On Effectively Exploiting Multiple Wireless Interfaces in Mobile Hosts

Cheng-Lin Tsao and Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
{cltsao,siva}@ece.gatech.edu

ABSTRACT

Most mobile devices today are equipped with multiple and heterogeneous wireless interfaces. In this paper we ask the following question: *what is the best approach to leverage the multiple interfaces available at a mobile device in terms of the performance delivered to the user?* In answering the question we argue that simple "bandwidth aggregation" approaches do not provide any meaningful benefits when the multiple interfaces used have highly disparate bandwidths as is true in many practical environments. We then present *super-aggregation*, a set of mechanisms that in tandem use the multiple interfaces intelligently and in the process is able to achieve a performance that is "better than the sum of throughputs" achievable through each of the interfaces individually. We prototype super-aggregation on both a laptop and the Google Android mobile phone and demonstrate the significant (up to 3x throughput) performance improvements it provides in real-world experiments.

Categories and Subject Descriptors

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

General Terms

Algorithm, Design, Performance

Keywords

Heterogeneous wireless networks, multi-homed mobile device, bandwidth aggregation

1. INTRODUCTION

Mobile devices, be it laptops or mobile phones, have gone through a sea-change in their capabilities over the last decade. One of the different dimensions along which they have evolved is that of connectivity. To cater to the requirement of mobile users of ubiquitous and best possible connectivity, most mobile devices today are equipped with multiple and heterogeneous wireless interfaces. For example, it is commonplace for a typical laptop to be equipped with WiFi, Bluetooth, and possibly even 2.5G/3G wireless interfaces. Similarly, popular mobile phones today ranging from the iPhone to the Google Android phone to the Blackberry all come equipped with multiple wireless data interfaces. Not surprisingly, an interesting and relevant question that arises in the context of such mobile devices is the following: *What is the best approach to leverage the multiple interfaces available at a mobile device in terms of the performance delivered to the user?*

This above question has not gone unanswered in related literature. Approaches that have tackled the question fall under two broad categories: "one-interface at a time" approaches and the more recently studied "simultaneous use of multiple interfaces" approaches. For obvious reasons including the fact that the wireless interfaces are innately limited and heterogeneous in capabilities, the latter class of approaches have significant promise in improving the performance experience by the user. However, many of the approaches that fall into this category have focused on what can be defined as *bandwidth aggregation* as the primary goal to achieve. In other words, if there are two interfaces I_1 and I_2 available with respective bandwidths of B_1 and B_2, the approaches focus on delivering the aggregate bandwidth of $B_1 + B_2$ to the user. We call the functionality provided by such approaches *simple aggregation* [10, 12, 13].

In this paper we argue that for most practical environments, simple aggregation, while achievable will provide no meaningful benefits to the user. We make this argument in the context of typical wireless interfaces today exhibiting a large degree of heterogeneity in terms of capabilities including bandwidth. Consider, for example, a mobile phone that supports both 3G and Wi-Fi data interfaces. 3G data rates support bandwidths of up to 100-500Kbps while Wi-Fi interfaces support 2-54Mbps. A simple aggregation of the bandwidths provided by the two interfaces will provide negligible improvement in terms of performance perceived by the user, with respect to a best-available-interface solution. In experimental field trials conducted with an Android phone that supports both 3G (HSDPA) and Wi-Fi (802.11g), the typical application layer data rates we observed on the two interfaces were 420 Kbps and 11.6 Mbps respectively. Performing simple aggregation of these two interfaces will achieve a net performance improvement of merely 3.6% to the user, with respect to using Wi-Fi by default.

In this context, we ask the following question in this paper: *Is there a more intelligent strategy for aggregation of multiple heterogeneous interfaces that will enable us to achieve "more than the sum of the parts" in terms of performance?* For example, is it possible to intelligently aggregate the aforementioned interfaces on the Android phone and achieve more than $0.42 + 11.6$ Mbps? In answering the question, we present a set of *super-aggregation* principles that with appropriate knowledge of the data being transferred over the interfaces do indeed enable an aggregate performance that is more than the sum of the parts. We call an approach that uses these principles as simply *super-aggregation*.

While we elaborate on the rationale behind the principles and their value later in the paper, briefly we introduce three *super-aggregation* principles: *selective offloading*: in spite of the fact that an interface has a much smaller amount of bandwidth, selectively offloading certain portions of the data transferred can have a significant impact on the performance; *proxying*: when an interface has only limited bandwidth but is up when the other interface is down, the limited bandwidth can be used for critical control information that in turn can serve to significantly improve the overall performance of the data transfer; and *mirroring:* for certain portions of the data being transferred intelligently mirroring the transfer on the interface with lower bandwidth can again have a profound impact on the perceived performance.

We present the *super-aggregation* strategy as a generic solution, but consider in detail one instantiation of it in the specific context of TCP acceleration over wireless data networks. We show how *super-aggregation* can be used to provide considerable performance improvements when using TCP. We later discuss other applications/protocols that may also benefit from super-aggregation. Also, we place special emphasis on the adoptability of *super-aggregation* and hence focus on a realization of *super-aggregation* that requires changes *only to the mobile device* and does not require any changes to the other communication end-point. We believe that such a design strategy may prove to be critical in the practicality of the solution. Finally, we implement *super-aggregation* on two platforms - a laptop and an Android mobile phone, and through real-world experiments that involve 3G (EVDO/HSDPA) and Wi-Fi (802.11g/b) interfaces demonstrate the efficacy of the principles and the overall solution.

Thus, in summary, the following are the key contributions of this work:

- We present a solution strategy called *super-aggregation* for mobile devices with multiple interfaces that achieves more than the sum of the parts in terms of aggregate performance when the multiple interfaces are used simultaneously. In this paper, we mainly focus on Wi-Fi and 3G as the two heterogeneous interfaces on the mobile device for all discussions and explore extensions to other technologies and more wireless interfaces later in the paper;

- We show that *super-aggregation* can be realized purely as a layer-3.5, mobile-device-only solution and how an instantiation of *super-aggregation* can be used to achieve TCP acceleration in wireless data networks; and

- We implement *super-aggregation* on two mobile platforms - a laptop and an Android mobile phone and in the process use real-world 3G and Wi-Fi wireless data networks to demonstrate the efficacy of the proposed *super-aggregation* principles and overall solution.

The rest of the paper is organized as follows: Section 2 presents the scope, motivation and goals of the work. Section 3 introduces the *super-aggregation* principles in the context of TCP acceleration. Section 4 discusses in detail the software architecture of *super-aggregation*, and Section 5 generalizes it to other applications/protocols with a case study. Section 6 describes the prototype implementation of *super-aggregation* on a laptop and an Android mobile phone and presents performance results. Section 7 and Section 8 discusses related work and issues, and Section 9 concludes the paper.

2. SUPER-AGGREGATION

2.1 Scope

The scope of this work is restricted to devices with multiple wireless interfaces. While several of the principles presented may be relevant for devices for multiple wired interfaces or heterogeneous wired and wireless interfaces as well, we do not delve into such scenarios in the paper. The devices themselves can be either mobile computing devices such as laptops and mobile smart phones that have data capabilities. In terms of the wireless interfaces, the principles and solutions presented are agnostic to the specific technologies used.

Without impacting the generality of the proposed solutions, we rely entirely on a laptop and a mobile phone (Google Android) as the devices in the experimental set-up used for performance characterization, evaluation and proof-of-concept demonstration. Similarly, we use 3G and Wi-Fi as the heterogeneous wireless interfaces in all our experiments. The laptop is equipped with an Atheros 802.11b/g card and a 3G (EVDO) USB stick. The Google Android phone has built-in 802.11g and 3G (HSDPA) interfaces. Further details of the test-bed can be found in Section 6.

While the super-aggregation principles presented are extensible to other applications and protocols, *we ground most of our initial discussions of super-aggregation in the context of TCP acceleration.* We later revisit the extensibility of super-aggregation to both other applications/protocols and to environments with more wireless interfaces later in the paper.

2.2 Problem Motivation

In this section we briefly illustrate the limitation of switching and simple aggregation and thus motivate the need for *super-aggregation.* Figure 1 shows measurement of TCP throughput as experienced by a multi-interface laptop that has 3G (EVDO) and Wi-Fi (g/b) connectivity. We emulate a perfect simple aggregation mechanism by opening two independent TCP connections through the two available interfaces, thus removing any synchronization bottlenecks. Hence, the result presented for simple aggregation is idealized.

The average throughput on the 3G link is much smaller than that on the Wi-Fi link and hence simple aggregation only gives marginal improvement: merely 3% in the 'g' mode and 16% in the 'b' mode. The measurement shows that simple aggregation is not an effective way of aggregating hetero-

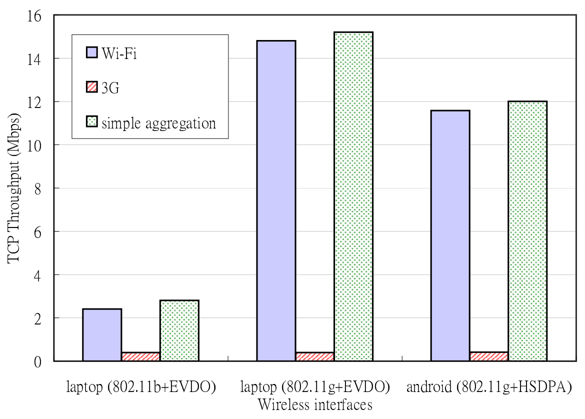


Figure 1: Marginal improvement of simple aggregation on heterogeneous interfaces.

geneous wireless interfaces in typical scenarios. The above result, while obvious, serves as the motivation for our investigation into super-aggregation principles that more intelligently leverage the multiple interfaces to achieve a throughput performance that is *better than the sum of the parts.*

2.3 Goals

The goals of this work, beyond the identification and development of super-aggregation principles, are as follows:

- **Deployability:** We believe that any aggregation solution for multi-homed mobile devices will have a significantly better adoption rate if it requires only mobile device side changes. Hence, one of our key goals will be to develop and realize super-aggregation through changes only at the mobile device.

- **TCP and beyond:** Since TCP is by far the most dominantly used transport layer protocol in today's Internet, we focus our goals almost entirely on TCP acceleration as the first application of super-aggregation. We however will revisit extending super-aggregation to other applications/protocols later in the paper.

- **Prototyping:** Another key goal for the work is prototyping and demonstrating super-aggregation on real-world platforms including a laptop and a mobile phone.

3. SUPER-AGGREGATION PRINCIPLES

In this section we present the different super-aggregation principles, and ground the discussions specifically in the context of TCP acceleration. At a high level, when aggregating a high-bandwidth interface with a low-bandwidth interface, a simple bandwidth aggregation strategy does not yield any significant improvements in performance. The super-aggregation principles on the other hand *explicitly leverage properties of the low-bandwidth interface that may be superior to those of the high-bandwidth interface to relieve any bottlenecks that prevent the effective utilization of the high-bandwidth interface.*

For each of the principles, we identify the rationale and present the high-level design. We arrive at the rationale by appropriately identifying an existing bottleneck in TCP's operation and show how super-aggregation may be used to relieve that bottleneck. For all the experiments we use the set-up discussed in Section 6 where both the laptop and the smartphone are connected to the backbone using a 3G link and an Wi-Fi link. A file server hosted in a major university is used as the fixed point from where content is downloaded or to where content is uploaded. We revisit the solution details including the generalization of the principles to beyond TCP in the next section. Also, we present the super-aggregation techniques for TCP acceleration in the form of a layer 3.5 solution in the protocol stack. For brevity, in the rest of the discussions we refer to the high bandwidth interface as the wi-fi interface and the low bandwidth interface as the 3G interface.

3.1 Selective Offloading - Tackling TCP Self Contention

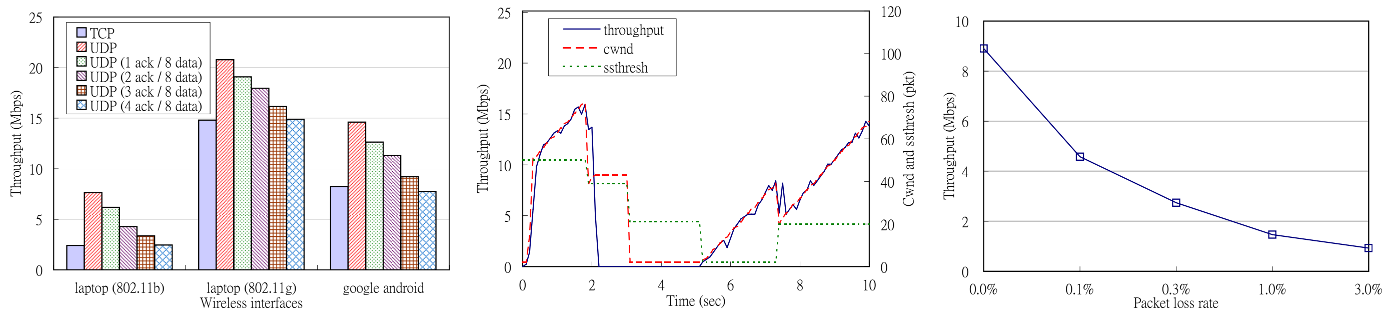
TCP Self Contention: Figure 2(a) shows the TCP throughput measurements for downstream traffic as observed at both the laptop and the smartphone. The figure also shows the throughput enjoyed by an application using UDP in a similar set-up. The UDP packet sizes are set to be the same as the TCP maximum segment size (MSS). The UDP throughput is 30% higher than the TCP throughput in the 802.11g network and as much as 70% higher in the 802.11b network.

Hence, the difference in performance is attributable to two possible factors: (a) upstream load imposed by TCP's ACK traffic and the resulting self-contention; and (b) TCP's congestion control algorithm potentially inhibiting the connection throughput. However, the experimental set-up is such that there is no contending traffic on the wireless legs, thus leading to the first factor as the dominating one. We refer to this cause as simply *ACK related self contention.* Although a typical TCP implementation sends an ACK for every two data packets, the ACK is significantly smaller with only a 20 byte IP payload as opposed to a 1480 byte IP payload for the data. *However, due to the overheads imposed by the 802.11 protocol, even small sized ACK frames end up contending on an equal footing to the data frames at the MAC layer.* This is because of both the byte overheads due to link layer headers and the preamble and other MAC operations such as inter-frame spacings and transmission of certain portions of the frames at base transmission rate.

We verify this self-contention hypothesis by explicitly sending bidirectional UDP traffic in the Wi-Fi network to mimic the behavior of TCP. We send 1464-byte UDP datagrams on the downlink, and 32-byte UDP datagrams on the uplink. As shown in Figure 2(a), we send one up to four ACKs per eight segments. The last case mimics self-contention of TCP ACKs, which reduces throughput from 20 Mbps to 15 Mbps in the 802.11g network. This is despite the fact that TCP ACKs take only 164 kbps. Other cases show that intermediate levels of self-contention also cause corresponding throughput reduction, and that TCP throughput can be increased by removing ACKs from the Wi-Fi network.

Selective Offloading: In this context, we propose an *offloading-ACK* mechanism to address self-contention in Wi-Fi networks. The key idea is to divert uplink ACKs to the 3G uplink to prevent them from contending with the downlink data in the Wi-Fi network[1].

[1]We address obvious issues such as impact of ingress filtering later in the paper.



(a) Impact from self-contention (b) Impact from blackout (c) Effect of random wireless losses

Figure 2: Motivation of *super-aggregation* for TCP

There are, however, two challenges that need to be addressed in order for *offloading-ACK* to be viable in a real environment: (i) The 3G uplink may not have sufficient bandwidth to send the required number of ACKs that will sustain the maximum TCP downlink throughput on the Wi-Fi network. The low uplink bandwidth has two impacts on TCP. ACKs may be dropped at the transmission buffer, which renders the TCP sender unable to increase its congestion window or have more bursty transmissions. (ii) The 3G link has a larger RTT, which increases the RTT observed by the TCP sender and hence slows down the growth rate of its congestion window and hence the overall throughput enjoyed by the TCP connection.

We address both the above challenges by adding intelligence to the offloading-ACK mechanism to better control *when the ACKs are offloaded* and *how many of the ACKs are offloaded*. First, offloading-ACK is performed only when the consequent RTT inflation does not adversely impact the growth of the congestion window. This occurs when the congestion window is large (and hence the connection throughput is less dependent on congestion window growth rate), which naturally is also when self-contention will be near its peak. A simple heuristic we use for the offload-ACK threshold is the *ssthresh* value that TCP uses in its congestion avoidance algorithm. The value is 20 segments in our prototyping. Secondly, offloading-ACK is performed only to that fraction of ACKs that are indeed sustainable by the low-bandwidth interface. The remaining ACKs are sent as-is through the default interface. ACKs in the beginning of a congestion window are preferably sent over the low-bandwidth interface as opposed to those toward the rear-end of the congestion window to offset the delay differences and also to mitigate any adverse impacts of out of order receipt of ACKs at the TCP sender[2].

We revisit other properties of the offloading-ACK mechanism and synergies with the other proposed mechanisms later in the paper.

3.2 Proxying - Overcoming Impact of Blackouts on TCP

Impact of Blackouts on TCP: A blackout for a wireless link occurs when the wireless channel experiences severe fading or the client undergoes a layer two or layer three handoff. TCP's performance is severely impacted by such blackouts, especially in vehicular Wi-Fi networks, as an experimental study shows average 75-second blackouts [6]. Figure 2(b) shows the state of a TCP connection during a blackout that occurs during a 2 second period. The TCP sender, unaware of the blackout, will lose all packets transmitted during the blackout and hence will experience a retransmission timeout. The sender subsequently will enter slow start and drop its congestion window to one. Since the TCP sender cannot know the exact time the blackout ends, it relies on retransmissions of the first segment in the congestion window followed by an exponential backoff in the RTO if no ACKs are received. It is very likely that when the blackout ends at the mobile device end, the TCP sender is still in the midst of waiting for the expiry of a now inflated retransmission timeout. This unnecessary idle period coupled with the TCP connection starting back from a congestion window of one and a very small *ssthresh* (due to the back to back timeouts) render the TCP connection crippled to a low throughput performance. In this particular example, the throughput of the TCP connection is roughly reduced by half because of the two second blackout. Note that any mechanism such as the mobile device gratuitously sending an ACK when the wireless interface comes out of blackout is unlikely to address all the above problems.

Proxying: In this context, we propose a super-aggregation technique called *proxying-blackout-freeze* in which the 3G link is used to notify the TCP sender about blackout events on the wi-fi link. The notification is in the form of a zero-window advertisement as if it were sent from the receiver with the wi-fi interface's IP address. The notification will freeze the TCP connection when blackout occurs. When the wi-fi interface recovers from the blackout, a resume notification in the form of a non-zero window advertisement is sent through the wi-fi interface. According to RFC 1122 [1], the TCP sender should enter persist mode upon receiving a zero-window advertisement. When it receives the window update at a later point it restarts sending segments from the first unacknowledged sequence number without reducing the congestion window or the slow-start threshold.

An important challenge that needs to be addressed is the actual blackout detection that must be done in real time and ideally with a low overhead. In the proposed super-aggregation technique, we implement a hybrid blackout detection mechanism to achieve a high responsiveness with a low overhead. An active link probing is performed when a passive link monitoring mechanism indicates that a blackout has likely occurred. The passive link monitor merely

[2]Note that a TCP sender that receives spurious ACKs (with sequence number less than a previously received ACK sequence number) will simply ignore the spurious ACKs.

tracks if any packet is seen at all on the wireless interface of interest. If no packets are seen for more than 200 ms, the wireless client sends an ICMP ping message to its default gateway on the wireless interface to verify if a blackout has actually occurred. The zero-window advertisement through the 3G interface is generated as soon as the blackout is verified. The monitor module continues sending of ICMP ping messages to the default gateway of the wi-fi interface until it receives a response to its query. As soon as recovery from the blackout is verified (through the receipt of response to the ping) a window update is sent to the TCP sender to resume the connection.

Another challenge is that TCP sender may not implement zero-window behaviors suggested by RFC 1122. For example, Linux (kernel 2.6.27) TCP implementation responds to zero window advertisement only when there is no outstanding packets. This can be resolved by combining the mechanism in next subsection. New ACKs are sent via 3G interface to make TCP sender believe all outstanding packets are received, and use *mirroring-loss-fetching* to recover those packets without impacting the performance on Wi-Fi.

3.3 Mirroring - Hiding Random Losses from TCP

Random losses: TCP is well known to suffer in the presence of random losses in wireless environments. Figure 2(c) shows the TCP throughput when introducing random wireless losses into the network. Even a 1% packet loss can reduce a TCP connection's throughput to less than 20% of the achievable performance. As identified by a multitude of prior works, the main cause of the degradation is TCP's interpretation of all packet losses to be due to network congestion and the consequent reduction of its sending rate by half. However, the above mentioned random wireless losses may occur due to a high bit error rate in the wireless network due to low signal-to-noise-ratios because of channel fading, large tx-rx distances, or interference.

Thus, if there is a reliable approach to distinguish congestion losses from random losses and have the TCP sender react only to congestion losses, considerable improvements can be achieved.

Mirroring: In this context we introduce a concept of *random-loss hiding* for TCP connections, wherein a loss classified as a random loss is not reported by the mobile device back to the TCP sender. If such losses are not reported (positive ACKs are sent as if such segments were received successfully) the TCP sender will not retransmit those segments thus compromising on the guaranteed reliability semantics. To facilitate such loss hiding without compromising on the reliability semantics the proposed super-aggregation technique establishes a *mirrored TCP connection* through the 3G interface with the goal of fetching only the segments lost due to random loss. Segments fetched using the mirrored connection are then inserted back into the byte stream of the original connection to fill the holes created by the random losses. Such loss hiding and lost segment fetching through the 3G interface can provide considerable benefits for the TCP connection established through the wi-fi interface. However, several key challenges need to be tackled. We now briefly elaborate on the challenges and the solutions proposed.

Loss distinction: How the proposed technique performs successful distinction of random and congestion losses is an important design element. However, note that the link layer at the mobile device will *receive corrupted frames* that it will then discard due to the errors unlike segments lost due to congestion that it will not receive in the first place. With an appropriate interface into the link layer the proposed super-aggregation technique gathers information on frames discarded due to corruption and consequently classifies losses as congestion losses or random wireless losses.

TCP connection mirroring: The mirroring of a TCP connection, in general, will require application layer knowledge. While performing such mirroring is one option, the proposed super-aggregation technique relies on a simple connection set-up replay for mirroring the TCP connection. In other words, the sequence of messages exchanged since the set-up of the original connection is replayed in order to mirror the original TCP connection. With typical Internet applications (http, ftp, smtp, cifs, p2p) we have considered thus far such a replay suffices.

Selected and Fast fetching: Since the 3G link has a considerably lower magnitude in terms of data-rate, a brute-force simple fetching of all the content just to retrieve the randomly lost segments is clearly not a viable strategy. Hence, the proposed mirroring mechanism performs selected and fast fetching, whereby the receiver proactively acknowledges segments that it does not need irrespective of whether it was received or not. Ideally, such segments will be purged from the TCP sender's buffer even before being transmitted by the sender. The only sequence numbers that the receiver does not acknowledge unless received correctly are the sequence numbers corresponding to the randomly lost segments. We place a guard time before fetching the desired segment to make the space for it to squeeze the narrow 3G link. The guard time is at least one segment size divided by data rate, and is configured as 256 ms in prototyping. We later show in the performance evaluation section on how such a fast and selected fetching scheme performs effectively. Note that some newer implementations of the TCP sender perform an explicit check for an incoming ACK sequence number being smaller than the right edge of the current congestion window. However, even under such conditions proactive ACKs paced correctly can appropriately accelerate the progress of the mirrored connection. However, some application level range-fetching (such as supported by http and ftp) would be required to eliminate redundant transmission of content on the mirrored connection.

Sequence number offsets: Since every new TCP connection establishment results in a new start sequence number, the mirroring mechanism appropriately offsets the sequence number of segments (or bytes) received on the mirrored connection to retrieve the sequence number pertinent to the original TCP connection.

3.4 Integrated Operations

The three mechanisms proposed for TCP acceleration can seamlessly work with each other. Moreover the traffic processing sequence would be as follows: *mirroring-loss-fetching*, *offloading-ACK*, and *proxying-blackout-freeze*. When segments arriving at the Wi-Fi interface have holes that have been attributed to random wireless losses, *mirroring-loss-fetching* hides them and generates a positive ACK for the latest segment. The generated ACKs are sent via the 3G interface using *offloading-ACK* mechanism. Finally, *Mirroring-loss-fetching* also establishes a mirrored connection to recover the randomly lost segments, and shares the 3G uplink capacity with *offloading-ACK*. If no packets are received for a period of time on the wi-fi interface, *proxying-blackout-freeze* detects the blackout and freezes the Wi-Fi TCP connection using a notification through the 3G interface.

Finally, each *super-aggregation* principle shown in this section does outperform simple aggregation in a setting with heterogeneous interfaces by leveraging the multiple interfaces intelligently. However, *super-aggregation* in principle may include as one of its mechanisms simple bandwidth aggregation as well especially when the interfaces have bandwidths of similar orders.

3.5 Super-Aggregation for Upstream Communication

Although the *super-aggregation* principles presented thus far were discussed in the context of downstream data transfer, they can be adapted for upstream data transfer in a straightforward manner. *Offloading-ACK* is realized by sending data packets on the Wi-Fi interface but labeling the 3G interface address as the source IP. Although data is sent on the Wi-Fi interface, ACKs will be received on the 3G interface to avoid self-contention. *Proxying-blackout-freeze* is similar to the downstream case, but zero-window advertisement and window updates are sent to the TCP sender in the local machine. *Mirroring-loss-fetching* for upstream traffic doesn't require establishment of a real mirrored connection. It merely retransmits random losses and masks the loss information from the local TCP sender.

4. SUPER-AGGREGATION ARCHITECTURE

4.1 Deployment Model

The *super-aggregation* principles presented in the last section can be implemented as a layer-3.5 software middleware in the mobile host. It can be implemented in the Linux kernel uses NetFilter to capture and process TCP packets traversing the network stack, or generate packets if necessary. The *super-aggregation* principles only require deployment at the mobile device and do not require any modification at the remote host or intermediate routers. The TCP implementations on the remote host and the mobile device are unaware of the *super-aggregation* principles that improve their performances transparently. With this deployment model, *super-aggregation* can enhance end-to-end performance of mobile host with any legacy TCP-based server.

4.2 Software Architecture

The implementation of each *super-aggregation* principle is divided into multiple components in the software architecture, as shown in Figure 3. We now briefly explain the components. *Offloading-ACK* is realized with the ACK Marker and Offloader components. *Proxying-blackout-freeze* is realized with the Blackout Detector and Blackout Handler. *Mirroring-loss-fetching* is realized with the Mirroring Manager, Loss Hider, and Fast Fetcher. Interface Characterizer is a common component used by *offloading-ACK* and *mirroring-loss-fetching*.

Figure 3 shows packet flow across the *super-aggregation* components. There are four types of TCP packets in the diagram: data packets and ACK of downstream traffic; data packets and ACK of upstream traffic. The four types of packets are illustrated with different arrows in the packet flow diagram. For ease of exposition we describe the *super-aggregation* operations in the order of upstream data (request from client), upstream ACK, downstream data (content from server), and downstream ACK.

Data of upstream traffic: Each packet generated by the TCP layer is recorded by the Mirroring Manager for establishing a mirroring connection on the 3G interface. The Mirroring Manager duplicates each data packet and puts the IP address of the 3G interface in the IP header. The data packets of the original connection are sent to the Wi-Fi interface. The duplicate packets are sent via the 3G interface in the same order to establish a mirroring connection and request for the same content from server.

ACK of upstream traffic: When a packet is received on the downlink of any wireless interface, it could be an ACK for previously sent data packets or data packets sent by server in response to user request. Both data packets and ACK packets may belong to the original connection or the mirroring connection. All packets received on any wireless interface are sent to the Mirroring Manager, which differentiate the four cases. It distinguishes packets belonging to the original connection and the mirroring connection by comparing destination IP addresses with the record of mirroring. Data packets and ACKs are differentiated from each other by calculating their TCP payload length. TCP ACKs belonging to the original connection are sent to TCP layer directly without any modification. ACKs belonging to the mirroring connection are processed by Mirroring Manager itself. It retransmits data packets if they are lost in the Internet so that requests on the mirroring connection is always the same as the original one.

Data of downstream traffic: Similar to ACKs of upstream traffic, downstream data packets belonging to the original connection and the mirroring connection are treated separately. The Mirroring Manager first checks if data seen on the original connection and the mirroring connection are identical. It sends notification to the Loss Hider to stop hiding loss if connection mirroring is not successful. Otherwise, the data packets belonging to the original connection are sent to the Loss Hider, and those of the mirroring connection are sent to the Fast Fetcher. Loss Hider sends all in-sequence data packets to TCP layer. If it finds a hole in the received data packets and the hole is attributed to a random wireless loss, the Loss Hider stores out-of-sequence packets in its buffer and notifies the Fast Fetcher the lost packets to fetch on the mirrored connection. It sends a TCP ACK for highest data sequence number it has seen in order to hide the losses from the TCP sender. The Loss Hider waits for the Fast Fetcher to recover the lost packets and delivers all recovered packets to TCP in sequence. Data packets belonging to the mirroring connection are sent to the Fast Fetcher. It sends packets lost on the original connection to the Loss Hider and ignores others. It uses the Fast Fetching technique to send ACKs to accelerate packet sending on the mirroring connection.

ACK of downstream traffic: ACKs of downstream traffic are generated by the TCP layer and several components, such as the Loss Hider and the Fast Fetcher. All those ACKs are sent to the ACK marker to perform appropriate *offloading-ACK*. It marks all ACKs as offloadable so that the Offloader knows they can be offloaded to 3G interface. If an ACK only contains cumulative information and can be replaced by newer ACKs, ACK Marker adds a mark of replaceable. The Offloader takes ACKs of both the original connection and the mirroring connection for *offloading-ACK* via 3G interface. If checks if the 3G interface has enough uplink capacity and if so offloads all ACKs to the 3G interface. If the 3G interface doesn't have enough uplink capacity for offloading, Offloader can discard replaceable ACKs without affecting TCP operations. Otherwise, it sends the remaining ACKs via the Wi-Fi interface.

Blackout handling: Blackout Detector on Wi-Fi interface monitors all activity events in Wi-Fi network to detect blackout and link recovery events. It sends probe message to default gateway if no activity is observed on Wi-Fi interface for a period of time. It sends blackout and link recovery notification to Blackout Freezer, which sends ACKs with zero-window advertisement and window update.. The ACKs are generated with latest ACK sent by ACK Marker, and have flow window size modified by Blackout Detector. The ACKs directly go through 3G interface since they are sent when Wi-Fi is in blackout.

Interface characterizer: All four types of packets traverse Interface Characterizer if they are sent through 3G interface. Interface Characterizer measures capacity on 3G uplink and downlink. It calculates remaining uplink/downlink capacity by subtracting data sent/received on 3G interface. Uplink and downlink capacity values are sent to Offloader and Fast Fetcher to let them respond to capacity decrease. Offloader reduces amount of ACKs sent to 3G interface if there is not enough uplink capacity. Fast Fetcher also increases the guard time to increase probability for lost packets to get through when 3G downlink capacity drops.

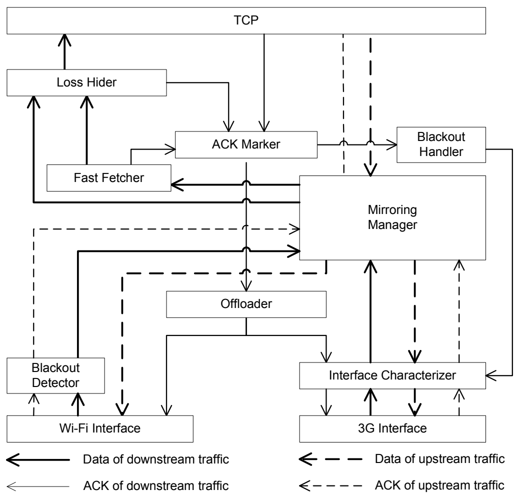


Figure 3: Data flow in *super-aggregation* architecture

5. SUPER-AGGREGATION BEYOND TCP

5.1 Generic Principles and Case Study

We now generalize the *super-aggregation* principles proposed in Section 3 and demonstrate their generic application with a case study of a non-TCP protocol. Each generic principle describes an approach of leveraging multiple interfaces of wireless devices that can increase throughput beyond the sum of the parts. We pick rate-adaptive video streaming [7] for the case study since it is a popular UDP application in the Internet. In such systems, the server sends video streams in the form of UDP datagrams to clients. To provide good video quality in response to capacity variation, the server adjusts its codec or sending rate based on available bandwidth to the client.

5.1.1 Selective offloading - tackling self-contention of client reports

Selective offloading chooses some packets to move from the Wi-Fi interface to the 3G interface. One design principle is moving some packets to the 3G interface to resolve self-contention in the Wi-Fi network, as in *offloading-ACK*. Moving small packets can provide significant improvements since overhead of sending them via the Wi-Fi interface is relatively higher. The other design principle is to use the 3G interface when overall performance is affected by some characteristics that the Wi-Fi interface performs poorer than the 3G interface. In rate-adaptive video streaming, clients keep sending reports of traffic characteristics to the sender to rate adaptation. The rate adaptation and overall throughput may be impaired by intermittent availability of the Wi-Fi interface, while 3G interface provides much higher availability. A video client on mobile host is unable to send reports during blackout and handoff. The server may interpret missing of reports as client disconnection or network congestion. Incorrect characterization causes improper rate adaptation of the video stream. *Offloading-report* moves report packets to the 3G interface with high availability enables continuous reporting, which allows server to do timely and accurate rate adaptation. It also reduces packet loss rate of video on downlink, since small report packets cause self-contention in Wi-Fi networks.

5.1.2 Proxying - improving reliability and timeliness of command packets

Proxying improves performance of the connection on Wi-Fi by masquerading packets from 3G, which serves as a proxy when Wi-Fi is temporarily unavailable. One design principle is to enable communication when the Wi-Fi interface has blackouts, as in *proxying-blackout-freeze*. Adding control packets via the 3G interface can help in preventing blackout's adverse effects to application. The other design principle is to improve reliability and timeliness of some packets by sending them to both interfaces. Heterogeneous interfaces provide diversity in packet losses, so sending a redundant packet to the 3G interface can effectively improve end-to-end reliability. Video streaming clients send command packets to perform control operations, such as pause/resume video delivery and updating configurations. Losing command packets degrades response time perceived by the client. *Proxying-redundant-commands* sends a duplicate copy of those command packets to improve the reliability. It improves response time to the client and may also improve other dimensions of performance since commands are delivered more timely.

5.1.3 Mirroring - reducing loss rate of baseline frames

Mirroring creates a independent connection on the 3G interface, and same or related content is downloaded to improve performance by leveraging loss diversity. One design principle is decoupling some of high-layer mechanisms to the mirroring connections, as in *mirroring-loss-fetching*. This helps the client to separate operation of two mechanisms that have adverse interaction in Wi-Fi networks. The other design principle is reducing packet loss rate by fetching redundant contents from both connections, especially essential portions in the original connection. Scalable Video Coding [15] is commonly considered in rate-adaptive video streaming since it encodes video content into different quality levels in a scalable way. All clients receive baseline frames and those with higher capacity also receive enhancement frames that rely on baseline frames. Baseline frames are more critical packets and requires less bandwidth than enhancement frames. *Mirroring-baseline-frames* establishes a mirroring connection via the 3G interface and requests for a baseline video stream of the same video content. Having duplicate baseline frames from server can significantly improve overall video quality, especially in a lossy environment.

5.2 Generic Architecture

We present the generalized software architecture in a modular fashion such that it is evident how to reuse the common components for generic *super-aggregation* principles. As shown in Figure 3, some components in the *super-aggregation* architecture are specific for TCP, while others provide common functionalities needed in other principles. For example, Offloader is used for *offloading-ACK*, Report Offloading, and Voice Offloading. *Offloading-ACK* uses ACK marker to notify Offloader that ACK packets should be offloaded. The Report Offloading and Voice Offloading should have their own component to mark report packets and voice frames. The Offloader takes all packets marked and split them according to available uplink bandwidth on 3G interface. Other common components include the Blackout Detector and the Interface Characterizer.

6. PERFORMANCE EVALUATION

In this section, we describe our prototyping on real multi-interface wireless devices and their performance in experimental field trials. We present performance of all *super-aggregation* principles on laptop in detail. Results on Google Android phone are summarized since they demonstrate similar performance.

6.1 Experimental Testbed

Figure 4 shows the topology of the experimental testbed. It includes both typical types multi-interface wireless device: laptop and smartphone equipped with Wi-Fi and 3G. The laptop is equipped with an Atheros 802.11a/b/g PCMCIA card and a Verizon USB727 EVDO stick. Its operating system is Fedora 9 with Linux kernel 2.6.27. The driver of the Wi-Fi interface and the EVDO interface are MadWifi 0.9.4 and wvdial, respectively. The smartphone is T-Mobile G1, which has embedded Wi-Fi and HSDPA interfaces. It runs with Google Android operating system and Linux kernel 2.6.25. The testbed includes a TCP server to provide bulk data transfer for downstream throughput measurement and has no background traffic. Connections last for 100 seconds in each experiment. It is a desktop with Fedora 9 and Linux kernel 2.6.27. Since original RTT between Wi-Fi interface and the TCP server is too small, we add a WAN emulator between TCP server and campus network. We conduct RTT measurement from the Wi-Fi interface to 20 popular websites, and the average value is 58.6 msec. The WAN emulator is configured with 50 msec RTT to make the testbed representative for multi-interface wireless devices.

6.2 Solution Prototyping

In order to evaluate *super-aggregation* performance with user-space implementation, we use a user-space TCP implementation so that *super-aggregation* can be realized below it. We select Atou (Almost TCP over UDP) [8], which is a validated tool for studying TCP performance. Both server and client program include bulk data transfer on top of Atou. Atou sends out segments and ACKs encapsulated in UDP datagrams, and TCP mechanisms are implemented based on standards. We use NewReno with SACK in Atou and verify that it gives almost the same performance as TCP (using iperf) as shown in the preliminary study in Section 3. Minor differences may appear between TCP and Atou for some scenarios. They may be caused by higher computation priority of TCP in kernel space or Atou's configurations not being exactly the same as that of a specific TCP implementation.

All three TCP-based *super-aggregation* principles are implemented in C and integrated with Atou. They are implemented only at the receiver-side source code and deployed to wireless clients. Sender-side codes are the original Atou, except some mechanisms are added for better TCP standard compliance (flow control and appropriate byte counting). We use gcc and ARM GNU/Linux cross-compiler to build the executable for laptop platform and Android platform, respectively.

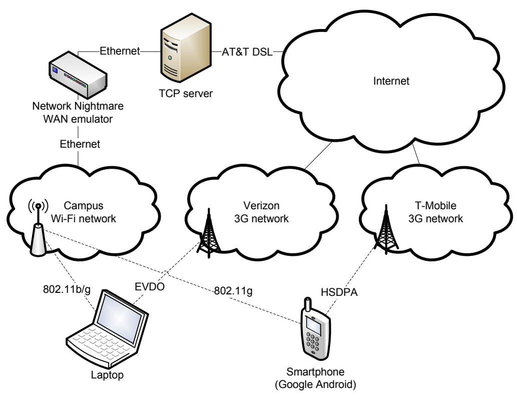


Figure 4: Topology of the experimental testbed

6.3 Offloading-ACK Performance

In experiments, *Offloading-ACK* resolves self-contention and improves TCP throughput with same magnitude as expected from the motivation. It improves TCP throughput by 37% and 152%[3] on the laptop client using 802.11g and 802.11b, as shown in Figure 5(a). Figure 5(b) is packet traces captured with tcpdump. It demonstrates self-contention of default TCP, in which data packets and ACKs don't overlap in time. It also shows capability of *Offloading-ACK* to allow TCP to fully utilize the 802.11g interface for downlink data. Although average throughput of *offloading-ACK* is affected by RTT of the 3G path, it is still able to achieve maximum throughput during a connection, as shown in Figure 5(c). It gives higher improvement magnitude when RTT on Wi-Fi path is longer since relative impact from 3G interface's

[3]To compare with TCP ACK aggregation techniques such as [3], we perform ACK aggregation (one for 12 packets) in 802.11g, and it only improves default TCP by 8.27% in experiments.

long RTT is smaller. The performance of integrated operations shows that *Mirroring-loss-fetching* can resolve the issue of long RTT on the 3G path.

6.4 Proxying-blackout-freeze Performance

Blackouts are generated every 20 seconds on average within the 100-second connection. Figure 6(a) demonstrates the effectiveness of *Proxying-blackout-freeze* by showing TCP states under blackout. TCP doesn't go to slow start or cut down congestion window when receiving zero-window advertisement via the EVDO interface. TCP throughput is also immediately resumed after receiving window update after link comes back. Compared to default TCP in Figure 2(b), *proxying-blackout-freeze* improves TCP throughput by 87%. Figure 6(b) shows that *proxying-blackout-freeze* gives close-to-ideal throughput with different RTT on Wi-Fi path, where ideal throughput values are calculated as throughput with no blackout times the ratio of link availability. Improvement magnitude is 161% when Wi-Fi RTT is 200 msec since TCP spends more time to recover its congestion window. Figure 6(c) shows that *proxying-blackout-freeze* gives more improvement with longer blackout duration. It improves TCP throughput by 136% when average blackout duration is 10 seconds.

6.5 Mirroring-loss-fetching Performance

Figure 7(a) shows that Fast Fetching technique is effective in fetching lost packets using the EVDO interface. It can recover lost segments 36 times faster than naive fetching with a normal TCP connection. As shown in Figure 7(b), *mirroring-loss-fetching* significantly outperforms default TCP. It keeps TCP throughput close to loss-less environment until loss rate is more than 3%. Figure 7(c) shows that *mirroring-loss-fetching* gives more improvement magnitude when the Wi-Fi path has longer RTT since TCP spends more time in recovering its congestion window. It improves TCP throughput by 175% when Wi-Fi RTT is 200 msec.

6.6 Performance of Integrated Operations

Figure 8(a) shows the experimental scenario for integrated operations. At first only Wi-Fi is available, then the client enters coverage of 3G network. It experiences two-second blackout when it moves out from first Wi-Fi network, across second one, and enters third Wi-Fi network. There is an interference sources in the third Wi-Fi network that causes one-percent random wireless losses.

Figure 8(b) shows instantaneous throughput of *super-aggregation* and default TCP as the client moves along. They have similar throughput when Wi-Fi is the only access at first, and *super-aggregation* starts to outperform default TCP once 3G is available because of *offloading-ACK*. Both throughputs drop to zero during blackouts, but *proxying-blackout-freeze* can immediately resume throughput while default TCP falls back to slow start. *Mirroring-loss-fetching* provides most significant improvement when random wireless losses happen in AP3. Overall, *super-aggregation* gives 189% improvement, which almost triples default TCP's throughput.

6.7 Performance on Google Android

Figure 8(c) shows throughput improvements of all super-aggregation principles on Google Android phone with 802.11g: 31% with *offloading-ACK*, 42% with *proxying-blackout-freeze* under 2-second blackouts, and 67% with *mirroring-loss-fetching* under 0.3% packet loss rate. Although throughput achieved on Android is typically lower than that of laptop, which may be due to different hardware capabilities, improvements from *super-aggregation* principles are significant.

7. RELATED WORKS

Simple aggregation mechanisms have been proposed in several related works. pTCP [10], WAMP [20], RMTP [13], MC^2 [18], and MAR [17] perform bandwidth aggregation at transport layer. R^{2}CP [12] and PRISM [11] also include mechanisms to address issues in wireless network, but they both require end-to-end deployment. They all provide linear throughput improvement by simply dividing traffic to multiple interfaces.

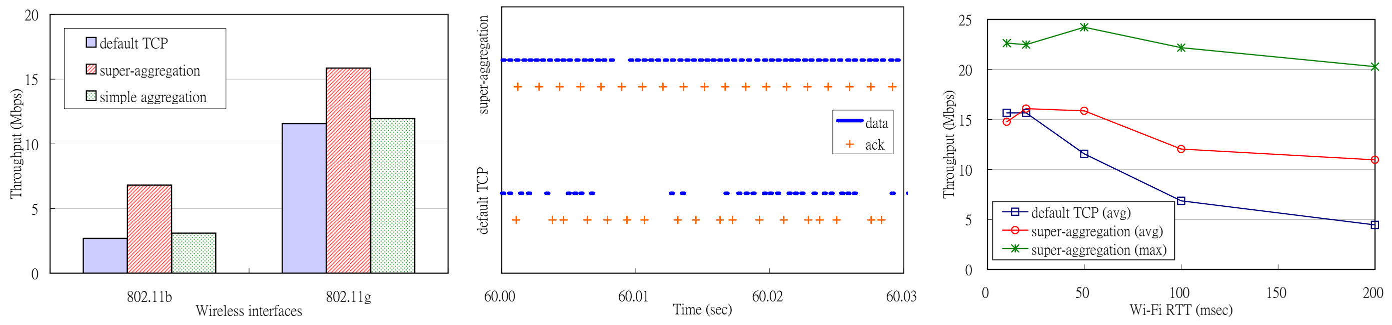
Some mechanisms are proposed to address similar issues of TCP over wireless networks. They typically consider just one wireless technology and cannot solve all issues with only client-side deployment. For example, Snoop [4] and WTCP [19] both address random wireless losses, but they are deployed at AP or both ends, respectively. Freeze-TCP [9] also uses flow-control mechanisms to freeze TCP connection before blackout happens. However, it requires appropriate TCP implementation on sender side and accurate prediction of blackout occurrence time to make the mechanism work. *Super-aggregation* principles leverage multiple interfaces to provide a more robust solution with simpler implementation.

Some mechanisms are proposed to use multiple interfaces to address issues other than throughput improvement. CoolSpots [14] uses Bluetooth-enabled access points to reduce energy consumption on wireless divides with both Wi-Fi and Bluetooth interfaces. Cell2Notify [2] uses cellular radio to wake up Wi-Fi interface on a smartphone when there is incoming traffic. It improves battery lifetime of VoIP over Wi-Fi by reduce energy consumption in idle time of Wi-Fi. Context-for-Wireless [16] proposes an adaptive radio selection strategy between Wi-Fi and cellular based on context.

8. ISSUES

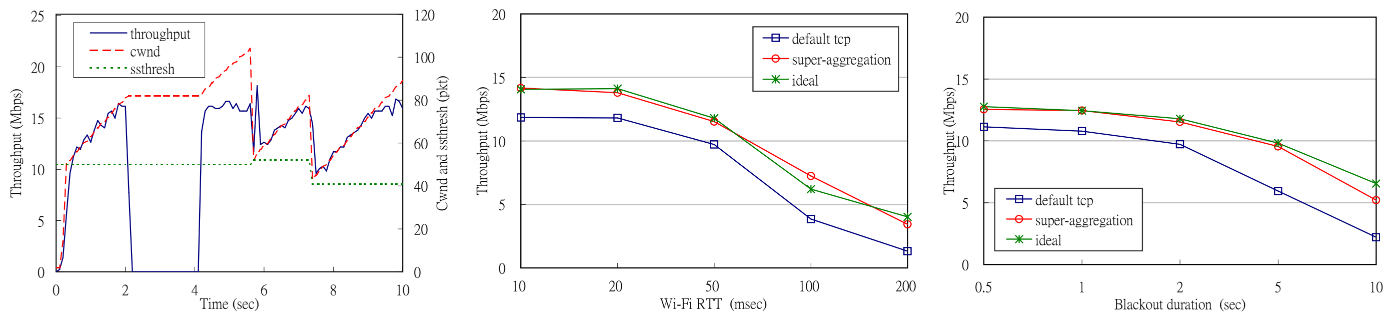
IP spoofing: *Offloading-ACK* and *proxying-blackout-freeze* require IP spoofing, which might be blocked by ingress filtering at routers. Experiments in [5] shows that around 25% of ASes allow spoofing, and 40% of clients in the Internet can spoof addresses up to a /8 netblock. There are two ways to support *offloading-ACK* and *proxying-blackout-freeze* when spoofed packets are blocked by ingress filtering. One is to tunnel packets to a proxy that can do IP spoofing. The proxy model is practical since forwarding light traffic for *offloading-ACK* and *proxying-blackout-freeze* can provide significant improvement. The other way is to introduce a new TCP option to allow sender to recognize other IP addresses belonging to the same wireless client.

Layer separation violation: Scope of *super-aggregation* principles span from link layer to transport layer, so they need to be carefully designed to not violating layer separation. We base the design of all three principles on standard operations and common properties of TCP, Wi-Fi, and 3G technologies. All principles work as enhancement to TCP with knowledge of underlying interfaces. It takes consideration to implement *super-aggregation* principles when functionalities in transport layer and link layer have non-standard implementation.



(a) Throughput improvement on 802.11b/g (b) Avoiding self-contention in 802.11g (c) Effect of RTT on the Wi-Fi path

Figure 5: Performance of *offloading-ACK*



(a) Freezing TCP during blackout (b) Effect of RTT on Wi-Fi path (c) Effect of blackout duration

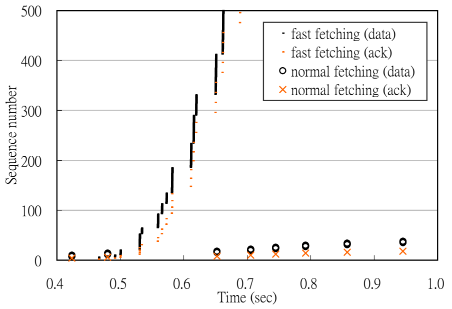
Figure 6: Performance of *proxying-blackout-freeze*

Extra resource consumption: The generic *mirroring* principle consumes extra resources when establishing the mirroring connection to server. The design of each *mirroring* principle should request for essential contents on the mirroring connection if possible. For example, *mirroring-loss-fetching* uses byte-range fetching technique if the functionality is supported by application-layer protocol of the traffic. This requires knowledge of application-layer functionalities and protocols, and it should be considered as add-on to principles implementation for better efficiency.

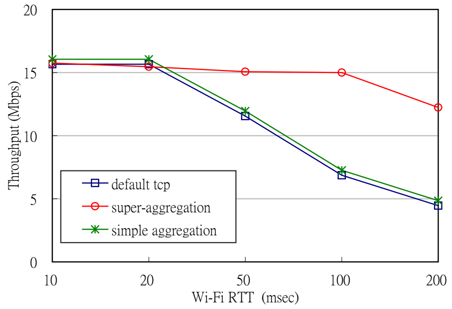
Connection mirroring: *Mirroring-loss-fetching* is mainly designed for state-less content services and should not be used in the following cases. If end-to-end semantics is critical for the application on top of TCP, hiding losses from sender side may cause inconsistency issues between two ends. It also assumes server to send identical data when receiving identical requests, but practical servers may give different response if the content provisioning includes randomness or other external information, such as system time. Besides, the mirroring connection cannot be established if the encryption used in the original connection have a different session key for each new connection. For those cases, *mirroring-loss-fetching* cannot be used and *super-aggregation* doesn't modify ACKs generated by the wireless client.

Extension to other wireless technologies: *Super-aggregation* can be applied to any combination of wireless technologies, as long as the interfaces exhibit heterogeneity in terms of three characteristics: capacity, connectivity, and loss rate. Since the different interfaces are likely to operate on different channels (to leverage the multiple interfaces in the first place) and hence will connect to different APs, they naturally will have uncorrelated connectivity and packet losses. Hence, there arise two possibilities based on whether or not there exists capacity heterogeneity: First, when two interfaces have heterogeneous capacities, the rule of thumb that must be applied is that the one with higher capacity acts as the primary interface. The other interface acts as the secondary to enhance performance through the *super-aggregation* principles. We do note that impact of the degree of heterogeneity on the performance gains with respect to simple aggregation is an interesting problem and something we leave for future research. Second, if the two interfaces have similar capacities, the interface with better connectivity is picked as the primary interface. The other interface is used as the secondary for the *super-aggregation* principles. Note that there may remain unutilized capacity on the secondary interface, and a simple aggregation technique can then be applied to use up the remaining bandwidth.

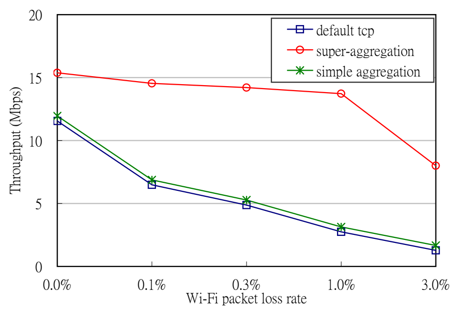
Thus, considering typical examples of recent wireless technologies such as 802.11n, WiMAX, 3G, and Bluetooth, we believe that the proposed principles will apply as-is to any combination of two heterogeneous wireless interfaces. For homogenous interfaces the applicability again is valid as long as capacity heterogeneity exists: for example, when two Wi-Fi interfaces on the same mobile device use different channels, and have different signal quality and hence data rates. If capacity is homogenous, such as two Wi-Fi interfaces with the same data rate, a combination of *super-aggregation* and simple aggregation will be required to maximize the throughput.



(a) Fast fetching

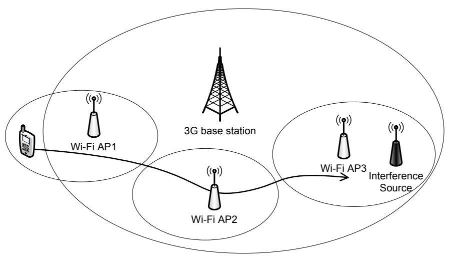


(b) Effect of packet loss rate

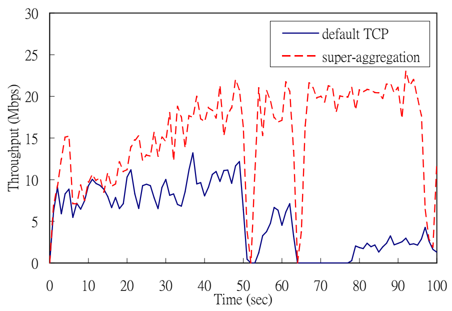


(c) Effect of RTT on Wi-Fi path

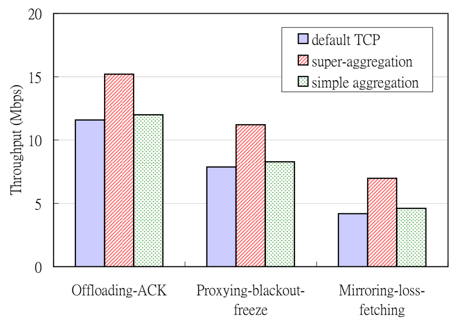
Figure 7: Performance of *mirroring-loss-fetching*



(a) Experimental scenario



(b) Performance of integrated operations



(c) *Super-aggregation* on Android phone

Figure 8: Integrated operations and performance on Android phone

Extension to three or more interfaces: Thus far in this paper, we have focused on the *super-aggregation* principles applying to only two interfaces. We now briefly discuss how the principles may be extended to apply to three or more interfaces: First, each mechanism is assigned to a different interface based on individual characteristics. Second, if an interface is underutilized, it is used to share the load of another interface. Finally, if any interface is still underutilized, simple aggregation is used along with *super-aggregation* to maximize throughput. This can be exemplified with a mobile device equipped with four interfaces using 802.11n, WiMAX, 3G and Bluetooth technologies respectively. For such a scenario, 802.11n will be selected as the primary interface for the highest capacity supported. WiMAX will then be assigned to *mirroring-lost-fetching* because of its relatively higher capacity. Bluetooth will be assigned to *proxying-blackout-freeze* for its short latency and low bandwidth, and 3G will be assigned to *offloading-ACK*. WiMAX may have extra uplink capacity since it is mainly used for downloading. It will then be assigned to share the loads of *offloading-ACK* on 3G. TCP ACKs with spoofed IP will be sent via both WiMAX and 3G, according to their uplink capacity. If WiMAX still has unutilized capacity, some data traffic will be split to it by using simple aggregation.

Battery lifetime: Although super-aggregation principles activate multiple interfaces simultaneously, we believe they don't consume more power than when using a single interface. Since throughput is improved by *super-aggregation*, a given amount of data will be transferred faster and energy saved with more sleep time. In this context, we have studied the energy consumed by downloading 10 MB of data on the Android phone with the following energy model, where the parameters are defined as follows: E_m^x is energy consumption per minute to maintain a connection on interface x (with or without power saving); E_t^x is energy consumption per byte transferred on interface x; T is end-to-end throughput; N_p and N_s are number of bytes transferred on the primary interface and that on the secondary one, respectively.

$$E = (E_m^{wifi} - E_m^{wifi-psm}) \cdot \frac{N_p}{T} + E_t^{wifi} \cdot N_p + E_t^{3g} \cdot N_s$$

Based on the empirical measurements of energy consumption in [16] (HTC Wizard in Table 2), *super-aggregation* (with the *offloading-ACK* mechanism) consumes 65.69 joules, which is better than that of default TCP (65.82) and simple aggregation (79.51).

9. CONCLUSIONS

In this paper we study if heterogeneous wireless interfaces can be aggregated with intelligent strategies to improve throughput beyond sum of the parts, as we call them *super-aggregation* principles. We propose three principles in the context of TCP that achieve *super-aggregation* benefits in Wi-Fi network when by adding a 3G interface with far smaller capacity. Each mechanisms are implemented and evaluated on laptop and smartphone platforms equipped both Wi-Fi interface and 3G interface. *Offloading-ACK* provides 30% throughput improvement by preventing self-contention in Wi-Fi network. *Proxying-blackout-freeze* uses 3G interface to maintain throughput of TCP during handoff between Wi-Fi APs. *Mirroring-loss-fetching* can improve

TCP throughput by 100% when one-thousandth random wireless loss occur in Wi-Fi network. We describe the generic software architecture that integrates all three principles and extends them for general traffic.

10. ACKNOWLEDGMENTS

This work was supported in part by funds from the Georgia Tech Broadband Institute and the NSF under grants CNS-0519733, CNS-0519841, and CNS-0721296.

11. REFERENCES

[1] Rfc 1122: Requirements for internet hosts - communication layers, 1989.

[2] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *Proceedings of ACM MobiSys*, pages 179–191, New York, NY, USA, 2007. ACM.

[3] E. Altman, I. Bp, and F. D. Ingeniera. Novel delayed ack techniques for improving tcp performance in multihop wireless networks. In *Proceedings of Personal Wireless Communications*, pages 23–25, 2003.

[4] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving tcp/ip performance over wireless networks. In *Proceedings of ACM MobiCom*, pages 2–11, New York, NY, USA, 1995. ACM.

[5] R. Beverly and S. Bauer. The spoofer project: inferring the extent of source address filtering on the internet. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI)*, pages 8–8, Berkeley, CA, USA, 2005. USENIX Association.

[6] 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 *Proceedings of ACM MobiCom*, pages 50–61, New York, NY, USA, 2006. ACM.

[7] P. de Cuetos and K. W. Ross. Adaptive rate control for streaming stored fine-grained scalable video. In *Proceedings of NOSSDAV*, pages 3–12, New York, NY, USA, 2002. ACM.

[8] T. Dunigan and F. Fowler. A tcp-over-udp test harness. Technical Report ORNL/TM-2002/76, Oak Ridge, TN, 2002.

[9] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments. In *Proceedings of IEEE INFOCOM*, volume 3, pages 1537–1545, Tel Aviv, Mar. 2000.

[10] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. *Wireless Networks*, 11(1-2):99–114, 2005.

[11] K.-H. Kim and K. G. Shin. PRISM: Improving the performance of inverse-multiplexed TCP in wireless networks. *IEEE Transactions on Mobile Computing*, 6(12):1297–1312, Dec. 2007.

[12] K.-H. Kim, Y. Zhu, R. Sivakumar, and H.-Y. Hsieh. A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces. *Wireless Networks*, 11(4):363–382, 2005.

[13] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *Proceedings of INCP*, pages 165–171, Nov. 2001.

[14] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *Proceedings of ACM MobiSys*, pages 220–232, New York, NY, USA, 2006. ACM.

[15] H. M. Radha, M. van der Schaar, and Y. Chen. The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP. *IEEE Transactions on Multimedia*, 3(1):53–68, Mar. 2001.

[16] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *Proceedings of ACM MobiSys*, pages 165–178, New York, NY, USA, 2007. ACM.

[17] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. Mar: a commuter router infrastructure for the mobile internet. In *Proceedings of ACM MobiSys*, pages 217–230, New York, NY, USA, 2004. ACM.

[18] P. Sharma, S.-J. Lee, J. Brassil, and K. G. Shin. Aggregating bandwidth for multihomed mobile collaborative communities. *IEEE Transactions on Mobile Computing*, 6(3):280–296, Mar. 2007.

[19] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Wtcp: a reliable transport protocol for wireless wide-area networks. *Wireless Networks*, 8(2/3):301–316, 2002.

[20] A. C. Snoeren. Adaptive inverse multiplexing for wide-area wireless networks. In *Global Telecommunications Conference, 1999. GLOBECOM '99*, volume 3, pages 1665–1672, Rio de Janeireo, 1999.