savings or gain in quality will depend on the fraction of time spent in either mode, and on the "default" static strategy).

We implemented this hint-driven topology maintenance protocol using rates of 1 and 10 probes per second for the stationary and mobile cases, respectively. An important issue in the protocol is how to handle the transition from mobile to stationary. Our protocol continues to send at the fast probe rate for one second after the node stops moving, ensuring that all packets in the history window are valid for the recent channel conditions.

Figure 4-6 compares the performance of our adaptive 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 1 probe per second strategy lags by multiple seconds.

To understand the impact of the hint-aware scheme, we now present a very simple

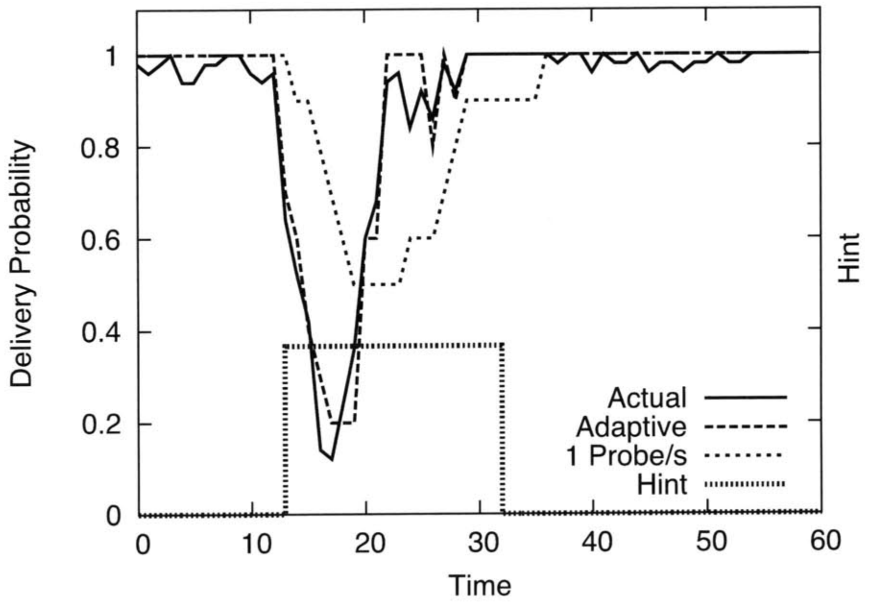

Figure 4-6: Delivery probability over time by probing strategy for static and mobile combined trace. A raised value of the movement hint indicates the receiver is moving.

analysis of what happens when the protocol does not correctly estimate the link quality. Suppose a node uses the ETX metric [8] to pick the next-hop in a mesh network, or to pick a good AP from among a set of choices. Suppose further that there are two choices, one with link delivery probability p_1 and the other with probability p_2, and without loss of generality, let $p_1 > p_2$. ETX would choose link 1, and the expected number of transmissions (ignoring the reverse link for the ACK) would be $1/p_1$. Suppose the error in the average link delivery probability estimate is δ.

The node would pick the wrong link if, and only if, $p_2 + \delta \geq p_1 - \delta$. In this case, the *penalty*, defined as the extra number of transmissions relative to the correct (optimal) value of $\frac{1}{p_1}$, is equal to $\frac{1}{p_2} - \frac{1}{p_1}$. The *overhead*, defined as the ratio of the penalty to the correct (optimal) value is therefore equal to $\frac{p_1}{p_2} - 1$. This number can be quite large for practical parameters; Figure 4-3 shows that δ can be larger than 0.25 in some cases. If we have two links, one with a delivery probability $p_1 = 0.8$ and the other with $p_2 = 0.6$, the overhead, for $\delta = 0.25$, is $5/12 = 42\%$ on that hop, a non-trivial quantity.

Chapter 5

Other Hint-Aware Protocols

In this chapter, we describe several other novel applications of sensor hints for improving various wireless protocols.

5.1 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.

5.1.1 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.

5.1.2 Evaluation

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 (in seconds). We divide the links into four buckets based on the difference in initial headings. Table 5.1 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.

[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 5.1: 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.

5.2 Access Point Policies

In this section we describe potential improvements to three basic WiFi access point (AP) functions using hints: association, packet scheduling, and disassociation.

5.2.1 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 generalize this example with an adaptive association protocol.

We suggest that clients include mobility hints—whether they are moving, their position, and their heading (say, in probe request packets). 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.

The score generation algorithm requires training. It can be implemented in two ways. In the first, access points initially score all augmented probe requests the same, but learn, over time, the hint values correlated with the longest associations. If APs are unmodified, client nodes instead report association durations and their relevant hint values to a central database that can be later queried by other nodes making an association decision.

We consider two different scenarios: 1. Outdoors and 2. Indoors. Outdoors, clients can use GPS to provide hints about its location, heading and speed which can be used to calculate the best association based on the method described above. Indoors, clients can use accelerometer to provide movement hint and compass or gyroscope to provide hint about heading. These hints can be used in conjunction with wireless signal strength information to predict the best AP to associate with.

5.2.2 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. Therefore, favoring the mobile node in the scheduling decisions could increase the total system throughput.

We can generalize this example with an adaptive scheduling protocol. Mobile nodes communicate their movement hint to the AP and the AP can then adjust its scheduling to dedicate a larger fraction of bandwidth to the mobile node.

5.2.3 Adaptive Disassociation

Another mobility-induced issue in infrastructure networks is pruning AP state when a client moves out of range. To see whether this pruning matters, we took a commercial AP and investigated what it does when a client moves out of range during a TCP transfer. Our experiment has two clients. One is static, while the other is initially static and then moves out of range. We calculate the received TCP throughput at each client in Figure 5-1. Initially, both clients roughly share the available bandwidth. One of the node moves away shortly before 35 seconds into the trace. Soon after, the throughput to the *remaining* static node drops precipitously and remains low for about 10 seconds, before recovering to use the entire bandwidth!

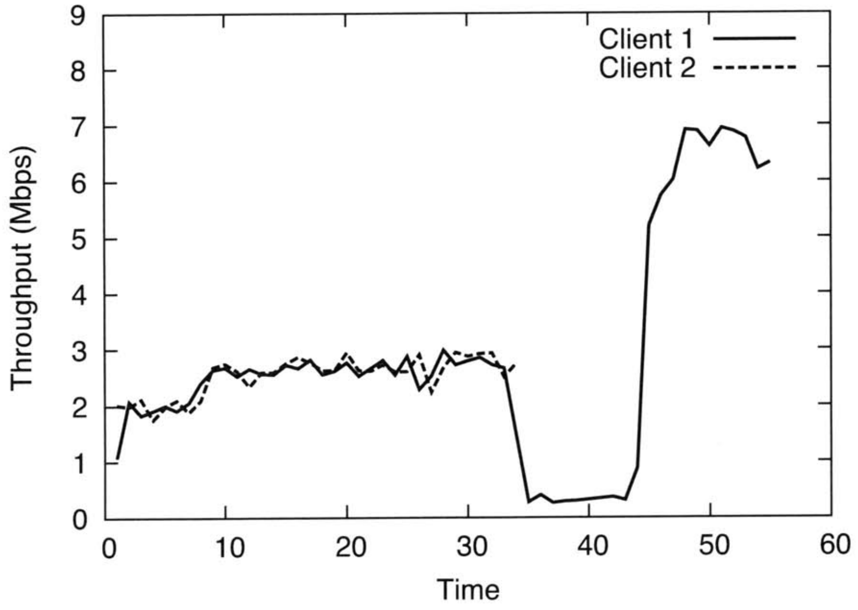

Figure 5-1: Throughput over time for two clients. Client 2 leaves range of the access point around 35 seconds.

What has happened here is that the AP was unaware of the movement of the first client, and continued to send packets to it. Of course, none of the link-layer frames got a link-layer ACK, so the AP re-sent them, before giving up and going to the next TCP packet, repeating the process for all the packets in the TCP window. Finally, after about 10 seconds of getting no response, the AP pruned the absent client.

But why did the throughput to the first client drop so dramatically? The answer has to do with the bit rate adaptation scheme and the fairness model in place at this AP. First, the absence of ACKs caused the bit rate to the first (moved) client drop to the lowest rate 1 Mbps. Meanwhile, the bit rate to the second client remains high, but the problem is that the AP implements frame-level fairness between clients, attempting to send an equal number of packets (more or less) to each client. In the absence of "time-based fairness" [22], the result is a significant drop in throughput. In fact, even if time-based fairness were in place, the resulting throughput to the first client would be only about 50% of what it should be.

The solution is simple: use the mobile hint protocol to have the client inform the AP of movement. When that happens, the AP does not simply attempt to send packets open-loop to a client that has provided such a hint, using a more careful protocol to only very occasionally probe to see whether the client has in fact come back into its vicinity. Of course, when the client sends a packet without any mobility hint, suggesting that it is stable once more, it can revert to the default aggressive behavior. This smarter hint-aware pruning scheme avoids the significant degradation observed in Figure 5-1 at relatively low cost in terms of messaging overhead.

5.3 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, as GPS does not work indoors.

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 [6,24]. 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.

5.4 Movement-based Power Saving

Another application to hints is in power saving. If a client node fails to find an access point for association and it receives a hint that it is not moving, it can power down its radio until it next receives a movement hint. Similarly, if it receives a speed hint that it is moving too fast for useful WiFi communication, it can power down the radio until its speed decreases. These approaches will save energy.

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

5.5 Cellular Networks

Mobile phones are equipped with a variety of sensors that could be used to give feedback in cellular networks. The same problems we observe in WiFi networks—high variations in loss rate due to mobility, difficulty in determining whether a client is still connected, and the frequency with which clients scan for connectivity—all arise in the cellular domain. For example, a cellular base station might adapt its bit rate rapidly using a protocol like RapidSample when interacting with a mobile client, or mobile clients might adapt the frequency with which they probe for nearby base-stations when they know they are (or are not) moving, or even hand-off to a better base station based on speed and location.

5.6 Using the Microphone

A changing environment (e.g., caused by pedestrians or driving cars) surrounding a *static* node can induce dynamic channel conditions similar to what would be experienced if the node itself were moving. In our experiments in such environments, RapidSample performed better than SampleRate. To detect such conditions, a microphone can be used to measure noise variation, which is likely to be highly correlated with nearby activity.

Chapter 6

Related Work

6.1 Hint-Aware Protocols

To the best of our knowledge, ours is the first practical work to explore the potential benefits of sensor hints in improving wireless performance. Related work that uses information outside the wireless networking stack has mostly focused on wireless power saving. For instance, WakeOnWireless [18] uses an additional low power radio that can be used for signaling to wake up the wireless radio. Cell2Notify [4] uses the cellular radio on a smartphone to wakeup WiFi interface for VoIP calls thus reducing the energy consumption of WiFi. BlueFi [4] uses GSM towers and nearby bluetooth devices to predict if WiFi connectivity is available, hence achieving power savings. In contrast to all the above work that leverages an additional radio, we propose a generalized architecture that uses hints from sensors such as GPS and accelerometer to improve performance.

In addition to power savings, hints from external sensors for wireless protocols has been used in the context of vehicular networking. There has been some work on using position and speed information to choose which AP to associate. Mobisteer [15] uses directional antennas in vehicles and uses location hints from GPS to find the best antenna orientation and the AP to associate with. CARS [17] 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 and performs well both indoors and outdoors.

6.2 Bit Rate Adaptation

Several rate adaptation protocols have been proposed in the past. These protocols can be roughly divided into three categories: frame-level, SNR-based and PHY-layer protocols.

Frame-level rate adaptation protocols use frame delivery and loss statistic as a metric to adapt bit rate. Recent frame-level schemes include SampleRate [5] and RRAA [25], which also provide a good survey of other frame-level schemes. SampleRate picks the bit rate that minimizes the average packet transmission time over a ten-second window. It periodically samples higher bit rates to adapt to changing channel conditions. As we showed in Chapter 3, SampleRate works well in static settings but performs poorly when a node is mobile. RRAA is more opportunistic than SampleRate and uses a short time window of frame loss statistics to choose the best bit rate. It also adaptively enables CTS/RTS to decrease collision losses. Though RRAA performs better than SampleRate in mobile settings, it still does not adapt to the rapidly changing channel conditions when a node is mobile. This point has been noted in [24] as well. In contrast to the above schemes, *RapidSample* is designed to perform well in mobile settings. It does not keep long histories and adapts bit rate more opportunistically than SampleRate and RRAA. But it should also be noted that, an aggressive scheme like *RapidSample* does not work well in static settings. Hence, our hint-aware protocol switches between SampleRate and *RapidSample* depending on whether the node is mobile or static.

SNR-based protocols use packet SNR estimate and a SNR-to-bit rate mapping to pick the best bit rate. [13] provides a good survey of these protocols. RBAR [10] uses RTS/CTS exchange to estimate the SNR at the receiver, and picks the bit rate accordingly. To avoid the overhead of RTS/CTS, CHARM [12] relies on the reciprocity of the channel and uses the SNR estimate of the packets overheard from the receiver. While RBAR uses the SNR of the last received packet (RTS packet), CHARM computes average SNR over a time window. It is well know that SNR-to-bit rate mapping changes depending on environment and hence these protocols require training. In contrast, *RapidSample* and our hint-aware protocol do not require any training.

SoftRate [24], AccuRate [16] are PHY-layer rate adaptation protocols that uses physical layer information to predict the best bit rate. While SoftRate estimates the channel BER, AccuRate uses a constellation based approach to pick the best bit rate. Though PHY-based schemes are superior to frame-based and SNR-based schemes, they cannot be implemented on today's wireless hardware that exists in smartphones and access points. In contrast, our hint-aware protocol can be readily deployed in current settings.

Chapter 7

Conclusion

Protocols optimized for stationary nodes work poorly when mobile, and protocols optimized for mobile nodes fall short of stationary performance when not moving. To circumvent this problem, we introduce an architecture that uses sensor hints to detect mobility, and propose that protocols use these hints to adjust their strategies based on the device's mobility state. To validate this concept we proposed, implemented, and evaluated three hint-aware solutions—a bit rate adaptation, a neighbor/topology maintenance protocol and a vehicular route selection protocol. All of them adapt their behavior based on mobility hints generated from commodity sensors, and we show they yield significant performance increases over existing solutions. We also described several other novel applications of this new architecture. We argue that the general use of external (to the network stack) hints has the potential to substantially alter how wireless networking operates.

Bibliography

[1] `http://qcn.stanford.edu/about/laptop.html`.

[2] Detecting device orientation. `https://developer.mozilla.org/en/Detecting_device_orientation`.

[3] Admob mobile metrics report, november 2009. `http://metrics.admob.com/wp-content/uploads/2009/12/AdMob-Mobile-Metrics-Nov-09.pdf`.

[4] G. Ananthanarayanan and I. Stoica. Blue-Fi: enhancing Wi-Fi performance using bluetooth signals. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 249–262. ACM New York, NY, USA, 2009.

[5] John Bicket. Bit-rate Selection in Wireless Networks. Master's thesis, Massachusetts Institute of Technology, February 2005.

[6] Joseph Camp and Edward Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-layer Implementation and Experimental Evaluation. In *MobiCom '08: Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, pages 315–326, New York, NY, USA, 2008. ACM.

[7] Tathagata Das, Prashanth Mohan, Venkata N. Padmanabhan, Ramachandran Ramjee, and Asankhaya Sharma. Prism: Platform for remote sensing using smartphones. In *In Proceedings of Mobisys 2010*, 2010.

[8] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A High-throughput Path Metric for Multi-hop Wireless Routing. In *MobiCom '03: Proceed-*

ings of the 9th annual international conference on Mobile computing and networking, pages 134–146, New York, NY, USA, 2003. ACM.

[9] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: vehicular content delivery using wifi. In *MOBICOM*, pages 199–210, 2008.

[10] Gavin Holland, Nitin Vaidya, and Paramvir Bahl. A Rate-adaptive MAC Protocol for Multi-Hop Wireless Networks. In *MobiCom '01: Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 236–251, New York, NY, USA, 2001. ACM.

[11] ipad internet usage patterns compared to smartphone and pc. `http://ipcarrier.blogspot.com/2010/06/ipad-internet-usage-patterns-compared.html`.

[12] Glenn Judd, Xiaohui Wang, and Peter Steenkiste. Efficient Channel-aware Rate Adaptation in Dynamic Environments. In *MobiSys '08: Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*, pages 118–131, New York, NY, USA, 2008. ACM.

[13] Katrina LaCurts. Measurement and Analysis of Real-World 802.11 Mesh Networks. Master's thesis, Massachusetts Institute of Technology, May 2010.

[14] Prashanth Mohan, Venkata N. Padmanabhan, and Ramachandran Ramjee. Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *In Proceedings of Sensys 2008*, 2008.

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

[16] Souvik Sen, Naveen Santhapuri, Romit Roy Choudhury, and Srihari Nelakuditi. AccuRate: Constellation Based Rate Estimation in Wireless Networks. 2010.

[17] P. Shankar, T. Nadeem, J. Rosca, and L. Iftode. CARS: Context aware rate selection for vehicular networks. In *The sixteenth IEEE International Conference on Network Protocols (ICNP 2008)*, pages 19–22, 2008.

[18] E. Shih, P. Bahl, and M.J. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171. ACM New York, NY, USA, 2002.

[19] Smartphone owners lead rise in mobile internet usage. `https://www.strategyanalytics.com/default.aspx?mod=ReportAbstractViewer&a0=5100`.

[20] More smartphones than desktop pcs by 2011. `http://www.pcworld.com/article/171380/more_smartphones_than_desktop%25%20_pcs_by_2011.html`.

[21] Smartphones grab 25% of u.s. mobile market. http://blog.nielsen.com/nielsenwire/online_mobile/android-soars-but-iphone-still-most-desired-as-smartphones-grab-25-of-u-s-mobile-market.

[22] Godfrey Tan and John Guttag. Time-based fairness improves performance in multi-rate wlans. In *Proc. of USENIX'04*, Boston, MA, June 2004.

[23] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Sivan Toledo, Jakob Eriksson, Sam Madden, and Hari Balakrishnan. Vtrack: Accurate, energy-aware traffic delay estimation using mobile phones. In *In Proceedings of Sensys 2009*, 2009.

[24] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. *ACM SIGCOMM Computer Communication Review*, 39(4):3–14, 2009.

[25] S Wong, H Yang, S Lu, and V Bharghavan. Robust Rate Adaptation for 802.11 Wireless Networks. In *Proc. of ACM MobiCom Conf.*, pages 146–157, Los Angeles, CA, September 2006.

[26] Smartphone sales up 24 percent. http://techcrunch.com/2010/02/23/smartphone-iphone-sales-2009-gartner/.