Augmenting Mobile 3G Using WiFi

Aruna Balasubramanian
University of Massachusetts

Ratul Mahajan
Microsoft Research

Arun Venkataramani
University of Massachusetts

ABSTRACT

We investigate if WiFi access can be used to augment 3G capacity in mobile environments. We first conduct a detailed study of 3G and WiFi access from moving vehicles, in three different cities. We find that the average 3G and WiFi availability across the cities is 87% and 11%, respectively. WiFi throughput is lower than 3G throughput, and WiFi loss rates are higher. We then design a system, called Wiffler, to augments mobile 3G capacity. It uses two key ideas—*leveraging delay tolerance* and *fast switching*—to overcome the poor availability and performance of WiFi. For delay tolerant applications, Wiffler uses a simple model of the environment to predict WiFi connectivity. It uses these predictions to delays transfers to offload more data on WiFi, but only if delaying reduces 3G usage and the transfers can be completed within the application's tolerance threshold. For applications that are extremely sensitive to delay or loss (e.g., VoIP), Wiffler quickly switches to 3G if WiFi is unable to successfully transmit the packet within a small time window. We implement and deploy Wiffler in our vehicular testbed. Our experiments show that Wiffler significantly reduces 3G usage. For a realistic workload, the reduction is 45% for a delay tolerance of 60 seconds.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication

General Terms

Design, Experimentation, Measurement, Performance

Keywords

3G augmentation, WiFi, Applications, Vehicular Networks

MobiSys'10, June 15–18, 2010, San Francisco, California, USA.

1. INTRODUCTION

Mobile Internet access today is suffering the curse of popularity. The ubiquitous deployment of cellular data networks has drawn millions of users to these networks, which is in turn creating immense pressure on the limited spectrum of these networks. Subscribers, especially in big cities, are experiencing deteriorating 3G quality because the network cannot cope with the high demand [25].

In response to this pressure, wireless providers are using methods such as imposing a limit of 5GB per month [26] and "educating" their users on "responsible" access [23]. We believe that such methods are in the end ineffective or at least insufficient. They are against the tide of users' desire for greater consumption.

We investigate the feasibility of a different method— augmenting 3G using WiFi. That is, reduce the pressure on 3G spectrum by using WiFi connectivity when possible for transferring data. At least one wireless provider is offering incentives to its subscribers to reduce their 3G usage by switching to WiFi at home [1]. In addition to reducing pressure on 3G spectrum, such augmentations also reduce the per-byte cost of data transfers, by 70% per one estimate [24].

In this paper, we focus on vehicular Internet access, a particularly challenging case for mobile connectivity. An increasing number of users today access the Internet from moving vehicles, either directly through their personal devices or through proxies inside transit vehicles [5, 11]. A range of other devices, such as navigation units, also need such connectivity.

However, using WiFi networks from moving vehicles is challenging. WiFi APs have a short range and are generally not deployed to provide coverage to roads. Even when APs are in range, the quality of connectivity may be poor [9, 12]. Thus, it is unclear if WiFi can usefully augment 3G, while providing the ubiquity and reliability that 3G subscribers expect.

To understand feasibility, we conduct a detailed study of joint 3G and WiFi access from moving vehicles, in three different cities. We find that on average 3G access is available 87% of the time, while WiFi access (through open APs) is available only 11% of the time. Although we do not understand why, our data suggests that the mutual availability is negatively correlated. In places where 3G is unavailable, WiFi is available roughly half the time. Thus, the combination is more available than if the two had independent availabilities. However, we also find that in half of the locations where WiFi is available, its throughput is much less than 3G. WiFi also experiences a much higher loss rate. To

our knowledge, our work is the first joint characterization of 3G and WiFi; prior works have measured the performance of one or the other [9, 13].

Our study thus suggests that straightforward methods to combine the two will reduce 3G load by at most 11%, and even that will come at the expense of poor application performance.

We design a system called Wiffler to overcome these availability and performance challenges. Its two key ideas are *leveraging delay tolerance* and *fast switching to 3G*. We observe that many applications such as email or file transfer can afford to delay data transfers without significantly hurting user experience. Wiffler leverages this observation to trade-off higher application latency for lower 3G usage. Instead of transmitting data immediately, it waits for WiFi to become available. By using a simple method to predict future WiFi throughput, it waits only if 3G savings are expected within the application's delay tolerance. Additionally, so that the performance of delay and loss sensitive applications is not hurt, Wiffler quickly switches to 3G if WiFi is unable to transmit the packet within a time window.

We implement and deploy Wiffler on a vehicular testbed. We evaluate Wiffler using the deployment and using trace-driven simulations. In our deployment, we observed that for transfers of size 5MB that can be delayed by at most 60 seconds, Wiffler reduces 3G usage by 30%. In simulation using realistic workloads, we find that Wiffler reduces 3G usage by 45% for a 60 second delay tolerance. Because of it's wait-only-if-it-helps strategy, the actual transfer latency is increased by only 7 seconds on average. For a VoIP application, we find that the time periods with good VoIP quality increases by 31% (absolute improvement) using fast switching, compared to a system that switches to WiFi irrespective of its quality. More importantly, the increase in quality is achieved even when 40% of the VoIP traffic was sent over WiFi.

2. 3G AND WIFI CHARACTERISTICS

The goal of this work is to augment 3G networks using WiFi in mobile environments. As a first step, we conduct a joint study of 3G and WiFi network characteristics. We seek to answer the following questions: *(i)* What is the availability of 3G and WiFi networks as seen by a vehicular user? *(ii)* What are the performance characteristics of the two networks?

2.1 Testbeds and methodology

We conducted measurements in three geographically-separate, outdoor testbeds that include effects present in real vehicular settings, such as noise, fading, interference, occlusions, and traffic patterns. We refer to the three testbeds as *Amherst*, *Seattle* and *Sfo*.

Amherst is located in the college town of Amherst, MA. It consists of 20 public transit vehicles that are equipped with a computer, an 802.11b radio, a 3G data modem, and a GPS unit. The 3G modem has HSDPA-based service via AT&T. The vehicles visit many locations multiple times each day. This set up allows us to analyze the stationarity of WiFi or 3G availability with respect to location and time of day. We collected more than 3000 hours of measurement data from *Amherst* over 12 days. Cumulatively, over 500 GB of data was transfered.

The vehicles in *Amherst* cover a large geographical area totaling 150 square miles. A 1.5 sq mile area of the testbed is covered by an experimental mesh of APs. When vehicles are in the mesh environment, they connect to the mesh APs. In the remaining areas, the vehicles connect to open, reachable APs. In our data, over 70% of the connections are through non-mesh APs. Details on the distribution and density of APs and on contact durations are presented in Soroush et al. [21].

The software on the vehicles includes two main programs. The first program scans the WiFi and 3G channels simultaneously and obtains an IP address whenever a connection is available. Once a connection is established on any interface, the second program sends and receives data to a server. Both the server and the vehicle log the characteristics of the duplex data transfer on the WiFi and the 3G interfaces.

In the case of WiFi, we verify that the connection through an AP is usable before the second program starts to send data. The verification is performed by pinging a known server. If the AP is not usable, the vehicle attempts to associate with a different AP. More than 55% of the APs that vehicles in *Amherst* encountered are closed. Nevertheless, the vehicles were able to successfully exchange data with more than 100 unique WiFi APs each day.

Seattle is located in the metropolitan region of Seattle, WA. It consist of one vehicle that is equipped with the exact hardware and software as the vehicles in *Amherst*. Measurements in *Seattle* include large portions of highway driving and we present results for data collected over 6 days. From the single vehicle, about 5GB of data were sent and received over the course of the experiment.

Sfo is located in a metropolitan region of San Francisco, CA. We presents results from 3 days of data, gathered from one vehicle using the same setup as the oher two testbeds.

The vehicles in both *Seattle* and *Sfo* exchange data with open APs that we did not deploy. Unlike *Amherst*, in which the buses scheduled routes, *Seattle* and *Sfo* data were collected using unscheduled driving patterns that did not follow a regular path.

2.2 Availability

To measure availability, the vehicle and the server periodically send data to each other over UDP. Availability is measured over 1 second intervals. In each interval, an interface (WiFi or 3G) is considered available if at least one packet was received in the interval. Availability is defined as the number of available 1-second intervals divided by the total number of intervals.

Figure 1 shows that availability in each testbed. We see that in *Amherst*, 3G is available 90% of the time and WiFi is available 12% of the time.

Interestingly, the percentage of time neither 3G or WiFi is available is only 5%. The combination of WiFi and 3G reduces unavailability significantly because of a negative cor-

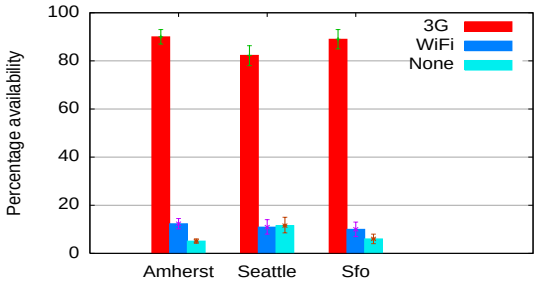


Figure 1: 3G and WiFi availability on the three testbeds.

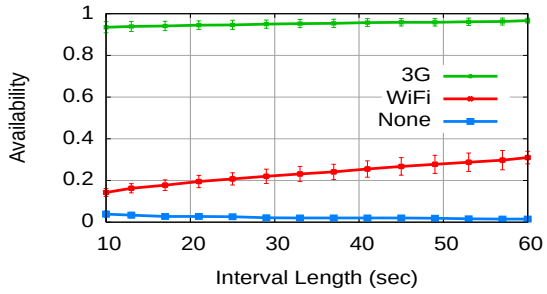


Figure 2: 3G and WiFi availability in *Amherst* at longer time intervals. The data are averaged all 8 days. Vertical bars show the 95% confidence interval.

relation between the availability of 3G and WiFi. Out of the 12%, 5% of the WiFi availability is when 3G is not available. If WiFi and 3G availability were completely independent, the overall unavailability even when both 3G and WiFi are combined would be $(1 - 0.90)(1 - 0.12) = 9\%$.

Figure 1 shows that the negative correlation between 3G and WiFi is not specific to *Amherst*, but can also be observed in *Seattle* and *Sfo*. In *Seattle*, 3G availability is only 82%, and the average WiFi availability is 10%. When 3G and WiFi are both considered, network unavailability is 11%. Again, if only the 3G interface is used, the unavailability would be roughly 16%. Similarly, in *Sfo*, 3G availability is 89% leading to an unavailability of 11%. But when combined with WiFi, the total unavailability reduces to 5%.

In summary, in all three testbeds, network unavailability is reduced by over 50% by combining WiFi and 3G compared to using 3G alone. We were not able to uncover the reason for the negative correlation observation.

2.2.1 *Availability over longer intervals*

For less-demanding applications, such as email or file transfer, intervals longer than 1 second are more appropriate for measuring availability. Figure 2 shows the availability of 3G and WiFi over longer time intervals, from 10 to 60 seconds. The results are based on *Amherst* measurements. 3G is available (i.e., at least one packet is received in an interval) close to 98% of the time with 60-second intervals. The availability of WiFi also increases with 60-second intervals, to 30%. We observed qualitatively similar effects in *Seattle* and *Sfo* (not shown in figure).

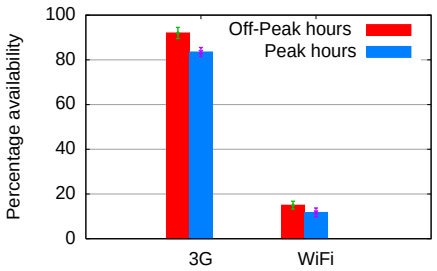


Figure 3: Comparing 3G and WiFi availability during peak and off-peak hours in *Amherst*.

2.2.2 *Availability in peak vs. off-peak hours*

We study availability of 3G and WiFi with respect to time of day, in particular, during *peak* and *off-peak* hours. We define peak hours to be from 8.00 am to 9.00 pm and off-peak hours to be from 9.00 pm to 8.00 am.

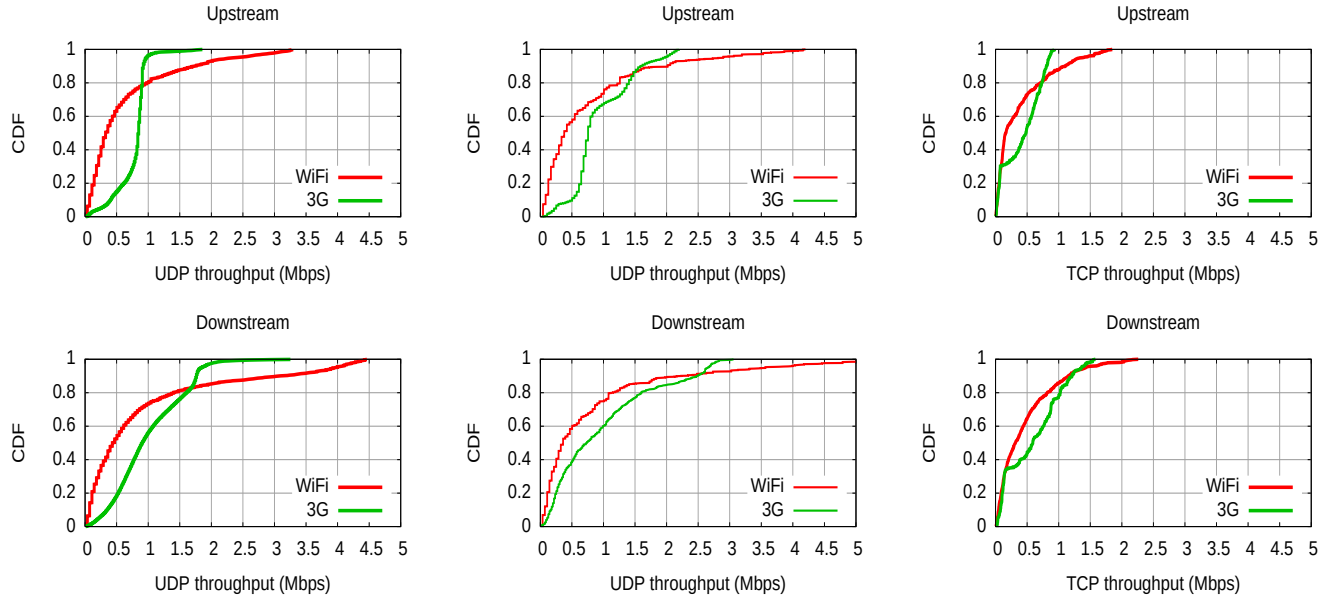
The vehicles in *Amherst* operate throughout the day but operate in fewer locations during off-peak hours. For a fair comparison, we only consider locations where the vehicle operates during both the peak and off-peak hours. We perform the experiment using days when the vehicles were operational during both peak and off-peak hours for at least 2 hours. The availability is computed as an average availability during 2-hour intervals.

Figure 3 shows the 3G and WiFi availability during peak and off-peak hours in *Amherst*. The results are computed over 4 days and the error bars show the 95% confidence interval. 3G availability during off-peak hours in *Amherst* is 9% more than the availability during peak hours. WiFi availability during peak hours is also lower, by 4%. We find that similar trends hold in *Seattle* as well. The difference in 3G availability between peak and off-peak hours is 6%, and the difference in WiFi availability is less than 3%.

The results indicate that the performance of 3G and WiFi networks suffers during periods of high spectrum use (i.e., during peak hours). But our experiments do not provide stronger evidence to show causality between availability and spectrum use.

2.3 Performance

We use three measures—UDP throughput, TCP throughput, and loss rate—to characterize the performance of 3G and WiFi networks. To measure the upstream and downstream UDP throughput, the vehicle and server each send 10 back-to-back 1500-byte packets every 20 ms. We measure UDP throughput in all three testbeds. To measure loss rate, the vehicle and server each send a 20-byte packet every 100 ms. To measure TCP throughput, the vehicle and the server each create a TCP connection and send 100KB data to each other repeatedly. At the end of a 100KB transfer, the TCP connection is closed and a new connection is created. We measure loss rate and TCP throughput only in *Amherst*. All performance results are based on at least 3 days of measurement data.



(a) Amherst *UDP throughput* (b) Seattle *UDP throughput* (c) Amherst *TCP throughput*

Figure 4: 3G and WiFi throughput measurements.

2.3.1 *UDP throughput*

Figures 4(a) and 4(b) show the 3G and WiFi UDP throughput in both directions. All CDFs are generated using measurements over 1-second intervals. They include points only for intervals with non-zero throughput (i.e., non-zero availability). The 3G lines thus have almost 10 times as many points as the WiFi lines.

In the upstream direction, 3G and WiFi achieve a median UDP throughput of 850 Kbps and 400 Kbps respectively in *Amherst*. In the downstream direction, the median 3G throughput is again about 2 times that of WiFi. For example, in *Amherst*, we observe a median 3G throughput of 1Mbps and a median WiFi throughput of 500Kbps. The median upstream and downstream UDP throughput is similar in *Seattle*. In both testbeds the median WiFi UDP throughput is about half of the median 3G throughput, but the top 20th percentile of WiFi outperforms 3G.

2.3.2 *TCP throughput*

Figure 4(c) shows the upstream and downstream TCP throughput of 3G and WiFi in *Amherst*. In the upstream direction, the median TCP throughput of 3G and WiFi are 500 Kbps and 200 Kbps, respectively. In the downstream direction, the median TCP throughput of 3G and WiFi are 600 Kbps and 280 Kbps, respectively. Thus, the median TCP throughput is only about half of the median UDP throughput for both the 3G and WiFi networks. However the relative TCP performance of 3G versus WiFi is similar to the relative UDP performance.

Taken together with the UDP measurements, the results above suggest that the throughput performance of WiFi in mobile outdoor environment is poorer than 3G. The result points to an important difference between stationary and mobile environments. In typical stationary settings, WiFi throughput is significantly higher than 3G throughput.

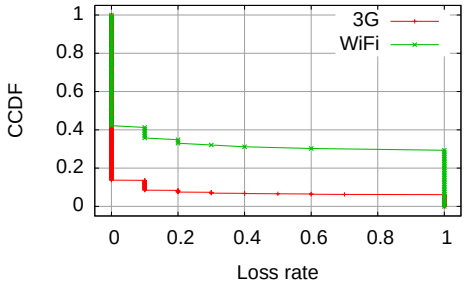


Figure 5: 3G and WiFi loss rate on *Amherst*.

2.3.3 *Loss rate*

Figure. 5 shows the loss rates over 1-second intervals for 3G and WiFi in *Amherst*. We see that 3G loses no packets in 93% of the intervals. WiFi has no packet loss in 78% of the intervals but loses all packets in 12% of the intervals. In other words, in 90% of the intervals WiFi delivers no packet or delivers all of them. This behavior is consistent with prior studies that have shown that WiFi losses are bursty in both indoor and vehicular settings [2, 22] and losses are bimodal. These bursty losses make WiFi offloading challenging, especially for applications with strict QoS requirements such as VoIP or video conferencing.

2.3.4 *Spatial distribution*

Using data collected in *Amherst*, we study the geographical distribution of 3G and WiFi performance. Our goal is

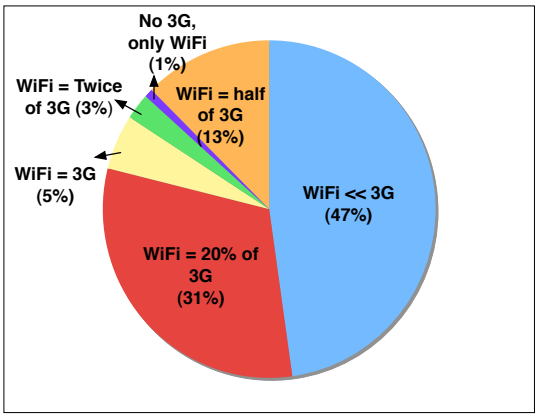


Figure 6: Comparing the total data sent over WiFi versus 3G in each grid location. The grid size is 0.5 miles × 0.5 miles. Results are averaged over all data collected in *Amherst*.

to characterize the locations where WiFi can augment 3G connectivity. We divide the geographical area into grids and compute the total data transfered over the 3G and WiFi per unit time spent in the grid, averaged over a day.

Figure 6 compares the performance of 3G and WiFi at different grid locations. In all, there were 120 grid locations in which packets were received on either WiFi or 3G at least once. In 47% of the grid locations, the total data sent on WiFi is insignificant compared to the data sent over 3G. In the remaining 53% of the grid locations, at least 20% of the 3G data could be shifted to WiFi. In 9% of the grid locations, equal or more data was sent over WiFi than 3G, i.e., all 3G traffic could be offloaded to WiFi.

2.4 Summary

In summary, the measurement study shows that

- The availability of WiFi is an order of magnitude poorer than 3G.
- The WiFi loss rate performance is also poorer compared to 3G. Therefore, leveraging WiFi to augment 3G may incur performance penalties. WiFi throughput is also much lower than 3G throughput.

Finally, our measurement results are consistent across three geographically diverse environments.

3. AUGMENTING 3G USING WIFI

This section presents the design of Wiffler. Its goal is to reduce 3G usage by leveraging WiFi connectivity when available, but to do so without affecting application performance.

The simplest policy for using WiFi is to send data on the WiFi network when available and switch to the 3G network when WiFi is unavailable. Results from our measurement study show, however, that this policy does not work well in practice because of two key challenges. First, the average availability of WiFi can be low—11% in our measurements—which severely limits the fraction of data that can be offloaded to WiFi. Second, WiFi loss rate is higher than 3G. For applications that are sensitive to losses, such as VoIP, using WiFi irrespective of its loss characteristics will degrade application quality.

Wiffler uses two ideas to address these two challenges: *leveraging delay tolerance* and *fast switching to 3G*. The key insight in the former is that delay tolerance allows applications to trade-off completion time for 3G usage. A user may be willing to tolerate a few seconds delay to send their email or complete a file transfer if it reduces 3G usage. Wiffler delays data transfers to reduce 3G usage, but only delays a transfer if the added delay results in 3G savings and is within the application's delay tolerance. Today, commodity phones such as iPhones provide an interface to specify application delay tolerance, but for energy benefits. For example, an iPhone user can set a delay tolerance threshold of 15 minutes, and new emails will begin downloading with a delay of up to 15 minutes.

For applications with strict quality of service requirements, such as VoIP and video, Wiffler uses the fast switching mechanism. When using WiFi, it promptly switches packets to 3G if WiFi cannot deliver them within a certain time period.

3.1 Wiffler API

Wiffler takes as input application data, which is characterized using S, the size of the transfer, D, the delay tolerance and an application-specified QoS metric. Based on these characteristics and those of the operating environment, it decides how to distribute the data across 3G and WiFi.

Our characterization of application data is flexible as a wide range of applications can be mapped to it. For example, a VoIP application might request (every 20 ms) for transferring a packet of 20 bytes with a delay tolerance threshold of 0 and a strict QoS requirement. A Web transfer might be 100 KB with a delay tolerance threshold of 20 seconds and no QoS requirement. We discuss in Section 3.5, how the characteristics of an application may be learned by Wiffler.

3.2 Leveraging delay tolerance

For applications that can tolerate small delays, the goal of Wiffler is to offloads as much data to the WiFi interface as possible. The simplest solution is to wait until the delay tolerance threshold to transfer data on WiFi when available, and to transfer the remaining data on 3G. However, this simple solution may significantly increase the completion time even when there are no 3G savings. For example, consider a scenario when there are no WiFi APs available until the delay tolerance threshold. The application will wait until the delay tolerance threshold for a WiFi offload opportunity even though delaying the transfer does not reduce 3G usage.

Wiffler estimates future WiFi throughput and delays transfers only if the estimate indicates that delaying transfer will reduce 3G usage. Wiffler's prediction method is described in Section 3.4. For now, assume that we have a predictor that yields a (possibly erroneous) estimate of WiFi capacity from the current time until a future time.

Wiffler uses the predictor to estimate offload capability of the WiFi network until the delay tolerance threshold.

D: earliest delay tolerance threshold among queued transfers
S: size in bytes to be transferred by D
W: estimated WiFi transfer size

if (WiFi is available):
- send data on WiFi and update S

if ($W < S \cdot c$ and 3G is available):
- send data on 3G and update S

Figure 7: Wiffler's prediction-based offloading.

The decision to either wait for a potential WiFi offload opportunity or to send immediately on 3G is made based on the predicted WiFi capacity and the application workload. For example, one possible strategy is to wait for WiFi only if all of the application data can be transfered over WiFi before the delay tolerance threshold. Since the estimate can be wrong, an alternative, more conservative strategy is to wait for WiFi only if the predictor estimates that twice the application data can be transfered over WiFi before the delay tolerance threshold. The completion time versus 3G savings trade-off for these two strategies is clearly different.

To capture this trade-off, we introduce a tuning parameter called the *conservative quotient*. The conservative quotient is a number between 0 and ∞. For a given value of c, the Wiffler offloading algorithm is shown in Figure 7. The algorithm considers the total data S that needs to be transfered within the earliest delay tolerance threshold, and the total data the node can transfer on WiFi, W. The next two steps are done in parallel. If WiFi is available, we use it immediately to transfer data. 3G connectivity is used only if we estimate that $W \leq S \cdot c$.

If $c < 1$, Wiffler will wait for WiFi offload opportunity even if only a fraction c of the total application data can be transfered on WiFi in expectation. Therefore, this strategy will offload more data on WiFi at the expense of completion time. On the other hand, if $c > 1$, Wiffler waits for WiFi only if the WiFi capacity is substantially more than the load. Therefore, the completion time of the strategy is likely to be lower, but it also has a lower offload potential. Unless stated otherwise, we set $c = 1$ in our experiments.

The conservative quotient can be set not only by the system or the application but also by the 3G provider. For example, during peak times when 3G spectrum pressure is high, the provider may decide to offload more data on WiFi at the expense of application latency and set c to a small value. But during the off-peak times, c can be set to a large value to improve application latency.

3.3 Fast switching to 3G

Applications such as video streaming and VoIP are sensitive to even small delays and losses. Because of a higher chance of loss, using WiFi to transfer such data can hurt application performance. Thus, if WiFi is losing or delaying packets, we should send them on 3G as soon as possible.

Wiffler uses low-level, link-layer information to enable fast switching to 3G in the face of poor WiFi conditions. Link layer information is needed because the WiFi NIC frequently takes a long time to complete retransmission attempts. For instance, the driver that we use in our testbed (*Madwifi*) retries packets 11 times, which even if we ignore medium access delays takes more than 120 milliseconds with the default 802.11b specification. This delay can affect performance of applications such as VoIP. Changing the default retransmission limit is not desirable either since other applications may actually desire retransmissions. Additionally, because of variable medium access delays, a low retransmission limit does not guarantee that WiFi would deliver the packet or declare failure in a timely manner.

Our fast switching mechanism is simple: it sends the packet on 3G if the WiFi link-layer fails to deliver the packet within a delay threshold. The motivation for this algorithm is that waiting for WiFi link-layer retransmissions incurs delays. In addition, when a packet is lost, there is a high chance that the retransmission will fail, since losses are bursty in the vehicular environment [2, 22]. Thus, it is better to send time-sensitive packets on 3G rather than waiting for likely more failures on WiFi. Choosing the delay threshold involves a trade-off between better application performance and 3G load. In Section 6, we analyze this trade-off in detail.

3.4 WiFi throughput prediction

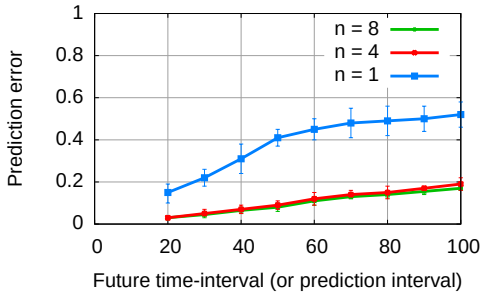
We predict WiFi offload capacity based on an estimate of the average throughput offered by an AP and a prediction of the number of APs that will be encountered until a given future time interval.

Our prediction of AP encounters is based on the observation that AP meetings occur in bursts. That is, if the mobile node meets APs frequently (e.g., because it is in a dense urban area with many APs), then the node is likely to meet the next AP within a short time interval. Similarly, if the mobile node has not met an AP for a long period of time (e.g., because it is on a highway), then the node is unlikely to meet an AP within a short time interval. An analysis of our measurement data shows that AP meetings in reality indeed have this property.

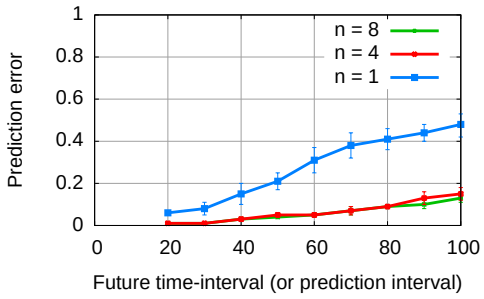
Based on this observation, we predict the number of AP encounters using a simple history-based predictor. The mobile node keeps track of the last n AP encounters and computes the average time between the encounters. Wiffler predicts the number of AP encounters until a future time interval using the average inter-meeting time of the past encounters. For example, if the average inter-meeting time of the past encounters is I seconds, then Wiffler predicts the number of AP encounters in a next T seconds to be $\frac{T}{I}$. Similarly, the average throughput is estimated based on the throughput observed by the vehicle at each AP encounter.

We study the accuracy of the AP encounter prediction using the traces we gathered from our testbeds. Figure 8 shows the AP prediction error for different values of n, the number of previous encounters used in the prediction. The prediction error is presented for different future time-intervals (or prediction intervals). We compare the predicted number of encounters with the actual number of encounters over 10-second time periods, and present the average.

Figure 8(a) shows for *Amherst* that if the prediction is based on only one previous AP encounter (n=1), the predic-



(a) Amherst



(b) Seattle

Figure 8: The relative average error between the number of APs predicted and the number of AP meetings observed in the measurement. Based on measurements collected from *Amherst* and *Seattle*. Vertical bars shows the 95% confidence interval around the mean.

tion accuracy is low. The prediction error is close to 20% even for predicting AP encounters until a small future time-interval of 20 seconds. On the other hand, when prediction is based on the previous 4 or 8 AP encounters, the prediction error is less than 5% up to a future prediction time-interval of 50 seconds. The prediction error increases to 20% for a prediction time-interval of 100 seconds. Figure 8(b) shows that AP prediction yields high accuracy in *Seattle* as well, even though the vehicle did not follow preset routes, unlike the vehicles in *Amherst*.

Since in both testbeds the accuracy of prediction based on 8 previous encounters is similar to the prediction based on 4, our experiments use n=4.

Our simple prediction framework allows us to estimate the WiFi offload capacity with no pre-programmed knowledge about the environment. More complicated prediction models that use additional information about the environment exist in the literature. For example, if the AP locations are available *a priori*, then the WiFi offload capacity can be predicted by predicting user mobility, instead of AP prediction [7, 16]. In Section 6, we show that the marginal improvement obtained by using location information is small.

3.5 Adopting Wiffler

In this section, we discuss a few issues relating to adopting Wiffler. First, Wiffler needs to know the delay tolerance threshold and the QoS requirement of each application that uses the network. One alternative is for the applications themselves to provide this information. We envision Wiffler to be implemented by exposing a simple API to applications. Alternatively, Wiffler can use application port information or binary names to infer delay tolerance. For example, packets corresponding to known email ports or "Outlook.exe" will be delayed for a pre-defined time.

Second, Wiffler requires proxy support, both to implement fast switching and the prediction-based offloading. A proxy will facilitate packet reception from multiple IP addresses (i.e., from the 3G and the WiFi interface) and allow switching between the interfaces. We note that proxy support is not needed for some applications. For instance, for HTTP, the client can use range requests to control when and how much data arrives on each interface using two separate connections.

Finally, while our measurements and experiments are based on WiFi APs that are open to all, Wiffler can be deployed and used with other APs as well. For instance, 3G operators such as AT&T and T-Mobile also operate a network of (closed) WiFi APs. By providing appropriate WiFi credentials to their 3G customers, these operators can encourage them to use Wiffler to reduce 3G usage when they are mobile.

4. Wiffler IMPLEMENTATION

We implemented Wiffler's prediction based offloading and fast switching on the same platform as our measurement study.

We implemented the protocol described in Figure 7 to make offloading decisions based on WiFi throughput prediction. Given pending data, at 1-second intervals, the vehicle runs the Wiffler offloading algorithm to determine if the data should be transferred over WiFi or over 3G. The vehicle and the server log delivery information. Offloading is implemented both for upstream and downstream traffic. The destination server also acts as a proxy to manage data coming from different IP addresses that the client acquires as it moves.

To implement Wiffler's fast switching, we added a signaling mechanism in the mobile node's driver that signals the application when the wireless card receives a link layer acknowledgement. The signal contains the identity of the acknowledged packet. The application matches the acknowledgement with its outstanding packets. If the application does not receive a link layer acknowledgement for a packet before a delay threshold, it sends the packet on the 3G interface. We set the delay threshold to 50 ms.

Implementing fast switching in the downstream direction is challenging. It needs either support from the APs or detailed information at the proxy on current WiFi conditions. Conveying current WiFi conditions from the mobile to the proxy can be time consuming. In this paper, we implement fast switching only in the upstream direction. Our trace-driven simulations study the benefit of fast switching in the downstream direction as well.

5. DEPLOYMENT RESULTS

We deployed Wiffler on the *Amherst* testbed. In this section, we present results from experiments that use this deployment. In the next section, we conduct a more detailed trace-driven evaluation.

	Completion time	% offloaded to WiFi
Wiffler **offloading**	45 sec	30%

Table 1: Deployment results of prediction-based offloading

	% time voice quality good	% offloaded to WiFi
Fast switching	68%	34%
WiFi when available	42%	40%

Table 2: Deployment results for VoIP using fast switching

5.1 Prediction-based offloading

This experiment uses a deployment of Wiffler on 20 nodes over a period of 2 days. Each node generates 5Mb of application data, uniformly at random, with the mean generation interval of 100 seconds. We set the delay tolerance threshold for data delivery to be 60 seconds. All data is destined to a known server that we control. The workload includes both upstream and downstream transfers. For downstream transfers, the vehicle sends the request to the server which then sends data to the vehicle.

Table 1 shows the results. For 5Mb transfers and a deadline of 60 seconds, Wiffler reduces 3G usage by 30%, even though the WiFi availability is only 12% (Section 2.2).

5.2 Fast switching

We evaluate fast switching in the context of VoIP. Although VoIP is a low-bandwidth application for which saving 3G usage may be less important, we chose it as an application representative of others such as video conferencing, real-time streaming, gaming, etc. Unlike the mean-opinion-score (MOS) for VoIP, there is no simple measure of video quality based on the loss and delay characteristics of the channel.

We assume that the VoIP application uses the popular G.729 codec and generates 20-byte packets every 20 ms. We calculate VoIP quality by using the standard MOS metric that ranges between 1 (unacceptable) and 5 (best). To evaluate VoIP performance in a quickly changing environment, we estimate the MOS value for 3-second intervals and quantify the overall quality as the fraction of intervals where the MOS value is more than 3.0 [2].

Table 2 shows the results using one vehicle in our deployment that operated in an area with high WiFi availability. Fast switching maintains good voice quality for over 68% of the time and reduces 3G usage by 34%. Instead, if we used WiFi whenever available, without switching to 3G during periods of bad WiFi quality, voice quality is maintained only 42% of the time, even though the 3G savings marginally increases from 34% to 40%.

6. TRACE-DRIVEN EVALUATION

We now present extensive trace-driven evaluation of Wiffler. We consider a range of different conditions as well as compare Wiffler to alternatives strategies for offloading data.

6.1 Evaluation Methodology: Prediction-based Offloading

To evaluate Wiffler's prediction-based offloading, we use the TCP throughput traces collected during our measurements. The traces provide information on how much data can be sent or received on 3G and WiFi at 1-second intervals. We show below that our trace-driven simulations yield results that similar to those in our deployment.

We characterize offloading performance using two metrics: *(i)* the fraction of data sent over WiFi, which measures the reduction in 3G usage; and *(ii)* the average completion time.

6.1.1 Alternative strategies

We compare Wiffler with several alternative strategies to offload data to WiFi. We consider three classes of alternative algorithms.

Algorithms without prediction: To understand the value of prediction, we evaluate two algorithms that do not use prediction. The *Impatient* algorithm uses a very simple policy: use 3G whenever WiFi is unavailable; else use WiFi. The *Patient* waits and sends data on WiFi until the delay tolerance threshold, and only switches to 3G if all of the data are not sent on WiFi before the delay tolerance threshold. *Patient* and *Impatient* present the two extreme points in the design space.

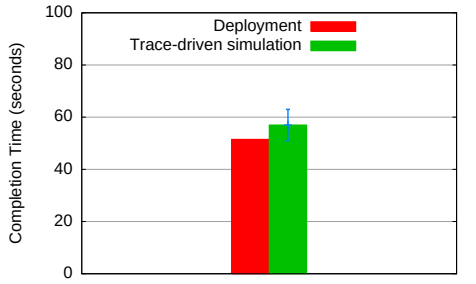
Algorithms with prediction: To understand the accuracy versus complexity trade-off, we compare Wiffler's simple prediction scheme against a more sophisticated prediction model that we call *Adapted-Breadcrumbs*. This model is similar to the Breadcrumbs system [16]. At each location grid, the system learns the available WiFi bandwidth and the probability of the client moving to an adjacent grid. It forecasts WiFi transfer sizes by taking the weighted average of expected transfers at each future grid. We use grid sizes of 0.2 miles $\times$ 0.2 miles and the learning phase uses the previous day of data.

Algorithm with future knowledge: To quantify the remaining room for improvement, we also consider an (impractical) algorithm with perfect future knowledge that we call *Oracle*. *Oracle* knows the exact amount of data that can be transfered using WiFi within the delay tolerance threshold, and uses this knowledge to make a decision about when to use the 3G network. It provides a lower bound on the lowest achievable completion time while minimizing 3G usage.

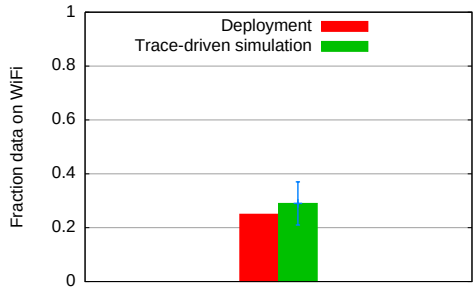
6.1.2 Workload

We conduct our experiments based on two workloads.

Realistic application workload: We obtained the workload from two corporate commuter buses that provide Internet access to the passengers. We sniffed the intra-bus WiFi network to capture packets that are sent and received by the riders. Based on the captured traces, we obtain distributions of connection sizes and inter-arrival times. We then generate realistic workloads based on the distributions. The average size of workload is 86 Kbps but it is highly bursty.

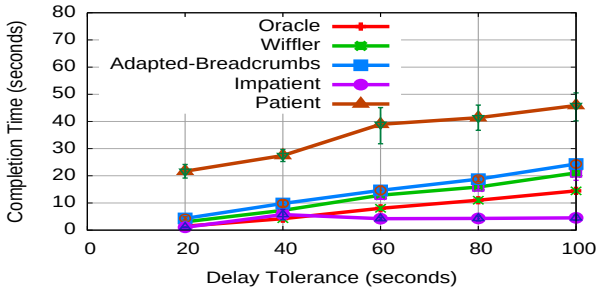


(a) *Completion time*

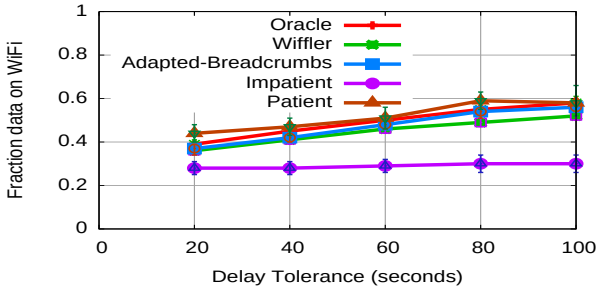


(b) *Fraction data offloaded to WiFi*

Figure 9: Comparing the deployment versus simulation results.



(a) *Completion time*



(b) *Fraction data offloaded to WiFi*

Figure 10: Offloading performance in *Amherst* with realistic application workload

More details on the workload are described in Mahajan et al. [14].

Synthetic workload: In order to experiment with a wider range of workload parameters, we generate a synthetic workload where a mobile node generates application data of size 5MB uniformly at random. The mean generation interval is set to 100 seconds. Similarly, a remote server generates transfers for each client at the same rate. Each experimental setting is run 10 times with different random seeds.

6.1.3 Validating trace-driven simulation

To validate the simulator, we collect throughput data during the deployment. During deployment periods when there are no application data to be sent or received, the vehicle transfers random data to the server both over WiFi and 3G and logs details of this transfer. As a result, the logs contain the throughput trace for the entire deployment duration. We conduct a trace-driven evaluation of Wiffler using this collected trace. We use the same packet generation parameters as the deployment. The simulation results are averaged over 10 runs with different seeds.

Figure 9 shows the performance of Wiffler observed in the deployment and in the simulator. Error bars show the 90% confidence interval. We see that the deployment results match well with the simulation results both in terms of completion time and percentage of data offloaded to WiFi.

6.2 Results: Prediction-based offloading

In this section, we evaluate the performance of Wiffler and the alternate offloading strategies. To simulate a range of different conditions, we vary three dimensions in our experiments.

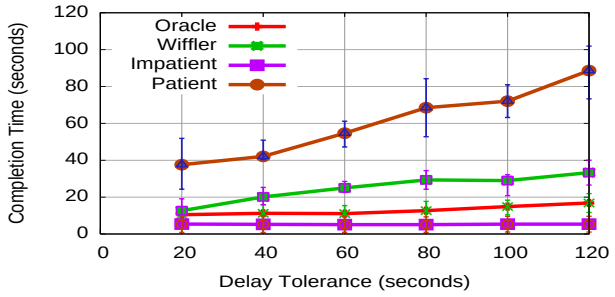
- *Workload:* We use realistic application workloads as well as synthetic workloads.
- *Location:* We use traces collected from *Seattle* and from *Amherst*. We also evaluate the performance of the protocols in areas with higher AP density.
- *Application conservativeness:* We use different conservative quotients and delay tolerance thresholds.

6.2.1 Realistic workload

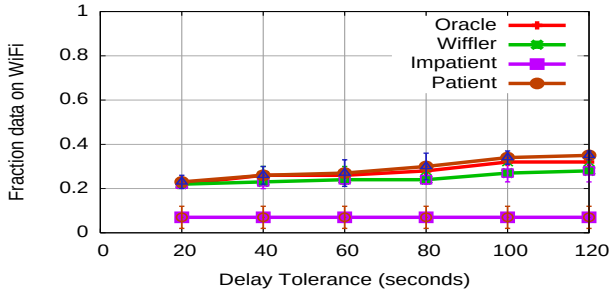
Figure 10 shows the performance of the different offload algorithms for varying delay tolerance threshold in *Amherst*. Wiffler offloads a significant fraction of data to WiFi. If users are willing to wait 60 seconds, they can reduce 3G usage by 45%. The offload fraction increases as delay tolerance increases.

The *Patient* protocol reduces 3G usage by the most, because *Patient* sends data on WiFi opportunistically until the delay tolerance threshold. As a result, Figure 10(a) shows that the completion time using *Patient* is significantly higher than all the other protocols. In terms of completion time, the *Impatient* protocol is the best performing since the protocol sends data on both 3G and WiFi and does not leverage delay tolerance. But as a result, *Impatient* reduces 3G usage by only 23% compared to the nearly 50% reduction achieved by other protocols, for a delay tolerance of 100 seconds.

Oracle, with complete future knowledge achieves the optimal balance between reducing 3G usage and decreasing completion time. Wiffler performs within 5% of both *Oracle* and *Patient* in terms of 3G savings, and is within 7 seconds of *Oracle* with respect to completion time. In contrast, the

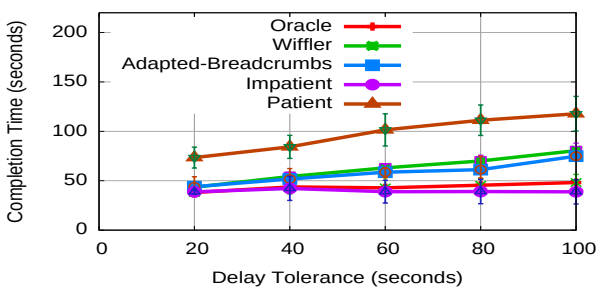


(a) *Completion time*

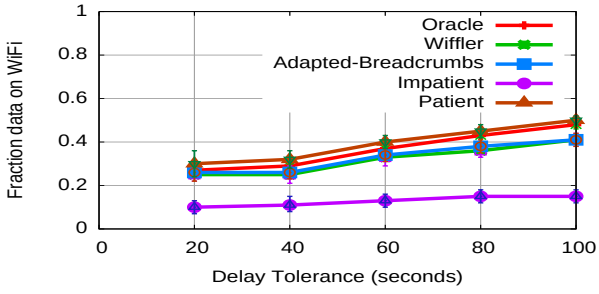


(b) *Fraction data offloaded to WiFi*

Figure 11: Offloading performance in *Seattle* with realistic application workload



(a) *Completion time*



(b) *Fraction data offloaded to WiFi*

Figure 12: Offloading performance in *Amherst* with synthetic workload

Patient scheme that uses no prediction has a completion time that is 25 seconds more than *Oracle* on average.

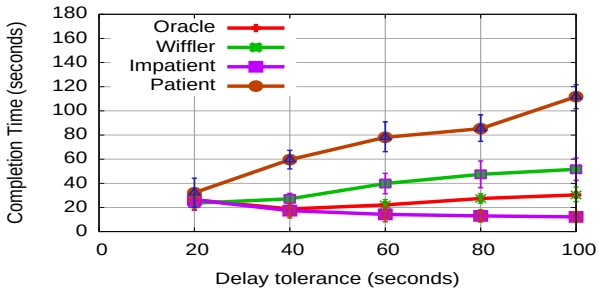
Figures 10 shows that *Adapted-Breadcrumbs* performs similar to Wiffler both in terms of completion time and 3G savings even though *Adapted-Breadcrumbs* uses a more sophisticated prediction algorithm that learns WiFi performance in each location.

Figure 11 shows the performance of the offload protocols in *Seattle*. Because we did not measure TCP throughput in *Seattle*, we use UDP throughput for this experiment. As in *Amherst*, Wiffler provides about the same amount of 3G savings as *Oracle*. The completion time of Wiffler is within 10 seconds of *Oracle*.

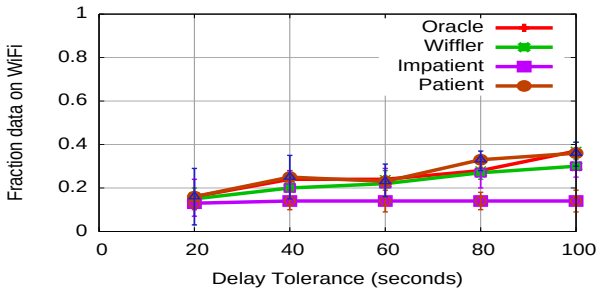
6.2.2 *Synthetic workload*

We repeated the experiments above for a synthetic workload of 5Mb file transfers, to understand the performance of the different protocols with larger transfer sizes. Figure 12 shows the performance results for *Amherst*. We observe that less than 22% of data is offloaded to WiFi for small delay tolerance threshold. But for a delay tolerance of 100 seconds, Wiffler offloads 40% of data over WiFi.

Not surprisingly, Figure 12(a) shows that the completion time for the synthetic workload is higher than the completion time for the realistic workload, because of the larger data sizes. The difference in completion time between Wiffler and *Oracle* is about 35 seconds compared to only 5 seconds with the realistic workload that had smaller data transfer sizes (Figure 10(a)). The completion time of *Patient* is nearly 75 seconds more than *Oracle*. As the data size increases, it is more likely that all of the data cannot be delivered using WiFi because of the lower throughput on WiFi and lower availability. As a result, in *Patient*, most transfers are



(a) *Completion time*



(b) *Fraction data offloaded to WiFi*

Figure 13: Offloading performance in *Seattle* with synthetic workload

completed only after the delay tolerance threshold, significantly inflating its completion time. Figure 13 shows that the results in *Seattle* are similar to those in *Amherst*.

6.2.3 *Impact of AP density*

3G cell towers are carefully placed to achieve near-complete coverage, but WiFi AP placement tends to be organic. In

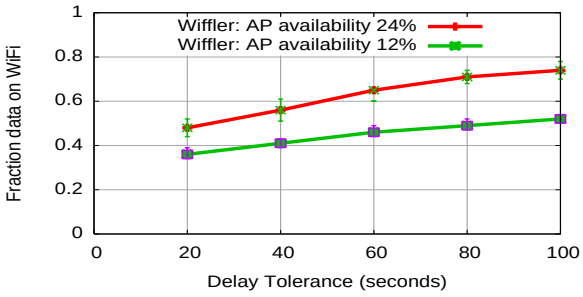


Figure 14: Comparing the fraction of data offloaded to WiFi under different AP availability conditions, in *Amherst* with realistic workload.

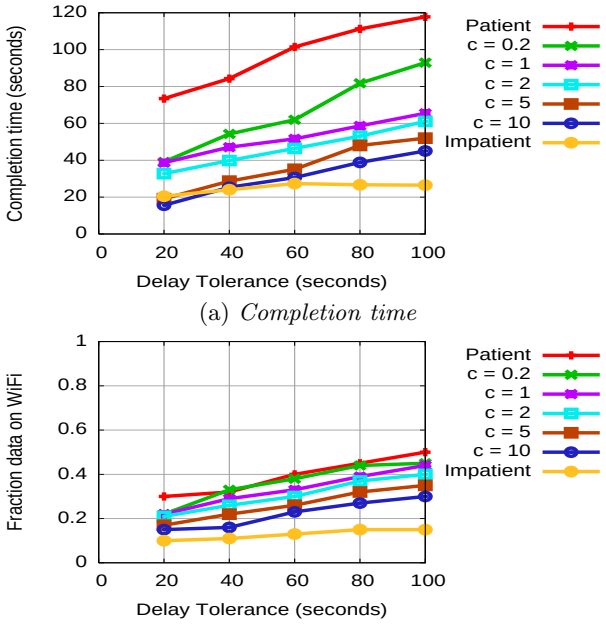
Amherst, certain areas have high AP density, but other areas have moderate to low AP density. As AP density is high typically in crowded downtown areas, where augmenting 3G capacity with WiFi is especially useful, we created a second data set. This *filtered* data set includes only measurements from a 15 sq. mile area with a high WiFi density. The availability of WiFi in this filtered data set is 24%, compared to 12% in the entire data.

Figures 14 show the performance of Wiffler in the total and the filtered data. In this experiment we used the realistic application workload. In areas with higher WiFi availability, 3G usage is reduced by 75% for a delay tolerance threshold of 100 seconds compared to a 50% reduction in 3G usage in regions with lower WiFi availability. The figure shows that even though the difference in WiFi availability in only 12%, the corresponding increasing in 3G savings is much higher. However, this is true only for large delay tolerance thresholds. For a lower threshold of 20 seconds, the difference in 3G savings between the two areas is only 9%.

6.2.4 *Impact of conservative quotient*

Wiffler uses prediction to trade-off completion time and 3G savings. As a result, the performance of Wiffler lies in between *Patient* and *Impatient*, the two extreme offloading strategies. In Section 3.2 we described an additional parameter called the *conservative quotient* c that allows Wiffler to achieve different trade-offs between completion time and 3G savings. Recall that Wiffler waits for WiFi only if the predicted WiFi capacity is c times the workload size.

Figures 15 shows the completion time and 3G savings for different values of c from 0.2 to 10. As the value of c increases, Wiffler starts sending data on the 3G interface much earlier instead of waiting for WiFi, which lowers completion time. On the other hand, the total data offloaded to WiFi when c=10 is significantly lower. When c=0.2, the total data offloaded to WiFi is 40% for a 100-second delay tolerance and the performance is close to the *Patient* protocol. On the other hand, the strategy has a poor completion time. The *conservative quotient* is thus an additional parameter that can be tuned to achieve different trade-offs. We find that c=1 offers a good trade-off between completion time and 3G savings.



(a) *Completion time*

(b) *Fraction data offloaded to WiFi*

Figure 15: Trade-off between application latency time and 3G usage, in *Amherst* with synthetic workload.

6.3 Evaluation Methodology: Fast switching

As with deployment experiments, we evaluate Wiffler's fast switching in the context of VoIP. We use two performance metrics: *(i)* fraction of time the mean opinion score (MOS) is good; and *(ii)* fraction of data offloaded to WiFi. As before, MOS values greater than 3.0 are considered good.

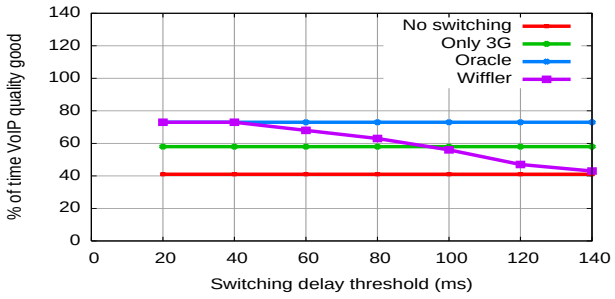
A goal of the evaluation is to understand the trade-off between VoIP quality and 3G savings for different values of switching delay threshold. Recall that Wiffler waits for a short period of time for a packet to be delivered over WiFi. If the packet is not delivered, Wiffler sends the packet over 3G. Clearly, a higher switching delay threshold increases 3G savings because there is a higher probability that the packet will be delivered using WiFi instead of 3G. However, the VoIP quality may be affected. Similarly, a lower switching delay threshold improves VoIP quality but may reduce 3G savings. We evaluate Wiffler for different values of this threshold.

6.3.1 *Alternative strategies*

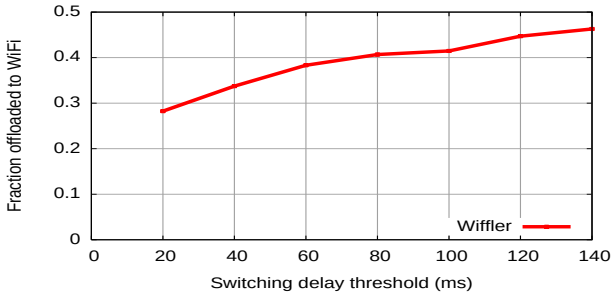
We compare the performance of fast switching in Wiffler to three other strategies. First, the *Only-3G* strategy supports VoIP using only 3G; the WiFi interface is never used. Second, the *No-switching* strategy does not switch away from WiFi as long as it is available. Finally, the *Oracle* strategy knows ahead of time if the packet will be lost on WiFi and, only in those cases, opts to send it on 3G. This strategy is impractical and serves as an upper bound on the performance.

6.3.2 *Workload*

For this trace-driven evaluation, we collected traces on *Amherst* by instrumenting one vehicle to send 20 byte packets



(a) *VoIP performance*



(b) *Fraction data offloaded to WiFi*

Figure 16: The performance of VoIP for varying switching time.

every 20 ms to a server over both WiFi and 3G. Unlike the implementation, packets are sent both in the upstream and downstream direction. We evaluate the fraction of time the voice quality is good in both directions. The traces are 1-hour long and from an area in *Amherst* with dense AP deployment.

6.4 Results: Fast switching

Figure 16(a) shows VoIP performance as a function of switching delay threshold. We see that for values below 60 ms, Wiffler is as good as the *Oracle*. It does not hurt VoIP quality if we can discover within that time that WiFi will lose or delay the packet. Of the four systems, *No-switching* performs the worst because of high loss rates. Wiffler improves voice quality to 73%, compared to 41% when using the *No-switching* system that switches to WiFi whenever available. Wiffler performs better than *Only-3G* because 3G frequently experiences high delays [13]. By dynamically deciding when to switch from WiFi to 3G, Wiffler provides both the low delays of WiFi and the high reliability of 3G.

Figure 16(b) shows that this advantage does come at the cost of a modest increase in 3G usage. Compared to *No-switching*, the increase in 3G usage is 10% if the switching delay threshold is 60 ms. Given the benefits of fast switching to application quality, we consider this increase to be a worthwhile trade-off.

7. RELATED WORK

Our work builds on previous research related to the use of multiple interfaces, predicting future connectivity, and characterizing connectivity from moving vehicles.

Using multiple interfaces. Many previous works propose mobile systems that leverage multiple interfaces. One thread of existing work optimizes for energy and exploits the differences in power consumed by different interfaces. For example, one method is to select the interface with low idle power consumption to wake up another interface [4, 20]. Zhong *et al.* [18] estimate the power consumption of different interfaces for various network activities. They use these estimates to switch between the different interfaces to save energy.

Other works use multiple interfaces to optimize performance. For example, vertical handoff techniques select the interface that currently offers the best performance [6]. Striping techniques multiplex data across different interfaces to balance load and improve performance [19].

In contrast to these works, our primary goal is not to optimize power consumption or performance. In our setting, a more expensive interface (3G) provides almost ubiquitous connectivity, but a cheaper interface (WiFi) is available intermittently. Our goal is to offload as many bytes on the second interface as possible, while satisfying an application-specific performance requirement. Thus, instead of responding purely to current conditions, we also base decisions on predictions of future conditions.

Predicting future connectivity. *Breadcrumbs* predicts future WiFi connectivity based on a model of the environment [16]. Similarly, Deshpande *et al.* [7] use WiFi prediction to improve mobile access. Both of these schemes rely on RF fingerprinting or an AP database for prediction. In addition, breadcrumbs uses client location information as part of its model, which means that clients must estimate the bandwidth available in different location grids and the transition probabilities between adjacent grids.

In contrast, our model does not require an external database or learning, and predicts based only on a short meeting history. In our evaluation, we compare the performance of Wiffler with a Breadcrumbs-like prediction model.

Characterizing vehicular connectivity. Several studies have characterized WiFi and 3G connectivity in isolation for vehicular settings. For WiFi, the CarTel study quantifies the frequency of AP encounters and the throughput that can be achieved using open APs [12]. Various researchers have since studied link layer characteristics [2, 15], TCP throughput [10], as well the performance of specific applications (e.g., web search [3]) and handoff policies [8]. Similarly, for 3G, several recent works have studied characteristics such as signal strength, loss rate, latency, and TCP throughput [13, 17].

In contrast, we conduct a joint characterization that enables a head-to-head comparison of 3G and WiFi. For any one technology, our results are qualitatively consistent with the studies above, but our joint characterization is crucial to understanding and leveraging their combined power. For instance, we uncovered a surprising finding that 3G and WiFi availability are negatively correlated, so WiFi can mitigate more issues for 3G than nominally expected.

8. CONCLUSIONS

Our work develops techniques to combine multiple interfaces with different costs and ubiquitousness, with the goal

of reducing the total cost of data transfer while meeting application requirements. We apply these techniques to the context of augmenting 3G with WiFi in mobile environments. 3G is costlier but more ubiquitous, and WiFi is cheaper but intermittently available. The design of Wiffler was informed by extensive measurements that jointly characterize 3G and WiFi across three different cities. The study found that WiFi was available only for roughly one-tenth of the time and had higher loss rates. We overcome these availability and performance challenges of WiFi by leveraging delay tolerance of applications and using a fast switching mechanism. Deployment experiments using 20 vehicles and trace-driven simulations showed the effectiveness of these techniques. For a realistic workload, Wiffler reduced 3G usage by almost half for a delay tolerance of 1 minute.

9. ACKNOWLEDGEMENTS

We thank Brian Levine for input during the initial discussions of this work, Brian Lynn for help with the measurement set up, Devesh Agrawal for help with implementation of fast switching, and Mira Dontcheva and Wilmot Li for help with collecting data in San Francisco. We also thank the MobiSys reviewers and our shepherd, Jakob Eriksson, for feedback on the paper. This work was supported in part by NSF award CNS-084585.

10. REFERENCES

[1] T-Mobile @ Home. http://support.t-mobile.com/doc/tm23449.xml.

[2] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. Levine, and J. Zahorjan. Interactive WiFi Connectivity For Moving Vehicles. In *Proc. ACM Sigcomm*, August 2008.

[3] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani. Web Search From a Bus. In *Proc. CHANTS workshop*, September 2007.

[4] N. Banerjee, M. D. Corner, and B. N. Levine. An Energy-Efficient Architecture for DTN Throwboxes. In *Proc. IEEE Infocom*, May 2007.

[5] T. Bishop and A. James. Microsoft Giving Workers Free Ride with its Own Bus Service. http://seattlepi.nwsource.com/business/330745_msfttranspo07.html, 2007.

[6] M. Buddhikot, G. Chandranmenon, S.J.Han, Y.W.Lee, and S. amd L.Salgarelli. Integration of 802.11 and Third Generation Wireless Data Networks. In *Proc. IEEE Infocom*, April 2003.

[7] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das. Predictive methods for improved vehicular wifi access. In *Proc. MobiSys '09*, June 2009.

[8] A. Giannoulis, M. Fiore, and E. W. Knightly. Supporting Vehicular Mobility in Urban Multi-hop Wireless Networks. In *Proc. MobiSys*, June 2008.

[9] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular Opportunistic Communication Under the Microscope. In *Proc. MobiSys*, June 2007.

[10] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular Opportunistic Communication Under the Microscope. In *Proc. MobiSys*, June 2007.

[11] M. Helft. Google Buses Help its Workers Beat the Rush. http://www.nytimes.com/2007/03/10/technology/10google.html, 2007.

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

[13] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3G Network: Interplay between the Wireless Channel and Applications. In *Proc. MobiCom*, September 2008.

[14] R. Mahajan, J. Padhye, S. Agarwal, and B. Zill. E PluriBus Unum: High Performance Connectivity On Buses. Technical Report 2008-147, Microsoft Research, 2008.

[15] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based Connectivity From Moving Vehicles. In *Proc. IMC*, October 2007.

[16] A. J. Nicholson and B. D. Noble. BreadCrumbs: Forecasting Mobile Connectivity. In *Proc. MobiCom*, September 2008.

[17] J. Ormont, J. Walker, S. Banerjee, A. Sridharan, M. Seshadri, and S. Machiraju. A City-wide Vehicular Infrastructure for Wide-area Wireless Experimentation. In *WiNTECH*, September 2008.

[18] A. Rahmati and L. Zhong. Context-for-wireless: Context-sensitive Energy-efficient Wireless Data Transfer. In *Proc. MobiSys*, June 2007.

[19] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MARS: A Commuter Router Infrastructure for the Mobile Internet. In *Proc. MobiSys*, June 2004.

[20] E. Shih, P. Bahl, and M. J. Sinclair. Wake on Wireless: An Event Driven Energy Saving Strategy for Battery Operated Devices. In *Proc. MobiCom*, September 2002.

[21] H. Soroush, N. Banerjee, A. Balasubramanian, M. D. Corner, B. N. Levine, and B. Lynn. DOME: A Diverse Outdoor Mobile Testbed. In *Proc. HotPlanet workshop*, June 2009.

[22] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The Beta-factor: Measuring Wireless Link Burstiness. In *Proc. SenSys*, October 2008.

[23] P. Svennson. AT&T: Tighter Control of Cell Data Usage Ahead. http://seattletimes.nwsource.com/html/businesstechnology/2010461891_apustecattdatausage.html, 2009.

[24] Economy + Internet Trends: Web 2.0 Summit. http://www.morganstanley.com/institutional/techresearch/pdfs/MS_Economy_Internet_Trends_102009_FINAL.pdf, 2009.

[25] J. Wortham. Customers Angered as iPhones Overload 3G. http://www.nytimes.com/2009/09/03/technology/companies/03att.html?_r=2&partner=MYWAY&ei=5065/, 2009.

[26] C. Ziegler. Sprint Falls in Line, Caps "Unlimited" Data at 5GB. http://www.engadgetmobile.com/2008/05/19/sprint-falls-in-line-caps-unlimited-data-at-5gb/, 2008.