Augmenting Mobile 3G Using WiFi: Measurement, System Design, and Implementation

Aruna Balasubramanian[‡], Ratul Mahajan[§], Arun Venkataramani[‡]
[‡]University of Massachusetts Amherst, [§]Microsoft Research

Abstract

We investigate if WiFi access can be used to augment 3G capacity. To understand the feasibility of 3G augmentation, we 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 testbeds is 87% and 11%, respectively. We also find that, unlike stationary environments, WiFi throughput is lower than 3G throughput in mobile environments, and WiFi loss rates are higher. We then design a system, called Wiffler, that uses two key ideas—*leveraging delay tolerance* and *fast switching*. For delay tolerant applications, Wiffler uses a simple model of the environment to predict WiFi connectivity, and delays applications to offload more data on WiFi. But Wiffler delays applications only if it results in 3G savings. 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. Both our implementation and trace-driven experiments show that Wiffler significantly increases 3G savings. For example, for a realistic workload, Wiffler reduces 3G usage by 45% for a delay tolerance of 60 seconds.

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 [30].

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 [28]. 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. 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 [2].

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 [6, 14]. 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 [12, 15]. 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. Interestingly, we find that their 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 than 3G. To our knowledge, our study is the first joint characterization of 3G and WiFi; prior works have measured the performance of one or the other [12, 17].

In summary, our study 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.* Our starting point is the observation 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 application latency for 3G usage. Instead of transmitting data immediately, it waits for WiFi to become available. But by using a simple method to predict future WiFi throughput, it delays data only if 3G savings are expected. 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 fast switching mechanism in Wiffler increases the time periods with good VoIP quality by 42%, 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. MEASUREMENT

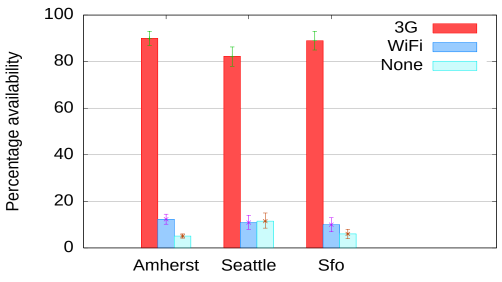
The goal of this work is to augment 3G networks using opportunistic WiFi. As a first step, we conduct a measurement study to jointly study the 3G and WiFi network characteristics. Specifically, we seek to answer the following questions: *(i)* What is the availability of 3G and WiFi networks as seen by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

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. In this paper, we refer to the three testbeds as—*Amherst*, *Seattle* and *Sfo*.

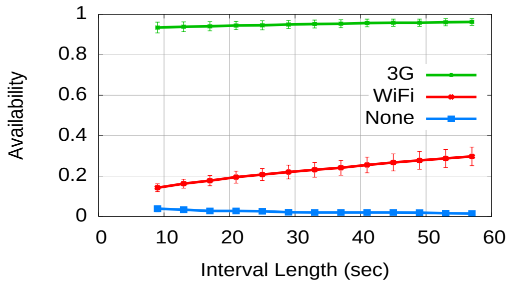
2.1 Testbeds and methodology

Amherst is located in Amherst, MA, a college town and 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 the same location 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. In all, over 500 GB of data was transfered over WiFi and 3G during the course of the experiment.

The vehicles in *Amherst* cover a large geographical area totaling 150 square miles. A 1.5 sq mile area of the testbed is dedicated to an experimental mesh that we deployed. When vehicles are in the mesh environment, they connect to mesh APs. In the remaining areas, the vehicles connect to open APs found on the road (that we did not deploy). More than 70% of the connections are through non-mesh APs. Details about the AP distribution, contact durations, and AP density are presented in [25].



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



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.

The software on the vehicles run 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 to an open AP and/or the 3G channel, the second program sends and receives data. Both our server and the vehicle's computer log the characteristics of the duplex data transfer on the WiFi and the 3G interfaces.

In the case of WiFi APs, we verify that the AP is open before the second program starts to send data. The verification is performed by pinging a known server. If the AP is closed, the vehicle attempts to associate with a different access point. Vehicles in *Amherst* encounter more than 55% closed APs during a day. Nevertheless, the vehicles were able to successfully exchange data with more than 100 unique open WiFi APs each day.

Seattle and *Sfo* are located in Seattle, Washington and San Francisco, California respectively. Both testbeds consist of one vehicle that is equipped with the exact hardware and software as the vehicles in *Amherst*. Measurements on *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 environment and we conduct a smaller scale measurement study in *Sfo*, and collect data for 3 days. In both *Seattle* and *Sfo*, vehicles exchange data with open APs that we did not deploy. Unlike buses in *Amherst* that follow a scheduled run, the *Seattle* and *Sfo* traces 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.

2.2.1 *Availability in the three testbeds*

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

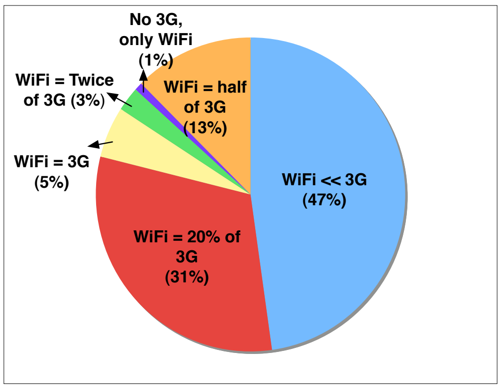
Figure. 1 shows that the negative correlation between 3G and WiFi is not specific to *Amherst*, but can be observed both in *Seattle* and *Sfo*. In *Seattle*, 3G availability is only 82%, and the average WiFi availability is 10%. When 3G and WiFi are both considered, network unavailability is 11%. Again, if only the 3G interface is used, the unavailability would be 18%. 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.

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 from moving vehicles over larger time intervals, from 5 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 to 30% with 60-second intervals. We observed qualitatively similar effects in *Seattle* and *Sfo* (not shown in Figure).

2.2.2 *Spatial distribution of 3G/WiFi availability*

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



3: 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*.

2.3 Performance

We measure three performance characteristics—UDP throughput, TCP throughput, and loss rate. 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 throughput in all three testbeds. To measure the upstream and downstream loss rates, the vehicle and server each send a 20-byte packet every 100 milliseconds. 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.

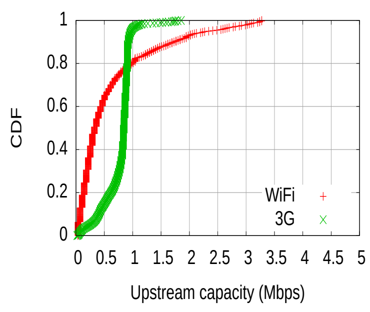
2.3.1 *Capacity*

Figure 4 shows the 3G and WiFi upstream and downstream UDP throughput. All CDFs are generated using measurements over 1-second intervals. They include points only for intervals with non-zero throughput, and so the 3G lines have almost 10 times as many points as the WiFi lines.

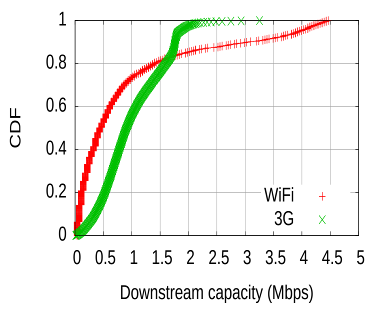
In the upstream direction, 3G and WiFi achieve a median UDP throughput of 850 Kbps and 400 Kbps respectively in *Amherst* (Figure. 4a). The median upstream UDP throughput is similar in *Seattle*. In both testbeds the median WiFi UDP throughput is about half of the median 3G throughput, but the top 20th percentile of WiFi outperforms 3G.

In the downstream direction, shown in Figures. 4(c) and 4(d), 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.

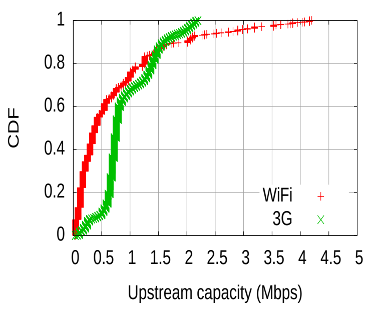
2.3.2 *TCP Throughput*



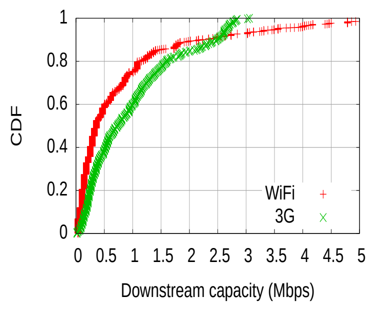
(a) Amherst*: Upstream UDP throughput*



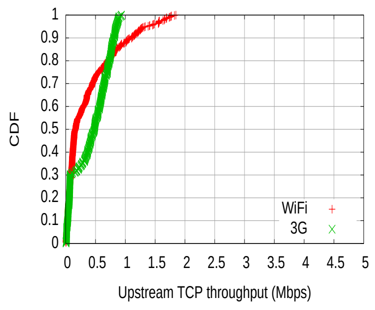
(b) Amherst*: Downstream UDP throughput*



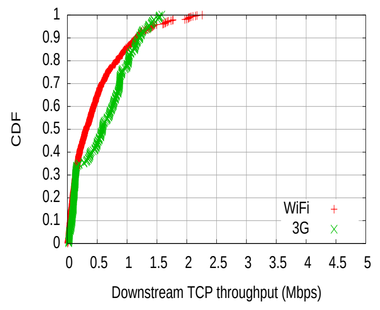
(c) Seattle*: Upstream UDP throughput*



(d) Seattle*: Downstream UDP throughput*



(e) Amherst*: TCP upstream throughput*



(f) Amherst*: TCP downstream throughput*

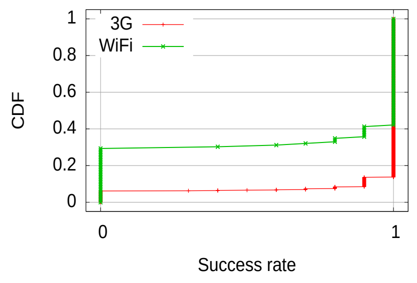
4: 3G and WiFi throughput measurements.

Figures 4(e) and 4(f) compare 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. We note that 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.

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 [3, 27] and losses are bi-modal. These bursty losses make WiFi offloading challenging, especially for applications with strict QoS requirements such as VoIP or video conferencing.



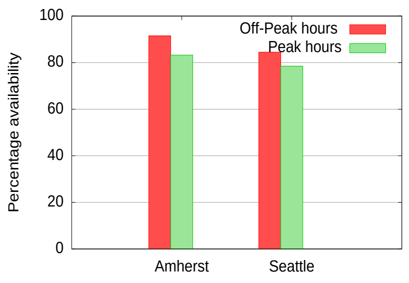
5: 3G and WiFi loss rate on *Amherst*.

2.4 3G availability during peak and off-peak hours

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

The vehicles in *Amherst* are operational during both peak and off-peak hours, but the vehicles are scheduled to operate in a smaller set of locations during off-peak hours. For a fair comparison, we only consider locations where the vehicle operates both during the peak hours and off-peak hours. We perform the experiment using days when the vehicles were operational both during peak and off-peak hours for at least 30 minutes. In *Amherst*, we present results averaged over 4 days and in *Seattle*, we present the results averaged over 2 days.

Figure 6 shows the 3G availability during peak and off-peak hours. 3G availability during off-peak hours in



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

Amherst is 9% more than the availability during peak hours. Figure 6 shows that the difference in availability between peak and off-peak hours extends to *Seattle* as well and the availability during off-peak hours is 6% more than during peak hours.

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

2.5 Summary

In summary, the measurement study shows that

- A non-trivial amount of WiFi is available, but the availability is an order of magnitude poorer than 3G.
- Unlike stationary environments, WiFi throughput is much lower than 3G throughput. The WiFi loss rate performance is also poorer compared to 3G. Therefore, leveraging WiFi to augment 3G may incur performance penalties.

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

3. Wiffler: AUGMENTING 3G USING WIFI

The goal of Wiffler is to reduce 3G usage by leveraging opportunistic WiFi, 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 that this simple policy does not work well in practice because of two key challenges. First, the average availability of WiFi in our measurement is only 11%, and therefore at most 11% of the data 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 leveraging delay tolerance 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 results in 3G savings. Wiffler leverages delay tolerance to reduce 3G usage, but only delays an application if the added delay results in 3G savings. 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, Wiffler uses the fast switching mechanism. Wiffler uses 3G whenever WiFi is unavailable; when using WiFi connectivity, Wiffler promptly switches packets over to 3G if WiFi fails to deliver it in 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.

3.2 Leveraging delay tolerance

For applications that can tolerate small delays, the goal of Wiffler is to offload as much data to the WiFi interface as possible. The simplest solution is to wait until the delay tolerance threshold to opportunistically 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 provide any 3G savings.

Wiffler uses WiFi throughput prediction to decide if data transfer should be delayed. Wiffler delays transfers only if the prediction indicates that delaying transfer will result in 3G savings. 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 capacity of the WiFi network until the delay tolerance threshold. 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.

```
$D$: 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 $S$
```

7: Wiffler's prediction-based offloading.

To capture this trade-off, we introduce a tuning parameter called the *conservative quotient.* The conservative quotient is a number between 0 and *Infinity* and for a given conservative quotient 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, 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 [3, 27]. 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 sending less data over the 3G network. In Section 6, we analyze this trade-off in more detail.

3.4 WiFi throughput prediction

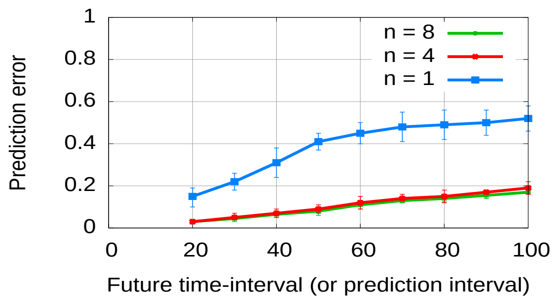
We predict WiFi offload capacity based on an estimate of the average throughput offered by an AP and a prediction of the number of APs that will be encountered until a given future time-interval.

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 hasn't 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.

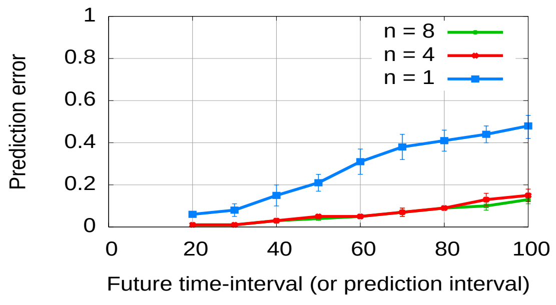
Based on this observation, we predict the number of AP encounters using a simple history-based predictor. The mobile node keeps track of the last n AP encounters and computes the average time between the encounters. Wiffler predicts the number of AP encounters until a future time-interval using the average 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 that if the prediction is based on only one previous AP encounter, (n = 1 in the figure), the prediction 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



(a) Amherst



(b) Seattle

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.

prediction yields high accuracy in the *Seattle* testbed as well, even though the vehicle did not follow a scheduled run, unlike the vehicles in *Amherst*. Since in both testbeds the accuracy of prediction based on 8 previous encounters (n = 8) is similar to the prediction based on 4 previous AP encounters (n = 4), in our experiments we predict based on 4 previous AP encounters.

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 [10, 20]. In Section 6, we show that the marginal improvement in performance obtained by using AP location information is small.

3.5 Adopting Wiffler

In this section, we comment on two issues relating to adopting Wiffler. First, to deploy Wiffler, each application needs to be associated with a delay tolerance threshold and a QoS requirement. One alternative is for the applications themselves to specify their delay tolerance threshold and their QoS requirement. We envision Wiffler to be implemented by exposing a simple API to applications. Alternatively, Wiffler can use application port information to infer delay tolerance. For example, packets from the well known "HTTP" port 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.

4. Wiffler IMPLEMENTATION

We implemented Wiffler's prediction based offloading and fast switching on the same platform that we use in our measurement study.

4.1 Prediction-based offloading

We implemented the protocol described in Figure 7 to make offloading decisions based on WiFi capacity prediction. Given pending data, at a 1-second interval, the vehicle runs the Wiffler offloading algorithm to determine if data is to be sent/received 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.

4.2 Fast switching

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 ID 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.

Unlike the prediction-based offloading, implementing fast switching in the downstream direction is challenging. It either needs support from APs and/or detailed information on current WiFi conditions at the proxy. Conveying current WiFi conditions from the mobile to the proxy can be time consuming. In this paper, we only implement fast switching in the upstream direction. In our trace-driven evaluation, we study the benefit of fast switching in the downstream direction as well.

5. DEPLOYMENT RESULTS

We deployed the Wiffler implementation in the *Amherst* testbed. In this section, we present results from the deployment experiments.

5.1 Prediction-based offloading results

This experiment uses a deployment of Wiffler on 20 nodes over a period of 2 days. Each node generated 5Mb of application data, uniformly at random, and the mean generation interval was set to 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 vehicle generates both upload and download requests. For downloads, the vehicle sends the request to the server which then transfers data to the vehicle.

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

1: Performance of Wiffler's prediction-based offloading in our deployment

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

2: Performance of VoIP using Wiffler's fast switching in our deployment

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 results

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 not a simple measure of video quality based on the loss and delay characteristics of the underlying 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 use the methodology we used in our previous work [3]. We estimate the MOS value for 3-second intervals. The overall quality is measured as the fraction of intervals where the MOS value was more than 3.0.

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

In this section, we present a trace-driven evaluation of Wiffler's prediction-based offloading and fast switching.

6.1 Evaluation Methodology: Offloading

To evaluate Wiffler's prediction-based offloading, we use the TCP throughput traces collected during our measurement. The traces provide information about data sent or received on 3G and WiFi at 1-second intervals.

We compare the performance of Wiffler with alternate offloading algorithms. We characterize the behavior of the offloading algorithms using two metrics: *(i)* the fraction of data sent over WiFi, or the reduction in 3G usage; *(ii)* the average completion time.

6.1.1 Alternate offloading strategies

We compare Wiffler against three other classes of 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 is not sent on WiFi before the delay tolerance threshold. *Patient* and *Impatient* present the two extreme points in the design space.

Algorithms based on forecasting: To understand the accuracy versus complexity trade-off, we compare Wiffler's simple prediction scheme against a more sophisticated prediction model, we call *Adapted-Breadcrumbs. Adapted-Breadcrumbs* is similar to the Breadcrumbs system [20]. 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.

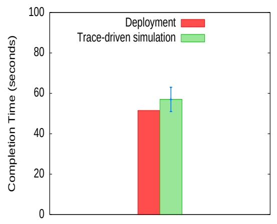
Algorithm with future knowledge: To quantify the remaining room for improvement, we also consider an (impractical) algorithm with perfect future knowledge that we call *Oracle*. *Oracle* knows the exact amount of data that can be transfered using WiFi within the delay tolerance threshold, and uses this 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

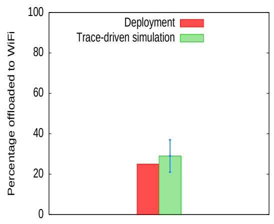
We conduct our experiments based on two workloads.

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. More details about the workload is described in [18].

Synthetic workload: In order to experiment 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 seeds.



(a) *Completion time*



(b) *Percentage data offloaded to WiFi*

9: Comparing the deployment versus simulation results.

6.2 Prediction-based offloading performance

First, we validate the trace-driven simulation using the deployment results. We then evaluate the performance of Wiffler and the alternate offloading protocols with respect to three dimensions

- *Varying workload:* Using realistic application workloads as well as synthetic workloads.
- *Varying location:* Using traces collected from both *Seattle* and *Amherst*. We also present separate experiments to evaluate performance of the protocols in areas with dense AP connectivity.
- *Varying application conservativeness:* Using different conservative quotients and delay tolerance thresholds.

6.2.1 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. The deployment results match well with the simulation results both in terms of completion time and percentage of data offloaded to WiFi.

6.2.2 Varying location: Realistic workload

***Amherst*:** Figures 10 shows the performance of the different offload algorithms for varying delay tolerance threshold. Wiffler offloads a significant fraction of data to WiFi; the offload fraction increases with longer delay tolerance. For example, if users are willing to wait 60 seconds, they can reduce 3G usage by 45%.

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. And as a result, *Impatient* only reduces 3G usage by 23% compared to nearly 50% 3G savings achieved by other protocols, for a delay tolerance of 100 seconds.

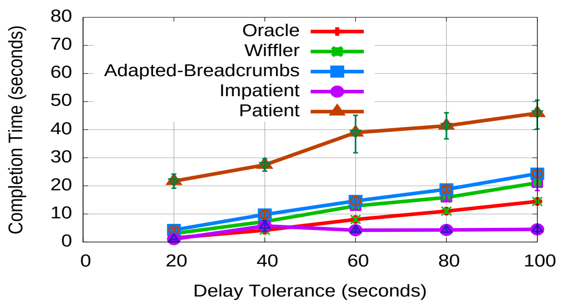
Oracle, with complete future knowledge achieves the optimal balance between reducing 3G usage and decreasing completion time. Wiffler performs within 5% of both *Oracle* and *Patient* in terms of 3G savings, and is within 7 seconds of *Oracle* with respect to completion time. In contrast, the *Patient* scheme that uses no prediction has a completion time that is on an average, 25 seconds more than *Oracle*.

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.

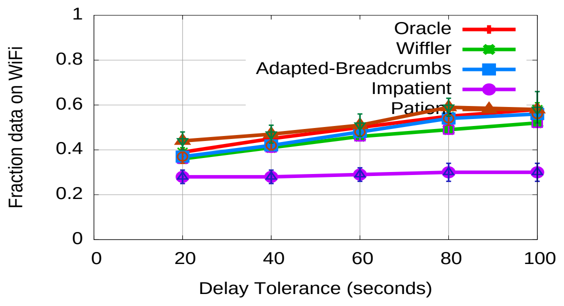
***Seattle*:** Figures 11 shows the performance of the different offload protocols for *Seattle* data for realistic workload. Since we did not collect TCP throughput traces in *Seattle*, we use the UDP traces for this experiment. Similar to the *Amherst* results, Wiffler provides about the same amount of 3G savings as *Oracle*. The completion time of Wiffler is within 10 seconds of *Oracle*.

6.2.3 Varying location: Synthetic workload

We repeated the above experiments, but for a synthetic workload of 5Mb file transfers. The goal of this experiment is to understand the performance of the different protocols when the transfer sizes are much larger. Figures 12 shows the performance results over *Amherst* data. For large file transfers, 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.

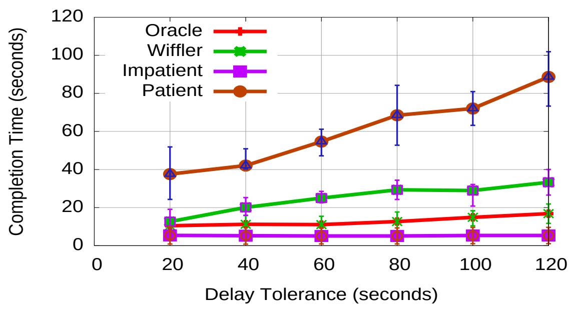


(a) Amherst*: Completion time*

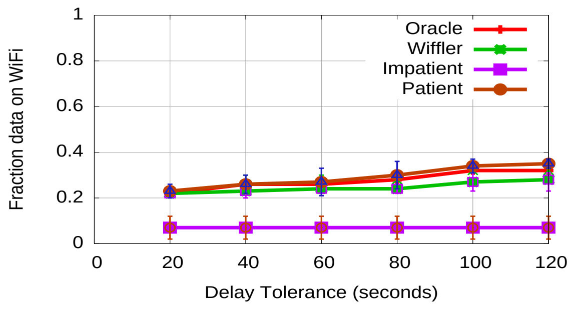


(b) Amherst*: Fraction data offloaded to WiFi*

10: *Amherst*: Realistic application workload

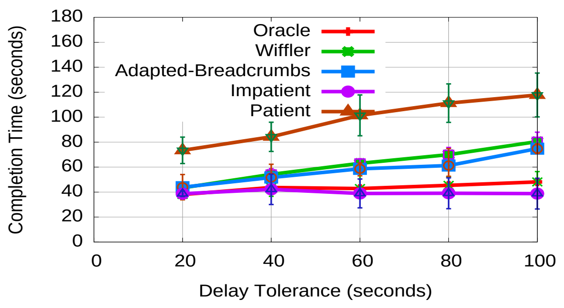


(a) Seattle*: Completion time*

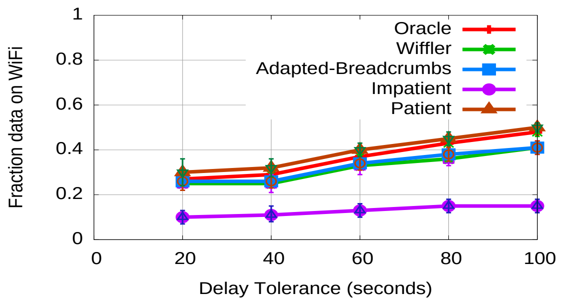


(b) Seattle*: Fraction data offloaded to WiFi*

11: *Seattle*: Realistic application workload



(a) Amherst*: Completion time*



(b) Amherst*: Fraction data offloaded to WiFi*

12: *Amherst*: Synthetic workload

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 for the realistic-workload experiments that had smaller data transfer sizes (Figures 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 completed only after the delay tolerance threshold, significantly inflating its completion time.

Figure 13 shows the performance of the different protocols using the synthetic workload over *Seattle* data. The results are qualitatively similar to the performance over *Amherst* data.

6.2.4 Varying AP density

3G cell towers are carefully placed to achieve near-complete coverage, but WiFi AP placement tends to be organic. In *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. The *filtered* data set includes only measurements from a 15 sq. mile area with a higher WiFi density. The availability of WiFi in this filtered data set is 24%, compared to 12% in the entire data.

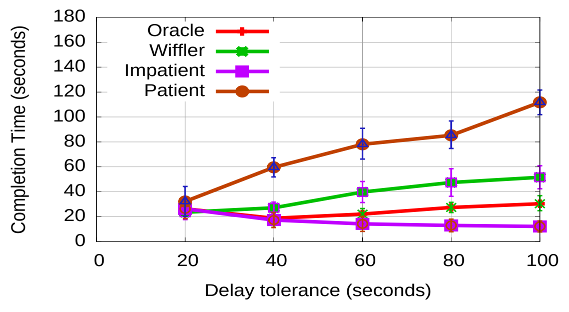
Figures 14 show the performance of Wiffler in the total and the filtered data. In this experiment we used the realistic application workload. In areas with higher WiFi availability, 3G usage is reduced by 75% for a delay tolerance threshold of 100 seconds compared to a 50% reduction in 3G usage in regions 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%.

6.2.5 Varying application conservativeness

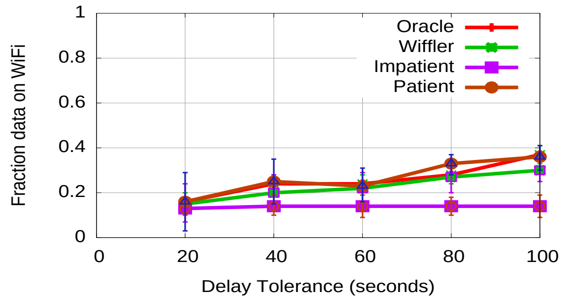
Wiffler uses prediction to trade-off completion time and 3G savings. As a result, the performance of Wiffler lies in between *Patient* and *Impatient*, the two extreme offloading strategies. In Section 3.2 we described an additional parameter called the *conservative quotient* that allows Wiffler to achieve different trade-offs between completion time and 3G savings.

Figures 15 shows the completion time and 3G savings for different values of the conservative quotient, starting from $c = 0.2$ to $c = 10$. Recall that, Wiffler waits for WiFi only if the predicted WiFi capacity is c times the workload size. As the value of c increases, Wiffler starts sending data on the 3G interface much earlier instead of waiting for WiFi, and as a result has lower 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 seconds delay tolerance and the performance is close to the *Patient* protocol. On the other hand, the strategy has poor performance in terms of completion time.

The *conservative quotient* is an additional parameter that can be tuned to achieve different trade-offs. We find that setting $c = 1$ offers a good trade-off between completion time and 3G savings.

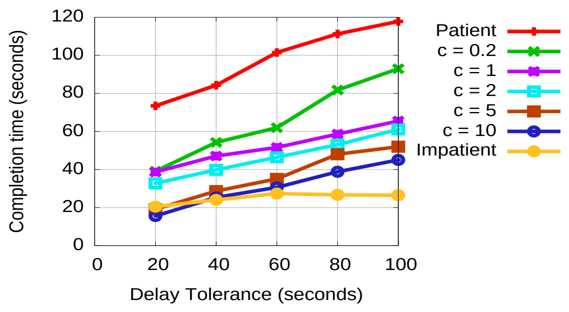


(a) Seattle: *Completion time*

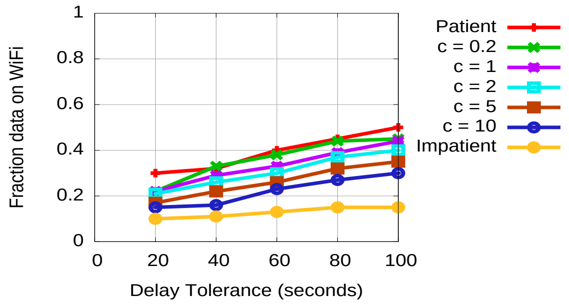


(b) Seattle: *Fraction data offloaded to WiFi*

13: *Seattle*: Synthetic workload

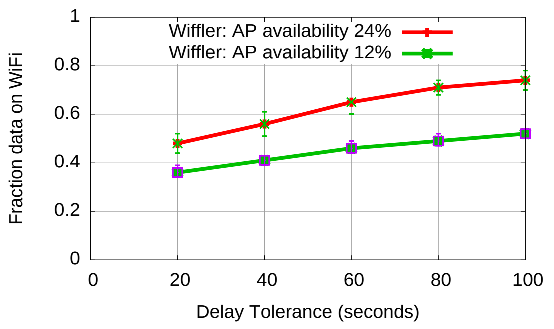


(a) *Completion time*



(b) *Fraction data offloaded to WiFi*

15: *Amherst*: Trade-offs between application latency time and 3G usage by varying the conservative quotient (Synthetic workload)



14: *Amherst*: Comparing the fraction of data offloaded to WiFi under different AP availability (realistic workload)

6.3 Evaluating fast switching

Similar to the deployment experiments, we evaluate Wiffler's fast switching in the context of VoIP. We evaluate based on two metrics — *(i)* Fraction of time the mean opinion score (MOS) is good. MOS represents the voice quality, and *(ii)* Fraction of data offloaded to WiFi or alternatively, the 3G savings. Similar to the deployment experiments we estimate the MOS value for 3-second intervals. The overall quality is measured as the fraction of intervals the MOS value was more than 3.0.

The goal of the evaluation is to understand the tradeoff between VoIP quality and 3G savings for different values of 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 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 delay threshold improves VoIP quality but may reduce 3G savings.

6.3.1 Workload

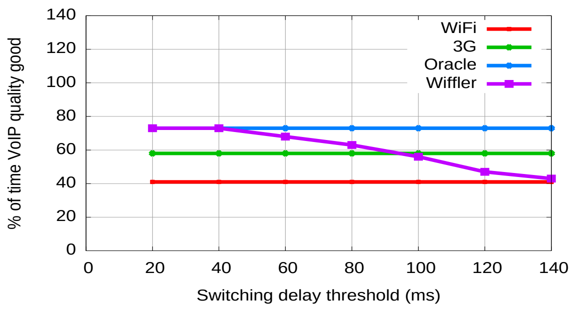
For this trace-driven evaluation, we collected traces on *Amherst* by instrumenting one vehicle to send 20 byte packets every 20 ms to a server over both WiFi and 3G. Unlike the implementation, packets are sent both in the upstream and downstream direction. We evaluate the fraction of time the voice quality is good in both directions. The traces are 1-hour long and the traces were collected from an area in *Amherst* with dense AP deployment.

6.3.2 Alternate strategies

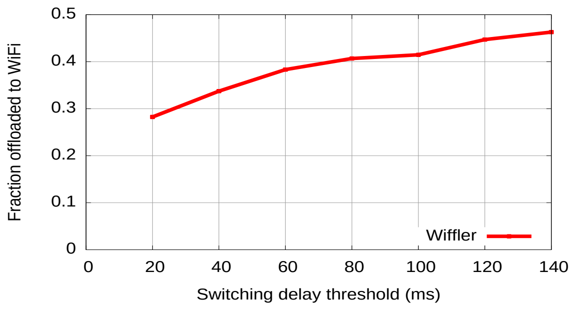
We compare the performance of Wiffler to four other system. The *Only 3G* system supports VoIP using only 3G. The *WiFi* system does not switch away from WiFi for as long as it remains available. *Oracle* knows ahead of time if the packet will be lost on WiFi and, only in those cases, opts to send it on 3G. We evaluate Wiffler for different delay thresholds.

6.3.3 Performance of fast switching

Figure 16 shows VoIP performance as a function of switching threshold. We see that for thresholds 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, using WiFi alone performs the worst because of high loss rates. Wiffler performs better than using only 3G



(a) *VoIP performance*



(b) *Fraction data offloaded to WiFi*

16: The performance of VoIP for varying switching time

because 3G frequently experiences high delays [17]. By dynamically deciding when to switch from WiFi to 3G, Wiffler combines the best of both worlds: low delays of WiFi and high reliability of 3G.

This advantage does come at the cost of a modest increase in 3G usage. We find that, compared to the WiFi-only system, the increase in 3G usage is 10% if the switching threshold is 60 ms and 20% if it is 20 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 [5, 24]. Zhong *et al.* [22] estimate the power consumption of different interfaces for various network activities. They use these estimates to switch between 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 [7]. Striping techniques multiplex data across different interfaces to balance load and improve performance [23].

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 a minimum 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 [20]. Similarly, Deshpande *et al.* [10] 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 the 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.

Characterizations of 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 the open APs [15]. Various researchers have since studied link layer characteristics [3, 19], TCP throughput [13], as well the performance of a specific application (e.g., web search [4]) and handoff policies [11]. Similarly, for 3G, several recent works have studied characteristics such as signal strength, loss rate, latency, and TCP throughput in vehicular [17, 21] as well as stationary [8, 9, 16, 29] settings.

In contrast, our measurement study enables a joint characterization and 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 understand and leverage 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 field measurements and system evaluation have demonstrated that systematically offloading data from 3G to WiFi can reduce demand on 3G networks without hurting application performance. Our measurement study jointly characterizes 3G and WiFi for two testbeds, finding that WiFi is available for a non-trivial amount of time, but has poorer quality in terms of loss rate and throughput. We present Wiffler, that uses WiFi to augment 3G using two key ideas: *leveraging delay tolerance* and *fast switching.* For applications that can tolerate a small delay, Wiffler waits for WiFi offload opportunities to reduce 3G usage. However, Wiffler waits for WiFi and delays applications only if it will result in 3G savings. For delay- and loss-sensitive applications, Wiffler switches proactively to 3G without incurring the high delay of WiFi when operating in a lossy environment. We implement and deploy Wiffler in our vehicular testbed. Both our implementation and trace-driven experiments show that Wiffler significantly reduces 3G usage. For example, for a realistic workload, Wiffler reduces 3G usage by 45% for a delay tolerance of 60 seconds.

9. REFERENCES

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

[2] Economy + internet trends: Web 2.0 summit. `http://www.morganstanley.com/institutional/techresearch/pdfs/MS_Economy_Internet_Trends_102009_FINAL.pdf`, 2009.

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

[4] A. Balasubramanian, Y. Zhou, W. B. Croft, B. N. Levine, and A. Venkataramani. Web Search From a Bus. In *Proc. ACM Workshop on Challenged Networks (CHANTS)*, pages 59–66, September 2007.

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

[6] T. Bishop and A. James. Microsoft giving workers free ride with its own bus service. `http://seattlepi.nwsource.com/business/330745_msfttranspo07.html`, 2007.

[7] 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*, 2003.

[8] R. Chakravorty, S. Banerjee, P. Rodriguez, J. Chesterfield, and I. Pratt. Performance optimizations for wireless wide-area networks: comparative study and experimental evaluation. In *Proc. MobiCom*, pages 159–173, 2004.

[9] M. Claypool, R. Kinicki, W. Lee, M. Li, and G. Ratner. Characterization by measurement of a cdma 1xevdo network. In *Wireless Internet Conference*, August 2006.

[10] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das. Predictive methods for improved vehicular wifi access. In *MobiSys '09: Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 263–276, New York, NY, USA, 2009. ACM.

[11] A. Giannoulis, M. Fiore, and E. W. Knightly. Supporting vehicular mobility in urban multi-hop wireless networks. In *Proc. MobiSys*, pages 54–66, 2008.

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

[13] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular Opportunistic Communication Under the Microscope. In *Proc. ACM Mobisys*, pages 206–219, June 2007.

[14] M. Helft. GoogleÕs buses help its workers beat the rush. `http://www.nytimes.com/2007/03/10/technology/10google.html`, 2007.

[15] 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. ACM SenSys*, pages 125–138, October 2006.

[16] Y. Lee. Measured tcp performance in cdma 1xev-do network. In *Passive Active Measurement Conference*, March 2006.

[17] 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*, pages 211–222, 2008.

[18] R. Mahajan, J. Padhye, S. Agarwal, and B. Zill. E pluribus unum: High performance connectivity on buses,. Technical Report MSR-TR-2008-147, Microsoft Research, 2008.

[19] R. Mahajan, J. Zahorjan, and B. Zill. Understanding WiFi-based Connectivity From Moving Vehicles. In *Proc. ACM IMC*, pages 321–326, October 2007.

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

[21] 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 '08: Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 3–10, 2008.

[22] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *Proc. MobiSys*, pages 165–178, 2007.

[23] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. MARS: A commuter router infrastructure for the mobile Internet. In *MobiSys*, June 2004.

[24] E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proc. MobiCom*, pages 160–171, 2002.

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

[26] Sprint capping unlimited 3G data service at 5GB. `http://gizmodo.com/391887/oh-no-sprint-capping-unlimited-3g-data-service-at-5gb`, 2008.

[27] K. Srinivasan, M. A. Kazandjieva, S. Agarwal, and P. Levis. The beta-factor: measuring wireless link burstiness. In *SenSys*, pages 29–42, 2008.

[28] P. Svennson. AT&T: Tighter control of cell data usage ahead. `http://seattletimes.nwsource.com/html/businesstechnology/2010461891_apustecattdatausage.html`, 2009.

[29] W. L. Tan, F. Lam, and W. C. Lau. An empirical study on the capacity and performance of 3g networks. *IEEE Transactions on Mobile Computing*, 7(6):737–750, 2008.

[30] 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.