Multi-Interface Connectivity On Modern Mobile Devices

Varun Anand

Abstract—**Modern mobile devices are equipped with multiple network interfaces such as Wi-Fi, 4G, bluetooth, NFC and others. However, current software allows us to use only one of them at any given time. This is done by assigning a static priority to each interface, and enabling the one with the highest priority among the interfaces available. This causes unnecessary network disruption especially since the users could be mobile, and does not provide a seamless experience that is desirable.**

INTRODUCTION

Imagine Alice leaving her office after work. Her daily routine involves listening to her favorite podcasts on her drive home. However, she always forgets to download them in advance. On her way out, she starts downloading a few of these podcasts. But the moment she steps into the elevator, her Wi-Fi signal is lost. This breaks the previous network connection, requiring her to restart the download on her mobile (4G) data connection. This is both a waste of her data connection, and delays the download of Alice's podcasts significantly. As soon as she reaches home, her phone connects to her Wi-Fi network at home. This stops her downloads again, potentially requiring her to restart some of them. This scenario resulted in multiple disruptions to the downloads while all through Alice's phone had data connectivity via one or more network interfaces. The ideal solution for Alice's problem would be to make use of all or a subset of the available network resources as per Alice's choice to give her a seamless user experience.

THESIS QUESTION: Can we enable **concurrent** use of multiple network interfaces for a single connection on mobile devices in an **efficient** and **seamless** manner?

DESIGN OBJECTIVES

- **Restrict modification to the client system:** Our design should not require any changes at the server end.
- **Support existing apps:** We want to provide the same interface to the existing applications, so that they don't need to modify their implementation.
- **Seamless user experience:** As mentioned in the scenario above, users should be oblivious to the network dynamics, especially when they are mobile.
- **Efficacy of network usage:** We should be able to make the best use of available network interfaces i.e., ideally, our observed throughput should be the sum of the throughputs of the used network interfaces.

DESIGN

Our design was driven by the objectives listed above. We chose to incorporate our changes to the Android Open Source Platform. We also decided to take advantage of the HTTP protocol to enable multi-interface connectivity. The reason for these choices are described below.

HTTP Protocol

Recent studies have shown that HTTP protocol dominate today's data traffic on smartphones. HTTP plus HTTPS account for more than 80% of the traffic on smartphones [1] and are about 97% of the handheld traffic as observed by campus Wi-Fi networks [2]. Given the predominant use of the HTTP protocol, we decided that this would be an ideal layer in the network stack to incorporate our changes.

The HTTP protocol provides an option for requesting data in **byte-ranges.** We leverage this feature to concurrently maintain multiple open connections and use as many active interfaces available at a given time per download. Currently, we maintain one connection per interface. By utilizing existing features of HTTP (such as byte-range request) we restrict our modifications to the client.

Android

We demonstrate our ideas on the Android Open Source Platform. Android has grown in popularity over the last few years and has become the dominant mobile platform for smartphones and tablets. It is also open-source allowing us to make the changes required.

To achieve our goals, we had to make several modifications to the Android Framework. Listed below are the major modifications.

CONTRIBUTIONS

This thesis makes the following contributions:

- Android, like most mobile platforms, gives preference to Wi-Fi over cellular interfaces such as 3G or 4G. Our first modification was to make changes to the framework as well as set up per interface routing tables to enable the use of multiple network interfaces concurrently.
- We have made changes to the HttpUrlConnection library to allow it to utilize multiple network interfaces such as

Varun Anand is a graduate student with the Department of Computer Science and Engineering, University at Buffalo, SUNY, NY USA webpage: (see http://www.varun-anand.com). This report is a condensed version of the Thesis work defended on May 08, 2014

Thesis Advisors: Karthik Dantu, Steve Y. Ko and D. Koutsonikolas. E-mail: {kdantu,stevko,dimitrio}@buffalo.edu

wi-fi and cellular data connectivity simultaneously for a single request.

- In our intent to maintain the same abstraction to apps, we had to buffer downloads in the HttpUrlConnection library. As a consequence, our system provides the ability to resume stalled connections when all the network interfaces on a mobile device are unavailable intermittently.
- The use of HTTP range requests allows our system to break a download into smaller *chunks* that can be downloaded by multiple interfaces in parallel. Increasing the chunk size increases the latency observed by the app, particularly on slow connections. Decreasing the size increases the overhead observed by our system. We identified ***Wait Time*** as a tuneable parameter that balances this tradeoff and allows for network interfaces with disparate network speeds to download chunks in parallel while keeping the app observed latency constant.
- We also discovered overheads when we enabled byte range requests in HTTP. By utilizing the ***pipelining*** feature in HTTP, we eliminated this overhead and demonstrate efficient use of multiple network interfaces concurrently.
- With extensive experimentation, we study the utility of each of our modifications, and finally demonstrate concurrent usage of two network interfaces with negligible overhead.

Challenges

The challenges we faced could be broadly classified under two headings.

Managing multiple interfaces concurrently

Our implementation provides a mechanism to enable concurrent use of multiple network interfaces. We make use of multiple kernel routing tables and add new functionality to the `ConnectivityService` component of the Android Framework. Together, they observe the network dynamics and enable all the available network interfaces at any given time. Figure 1 shows the high level changes made to the `ConnectivityService` component.

Maximize bandwidth utilization using HTTP

Once we had the platform that could support multiple interfaces concurrently, we studied ways to utilize the interfaces efficiently. Figure 2 shows the high level changes made to the `HttpURLConnection` library.

Experimental Validation

Through extensive experimentation, we analyze the utility of all our changes and demonstrate that our system achieves the design objectives we listed initially.

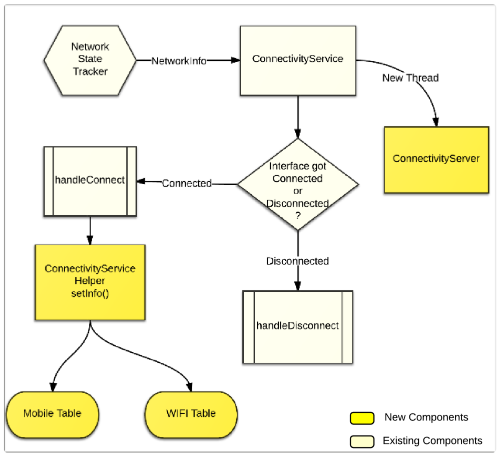

Fig. 1. Changes made to the `ConnectivityService` component

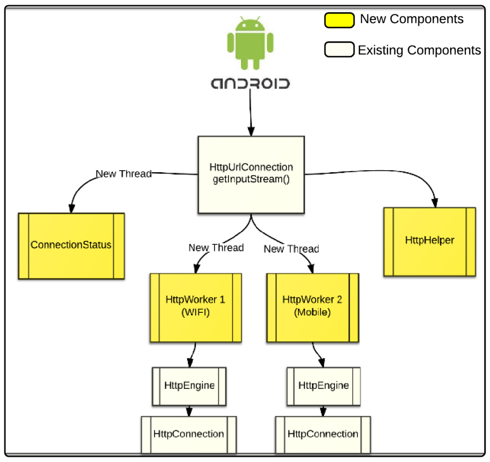

Fig. 2. Changes made to the `HttpURLConnection` library

Efficiency

Figure 3 shows the individual as well as the combined average throughput over 4G and WiFi interfaces. The results shown are averaged over twenty trials each. We downloaded a 100MB file, and shown in the figure are the average download speeds as well as the variance observed.

The combined throughput with both interfaces enabled is slightly greater than the sum of the individual throughputs observed over 4G and WiFi. Firstly, this demonstrates that we can utilize multiple network interfaces **concurrently, and with minimal overhead**. We attribute the slight increase in throughput to the large variance in the WiFi network.

Seamlessness

Figure 4 demonstrates seamlessness and resumability provided by our system. We downloaded a file of 500 megabytes

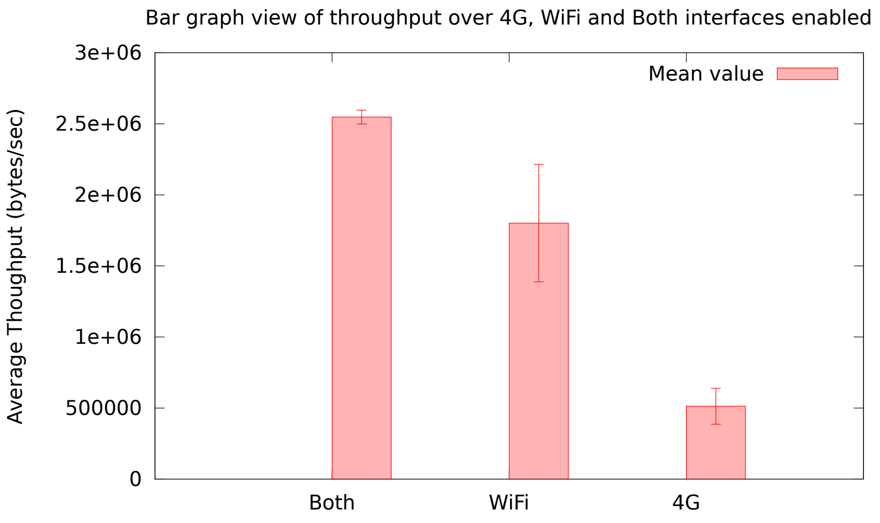

Fig. 3. Combined average throughput

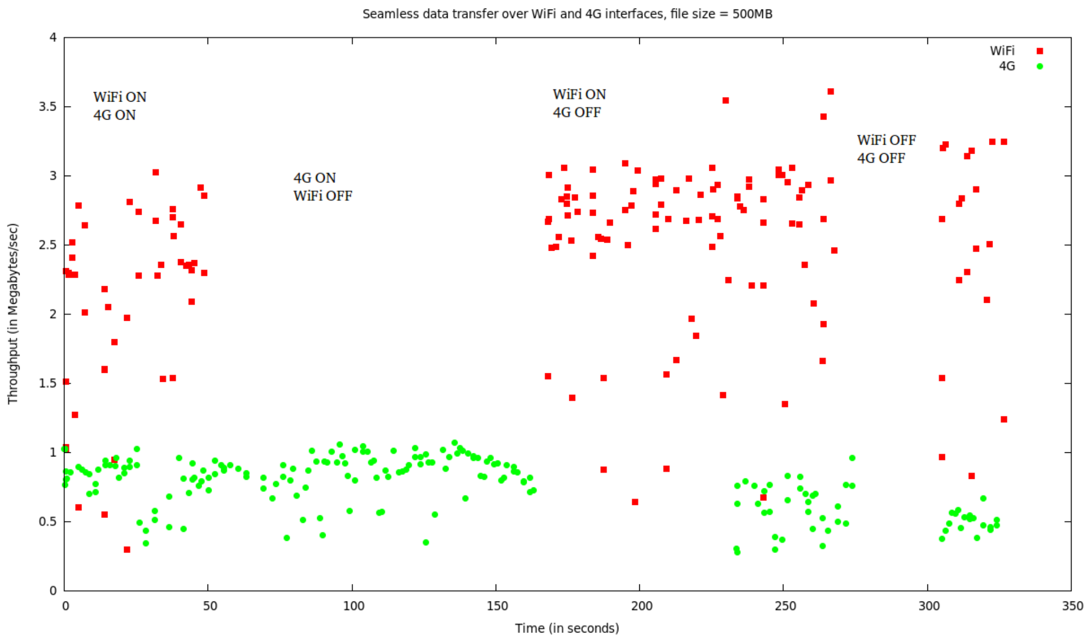

Fig. 4. Results for seamlessness with resume capability

over WiFi and 4G interfaces and plot the throughput over both the interfaces with time. We label 4 different regions according to the connectivity change observed by the mobile device. We clearly see the **seamlessness** provided by our implementation. Even though the network characteristics are changing, our implementation makes use of all available interfaces to download the file without the need for user intervention. Furthermore, when all the interfaces are down, our implementation buffers the previously downloaded portion of the file and waits until the network conditions change. It then resumes the download and completes it.

REFERENCES

[1] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, D. Estrin, *A First Look at Traffic on Smartphones*, In proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC'10.

[2] A. Gember, A. Anand, A. Akella, *A Comparative Study of Handheld and Non-handheld Traffic in Campus Wi-Fi Networks*, In proceedings of the 12th International Conference on Passive and Active Measurement, PAM'11.