Multi-Network Control Framework for Mobile Applications

Shuo Deng, Alvin Cheung, Sam Madden, Hari Balakrishnan
Massachusetts Institute of Technology

ABSTRACT

Mobile phones now come equipped with an assortment of sensors capable of generating data at high rates, as well as various wireless networks over which to transmit this data. In this paper, we introduce the design for a multi-network control framework for mobile applications. Our goal is to build a deployable framework that is compatible with current network architectures. It enables mobile application developers to use multiple networks easily and expressively via a comprehensive set of APIs for them to express their requirements. We also show some initial results demonstrating potential network performance (throughput, delay, etc.) improvement this framework can bring to mobile applications.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.3 [**Computer-Communication Networks**]: Network Operations—*Network Management*

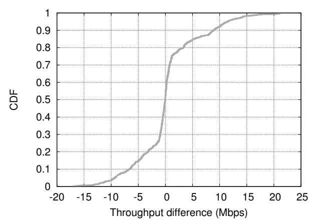
KEYWORDS

Multi-Network, Mobile Device

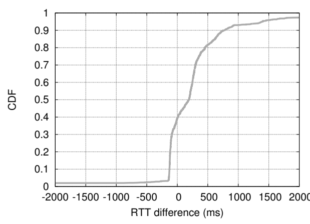
1. INTRODUCTION

Advances in mobile phones now allow phone users to capture a variety of information, such as high resolution videos, images, and sound. Mobile devices are increasingly used as generators of data in a wide range of applications, rather than mere consumers. Along with the increasing amount of mobile data generated and increasing needs of data sharing, mobile devices come equipped with multiple different networks, including wide-area cellular networks like LTE or 3G, and a variety of local-area wireless networks like WiFi in the 2.4 GHz and 5 GHz bands, WiFi-Direct (a "peer-to-peer" mode for WiFi that does not require transmissions via a separate access point), and Bluetooth.

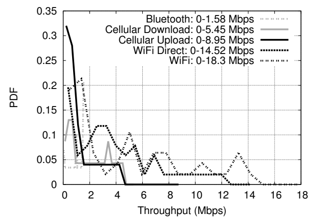
Unfortunately, two significant problems prevent today's applications from harnessing the potential benefits of these different network technologies to handle the requirements of data sharing and uploads. First, thanks to decisions made decades ago during the design of TCP and the standard socket API, mobile applications running over TCP (the common case) on stock mobile devices are able to use only one of these networks at a time and rarely have the ability to pick the network. Second, even if the applications can control multiple networks at the same time, there is no obvious solution as to *when to use which network(s)* because 1) the network condition varies over time, location, and load; 2) there is no simple and direct correlation between the applications' requirements. For example, "continuous connection", "minimize energy", and the network status information that the application can acquire. Thus, solving the problem of *when to use which network(s)* requires full knowledge of the network status as well as full control of multiple networks on a mobile device, which is beyond a single applications capability.



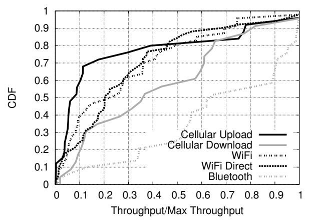
(a) WiFi and cellular throughput difference.



(b) WiFi and cellular RTT difference.



(c) PDF of peer-to-peer throughput.



(d) CDF of P2P throughput/max throughput.

Figure 1: WiFi vs Cellular Throughput measurement results.

In fact, multi-network coordination has been studied over the past decade[1–8]. But none of the proposed systems has been widely deployed onto commercial mobile devices, mainly for two reasons: 1) the system is not compatible with current network architecture – it either requires changes to network stacks or requires new hardware deployment; 2) each system tries to solve a specific problem, for example, minimizing energy consumption, or maximizing bulk transfer throughput, so that the system is not generalized enough to meet all possible requirements from applications.

In this paper, we introduce the design for a multi-network control framework for mobile applications. Our goal is to build a deployable framework that is compatible with current network architectures. It enables mobile application developers to use multiple networks easily and expressively via a comprehensive set of APIs. The rest of this paper is structured as follows. Section 2 shows measurement results that motivate the design of this framework. Section 3 describes the overview of the framework. To solve the problem of when to use which network(s), in Section 4, we develop two algorithms, which are evaluated in Section 5. Section 6 concludes and discusses future work.

2. MOTIVATION

In this section, we describe a series of real-world measurements on throughput and delay variation across networks in different environments. First, we measured TCP-send/-receive throughput and ping RTT between a server in our lab and smartphones. We car-

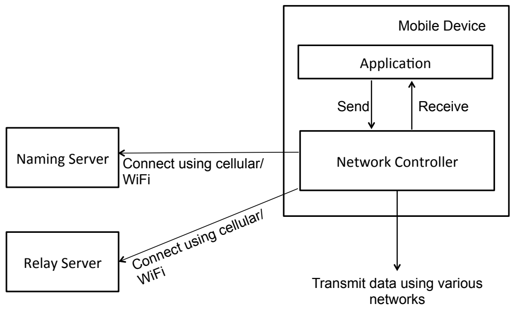

Mobile Device
Application
Send
Receive
Network Controller
Naming Server
Connect using cellular/ WiFi
Relay Server
Connect using cellular/ WiFi
Transmit data using various networks

Figure 2: Framework overview.

ried out the measurements using HTC EVO 4G LTE (Sprint), and Galaxy Nexus (Verizon) smartphones with Android firmware version 4.0 or higher, at 20 different locations where both WiFi and cellular network are available, including shopping malls, WiFi covered downtown area, and university campus. At each location, we measured the throughput for each direction (send/receive) and for each network (WiFi/Cellular) for 5 runs. Each run is a 30-second bulk TCP transfer. Before each TCP transfer, the phones ping the server to measure RTT. Figure 1(a) and 1(b) shows the CDF of the throughput and RTT difference between WiFi and cellular network. Difference less than zero means that WiFi throughput/RTT is less than cellular network throughput/RTT, and vise versa. For both figures, the parts of the curve far away from $x = 0$ corresponds to cases where throughput/RTT difference between WiFi and cellular network is big, and choosing the better network can help to improve the network performance significantly.

Then, we measured the throughput of peer-to-peer transfers between two phones using WiFi, Bluetooth, WiFi Direct, and cellular network. For each experiment with WiFi, Bluetooth, and WiFi Direct, we measured 1 minute TCP bulk transfer throughput. For cellular network, we measured the uplink/downlink TCP throughput from a mobile phone to our server. Figure 1(c) shows that there is no one single network that dominates the others in throughput. Meanwhile, Figure 1(d) shows that for each network, the measured throughputs are fairly evenly distributed between zero and the maximum observed throughput.

As indicated by the measurement results, there is a clear opportunity to improve network performance (e.g. throughput, delay) by enabling the phones to choose the best network for each data transmission.

3. SYSTEM OVERVIEW

To exploit the opportunities mentioned above, we built a multi-network control framework shown in Figure 2. The framework consists of two components: a network controller that runs on each device instance, and two remote servers that provide naming and relaying services. The network controller is responsible for initiating and receiving data transmission requests on the device. The naming server stores network information about each device that has the framework installed. The relay server is responsible for P2P data transmission when infrastructure-based (WiFi or cellular) networks are used.

An application running on top of the framework sends network requests by calling `send(UID,data,property)`. `UID` is a unique identifier, to be resolved by the Naming Server, that is assigned to each mobile device/server in the system. `data` is the piece of message, or a file to be sent. `property` is the network property required by the application. Our current framework provides the following properties: `continuous`, `min_energy`, `min_delay`, `min_data`, `max_tput`, and `delay_tolerant`.

4. ALGORITHMS

The framework selects the best network(s) for applications' network requests based on current network status. In this section, we describe two network selection algorithms to maximize throughput (i.e. `max_tput`) for client-server and peer-to-peer data transfer.

4.1 Client-Server: WiFi/Cellular Selection Algorithm

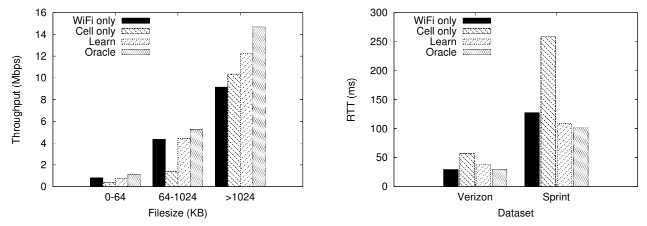
Our framework includes an algorithm to choose between WiFi and cellular network when applications need to send/receive data to a remote server. It uses a machine learning algorithm called "random forest" to decide which network to use based on information provided by the application and some light-weight measurement. We chose "random forest" because during our cross validation test, it outputs higher prediction accuracy than other tree-based learning algorithms.

When collecting throughput data for the 20 locations (see Section 2), we also collected information including data transfer direction, signal strength of WiFi/cellular network, cellular network type(GPRS/3G/LTE), ping RTT, DNS lookup time, available WiFi AP numbers, WiFi link speed, WiFi/cellular UDP throughput and loss rate before each TCP bulk transfer. We call this information "light-weight measurement". During the transfer, we ran `tcpdump` on both the phone and the server to keep track of the average throughput after 10KB, 20KB, 30KB, ... of data was transmitted. From this raw dataset, we generated 705,184 samples for Verizon network and 241,348 samples for Sprint. We fed 5% of the samples into the "random forest" algorithm to train the model. During the training process, we use *WiFi throughput higher than cellular network* as a binary output, and use different subsets of light-weight measurements as the input. We found using WiFi/cellular signal strength, data transfer direction, and file size as the input subset gives the highest accuracy. We used the "random forest" model trained from this input subset as our network selection model. The selection process works as follows: upon receiving a sending request from the application, if the framework confirms the destination UID is a remote server, it measures the signal strength of both WiFi and cellular network. Then our framework feeds the measurement results into the trained model. If the output is "yes", the framework chooses WiFi to transmit, otherwise chooses cellular network.

4.2 P2P Network Selection Algorithm

For P2P transfer, we developed an algorithm called "ShareWell" to maximize the overall throughput when multiple phones are transmitting data between each other, especially when some/all the phones are in close proximity. ShareWell works as follows. Suppose sender s would like to transmit data to a receiver r. After querying the Naming Server for available networks on r, s determines the network to use for transmission in the following manner:

1. **Broadcast intent to transmit.** For each network j that is available for transmission, s broadcasts a message querying the current average throughput from other send-receive pairs that are transmitting using network j. The message is received by others that are in close proximity, i.e., those that are connected to the same WiFi access point, or those that are in the same ad hoc network in the case of WiFi Direct.
2. **Probe to estimate available bandwidth.** After receiving answers from all pairs (or after a predefined time out of 100ms), s estimates the bandwidth available on each network. This is done by transmitting data on each network j for a short period of time (1 second in the current implementation). Meanwhile, other pairs that are transmitting on network j also measure their throughput during this probe period. The goal is to estimate the amount of throughput available to each pair assuming s will also transmit on the same network. We refer to this as the "shared throughput."



(a) Throughput (b) Delay

Figure 3: Median throughput and delay when using client-server network selection algorithm based on learning (marked as "learn") compared with using WiFi/cellular network only and oracle strategy.

3. **Determine the optimal network to transmit.** After collecting the shared throughput values from each available network, s performs the following calculation to choose the network to transmit. Suppose there are $n-1$ pairs of transmission on-going, and the nth pair with transmitter s arrives. Let $\text{tput}_{i,j}$ be the throughput for transmitting pair i's throughput in network j ($\text{tput}_{i,j}$ is set to 0 if pair i is not transmitting in network j), and $\text{tput}'_{i,j}$ be the "shared throughput" for a current transmitting pair i when s is probing network j. Then s will choose the network that maximizes the total throughput across all transmitting pairs, i.e.,:

$$\operatorname*{argmax}_k \left(\text{tput}_{n,k} + \sum_{i,j=k} \text{tput}'_{i,j} + \sum_{i,j\neq k} \text{tput}_{i,j}\right)$$

Data transmission commences after s chooses the network on which it decides to transmit.

4. **Re-probe when other transmission terminates.** Each sender broadcasts a message when it finishes transmission. If s receives such a message while transmitting, it repeats the probing process to determine if it should switch to another network to continue the rest of the transmission.

5. EVALUATION

We use the data collected at 20 locations to evaluate the WiFi-Cellular Selection Algorithm (Section 4.1). Figure 3(a) shows the result for Verizon dataset. The learning based strategy improves median throughput by 29% over Cell-Only, by 11% over WiFi-Only across different filesize ranges. For Sprint dataset (not shown in figure), our algorithm performs almost the same as the WiFi-only strategy and also close to oracle. This is because most of time during our measurement, the Sprint phone has only 3G available, and the 3G throughput is always lower than WiFi. We also used a similar learning process to learn which network to use to minimize delay (i.e., `min_delay` property). Here, we define the delay and the time between the phone sending out a TCP SYN packet and receiving a SYN ACK packet. Figure 3(b) shows for Verizon dataset, the learning based strategy reduces deley by 18 ms (32%) over Cell-Only, -9 ms (-32%) over WiFi-Only. For Sprint dataset, the learning based strategy reduces deley by 150 ms (58%) over Cell-Only, 19 ms (15%) over WiFi-Only.

To evaluate the P2P algorithm, ShareWell (Section 4.2), we implemented a time-slot-based simulator that randomly generates the amount of data to be transmit between pairs of phones and time when each pair starts transmitting. We then measured real-world throughput when three pairs of phones are transmitting using different networks and fed throughput values into the simulator. We compare ShareWell with other schemes, including WiFi 2.4GHz only,

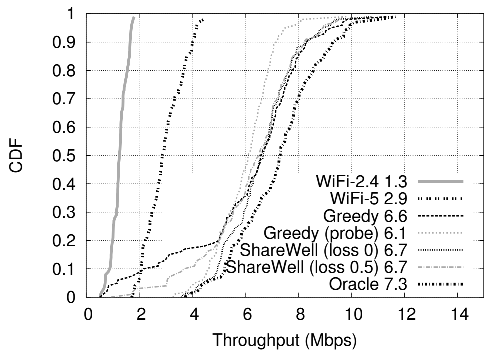


Figure 4: CDF of average throughput across 3 pairs of phones choosing among 3 networks (WiFi 2.4 GHz, WiFi 5 GHz and Bluetooth). The median value for each CDF is shown in the legend (in Mbps).

WiFi 5GHz only, and a greedy algorithm that the senders choose the best network by probing once (shown as Greedy in Figure 4) or probing periodically (shown as Greedy (probe)), without notifying other phones sharing the same network. For ShareWell, we considered cases when the notifications sent between phones are lossless (shown as ShareWell (loss 0)) and lossy (shown as ShareWell (loss 0.5) for lossrate=0.5). Figure 4 shows the average aggregate throughput for three pairs of phones. The lossless ShareWell achieves 5.1× gain over using WiFi 2.4 GHz only, and 2.3× gain over using WiFi 5 GHz only, and outperforms the greedy algorithm.

6. CONCLUSION AND FUTURE WORK

In this paper, we present a design of a multi-network control framework for smartphones. We also show some initial results demonstrating potential network improvement this framework can bring to mobile applications. We plan to focus on the following direction for future work: 1) algorithms on efficient network measurement and network selection for different network properties; 2) algorithms for global optimization when multiple applications are running simultanously on a single device require different (or even contrary properties); and 3) TCP/UDP based protocol for seamlessly network switching.

REFERENCES

[1] Y. Agarwal, T. Pering, R. Want, and R. Gupta. SwitchR: Reducing System Power Consumption in a Multi-Client, Multi-Radio Environment. In *Wearable Computers*, 2008.

[2] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath. Combine: Leveraging the Power of Wireless Peers through Collaborative Downloading. In *MobiSys*, 2007.

[3] P. Bahl, A. Adya, J. Padhye, and A. Walman. Reconsidering Wireless Systems with Multiple Radios. *SIGCOMM CCR*, 2004.

[4] C. Carter, R. Kravets, and J. Tourrilhes. Contact Networking: a Localized Mobility System. In *MobiSys*, 2003.

[5] R. Chandra and P. Bahl. MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a Single Wireless Card. In *INFOCOM*, 2004.

[6] B. D. Higgins, A. Reda, T. Alperovich, J. Flinn, T. J. Giuli, B. Noble, and D. Watson. Intentional Networking: Opportunistic Exploitation of Mobile Network Diversity. In *MobiCom*, 2010.

[7] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic. MultiNets: Policy Oriented Real-Time Switching of Wireless Interfaces on Mobile Devices. In *RTAS*, 2012.

[8] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces. In *MobiSys*, 2006.