Moving Bits from 3G to Metro-Scale WiFi for Vehicular Network Access: An Integrated Transport Layer Solution

Xiaoxiao Hou, Pralhad Deshpande, Samir R. Das
Computer Science Department, Stony Brook University, Stony Brook, NY 11794-4400, U.S.A.
Email: xhou, pralhad, samir@cs.stonybrook.edu

***Abstract*—We investigate a transport layer protocol design that integrates 3G and WiFi networks, specifically targeting vehicular mobility. The goal is to move load from the expensive 3G network to the less expensive WiFi network without hurting the user experience. As the test platform we choose a nationwide 3G network and a commercially operated metro-scale WiFi network. We exploit the often complementary characteristics of these networks for a hybrid design at the transport layer. To this end, we modify the stock Linux SCTP implementation to support 'striping' across multiple interfaces and the ability to handle frequent path failures and recovery in a seamless fashion. Instead of simply striping data over two network connections, we develop a utility and cost-based formulation that decides the right amount of load that can be put on the 3G network to maximize the user's benefit. We develop and experiment with a transport level scheduler to do this. We call the new SCTP design as oSCTP, meaning 'SCTP to be used for offloading.' We demonstrate the effectiveness of oSCTP and show that it is able to deliver superior network throughput and user experience, while significantly reducing the load on the 3G network.**

I. Introduction

Many metro-scale WiFi deployments [1] have succeeded in making WiFi ubiquitous providing very good coverage in urban spaces. This makes it viable to access WiFi outdoors even in a mobile scenario at vehicular speeds (see, e.g., [2], [3], [4]). However, prior studies used either open WiFi access points (AP) in a metro area or campus networks, where road coverage may not always be ubiquitous. Our recent experience with a head-to-head comparison study with metro-scale WiFi and 3G networks [5] shows that it is possible for WiFi to deliver competitive or superior performance when continuous coverage is available, even at vehicular speeds. In general, WiFi and 3G networks exhibit somewhat complementary characteristics [6], [5]. Thus, it is indeed possible to exploit WiFi – which is often free or low cost – to reduce the load on the expensive 3G networks.[1]

This idea certainly is not new, and has been promoted by both networking [6] and policy researchers [7]. Our goal in this paper is to design and evaluate a *hybrid access network protocol* that uses both 3G and WiFi for the best possible user experience while reducing load on the 3G link. There are quite a few technical challenges. We address many of them, *specifically focusing on the vehicular networking scenario* – the most challenging scenario for WiFi. The challenges are as follows: (i) The WiFi physical layer does not suit outdoor usage, particularly in long distances. (ii) Transmit power limitations limit coverage and/or bit rate. (iii) Coverage limitations give rise to frequent handoffs in a mobile scenario. (iv) Coverage 'holes' may not be uncommon. Thus, the best way to exploit WiFi would be to exploit 'diversity' – where WiFi is used opportunistically when available, while using 3G as a backup when WiFi is either not available or provides a poor bandwidth [6]. Our work takes this approach with a goal to provide an excellent user experience while saving bits on the 3G network.

Our work is inspired by the experience we reported in [5], where WiFi access with vehicular mobility is studied using a commercially operated 'metro-scale' WiFi network. This study is in contrast with prior studies that only considered open APs in the wild [3], [2] or a limited WiFi deployment [8], [4], [9], [6]. This study lays down the salient features of the two access networks to indicate how a hybrid access network should be designed. We summarize the results from this study in Section III. Based on this experience we design and implement the hybrid network access technique. The novelty here lies in exploiting network utility and cost functions (Sections III) for a seamless design. *The broad advantage of this approach is that it can support both networks seamlessly even with interactive traffic such as streaming, as opposed to just opportunistic use of WiFi [6] that can only support non-interactive traffic.*

We have spent a considerable effort in the implementation. The implementation is done at the transport layer using SCTP (Stream Control Transmission Protocol) [10]. We choose SCTP for its ability to support multihoming natively. However, multiplexing transmissions on multiple interfaces (striping) is not natively supported in SCTP. Thus, we augment SCTP to support our hybrid access protocol (Section IV) providing the first such testbed implementation and evaluation that is reported in literature. We call our augmented transport layer *oSCTP*, meaning 'SCTP to be used for offloading.' oSCTP can be best described as concurrent multipath SCTP with resilience to path failure. We evaluate the performance of *oSCTP* under vehicular mobility and demonstrate that the hybrid access provides significantly greater user experience while saving bits

[1] We use the term '3G' somewhat loosely in this paper – to mean current generation cellular data networks. It does not exclude the so-called '4G' networks, or even lower speed '2G' or '2.5G' networks.

on the 3G link (Section V).

II. Related Work

A. Vehicular WiFi Access

The potential of using intermittently available WiFi connectivity from moving vehicles for data transfers has been explored in several experimental studies. In one of the earliest such attempts, the Drive-thru Internet project [11], [12] has performed controlled experiments with a single car driving past a single access point to measure range and connectivity in an intermittent network. In a more recent work, TCP performance issues have been analyzed in a similar context [13].

Upload performance in the wild using open APs has been examined in the CarTel project [2]. The ViFi project [4] has explored link layer performance issues by exploiting macrodiversity (using multiple APs simultaneously), and opportunistic receptions by nearby APs. By the nature of the work, both CarTel and ViFi focus on upload.

The Cabernet project [3] has studied downloads and intermittent connectivity. They improve handoffs and also propose a new transport protocol. In general, improving handoff performance under vehicular mobility is an important area of research. In addition to the Cabernet project [3], Deshpande et al. [9], and Giannoulis et al. [8] have developed and evaluated optimized handoffs techniques. One or more of these strategies can nicely complement our work.

Intermittent connectivity causes problems for applications that require maintaining a session. This issue is addressed in some papers [3], [12] by creating a transport layer protocol that maintains sessions transparently to changing IP addresses. In our work, this issue does not arise as we use a metro-WiFi provider where the IP address does not change across handoffs.

B. Mobile Cellular Data Access

Since cellular data networks are closed systems, the scope of interesting studies is somewhat limited. The studies have been typically based on measurements on end systems. Some examples are as follows. In [14] mobile experiments are done in EVDO networks characterizing cross-layer aspects with TCP. Bandwidth predictability is evaluated for HSDPA networks in [15]. A measurement system for city-wide measurement of WiFi and EVDO networks was developed in [16].

C. Using Multiple Network Interfaces

'Striping' across multiple network interfaces (possibly using different technologies) is a popular method to aggregate bandwidth. This can be done in the link layer [17], network layer [18] and transport layer [19] depending on the available facility and application.

There have been efforts to use striping over multiple WWAN data links. See, for example, the MAR project in [20] and the PRISM project in [21].

The Stream Control Transmission Protocol (SCTP) naturally supports multihoming but does not support the striping functionality. Aggregation efforts over SCTP have been reported in the cmtSCTP (Concurrent Multipath Transfer SCTP) project [22], [23]. However, the studies there rely on ns2 simulations that gloss over realistic network behaviors. Our work is also based on the multihoming capability of SCTP. But we develop a prototype system that is specifically optimized for intermittent WiFi connectivity on one of the paths. Our implementation of SCTP also now includes a scheduler specifically for the utility model we develop.

A recent paper [6] has addressed the issue of augmenting mobile 3G using WiFi. The idea is to offload data on WiFi whenever possible hence avoiding using the 3G link when WiFi is available. In contrast, our work focuses on using both 3G and WiFi links concurrently, but using a utility model for scheduling packets on the 3G link. Several papers also provide head-to-head comparison of 3G and WiFi performances in the vehicular context showing the promise of the offloading approach [6], [5].

III. Augmenting 3G with WiFi

It is instructive to start the technical part of the paper by summarizing our recent measurement study in [5]. This study has performed a head-to-head comparison of a 'metro-scale' WiFi network and a 3G network under vehicular mobility. The metro-scale WiFi network is Cablevision's (a regional ISP in the Long Island area of New York) 'Optimum WiFi' [24] providing more than 10,000 access points in the region. The chosen 3G network is Verizon's (a nation-wide cellular provider) EVDO Rev. A network. The experimental component of the current paper uses the same network. The relevant observations from the measurement study in [5] are as follows.

- Instantaneous throughputs on WiFi can be zero occasionally because of lack of coverage on roads (roughly one-third of the times). But, roughly one-third of the times it can deliver significantly high throughput (over 2.5 Mbps). The goal is to exploit these high throughput regions effectively to move load away from 3G.
- There is a good temporal correlation for the instantaneous throughputs on both networks, at least for short lags. This implies a good predictive ability. This will be exploited in our design.
- Correlation between WiFi and 3G throughputs is poor signifying that diversity techniques could be successful.

It is quite clear that compared to using 3G alone, a *hybrid access network* using both 3G and WiFi has several potentials. First, it can improve long term average throughput (by approximately a factor of 2 in our experience). Second, with a careful design, it can reduce the load on the 3G network by simply choosing to transmit over WiFi as much as possible. The hybrid access should not use a straightforward striping over multiple interfaces, as the cost models for these networks could be very different. On the other hand, avoiding 3G network to the extreme (e.g., using WiFi opportunistically by postponing data transfer until WiFi is available [6]) is not appropriate either. This approach cannot support interactive use very well, as this provides zero throughput when WiFi is not available. We focus on a *flexible hybrid design that uses both interfaces*

intelligently and can – mostly using WiFi except augmenting it with 3G when appropriate and only to the extent appropriate.

But what is really appropriate? The answer to this question entirely depends on the application. In order to provide a general design framework we take the help of network utility function [25] that models an application's utility of the available network throughput. We also model cost of accessing the network as a function of network throughput. Typically, a tradeoff exists between utility and cost. Higher throughputs may not provide any tangible benefit if the additional utility gained is small, but the cost is considerable.

A download application (elastic traffic) may completely avoid the 3G link and can still provide acceptable user-perceived performance. On the other hand, media streaming with QoS needs will freeze when WiFi is not available or provides very low bandwidth. This could happen even when playout buffers are used, though buffers certainly soften the impact.

Also, the cost model for the links are important input to the design decision. When the cost of the 3G network is very low, striping across both networks may be appropriate for most applications. When it is very high, 3G must be avoided except when really necessary.

This approach allows us to develop a general-purpose technique that is not tied to specific applications or cost models. The existing research (see, e.g., [26], [27]) does develop similar models in the context of multiple interfaces. However, they are not directly useful in our context because they either require perfect future knowledge [26] or minimizes cost while provisioning only minimal performance guarantees (simply stabilizing network queues) [27].

A. Modeling User Benefit

Utility of a flow $U(x)$ is expressed as a function of throughput x [25]. $U(x)$ describes how much the user values throughput. For elastic flows, such as TCP downloads – one of the most prevalent form of traffic on the current generation Internet, the utility function has a diminishing marginal rate of increase with increasing throughput. This presents a strictly concave function. Modeling $U(x)$ as a logarithmic function has been common in networking literature. We will also take this approach here. *However, our general approach does not depend on the exact nature of the utility function so long as it can be specified by the user for the specific service to be used.* Designing utility functions for specific services or applications is beyond the scope of this paper.

The network service using different types of link defines a cost function $C(x)$ that also depends on the throughput x on that link. The difference of $U(x)$ and $C(x)$ models the user's benefit that we want to maximize.

For our specific problem, assume that x_w and x_g are instantaneous throughputs on the WiFi and the cellular 3G link respectively. Assume that these throughputs are controllable by a throttle; however, certain maximum bounds exist, i.e., $x_w \leq X_w$ and $x_g \leq X_g$. Assume that the costs on the two networks are denoted by $C_w(x)$ and $C_g(x)$, respectively. Then,

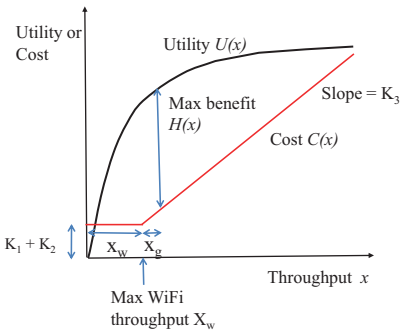


Fig. 1. Example plot of utility, cost and benefit functions, demonstrating the benefit maximization approach.

we want to maximize

$$H(x_w, x_g) = U(x_w + x_g) - C_w(x_w) - C_g(x_g).$$

See Figure 1. As an example, assume $C_w(x)$ to be a constant, i.e.,

$$C_w(x) = K_1,$$

and $C_g(x)$ to be a constant plus a linear function of x, i.e.,

$$C_g(x) = K_2 + K_3 x.$$

Assume a logarithmic utility function suitable for elastic traffic:

$$U(x) = K_4 \log(x).$$

With a simple exercise of multivariate optimization it can be shown that $H(x_t)$ is maximized when

$$\begin{aligned} x_w &= X_w \\ x_g &= \begin{cases} min(X_g, \frac{1}{K_3} - X_w), & if X_w < \frac{1}{K_3} \\ 0, & otherwise \end{cases} \end{aligned} \quad (1)$$

Note that X_w and X_g may not be constant as they may be dependent on wireless link qualities or available bandwidth. In that case, the above relationship must hold at all points of time. If the cost and utility functions are as above, then the optimal strategy is as follows (see Figure 1).

Strategy 1

1) Use the maximum possible throughput on WiFi ($x_w = X_w$) at all points of time.
2) If the throughput on WiFi is not sufficient (i.e., $X_w < \frac{1}{K_3}$), use 3G only to fill in the slack (i.e., $x_g = \frac{1}{K_3} - X_w$), but no more. Thus, the 3G link is to be throttled unless the maximum throughput on 3G is already less than $\frac{1}{K_3} - X_w$.

The 'combined throughput target' is related to the maximum possible WiFi throughput (X_w) and the slope of the 3G cost function (K_3). *Our goal is to develop a striping mechanism along with a scheduler that implements the above strategy once per scheduling interval.* The design of the scheduler will be discussed in the next section.

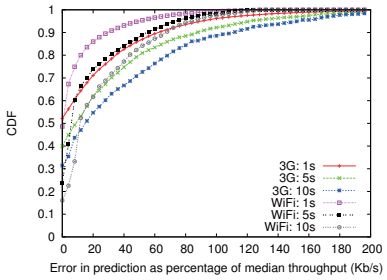


Fig. 2. CDF of errors in throughput prediction (as a percentage of median throughput on the respective link) for different prediction intervals for WiFi and 3G.

While we have used specific formulations, the above technique allows for modeling any type of utility and cost functions. For example, the utility function can use a simple thresholding or the cost function could be tiered to reflect more realistic pricing plans of the 3G networks. Also, note that we are only concerned here about a single flow's utility from the user's point of view. The technique can be extended when multiple concurrent flows are present.

B. Scheduling Interval and Throughput Prediction

We need to use throughput prediction for implementing Strategy 1. This is because in a scheduling interval i, the 3G link may need to be throttled depending on the WiFi throughput. However, WiFi throughput in scheduling interval i is not known in advance and thus must be predicted. The autocorrelation analysis in [5] has shown excellent temporal correlation for instantaneous throughput values over short time lags for both 3G and WiFi. We build upon this analysis to estimate how well recent throughputs can predict the throughputs to be seen in the near future. Our prediction model of choice is the autoregressive (AR) model. The analysis in [5] has shown that prediction based on even the single most recent interval can be very accurate and performs no worse than predicting over multiple past intervals.

However, the duration of the scheduling interval matters. In Figure 2 we compare the prediction errors for different scheduling intervals for both networks. The throughput data collected in [5] are used. Note that the prediction is poorer with larger intervals. As expected [5], the 3G throughput can be better predicted than WiFi.

Looking at Figure 2, errors appear reasonable for WiFi for 1-5 sec intervals, with median error being within about 5% of the median throughput. For 10 sec. interval, the error is higher, with median roughly coming within 12%. In the reported experiments that will follow, we keep the scheduling interval at 1 sec to minimize prediction errors. We do note that use of historical location-tagged throughput data (similar in spirit as in [9]) can improve predictions significantly and allow use of longer scheduling intervals.

IV. Implementation Using SCTP

The best way to implement the technique outlined above is to perform striping on the WiFi and 3G interfaces, and then throttle the 3G link as appropriate. We have adopted a transport layer solution based on SCTP. Adopting a transport layer solution makes the technique independent of applications. Moreover SCTP's socket interface is quite similar to TCP. Current TCP based applications can be easily ported to SCTP. Also, there are utilities like *withsctp* which translate TCP library calls to SCTP allowing TCP binaries to run over SCTP directly.

A. Extending SCTP

SCTP provides reliable transmission and flow and congestion control just like TCP. Additionally it supports multihoming and multiple paths natively. One SCTP association (analogous to a TCP connection) can bind multiple IP addresses at each endpoint. However, SCTP's multihoming does not provide any striping functionality. SCTP chooses a 'primary' path for communication by default, switching over to others only when the primary path fails. Obviously 'stock' SCTP does not satisfy our requirement. We modified Linux Kernel SCTP (lksctp) [28] to perform striping over multiple interfaces for bandwidth aggregation. Further, we have enabled throttling of throughputs on individual paths based on the utility model described above. We call this version of SCTP that provides striping and throughput throttling capability *oSCTP*.

Note that currently an SCTP path is defined by a destination IP address instead of a source-destination IP address pair. For our scenario in Figure 3, the server has two paths to the client whereas the mobile client has only one path to the server. Therefore, our striping and scheduling techniques only focus on the streams from server to client, i.e., download streams. We expect that downloading will be the dominant behavior for in-car applications. However, there is nothing fundamental in our technique depending on downloads. It can be extended to uploads as well. Further, we consider a noise-limited rather than an interference-limited network. This means that when link performance is poor, this is only due to noise rather than interference from competing transmissions. Thus, we do not consider scheduling across multiple vehicular clients.

Several technical issues crop up when providing striping functionality to SCTP. Only scheduling packets on two paths alternatively without other changes results in very poor performance. Existing literature has addressed some of these issues (discussed in Section II-C) but only in a disjointed fashion. It is unclear how these techniques will work when combined together. Moveover, their performance has most commonly been evaluated on simulators like ns2. Striping performance using SCTP on a testbed in an open multihomed environment using heterogenous technologies is largely unknown.

In the remainder of this section we describe our two testbeds on which we evaluate *oSCTP*; describe how we handle issues related to congestion control when *oSCTP* uses multiple paths concurrently; devise an algorithm that schedules packets across multiple links; expose problems with SCTP's path recovery procedure and propose techniques for quick path recovery. Finally we describe our implementation of **Strategy 1** from Section III-A to maximize users benefit while using *oSCTP*.

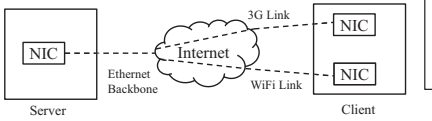


Fig. 3. System model.

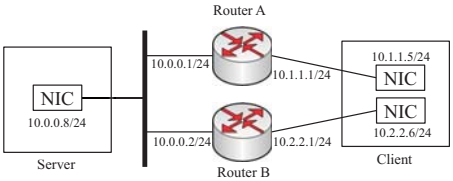


Fig. 4. Emulation testbed used for analyzing SCTP performance. 'Linux Advanced Routing & Traffic Control' and `netem` are used on the routers to emulate different transport level data rates, delays and loss rates.

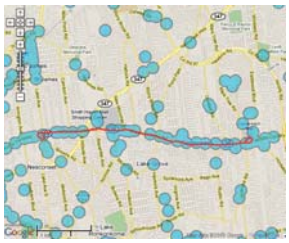


Fig. 5. Map of the road stretch used for driving experiments (part of Route 25), along with the route shown in red. Approximate WiFi coverages are shown (from [24]).

B. Experimental Testbeds

We have used two types of testbeds. The 'emulation testbed' can control rates and latencies of different paths in a controlled setup in the lab. The 'vehicular testbed' is used for field experiments.

1) Emulation Testbed: The network environment for this testbed is shown in Figure 4. The two routers are Dell desktops with two ethernet interfaces using 'Linux Advanced Routing & Traffic Control' and `netem` to emulate different bandwidths, delays and loss rates. The server and client are Dell desktops running Linux. They are connected over 100 Mbps ethernet links to the routers. The emulation testbed provides a controllable and repeatable environment for evaluation.

2) Vehicular Testbed: Our vehicular testbed uses the same hardware setup as in our prior work [5]. Briefly, we use a Dell Latitude D510 laptop running Linux as the mobile client. The original miniPCI WiFi interface from the laptop is removed and replaced by a carrier-grade interface (Ubiquity XR2 [29]) with transmit power set to 25 dBm. The WiFi card is connected to a high-gain (12 dBi) omni-directional antenna. The interface uses Atheros chipset supported by the madwifi driver. The laptop also carries Verizon's USB-based USB760 EVDO Rev. A Modem as the second network interface. The client accesses two heterogeneous wireless networks – Optimum WiFi's metro-scale deployment [24] and Verizon's EVDO network. Figure 5 shows a stretch of road of approximately 9 miles where the driving experiments were done. Some of our experiments, wherever indicated, were done in stationary settings along the path.

C. Congestion Control

1) SACK-Driven Fast Retransmission: SCTP uses TCP's Reno-like congestion control that uses fast retransmission. Three duplicate CACKs (Cumulative ACKs) trigger the sender to fast retransmit the unACKed packet with the smallest TSN (Transmit Sequence Number). We call this **CACK-Driven Fast Retransmission (CD-FRT)**. For single path situation, duplicate CACKs mean out-of-order packet arrivals, indicating possible packet loss. When transmitting on multiple paths concurrently, out-of-order packets may arrive in a normal course due to different delays of different paths. This could trigger multiple duplicate CACKs and fast retransmission, even in absence of any packet loss. Our emulation experiment shows

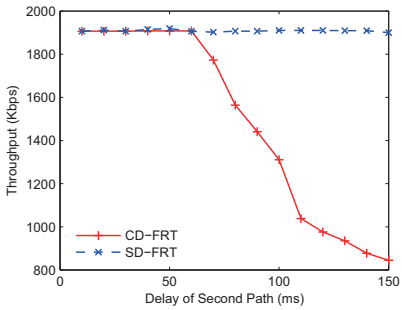
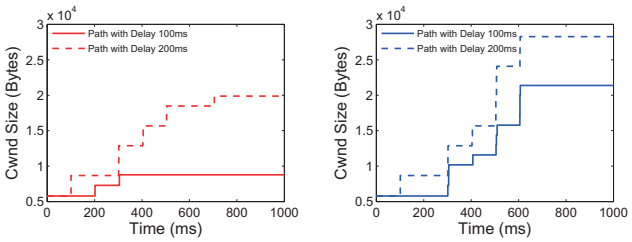


Fig. 6. Throughputs for CACK-driven and SACK-driven fast retransmission schemes. The rates of the paths are set to 2 Mbps and 1 Mbps in the emulation testbed. The delay of the 2 Mbps path is fixed at 50 ms. The delay of the 1 Mbps path is varied.

that with the difference of path delays increasing, the fraction of packets fast retransmitted increases even though there are no packet losses. The unnecessary fast retransmissions not only waste the bandwidth, but also prevent congestion window from advancing normally (when fast transmission happens, the congestion window of the corresponding path will be decreased to half) thus adversely affecting the throughput for the CD-FRT scheme as shown in Figure 6.

With multiple paths with delay differences, duplicate CACK is no longer a sign of packet loss. We provide an alternate mechanism to indicate out-of-order packet arrival for each path individually. Besides CACKs, when the receiver detects TSN gaps, each received packet after the gap is acknowledged by a SACK (Selective ACK). We modify the fast retransmit procedure to work as follows – if three packets sent over a path with TSN greater than smallest unACKed TSN sent over the same path have been ACKed then we consider this to be a sign of packet loss. The smallest unACKed packet on that path is then fast retransmitted. We call this approach **SACK-driven fast retransmission (SD-FRT)**. Note this approach retains the general idea of fast retransmit except that it treats each path separately.

We have evaluated the benefit of our SD-FRT procedure on our emulation testbed. Figure 6 shows the throughputs for the CD-FRT and SD-FRT schemes. In case of SD-FRT, the unnecessary fast retransmissions are eliminated resulting in the aggregated throughput being stable over different delays



(a) CACK-driven congestion window advancement.

(b) SACK- and CACK-driven congestion window advancement.

Fig. 7. Example trace of different congestion window advancement methods on the emulation testbed. The rates of both paths are set to 2 Mbps and the delays are set to 100 ms and 200 ms.

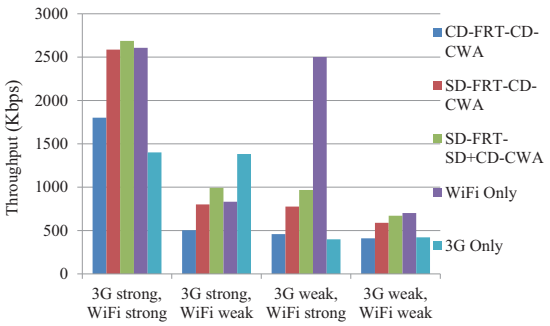


Fig. 8. Average throughputs in four different scenarios with different signal qualities on the WiFi and 3G links as evaluated on the vehicular testbed. Three different combinations of the two fast retransmit and congestion window advancement methods are shown along with 'WiFi only' and '3G only' for baseline comparisons.

for the 1 Mbps path.

2) SACK- and CACK-Driven Congestion Window Advancement: Out-of-order packet arrival also causes incorrect congestion window advancement. SCTP maintains a congestion window for each path. By default, congestion window advancement is driven by incoming new CACKs. We call this default mechanism as **CACK-Driven Congestion Window Advancement (CD-CWA)**. As explained before, in *oSCTP* the delay difference of the two paths leads to out-of-order packet arrival at the receiver. This means the receiver will generate SACKs for each path frequently but generate new CACKs infrequently. Due to infrequent CACKs the congestion windows will not advance normally. Our solution to this problem is to make the congestion window of a path advance if the unACKed packet with the smallest TSN on a path is acknowledged by either a SACK or a CACK. We call this approach **SACK-Driven and CACK-Driven Congestion Window Advancement (SD+CD-CWA)**. Again, we retain the general idea of congestion window advancement except each path is treated separately.

Figure 7 shows the performance advantage of SD+CD-CWA with respect to CD-CWA. Note that CD-CWA limits the congestion window advancement (Figure 7(a)), while SD+CD-CWA allows the congestion window to advance more freely (Figure 7(b)).[2] In this example, the SD+CD-CWA results in a throughput gain of 40% over the CD-CWA scheme.

We also evaluated the above two approaches on the vehicular testbed. A set of combined notations are used (e.g., CD-FRT-CD-CWA, etc.) which should be self-explanatory. Since actual driving conditions cannot be repeated for protocol comparisons, we compare our techniques in four stationary scenarios (with the vehicle stopped at the roadside), where the WiFi and 3G signal qualities are either weak or strong resulting in throughputs on the individual paths being low or high. These four scenarios represent the extreme scenarios that the mobile client might face. Figure 8 shows the results. Note that eliminating unnecessary fast retransmission improves the throughput roughly by 40-60%. Altering the congestion window advancement procedure to respond to SACKs along with CACKs improves the throughput by approximately 5-20% over the CACK only approach. Also, 'WiFi only' or '3G only' provides higher throughput relative to use of both links when the other link is weak. This is because of the use of round-robin scheduling that disregards quality of the links. We will now address scheduling in the following subsection.

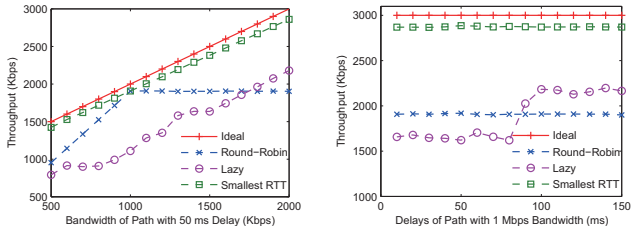
[2]Unlike TCP, SCTP increases congestion window only when the current window is exhausted. Thus, the usual sawtooth pattern of TCP's congestion window is not seen in Figure 7.

D. Scheduling Paths

Until now, in all our experiments, packets arriving from the upper layer are scheduled alternately on the two paths in *Round-Robin* fashion. As we have seen in Figure 8, this simple strategy is not efficient. The throughput tends to get bounded by two times the throughput of the path with smaller bandwidth. If one thinks carefully, there are two pieces to this problem. (i) Packets must be scheduled on a path as long as the congestion window on the path is open. (ii) Sequence number holes in the receive window should be avoided. Since the receiver relays packets to the upper layer in sequence, existence of holes simply prevents the receive buffer from flushing. The simple *Round-Robin* scheduler does not address any of these two issues. We propose two strategies which try to schedule packets so as to address the above two concerns.

- *Lazy:* Schedule packets on the current path until the congestion window exhausts and then switch to the other.
- *Smallest RTT:* Always schedule packets on the path with the smallest RTT that has congestion window open.

We have implemented the above described schedulers as part of *oSCTP* and compared their performance to the *Round-Robin* scheduler. Figure 9(a) shows the throughputs of the three schedulers on the emulation testbed with different path characteristics. The *Ideal* plot is simply the sum of the bandwidth of the two paths. Note that in both Figures 9(a) and 9(b) the *Round-Robin* scheme provides an aggregated throughput of two times the path with lesser bandwidth. The *Lazy* scheduler tends to out-perform the *Round-Robin* scheduler only when the disparity in the bandwidths of the two paths or the delays of the two paths is large. The *Smallest RTT* scheduler's throughput is almost equal (closer than 90%) to the *Ideal* as it addresses the above two issues.



(a) Path 1: (100 ms & 1 Mbps); Path 2: (50 ms & varying rate).

(b) Path 1: (50 ms & 2 Mbps); Path 2: (varying delay & 1 Mbps).

Fig. 9. Performance of different schedulers on the emulation testbed. The characteristics of the two paths are shown in the subheadings.

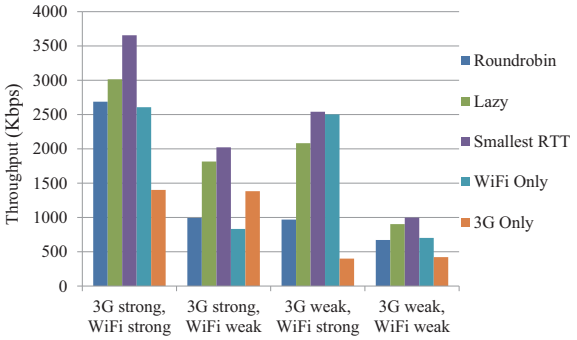


Fig. 10. Throughput comparison of the three schedulers on the vehicular testbed in four different scenarios with different signal qualities on the WiFi and 3G links.

Figure 10 shows the throughput comparison of the three schedulers on the vehicular testbed in four different scenarios as in Section IV-C2. The *Smallest RTT* scheduler always has the best performance. It achieves almost about 90% of the combined throughputs of WiFi only and 3G only in most scenarios. Our analysis shows *Smallest RTT* to be the best choice for the scheduler. In the remainder of our analysis we choose *Smallest RTT* as our default scheduler.

E. Resilience to Path Failure

Both WiFi and 3G connections oscillate greatly in the moving vehicle. SCTP recovers from path failure by choosing an alternate path to its peer node. This recovery, however, is very slow. There are two parts to this problem. We elaborate on each of them and provide solutions.

1) Recovery from Path Failure: The default mechanism of SCTP to detect path failure is painfully slow for vehicular scenarios where frequent failures may occur on the WiFi link. In stock SCTP, a path is considered 'failed' only when 4 successive re-transmission attempts fail to elicit an ACK. Since these re-transmission attempts happen after exponentially increasing backoffs, the failure detection can take a long time, often more than 60 sec.

Since the packets scheduled on the failed path do not arrive at the receiver, this creates holes in the receive buffer preventing it from flushing. This makes the aggregated throughput drop to zero until the server identifies that the path has failed and re-schedules the packets on the other path. An

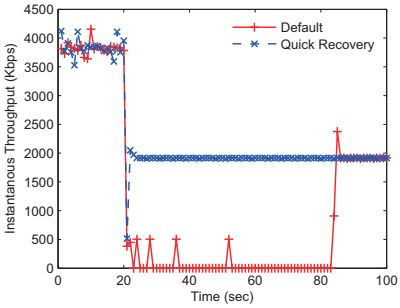


Fig. 11. Trace of instantaneous SCTP throughput showing impact of path failure. Both paths are set to 2 Mbps with 50 ms delay on the emulation testbed. At 20 sec, one path is forced to fail.

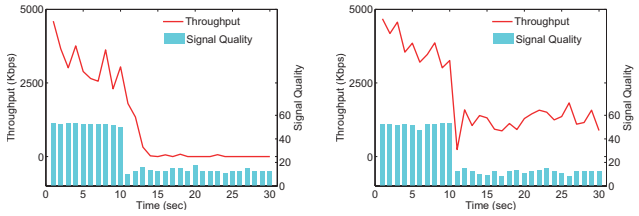
experiment on the emulation testbed demonstrates this. See Figure 11. With the default scheduling, one path failure leads the throughput to zero. Only after 60 seconds, the throughput settles back at the bandwidth provided by the other, live, path.

Our implementation of *oSCTP* designates a path on which a retransmission timeout has occurred as failed. In addition to SCTP's default procedure to re-designate a path as 'alive,' *oSCTP* also re-designates the path as alive if any outstanding packet is ACKed on such a failed path.[3] With this improvement, upon path failure the aggregate throughput settles to that provided by the live path. See Figure 11.

2) Preventing Loss of ACKs: Recall that in SCTP a path is defined by the destination address only. This means that in our system model in Figure 3, from SCTP's perspective, there are two paths from the server to the client, but only one path from the client to the server. A packet's source IP address is determined by the source node's IP layer and not by SCTP. Thus, the client's IP layer can choose any one of the two paths for sending back ACKs. If the path chosen to send back ACKs fails and the IP layer remains oblivious to this fact, then it will continue to schedule the ACKs on this path leading to zero throughput. In this case, since the server does not receive ACKs for its transmissions on either paths, the server considers the entire SCTP association to be broken. Figure 12(a) demonstrates this problem showing the aggregate throughput going down to zero as the client moves away from a WiFi AP on our vehicular testbed.

We address this issue by making *oSCTP* on the client explicitly choose a source IP address for the outgoing ACKs. *oSCTP* chooses the source IP address of an ACK to be the last received packet's destination IP address. This ensures that the ACK is sent over a live path. Figure 13 shows a trace of ACK throughputs at the server. The path carrying the ACKs fails at 10 sec. When one path fails, data transfer continues on the other path unhindered when the ACKs have an appropriate source address. The resulting improvement in throughput is shown in Figure 12(b), as client continues to use the available 3G path to deliver ACKs.

[3]SCTP's default mechanism to test whether a failed path has been back to life is to send heartbeat messages on it every 30 sec. We have taken a more proactive approach by having *oSCTP* to send heartbeats every 1 sec.



(a) Default. (b) Choosing correct source address for ACKs.

Fig. 12. Trace of instantaneous throughputs and signal quality demonstrating impact of path failure and choosing the right path for the ACKs. The vehicular testbed is used. The WiFi path fails at around 10 sec.

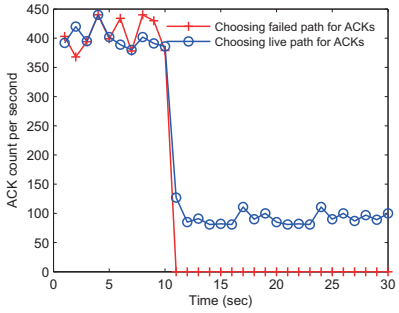


Fig. 13. Trace of instantaneous ACK throughputs measured at the server. The path carrying the ACKs fail at 10 sec. The emulation testbed is used.

The overall gain of adding resiliency to path failures to *oSCTP* is shown in Figure 14. To emulate intermittent connections caused by vehicular mobility, one path is taken down at 60 sec and remains down for an interval of time and then is brought back up for 60 sec again. This experiment is repeated 10 times and the average throughput is reported for different failure duration. By employing resilience to path failure the aggregate throughput obtained is around 95% of the ideal throughput.

F. Throughput Prediction and Packet Scheduling

Our goal is to implement **Strategy 1** described in Section III-A on the SCTP server in each scheduling interval (chosen to be 1 sec). For each scheduling interval i, the

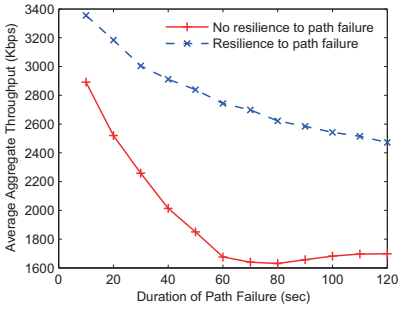


Fig. 14. The average throughput when one path fails. Both paths are 2 Mbps with 50 ms delay. One path fails at the 60 sec, remains down for a varying interval and then comes back up for 60 sec again.

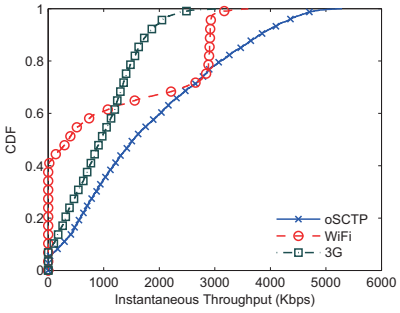


Fig. 15. CDF of instantaneous throughputs for WiFi, 3G and *oSCTP* in driving experiments.

available bandwidth X_w^i on the WiFi path is measured on the server side by simply calculating the number of bytes acked in the i-th interval. x_g^i is determined using Equation 1 (or, **Strategy 1**). Note that in step 2, X_w^i must now be estimated as this value needs to be known at the start of the interval i. The measured value of X_w^{i-1} in the prior interval is used to estimate X_w^i. See the discussion in Section III-B. Since the scheduling interval is 1 sec, it just means that the 3G path must carry a maximum of x_g^i bits in the interval i. This can be achieved via a simple modification of path scheduling described in Section IV-D. The 3G path is used only until x_g^i bits are transmitted in each scheduling interval.

Ordinarily, *oSCTP* does not know which path is the 3G link. We address this issue by letting the 'primary path' to always denote that path where bits should be saved. The user can easily set 3G link as 'primary path' in *oSCTP*'s socket API.

V. Evaluation of Hybrid Access

We now evaluate *oSCTP* for hybrid WiFi and 3G access under vehicular mobility. The vehicular testbed as described in IV-B is used. In the reported experiments, no optimization is done for the WiFi link layer handoff. Stock `madwifi` driver implementation is used. Thus, the WiFi performance results could only be improved if handoff or packet forwarding optimizations [30], [31] are used. The WiFi network retains the IP address across handoffs, even after temporary disconnections. Thus, there is no network layer handoff latency. The interested reader can look at [5] for further details about the network environment. The driving experiments are done on the stretch of road showed in Figure 5. The average driving speed is about 40 mph and the highest speed can be as fast as 50 mph.

A. Baseline Striping Performance

We first evaluate striping performance with *oSCTP*. Here, both interfaces are used together to obtain the maximum possible aggregate throughput. The utility-cost model is not considered as there is no throttling on the 3G path. For the performance analysis, three different sets of drives are done on the same stretch of road in the same direction, first measuring WiFi and 3G performance in an isolated fashion, and then measuring striping performance with *oSCTP* as implemented

Combined throughput target	2 Mb/s	900 Kb/s	600 Kb/s
Avg. aggregate throughput	1.43 Mb/s	1.27 Mb/s	1.21 Mb/s
How often 3G fired	77%	62%	52%
Fraction of load carried on 3G	37%	26%	19%

TABLE I
AVERAGE THROUGHPUTS AND 3G LINK USAGE FOR THE THREE THROUGHPUT TARGETS FOR OSCTP-BASED SCHEDULING. AVERAGE THROUGHPUT WOULD BE APPROXIMATELY 700 KBPS IF ONLY 3G LINK USED.

in the previous section. Instantaneous throughputs (average of per-second throughputs) are logged.

Figure 15 shows the CDF of instantaneous throughputs for the three experiments. Note zero WiFi throughput for about 40% of times (likely because of lack of coverage), but superlative throughput (close to 3 Mbps) for about 25% of times. The aggregated throughput achieved via striping with *oSCTP* is obviously far superior. The median is roughly the sum of the median of WiFi and 3G individually. The average aggregated *oSCTP* throughput is also similar – roughly 85% of the sum of the average of WiFi and 3G.

B. Scheduling Performance

We now evaluate the performance of the *oSCTP*-based scheduler implementing **Strategy 1**. For this evaluation, the only input parameters are the 'combined throughput target' or the value of $\frac{1}{K_3}$ (see Section III-A) and the scheduling interval (see Section III-B). We choose three different values for this target, viz., 2 Mbps, 900 Kbps and 600 Kbps. These target values are respectively much higher, somewhat similar and lower than the average 3G throughput. In each case the scheduling interval is chosen to be 1 second. We performed two drives for each of these targets and present averaged results.

Table I summarizes the results of the driving experiments. Note that the average throughput is much less than the target for 2 Mbps, and is much better for the other two targets. This is simply because the network overall often lacks 2 Mbps capacity, but more frequently can offer the other two lower throughput targets. This is also apparent from Figure 15. Note that we are able to reach much higher averages for the two lower targets, simply because WiFi sometimes provides a superlative throughput and our strategy never throttles the WiFi link. Also note that for larger throughput targets, 3G is fired in more scheduling intervals and the fraction of total load carried by 3G is also more. But, overall this fraction is small and varies between 19–37% for our chosen range of throughput targets. This obviously means a significant saving of bits on the 3G network.

We also investigate the performance of the underlying scheduler. See Figure 16. Two types of performance metrics are presented. The first one is the prediction error. Note that in implementing **Strategy 1**, we use the WiFi throughput in the previous scheduling interval to estimate the same in the current scheduling interval. The CDF of the prediction error (normalized by the target throughput) is presented in Figure 16(a). We note that a large zero error region is due to the fact the zero throughput on WiFi is typically very accurately predicted and such zero throughput regions are frequent (about 40% of the times).

The second metric we evaluate is actual scheduling performance. This is related to how much load the 3G link actually carries versus how much load the scheduler should schedule on the 3G link in each scheduling interval. We normalize their difference ('actually carried' − 'should carry') by the throughput target and present the CDF in Figure 16(b). Note that the negative differences are significant for higher throughput targets. This is attributed to 3G link lacking capacity and not being able to reach its target $\frac{1}{K_3} - X_w$. For the lowest throughput target (600 Kbps) this issue is less of a problem and the difference reduces and can be explained mostly by the prediction errors. For the case of positive differences, it is mostly explained by the prediction error.

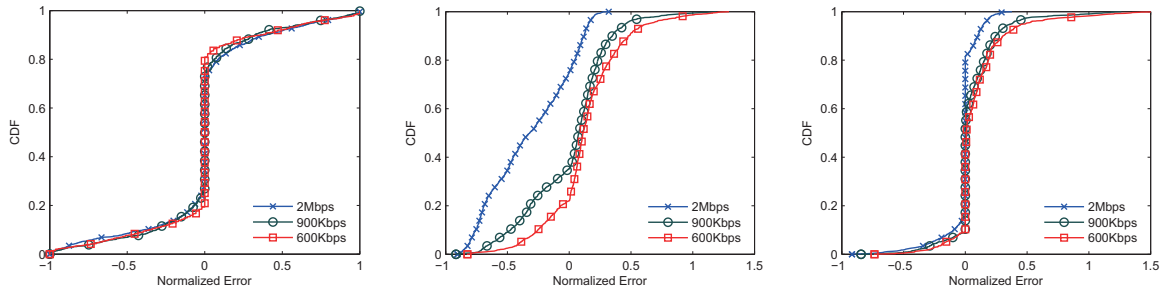
It is also of interest evaluating how much scheduling error our implementation introduces. This is a similar normalized difference as above. However, in the intervals where 3G is fired but the aggregate throughput is less than the throughput target, the inability to reach target is typically not an error introduced in the scheduler. This is due to the lack of enough capacity in the network. Assuming the scheduling error to be zero in such intervals, we plot another CDF for the same quantity as in Figure 16(b). The new plot is in Figure 16(c). Note that the roughly 90% of the times the error within ± 0.2.

VI. CONCLUSIONS AND FUTURE WORK

We have developed a transport layer protocol based on SCTP to move bits from expensive cellular data networks to relatively cheaper WiFi networks. The focus has been to support vehicular mobility. We have developed a general purpose utility and cost function-based formulation and a scheduling system that follows such models. The scheduling system is designed to delivers the optimal benefit to the user based on the chosen utility and cost models. We implement our scheduling system as part of our modified Linux SCTP implementation (called *oSCTP*) that allows for concurrent use of both paths. The utility function may be application specific and can be chosen from a library of functions, by associating applications with utility functions. It is also possible to provide the user with an interface to dial up or down the utility dynamically.

We have addressed a range of issues that arise when building an SCTP implementation that supports striping in a mobile context, e.g., congestion control, path scheduling and overcoming path failure. We have also provided striping performance results in an open multihomed environment using different technologies. Our driving experiments using *oSCTP*-based scheduler shows roughly 65%-80% overall load reduction on the 3G network by exploiting a metro-scale WiFi network.

While our experiments are indeed limited with straightforward choices of utility and cost models, the general trend they demonstrate is very promising. We hope that our experience will encourage service providers to deploy more metro-scale



(a) CDF of prediction errors. (b) CDF of normalized scheduling errors. (c) CDF of normalized scheduling errors with corrections.

Fig. 16. Evaluation of the *oSCTP*-based scheduler for three different throughput targets in the driving experiments. See the text for explanation.

WiFi networks with even better road-side coverage and integrate WiFi and 3G networks more tightly.

Several unaddressed issues remain. WiFi being on a license-free band can potentially become congested even when the operator has provisioned the network well. This will likely limit its throughput benefits. On the other hand, new technologies such as 802.11n and 802.11r can provide potential for higher throughputs and faster handoffs making WiFi use even more productive than reported here. Finally, use of multiple interfaces concurrently increase the energy cost. This will be an issue for a smartphone or tablet like device. However, energy itself can be a part of the utility-cost model. We will investigate this in a future work.

ACKNOWLEDGEMENT

This work was partially supported by NSF grants CNS-0751121, CNS-0831791 and CNS-1117719.

REFERENCES

[1] "Muniwireless list of cities and counties with large WiFi networks," http://www.muniwireless.com/reports/Mar-28-2009-list-of-cities.pdf.
[2] V. Bychkovsky, B. Hull, A. Miu, H. Balakrishnan, and S. Madden, "A measurement study of vehicular Internet access using in situ Wi-Fi networks," in *Proc. ACM MobiCom Conference*, 2006.
[3] J. Eriksson, H. Balakrishnan, and S. Madden, "Cabernet: A WiFi-Based Vehicular Content Delivery Network," in *Proc. ACM MobiCom Conference*, 2008.
[4] A. Balasubramanian, R. Mahajan, A. Venkataramani, B. N. Levine, and J. Zahorjan, "Interactive WiFi connectivity for moving vehicles," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 427–438, 2008.
[5] P. Deshpande, X. Hou, and S. Das, "Perfomance comparison of 3G and metro-scale WiFi for vehicular network access," in *Proc. Internet Measurement Conference (IMC)*, 2010.
[6] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. ACM MobiSys Conference*, 2010.
[7] W. Lehr and L. McKnight, "Wireless Internet access: 3G vs. WiFi?" *Telecommunications Policy*, vol. 27, no. 5-6, pp. 351–370, 2003.
[8] A. Giannoulis, M. Fiore, and E. W. Knightly, "Supporting vehicular mobility in urban multi-hop wireless networks," in *Proc. ACM MobiSys Conference*, 2008.
[9] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive methods for improved vehicular WiFi access," in *Proc. ACM Mobisys Conference*, 2009, pp. 263–276.
[10] R. Stewart, "Stream control transmission protocol," Internet RFC 4960.
[11] J. Ott and D. Kutscher, "Drive-thru internet: IEEE 802.11b for automobile users," in *Proc. IEEE Infocom*, 2004.
[12] ——, "A disconnection-tolerant transport for drive-thru internet environments," in *Proc. IEEE Infocom*, 2005.
[13] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular opportunistic communication under the microscope," in *Proc. ACM MobiSys*, 2007.
[14] 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. ACM MobiCom Conference*, 2008, pp. 211–222.
[15] J. Yao, S. S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *Proc. ACM WiNTECH*, 2008, pp. 11–18.
[16] J. Ormont, J. Walker, S. Banerjee, A. Sridharan, M. Seshadri, and S. Machiraju, "A city-wide vehicular infrastructure for wide-area wireless experimentation," in *Proc. ACM WinTech Workshop*, 2008, pp. 3–10.
[17] K. Sklower, B. Lloyd, G. McGregor, D. Carr, and T. Coradetti, "The PPP Multilink Protocol (MP)," *RFC 1990*, August 1996.
[18] D. Phatak and T. Goff, "A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments," in *Proc. Infocom Conference*, vol. 2, 2002, pp. 773–781.
[19] L. Magalhaes and R. Kravets, "Transport level mechanisms for bandwidth aggregation on mobile hosts," in *Proc. IEEE ICNP Conference*, Riverside, CA, November 2001, pp. 165–171.
[20] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee, "MAR: a commuter router infrastructure for the mobile internet," in *Proc. ACM MobiSys Conference*, 2004, pp. 217–230.
[21] K.-H. Kim and K. G. Shin, "PRISM: improving the performance of inverse-multiplexed TCP in wireless networks," *IEEE Trans. on Mobile Computing*, vol. 6, no. 12, pp. 1297–1312, Dec. 2007.
[22] J. R. Iyengar, P. D. Amer, and R. Stewart, "Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths," *IEEE Trans. on Networking*, vol. 14, no. 5, pp. 951–964, Oct. 2006.
[23] P. Natarajan, N. Ekiz, P. D. Amer, and R. Stewart, "Concurrent multipath transfer during path failure," *Computer Communications*, vol. 32, pp. 1577–1587, 15 September 2009.
[24] "Optimum WiFi," http://www.optimum.net/MyServices/WiFi/.
[25] S. Shenker, "Fundamental design issues for the future Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 7, pp. 1176–1188, 1995.
[26] M. A. Zaharia and S. Keshav, "Fast and optimal scheduling over multiple network interfaces," http://src.acm.org/Matei/matei.html, 2007, tech. Report.
[27] M.-R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. ACM MobiSys '10*. New York, NY, USA: ACM, 2010, pp. 255–270.
[28] "Linux kernel stream control transmission protocol (lksctp) project." [Online]. Available: http://lksctp.sourceforge.net/
[29] "Ubiquity Networks, Inc." http://www.ubnt.com.
[30] J. Jeong, S. Guo, Y. Gu, T. He, and D. Du, "Tsf: Trajectory-based statistical forwarding for infrastructure-to-vehicle data delivery in vehicular networks," in *ICDCS'10*, Genoa, Italy, 2010.
[31] F. Xu, S. Guo, J. Jeong, Y. Gu, Q. Cao, M. Liu, and T. He, "Utilizing shared vehicle trajectories for data forwarding in vehicular networks." in *INFOCOM'11*, Shanghai, China, 2011.