Quality-Aware Traffic Offloading in Wireless Networks

Wenjie Hu and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
University Park, PA, 16802
{wwh5068, gcao}@cse.psu.edu

ABSTRACT

In cellular networks, due to practical deployment issues, some areas have good wireless coverage while others may not. This results in significant throughput (service quality) difference between wireless carriers at some locations. Through extensive measurements, we have validated the existence of such service quality difference. Then, through peer to peer interfaces such as WiFi direct, a mobile device (node) with low service quality can offload its data traffic to nodes with better service quality, to save energy and reduce delay. To achieve this goal, we propose a *Quality-Aware Traffic Offloading (QATO)* framework to offload network tasks to neighboring nodes with better service quality. QATO can identify neighbors with better service quality and provide incentive mechanisms to motivate nodes to help each other. To validate our design, we have implemented QATO on Android platform and have developed a web browser and a photo uploader on top of it. Experimental results show that QATO can significantly reduce energy and delay for both data downloading and uploading. Through trace-driven simulations, we also show that all users can benefit from data offloading in the long run.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*; C.4 [**Performance of Systems**]: *Measurement techniques*

Keywords

Data offloading; Energy Saving; Cellular Networks; Smartphone

1. INTRODUCTION

In cellular networks such as 3G, 4G and LTE, mobile devices are served by base stations in that area. Due to practical deployment issues, some areas have good coverage while others may not. As a result, the wireless signal strength of a mobile device varies based on its location. Moreover, the data throughput in an area also depends on the number of people in the area and the backhaul network of the wireless carrier [12]. When the service quality (in terms of signal strength and throughput) is low, it takes longer time to transmit the same amount of data and consumes more energy.

Some existing work has addressed the service quality difference at different locations. Schulman *et al.* proposed to defer data transmission to save energy when the service quality is poor [16]. However, this solution only works when it is known that the user will quickly move to a location with better service quality. There are also solutions on offloading cellular traffic to WiFi network to save energy and improve service quality [15, 10]. However, WiFi access may not always be available.

In this paper, we address the service quality difference from a different perspective. Through extensive measurements, we observe that mobile devices (nodes) within an area may have different service quality (e.g., a node may consume multiple times of energy and delay to download the same amount of data), especially when different service providers are used. Then, through peer to peer interfaces such as WiFi direct, a node with low service quality can offload its traffic to the node with better service quality, to save energy and reduce delay. Based on this finding, we propose a *Quality-Aware Traffic Offloading (QATO)* framework, where nodes with low service quality may offload their data traffic to those with better quality via the WiFi direct interface. QATO can identify neighbors with better service quality through service discovery, and provide incentive mechanisms to motivate nodes to help each other.

We have implemented the QATO framework on Android platforms. To evaluate its performance, we have developed two applications based on QATO: a Web browser application which is mainly used to test the performance of download offloading and a photo uploader application which focuses on upload offloading. Experimental results show that QATO can reduce energy by 38% in downloading and 70% in uploading, and reduce delay by 45% in downloading and 88% in uploading. Trace-driven simulations are used to evaluate the performance in a larger scale, and the results show that all nodes can save energy and reduce delay in the long run. Our contributions are as follows.

- We introduce the idea of leveraging data throughput difference between nearby nodes to save energy and reduce delay.
- We design a quality-aware traffic offloading framework to automatically detect neighbors providing the same service and offload traffic to nodes with better throughput. Also, we design proper incentive mechanisms to encourage users to help each other.
- We implement the framework on the Android platform and develop two applications to demonstrate its effectiveness.

The rest of the paper is organized as follows. Section 2 introduces the background of different cellular networks and their

MobiHoc'14, August 11–14, 2014, Philadelphia, PA, USA.

http://dx.doi.org/10.1145/2632951.2632954 .

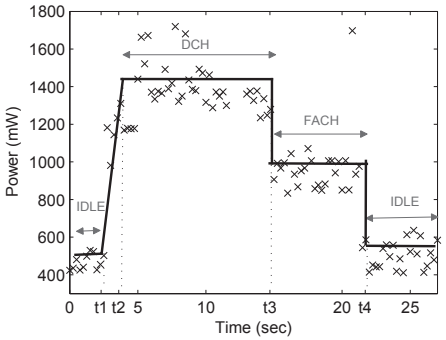


Figure 1: The power level of using the UMTS cellular interface (with screen on)

energy and delay model. Section 3 provides the motivation for quality-aware traffic offloading. We present the design and implementation of the traffic offloading framework in Section 4 and Section 5, respectively. The performance of QATO is evaluated in Section 6. Section 7 discusses related work, and Section 8 concludes the paper.

2. PRELIMINARIES

In this section, we first give a short description of different cellular networks, and then introduce the energy and delay model.

2.1 UMTS, HAPA+ and LTE Network

UMTS (the Universal Mobile Telecommunication System) is a popular 3G standard developed by 3GPP. Other than TDMA used by GSM, UMTS uses Direct Sequence CDMA (DSCDMA) technologies and provides a maximum bit rate of 384 Kbps to a single user at its first version, release 99. To support higher data rate, High Speed Downlink Packet Access (HSDPA) has been introduced in UMTS, which uses a new channel called the High Speed Downlink Shared Channel (HS-DSCH) and applies various techniques such as adaptive modulation, 16 QAM, and HARQ to provide a higher downlink data rate up to 14 Mbps. Later, High Speed Uplink Packet Access (HSUPA) was added as a complementary technique to improve the uplink speed. HSDPA and HSUPA are then merged and the enhanced version is called HSPA+, also referred to as 4G. In HSPA+, enhanced techniques such as 64 QAM, Multiple-Input Multiple-Output (MIMO), are used to increase the data rate up to 84 Mbps [21].

LTE (the Long-Term Evolution) is the latest extension of UMTS, and it enhances both the radio access network and the core network. The core network architecture of LTE is based on all-IP network and can support other non-3GPP radio access networks such as WiMAX and CDMA2000, which enables these networks to adopt LTE as their future radio access network. LTE can provide much higher bandwidth than 3G [8].

2.2 Energy and Delay Model

The power model of a typical data transmission in UMTS is shown in Fig. 1. Initially, a node stays at the IDLE state, which consumes little power. When a data request arrives, it promotes to the data transmission state (DCH), where data can be sent/received at high speed. After a data transmission, it stays at the TAIL state (FACH) for a period of time, in case another data request comes soon. In HSPA+ and LTE, the power models are similar. Therefore, we use a general cellular network power model presented by previous work [23].

The power consumption of the UMTS/HSPA+/LTE cellular interface can be generalized into three states: *promotion, data transmission* and *tail*, and the power consumption of these states are denoted as P_{pro}, P_{cell} and P_{tail}, respectively. The energy consumption of a task in cellular network can be modeled as follows. Suppose task T_i arrives at t_i with data size D_i, and the most recent task on the same node is T_j. The data throughput at the given node at t_i is r_{cell}. The time interval between task T_i and T_j is $\Delta t = t_i - t_j - D_j/r_{cell}$. There are three cases to compute T_i's energy consumption depending on Δt. 1) If Δt is larger than the tail timer t_{tail}, i.e., the cellular interface is in IDLE state before T_i arrives, then T_i will consume extra promotion and tail energy, besides the data transmission energy. 2) If Δt is smaller than t_{tail} but bigger than 0, there is tail energy but no promotion energy. 3) If Δt is smaller than 0, T_i will be overlapped with T_j for some time, and there will be no additional tail energy.

Table 1: Mobile Devices and Network Types

Device	Provider	Network[1]	Network[2]
Samsung Galaxy S3	Carrier 1	HSPA+	LTE
Samsung Galaxy S4	Carrier 2	LTE	LTE

[1] Network in City 1; [2] Network in City 2

$$E^i_{cell}(T_j) = \begin{cases} P_{pro} \times t_{pro} + P_{cell} \times \frac{D_i}{r_{cell}} + P_{tail} \times t_{tail}, & \text{if } \Delta t > t_{tail} \\ P_{cell} \times \frac{D_i}{r_{cell}} + P_{tail} \times \Delta t, & \text{if } 0 < \Delta t \leq t_{tail} \\ P_{cell} \times \max\{\Delta t + \frac{D_i}{r_{cell}} - t_j, 0\}, & \text{Otherwise} \end{cases} \quad (1)$$

The delay to complete a task is also related with Δt. If Δt is smaller than t_{tail}, the delay is only the data transmission time. Otherwise, the delay should include additional promotion delay, as shown in Eq. 2.

$$d^i_{cell}(T_j) = \begin{cases} D_i/r_{cell} + t_{pro}, & \text{if } \Delta t > t_{tail} \\ D_i/r_{cell}, & \text{Otherwise} \end{cases} \quad (2)$$

3. THE MOTIVATION FOR QUALITY-AWARE TRAFFIC OFFLOADING

In this section, we introduce our method to measure the service quality (mainly in terms of data throughput), and give the motivation of quality-aware traffic offloading.

3.1 Measurement Methodology

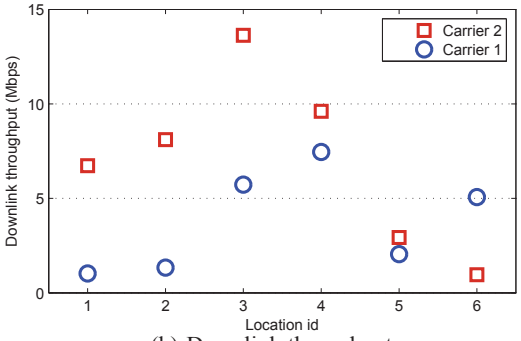
We use two types of smartphones (Samsung Galaxy S3 and S4) to measure the throughput of two cellular providers (denoted as *Carrier 1* and *Carrier 2*) at two cities (denoted as *City 1* and *City 2*). A brief description of the devices and networks are shown in Table 1.

To measure the throughput, we ported *iperf* to smartphones and added timestamp to record the start and end of a data transmission and extended it to support WiFi direct. With iperf, the smartphone establishes a TCP connection to our backend server and measures the downlink and uplink throughput for 30 seconds, respectively. To measure the power consumption, we use the Monsoon power monitor to provide power supply for smartphones, which can provide constant voltage and measure the current at a rate of 5000Hz. Based on the power measurement trace and the start/end time from

Table 2: Power consumption of different networks

	State	Power (mW)	Duration (s)	Download throughput (Mbps)	Upload throughput (Mbps)
HSPA+ (Carrier 1)	Promotion	1422.2 ± 34.1	2.3 ± 0.5	-	-
	Data	1990.9 ± 44.2	-	4.5 ± 1.3	1.3 ± 0.2
	Tail	1622.6 ± 39.6	11.4 ± 1.4	-	-
LTE (Carrier 1)	Promotion	1214.8 ± 24.3	0.25 ± 0.4	-	-
	Data	1865.8 ± 25.6	-	51.11 ± 16.9	17.9 ± 5.1
	Tail	1125 ± 22.3	11.5 ± 0.56	-	-
LTE (Carrier 2)	Promotion	1567.9 ± 47.8	0.34 ± 0.04	-	-
	Data	2224.7 ± 53.1	-	15.4 ± 5.5	5.3 ± 2.5
	Tail	1757.5 ± 97.5	3.37 ± 0.08	-	-
WiFi direct	Data	1323.6 ± 13.9	-	29.5 ± 1.2	29.5 ± 1.2

(a) Locations



(b) Downlink throughput

Figure 2: Downlink throughput in different locations. There are coverage blind spots in location 1 and location 2 for Carrier 1, and location 6 for Carrier 2, which motivates the use of traffic offloading to improve service quality.

the iperf trace, we can get the power consumption at different network state. The results are shown in Table 2, where the power consumption is measured as the whole phone's power consumption when the screen is on.

3.2 Understanding Quality Difference between Different Carriers

3.2.1 *Micro Perspective: Coverage Blind Spots of Different Carriers*

Within the coverage of a cellular base station, the data rate within an area varies greatly due to many reasons, such as the distance from the base station, obstacles on the way, interference from other devices, etc. It is common that one carrier may have some coverage blind spots at some locations, which means that the throughput is extremely low and the quality of experience is poor. This problem is especially worse in indoor environments.

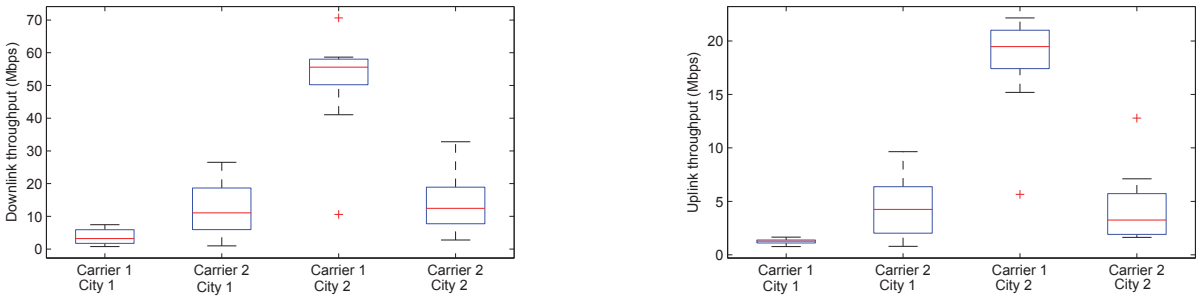
We picked 6 popular locations in our university, including Lab, library, classroom, and Cafeteria, as shown in Fig. 2(a). In each location, we measured the data throughput of different carriers at the same time. The comparison results are shown in Fig. 2(b). As can be seen, both carriers have high throughput to satisfy users' requirements in locations 3 and 4, but low throughput in location 5. Here we are more interested in other locations. In locations 1 and 2, Carrier 1 has extremely low data throughput but carrier 2 has much better service quality. Similarly, in location 6, Carrier 2 has very low data throughput but Carrier 1 has better service quality. In these blind spots, there is strong motivation for nodes to offload traffic to neighbors using different carrier. For example, at location 6, nodes served by Carrier 2 have motivations to offload traffic to nodes served by Carrier 1 to save energy and reduce delay.

3.2.2 *Macro Perspective: Quality Complementary between Carriers*

Different carriers have different priorities when deploying cellular networks in different cities. One may provide better service quality in one city but lower quality in another city, while the reverse is true for another carrier. From the macro perspective, different carriers may complement each other in many cities, and provide opportunities to offload traffic to neighboring nodes with better service quality.

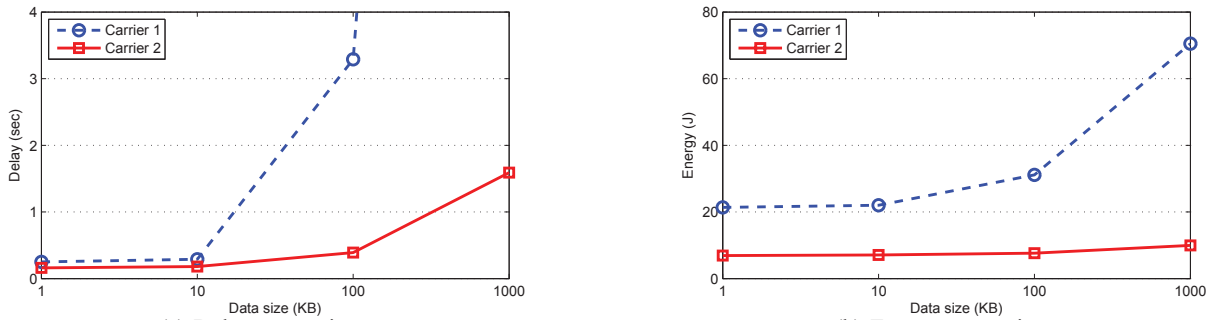
To verify this hypothesis, we collect data throughput of two carriers in two cities, as mentioned in Table 1. In each city, we collect the throughput in multiple locations at different time. The downlink and uplink throughput of both carriers in these two cities are shown in Fig. 3(a) and Fig. 3(b), respectively. In the boxplot figure, the middle line of a box indicates the median, and the lower and upper side of the box are the first (25%) and third (75%) quartile, which are denoted by $Q1$ and $Q3$. The outliers are values outside $1.5 \times (Q3 - Q1)$ range above $Q3$ or below $Q1$.

Both downlink and uplink throughput show the same trend. Carrier 1's HSPA+ network provides lower service quality than Carrier 2's LTE network while Carrier 1's LTE network outperforms that of Carrier 2. Thus we have two findings. First, there



(a) Downlink (b) Uplink

Figure 3: Throughput of different carriers in City 1 and City 2. Carrier 2 provides better service quality in City 1 while Carrier 1 provides better service quality in City 2.



(a) Delay comparison (b) Energy comparison

Figure 4: Energy and delay comparison between different carriers with different throughput.

is high throughput difference between carriers, which can be as much as *eight times*. Second, the service quality provided by different carriers varies in different cities and complements each other, which provides opportunity and motivation to help each other. For example, suppose a group of commuters using these two cellular carriers periodically travel between these two cities, then users of Carrier 2 may share their services with users of Carrier 1 in City 1, while utilize the service from Carrier 1 in City 2. Thus, with data offloading, all users can benefit in the long run.

3.3 Delay and Energy Comparisons

Under different service quality, the data access delay and the energy consumption to accomplish a network task are different. To measure this difference, we use GS3 with HSPA+ data plan from Carrier 1 and GS4 with LTE data plan from Carrier 2 as mentioned in Table 1 to download a given size of file from our server in City 1. The delay is the time from the start of downloading a file to the time when the file is completely received. The energy consumption is measured as the whole phone's energy consumption during downloading (including tail energy) when the screen is on.

Figure 4 shows the results. As can be seen in Fig. 4(a), the data access delay is similar for both carriers when the data size is small. As the data size increases, the data access delay with Carrier 1 increases much faster than that of Carrier 2. This result is consistent with our experience in real life; i.e., when updating emails or browsing simple webpages, LTE feels the same as 3G network; however, when watching movies, LTE significantly outperforms 3G. This result indicates that offloading large data to nodes with better quality can significantly reduce the delay.

Figure 4(b) compares the energy of data transmission with different service quality. Since Carrier 1 takes longer time to transmit the data, and has larger promotion and tail energy, it consumes much more energy than Carrier 2. Fortunately, the energy consumption does not increase linearly with the data size. For example, when the data size increases from 10KB to 1000KB, the energy only increases 3.2 times for Carrier 1 and 1.4 times for Carrier 2. It means that aggregating a large amount of data on a node with higher throughput does not increase the energy too much.

4. QATO DESIGN

In this section we introduce the design of our data offloading framework QATO. We first give an overview of QATO and then describe its major components.

4.1 QATO Overview

Consider that a group of users stay together for a relatively long time, such as in a Lab or on a commuting bus from one city to another. They use different cellular carriers and are willing to share data services with others to trade for better service from others at a later time. In such scenarios, QATO enables phones to offload their network tasks to a neighboring node with better service quality. Here a network task means an independent task to fulfill one user request, such as downloading one webpage or uploading one photo, which may contains lots of data packets.

The architecture of QATO is shown in Fig. 5. In the original smartphone system, all network tasks are buffered in the local queue and then scheduled based on the FIFO order. With QATO, the network tasks are guided into the offload engine module first. By taking into account the local network information and neighbor

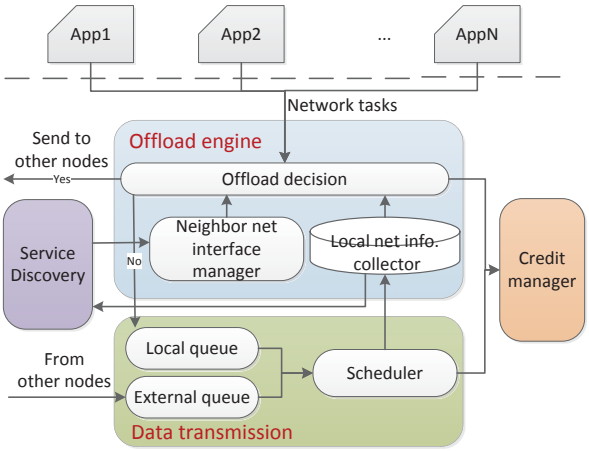


Figure 5: QATO architecture

network information collected by the service discovery module, the offload engine module determines whether to offload the network tasks and to which node to offload. The data transmission module maintains a local task queue and a remote task queue, and properly schedule them to reduce energy and delay. Since users may be selfish, we also design a credit based mechanism to motivate users to share their cellular service. The credit manager module manages credits; i.e., collecting credits from nodes using the service and pay credits to nodes providing the service.

4.2 Service Discovery

Detecting neighbors nearby has been studied in recent works, but most of them are based on Bluetooth interface [7]. To the best of our knowledge, QATO is the first work to discover neighbors using the WiFi direct interface. Moreover, we also need to know which neighbors support QATO, i.e., are willing to offload traffic for others. To solve this problem, we introduce the DNS (Name Domain System) based service discovery (DNS-SD) [2]. It allows nodes to discover neighboring nodes supporting a specific service within the local network. Different from the traditional DNS service that maps host name to IP address, DNS-SD can be used to find neighbors without support of the central server.

Android begins to support DNS-based service discovery since Android 4.1 (Jelly Bean), and it supports DNS-SD to be deployed on WiFi direct interface, without connecting to any Access Point. On each node, we register "QATO" as a service, with "_http._tcp" as the service type. A local port is also assigned for this service. After successful registration, the node will be able to respond to the "QATO" requests from neighbors. In the response, it also includes its IP address and port number. Based on these information, two nodes can create connection via the WiFi direct interface and then exchange the network quality information. The network quality information is organized in an XML based profile, which contains cellular network information such as cellular network type, signal strength and downlink/uplink throughput, and task related information such as the local task arriving rate, remote task arriving rate and total remote task size. Some frequently used notations are listed in Table 3.

Table 3: Notations

r_{p2p}, P_{p2p}	Throughput and power of the P2P network
t_{pro}, P_{pro} *	Promotion delay and power of cellular network
t_{tail}, P_{tail} *	Tail time and power of cellular network
r_{cell}, P_{cell} *	Throughput and power of cellular network
λ_l, λ_r *	Data arriving rate of local and remote task queue
S_r, $\overline{D_r}$ *	Total and average data size in remote task queue
Γ	Data size threshold for scheduling remote tasks

*: notations with superscript p indicate the corresponding value of the proxy node

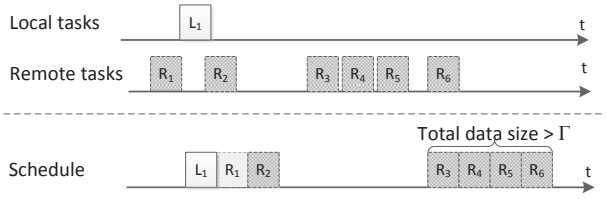


Figure 6: The scheduling of local tasks and remote tasks

After service discovery, a node collects a list of network quality profiles from neighbors. Then, the neighbors with higher data throughput than the node itself are recorded in a *proxy list*, i.e., the potential nodes to offload network tasks for this node. To save energy, service discovery is run when a node first comes to a new location and has no neighbors. Later on, it is only executed when the selected proxy disappears or when the proxy's service quality becomes worse than itself.

4.3 Data Transmission Schedule

In QATO, a node maintains two task queues, local task queue Q_l and remote task queue Q_r, to store the network tasks generated locally and received from neighbors, respectively. During data transmission schedule, i.e., task schedule, we consider two factors. First, local tasks should not be affected by remote tasks. Second, remote tasks should be scheduled only when their introduced tail energy is negligible. To solve this problem, we design a scheduling algorithm for the proxy node, as shown in Algorithm 1. A local task is scheduled when it is generated since it has higher priority. For a remote task, it is scheduled based on the following two cases to save energy:

- **Case 1:** A remote task is executed when the cellular interface is already in the data transmission state, either triggered by local tasks or by previous remote tasks. In this case, we record the tail ending time. After scheduling a task, the tail ending time is also extended.

- **Case 2:** Remote tasks are executed in a bunch when the accumulated data size is larger than a threshold Γ. This ensures that remote tasks are scheduled when there is no pending local task.

An example is shown in Figure 6. Remote task R_1 comes first but is not scheduled immediately. Local task L_1 is scheduled when it is generated. After that, R_1 is scheduled as the cellular interface is on the data transmission state, as illustrated in Case 1. The same happens to R_2. When R_3 comes, the cellular interface is moved to the IDLE state, and it waits for future tasks. When R_6 comes, the accumulated remote tasks are more than Γ, and these four tasks are scheduled together, as discussed in Case 2.

The selection of Γ affects energy efficiency, in terms of energy per byte. Larger Γ means more tasks are scheduled together

Algorithm 1 Task Schedule on Proxy Node

```
function TaskSchdule(Q_l, Q_r)
    S_r ← 0, TailEnd ← 0
    order all tasks by task time
    for each task T_i ∈ Q_l ∪ Q_r do
        if T_i ∈ Q_l then
            ExecuteTask(T_i)        /* Get T_i by cellular interface */
            Q_l ← Q_l \ T_i          /* Remove T_i from local task queue Q_l */
            TailEnd ← t_i + t_pro + D_i/r_cell + t_tail
            ExecuteRemoteTask_Case1(Q_r, TailEnd)
        else
            S_r ← S_r + D_i
            if S_r > Γ then
                ExecuteRemoteTask_Case2(Q_r, S_r)
                S_r ← 0
            end if
        end if
    end for
end function

function ExecuteRemoteTask_Case1(Q_r, TailEnd)
    for T_j ∈ Q_r do
        if t_j < TailEnd then
            ExecuteTask(T_j)
            TailEnd ← t_j + D_j/r_cell + t_tail
            Q_r ← Q_r \ T_j
            S_r ← S_r − D_j
        end if
    end for
end function

function ExecuteRemoteTask_Case2(Q_r, S_r)
    for T_j ∈ Q_r do
        if S_r > 0 then
            ExecuteTask(T_j)
            Q_r ← Q_r \ T_j
            S_r ← S_r − D_j
        end if
    end for
end function
```

to amortize the tail energy, and therefore improves the energy efficiency, but it also increases delay. To find out the proper Γ value, we upload different size of data using both carriers in City 1. As shown in Fig. 7, when the data size is larger than 500KB, there is a big drop of energy per byte for both carriers. The same trend exits during downloading. Thus, it is better to set Γ bigger than 500KB.

4.4 Offload Engine

As mentioned before, each node maintains a proxy list after service discovery. When a network task comes, the offload engine will determine whether to offload the task. If so, it selects the proxy node p from the list considering both energy and delay.

4.4.1 Energy Consideration

Suppose task T_i is generated at time t_i, with data size D_i. If this task is executed locally, the energy consumption E^i_{local} can be computed using Eq. 1. Otherwise, if this task is offloaded to p, the data transmission energy should be the data transmission energy on node p, plus the additional energy of the P2P interface. Considering the two cases of scheduling a remote task, in Case 1, there is no additional promotion energy and tail energy for T_i; in Case 2, T_i shares part of these energy proportional to its size. As we are not sure when the future tasks will be scheduled, thus the total energy is computed based on the worse case, as Eq. 3.

$$E^i_{remote} = \frac{D_i}{r_{p2p}} \times P_{p2p} + \frac{D_i}{r^p_{cell}} \times P^p_{cell} + \frac{D_i}{\Gamma} \times (P^p_{tail} \times t^p_{tail} + P^p_{pro} \times t^p_{pro}) \quad (3)$$

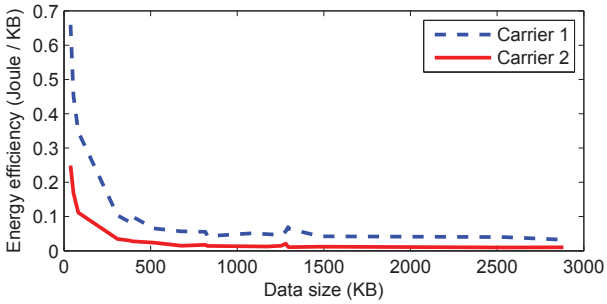


Figure 7: The selection of Γ

4.4.2 Delay Consideration

If task T_i is executed locally, the delay d^i_{local} can be computed using Eq. 2. When offloaded to proxy node p, the delay contains four parts: the time to execute the first task in the remote queue, queue delay, the time to get T_i via cellular network and the time to transmit it back via the P2P interface. Similarly there are also two cases to compute the delay of the first remote task. In Case 1, the delay of the first task in the remote queue can be estimated by $1/\lambda^p_l$. In Case 2, the first task of the remote queue needs to wait for a while until the total data size of the remote queue is larger than Γ, which is $\frac{\Gamma - S^p_r}{\lambda^p_r \times \overline{D^p_r}}$. The queue delay depends on the total remote task size S^p_r. The cellular network delay and the P2P delay depend on D_i. Putting them together, the delay to execute T_i at the proxy node p is:

$$d^i_{remote} = \frac{D_i + S^p_r}{r^p_{cell}} + \frac{D_i}{r_{p2p}} + max\{1/\lambda^p_l, \frac{\Gamma - S^p_r}{\lambda^p_r \times \overline{D^p_r}}\} \quad (4)$$

4.4.3 Proxy Selection and Congestion Avoidance

All proxy nodes are ordered in two lists by their downlink and uplink throughput, respectively. Then a node will select the proxy from up to down in the corresponding list according to its flow direction. If any proxy node p can save both energy and delay for T_i, then T_i will be offloaded to p.

This method can also help to avoid congestion at the proxy node. If one node has higher throughput, its remote queue list will be longer. For later coming tasks, the expected delay at this proxy will be longer, so they will be scheduled to other proxy nodes.

4.5 Credit Manager

We design a credit-based scheme to motivate nodes to offload data for others. The credit is a kind of virtual money, which can be used to pay for the network service provided by others. It can also be extended to exchange with real money, but this is out of the scope of this paper. The credit value uses three parameters ρ_e, ρ_d and ρ_b, to adjust the unit and combine energy, delay and bandwidth together, so that $\rho_e Energy = \rho_d Delay = \rho_b Bandwidth$. These three parameters vary due to the network quality. The credit manager computes credits based on the node with poor network quality. For example, when a node uses Carrier 1 in City 1 wants to offload its downlink task, we would use the values in Table 4 to compute the credit, which means saving 1 second of transmission time is equivalent to saving 1.9 Joule energy and saving 4.5M bit (576K Byte) bandwidth.

If task T_i is executed locally, a node will cost energy, delay and bandwidth. By offloading T_i, a node can save some of the cost, which is denoted as C^i_{save}. This is the maximum credit a node wants to pay to the proxy when offloading T_i. On the other hand, the proxy

Table 4: Relationship between energy, delay and bandwidth

Energy (J)	Time (sec)	Bandwidth (KB)
1.9	1	576

```
// original interface
Socket socket = new Socket (serverIP, serverPort);
InputStream in = socket.getInputStream();
OutputStream out = socket.getOutputStream();
.....
// QATO interface
QATOSocket socket = new QATOSocket (serverIP, serverPort,
                                    proxyIP, proxyPort);
InputStream in = socket.getInputStream();
OutputStream out = socket.getOutputStream();
.....
```

Figure 8: QATO TCP interface

has to pay extra energy and bandwidth to get T_i, and the total cost is C^i_{cost}. A proxy should get at least C^i_{cost} credit back when helping others to offload T_i.

$$C^i_{save} = \rho_e \times E^i_{local} + \rho_d \times (d^i_{local} - d^i_{remote}) + \rho_b \times D_i \quad (5)$$

$$C^i_{cost} = \rho_e \times E^i_{remote} + \rho_b \times D_i \quad (6)$$

As task T_i is offloaded only when it saves both energy and delay, we have $C^i_{save} > C^i_{cost}$. To motivate both users to use QATO, some credits in-between should be paid to the proxy as shown in Equation 7, where α is a parameter to balance the motivation, and it is set to 0.5 in our system. Note that other incentive scheme like [24, 25] can also be applied.

$$C^i_{paid} = C^i_{cost} + \alpha(C^i_{save} - C^i_{cost}) \quad (7)$$

5. QATO IMPLEMENTATION

QATO is implemented on the Android platform. In this section we introduce the interface of QATO and two applications designed based on QATO.

5.1 QATO Interface

All components, including service discovery, offload engine, and credit manager are wrapped into classes and run in the background when QATO is turned on. For developers, QATO provides a simple interface. Using TCP as an example (UDP works similarly), the QATO interface is shown in Fig. 8. Originally a client connects to the server directly. After successful connection, a client can manipulate the input/outputstream. In QATO, we provide a new QATOsocket with two more parameters: proxy IP address and port.

All data traffic using QATOsocket will be forwarded to QATO. The offload engine decides whether to offload the traffic. If not, the connection works the same as the original one. Otherwise, the node first creates a connection to the proxy via the WiFi direct interface, and then the proxy creates a new connection to the real server. Later on, the proxy connects the input/outputstream of one connection to the output/inputstream on the other side. In this way, the proxy works like a tunnel to transmit data, without storing the user data, and as a result user privacy is also protected. For developers, the whole process is transparent and they can use the inputstream and outputstream as normal.

5.2 Applications

We develop two applications on top of QATO: a web browser focusing on downlink offloading, and a photo uploader focusing on uplink offloading.

Table 5: Webpage benchmarks

Name	URL	# of Files	Size (KB)
Google	www.google.com	2	10
Yahoo	www.yahoo.com	165	808
Amazon	www.amazon.com	9	126
Wiki	www.wikipedia.org	18	117
Ebay	www.ebay.com	13	221
Bing	www.bing.com	2	29
Craigslist	www.craigslist.com	7	409
Go	www.go.com	18	980
Espn	www.espn.go.com	53	598
CNN	www.cnn.com	21	396

5.2.1 Web Browser

In the web browser application, the opening of a webpage is treated as downloading multiple files. To eliminate the effect of congestion on web server, all files (including embedded objects) of the webpage are downloaded to a server in our lab. When the phone opens a webpage, we use the idea in [23, 22] by downloading all object files first and then rendering them. It works as below. After a user enters a URL and clicks the "go" button, the web browser downloads the main webpage. Then it parses the whole content, detects all embedded object files, including CSS, images, javascript files, and downloads them together. After all files are downloaded, the web browser modifies the object links to the local files and displays the webpage. By downloading all webpage files in a bunch, the tail energy can be significantly reduced.

5.2.2 Photo Uploader

With cameras on smartphones, users are generating more and more photos, and uploading them to Facebook, flickr, google, etc. Since photos are very large, it may take much more time to upload photos and consume lots of energy when the wireless signal is not good. Thus, it is better to offload such tasks to neighboring nodes with better network quality. To achieve this goal, we design a photo uploader based on QATO. Users can take photos or select photos from the gallery and then upload them to a self defined server. Since photo uploading is delay tolerant, the proxy can adjust Γ to achieve a balance between saving energy and reducing delay.

6. PERFORMANCE EVALUATIONS

In this section, we first run some experiments to evaluate the benefits of traffic offloading, and then use trace driven simulations for more tasks. We compare the performance of QATO, denoted as "`Ours`", to the original approach (without data offloading), denoted as "`Original`".

6.1 Real Experiments

We have implemented QATO on two smartphones as listed in Table 1 and run experiments in City 1. All phones have Android 4.2.2 and have pre-installed two applications: web browser and photo uploader. As LTE has larger downlink throughput than HSPA+, the GS4 phone is selected as the proxy.

To evaluate the performance of QATO, we first run each application using the GS3 phone individually. Then we turn on QATO on both phones and put them within communication range. For both approaches we use Monsoon power monitor to measure the energy as described in Section 3.1. To be fair for different approaches, the base energy is removed, which is the average base

Table 6: Photo benchmarks

ID	Data Size (KB)	ID	Data Size (KB)
1	38	11	1028
2	55	12	1092
3	83	13	1171
4	309	14	1255
5	382	15	1280
6	392	16	1293
7	393	17	1297
8	519	18	1479
9	678	19	2539
10	817	20	2882

power multiply the data transmission time. For the original method, we only consider the data transmission energy of GS3. When QATO is started, we also consider the data transmission energy of the proxy node (i.e., the GS4 phone in our case).

6.1.1 Web Browser

We pick 10 most popular websites from the Alexa website [1], as listed in Table 5 to test the performance of download offloading. These websites have different numbers of embedded objects. Some contain one image while others contain hundreds of images. In this application, the delay is defined as the time from a user pressing the "go" button to the time when the webpage is totally downloaded. Since web browser is not delay tolerant, we set Γ to the minimum webpage size so that all offloading requests can be executed immediately.

Figure 9 compares the energy and delay of the two methods. When downloading webpages, the original method takes more energy and time than ours since QATO can offload traffic to the GS4 phone, which has much higher downlink throughput. On average, our method can save energy by 38% and reduce delay by 45%.

6.1.2 Photo Uploader

We use 20 photos with different sizes to evaluate the performance of upload offloading. The data size is listed in Table 6, which ranges from several kilobytes to several megabytes, with an average value of 939KB. The photos are roughly divided into two categories: small photos with data size smaller than 1MB, and large photos larger than 1MB.

The comparison results of the original method and our method are shown in Fig. 10. The dotted line shows the average value of small photos and large photos of the original method. In the original method, the energy and delay of the small photos are much smaller than that of the large photos. More specifically, the average energy of the small photos is 23.4J, while it is 54.1J for large photos. The average delay of small photos is 3.57 seconds while it is 17.63 seconds for large photos. This is because the uplink bandwidth of HSPA+ is relatively small. However, when using QATO, the energy and the delay are both reduced significantly, since the GS4 phone has much higher uplink bandwidth. For all photos, our method can save 70% of energy and 88% of delay on average.

The data size threshold Γ affects the performance of data offloading. Since photo uploading is delay tolerant, adding more delays (i.e., increasing Γ) can be used to save more energy. Suppose a user selects 20 photos to upload and the duration of selecting one photo is 5 seconds, the energy and delay of using QATO with different Γ are shown in Fig. 11. When Γ increases, more energy is saved since more data can be aggregated to transmit at once and more tail energy is saved. However, the delay also increases since there will be more queue delay and promotion delay.

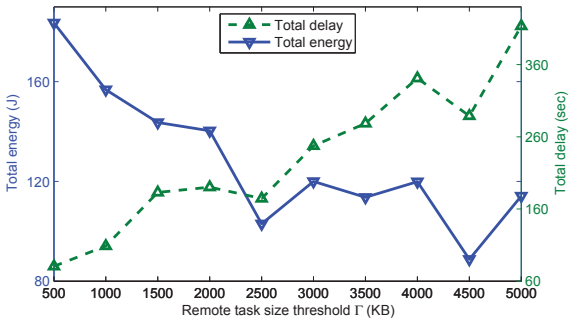


Figure 11: The impact of Γ on the performance of photo uploading

6.2 Trace-driven Simulations

In this section, we aim to show that the credit manager can benefit all nodes in the long run. We collect network traces from 4 users in one month. Then we assume two commuters, user 1 with Carrier 1's data plan and user 2 with Carrier 2's data plan, always travel between City 1 and City 2. We feed trace 1 and trace 2 to user 1 in City 1 and City 2, respectively. Similarly trace 3 and trace 4 are fed to user 2 in two cities. As shown in Table 2, user 1 offloads traffic to user 2 in City 1 and user 2 offloads traffic to user 1 in City 2. In these two month periods, we compare the performance with/without QATO and the results are shown in Fig. 12.

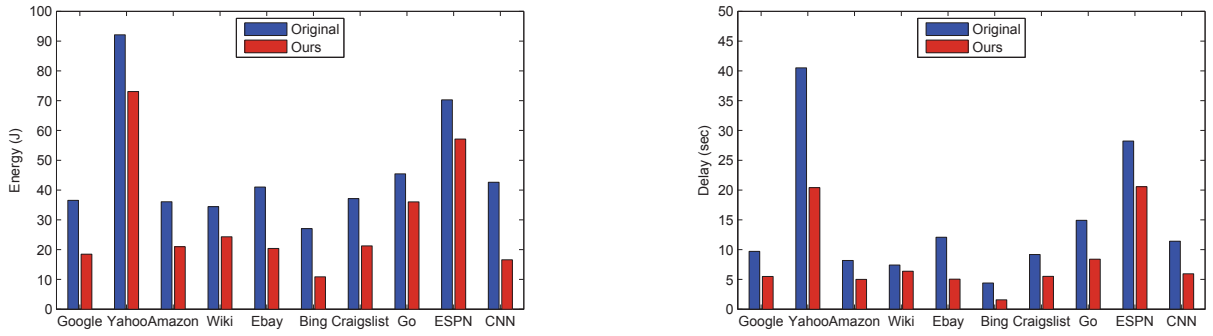
Fig. 12(a) and Fig. 12(b) compare the energy and delay of the original method and QATO. As can be seen, both users benefit from using QATO by saving energy and reducing delay. User 1 consumes much more energy and time than user 2 in the original method because he has more network tasks. Due to the large number of tasks, user 1 has more opportunity to save energy and reduce delay by offloading them to user 2. In total user 1 saves 62% of energy and user 2 saves 36%, and user 1 reduces delay by 50% and user 2 reduces the delay by 25%.

The credit (cost) value considers energy, delay and bandwidth. As mentioned before, when offloading a task, a node saves some cost which is more than the credits paid to the proxy. From this point of view, we say a node benefits from QATO if the saved cost is more than the paid credits in the long run. To verify this assumption, we compare the total saved cost and the paid credits during the two month periods for both users and show the results in Fig. 12(c). It clearly verifies our assumption since both users paid less credits than their saved cost. The benefit (the difference between saved cost and paid credit) of User 1 is larger since User 1 offloads more tasks. On the other hand, User 2 also earns some credits (which is not shown in the figure) by being a proxy.

7. RELATED WORK

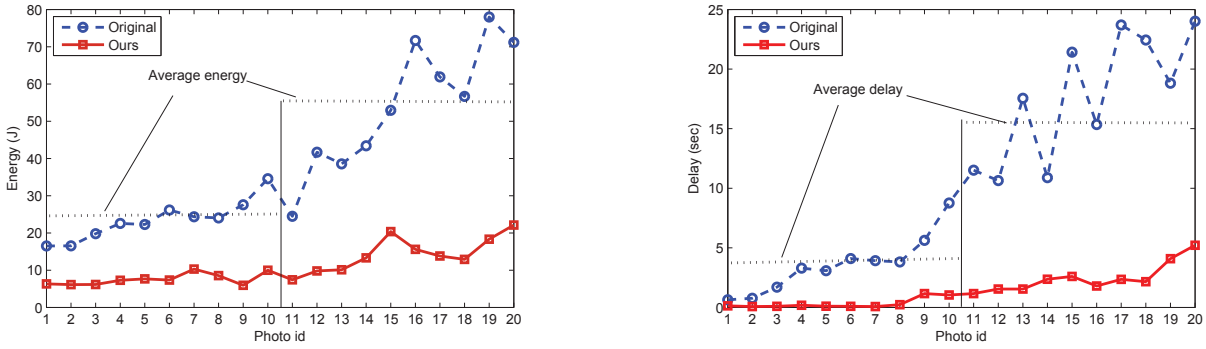
Our work aims to save energy and delay by offloading traffic to neighbors with better service quality. It is related with three categories of work.

Power saving in cellular networks: In cellular networks, including GSM, UMTS, HSPA, and LTE, the radio interface on smartphone is kept in the high power state for a long time (called the long tail problem) after data transmission. One advantage of this



(a) Energy (b) Delay

Figure 9: Energy and delay comparisons with/without QATO when downloading webpages.



(a) Energy (b) Delay

Figure 10: Energy and delay comparisons with/without QATO when uploading photos.

approach is that it can reduce the latency for next possible data transmission, but it also wastes lots of energy. To solve this problem, some researchers introduce methods to aggregate the network traffic to amortize the tail energy [6], or turn the radio interface off quickly by predicting the end of communication [3, 13]. The tail energy can also be saved by aggregating traffic together, if the application is delay tolerant [4].

Quality aware data access: The service quality difference of cellular network within an area has drawn researchers' attention. The Bartendr project [16] studies the relationship between signal strength, throughput and energy. It indicates that the energy of cellular network interface increases and the data throughput decreases when the signal is weak. Based on this finding, they propose to predict the signal strength considering both location and moving direction, and then defer data transfer until reaching a location with better signal. By moving one step further, a travel trip can be planned considering network quality [18]. However, these approaches all require the knowledge of users' movement, which is not easy to achieve in many cases. There are also studies on helping static users to improve the signal quality [20] or increasing network capacity [14] of cellular network via the P2P links. Different from them, we leverage users' cooperation to save energy and reduce delay.

Offloading: In the past several years, there has been lots of research on 3G offloading which focuses on offloading 3G traffic to WiFi network or opportunistic mobile network to save energy or 3G bandwidth [15, 10, 24, 25]. The spider project [19] uses concurrent WiFi connections to improve the throughput and connectivity. However, WiFi access may not always be available.

To leverage neighboring nodes with good signal in 3G networks, UCAN relays data to nodes with higher throughput via the 802.11 interface [11]. Our work is different from it in three perspectives. First, based on measurements, we give motivations for node cooperation even at the same location. Second, we have implemented the real system based on smartphones where previous work is limited to simulations. Third, our work considers many practical scheduling issues related to the long tail problem, which are not considered in UCAN.

There are also works on mobile clouds [5, 9] which aim to offload complex computations to cloud to save energy. Recently, many researchers also consider offloading computations to nearby mobile devices [17] to save energy. Their idea of leveraging neighbors' resource inspires our work, but their works focus on computation offloading whereas our work focuses on communication offloading.

8. CONCLUSIONS

In this paper, based on real measurements, we demonstrated the existence of significant throughput difference between wireless carriers at some locations, and then motivated the necessity of node collaboration to save energy and reduce delay in cellular networks. We proposed a data offloading framework QATO to offload network tasks to neighboring nodes with better throughput, to save energy and reduce delay. QATO can identify neighbors with better service quality through service discovery, and provide incentive mechanisms to motivate nodes to help each other. To validate our design, we have implemented QATO on Android platform and

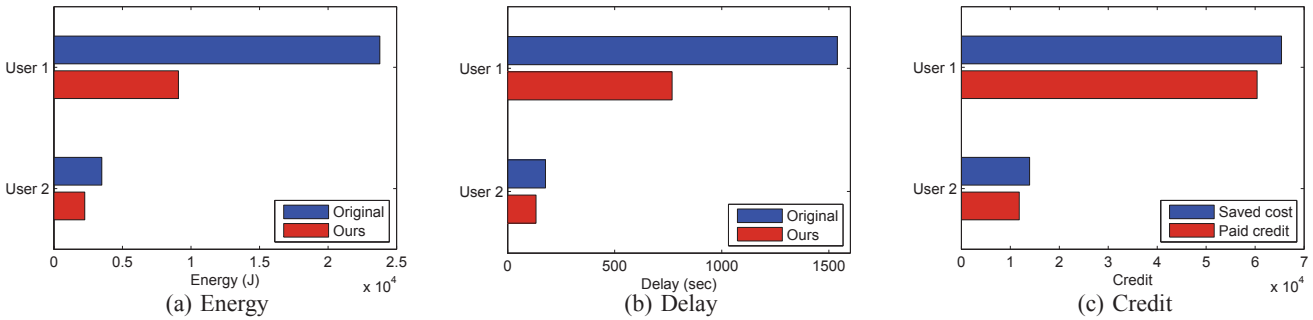


Figure 12: Performance comparisons with/without data offloading

developed a web browser and a photo uploader on top of it. Experimental results show that QATO can reduce energy by 38% in downloading and 70% in uploading, and reduce delay by 45% in downloading and 88% in uploading. Through trace-driven simulations, we also show that all users can benefit from data offloading in the long run.

Acknowledgment

We would like to thank anonymous reviewers for their insightful comments and helpful suggestions. This work was supported in part by the National Science Foundation (NSF) under grant number CNS-1218597, and by Network Science CTA under grant W911NF-09-2-0053.

References

[1] Alexa top sites. `http://www.alexa.com/topsites`.

[2] Rfc 6763: Dns-based service discovery. `http://tools.ietf.org/html/rfc6763`.

[3] P. K. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. N. Padmanabhan, and G. Varghese. RadioJockey: Mining Program Execution to Optimize Cellular Radio Usage. In *ACM MobiCom*, 2012.

[4] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *ACM IMC*, 2009.

[5] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making Smartphones Last Longer with Code Offload. In *ACM MobiSys*, 2010.

[6] W. Hu and G. Cao. Energy Optimization Through Traffic Aggregation in Wireless Networks. In *IEEE INFOCOM*, 2014.

[7] W. Hu, G. Cao, S. V. Krishanamurthy, and P. Mohapatra. Mobility-Assisted Energy-Aware User Contact Detection in Mobile Social Networks. In *IEEE ICDCS*, 2013.

[8] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *ACM MobiSys*, 2012.

[9] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang. ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading. In *IEEE INFOCOM*, 2012.

[10] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong. Mobile Data Offloading: How Much Can WiFi Deliver? *IEEE/ACM Transactions on Networking*, 21(2):536–550, 2013.

[11] H. Luo, R. Ramjee, P. Sinha, L. E. Li, and S. Lu. UCAN: A Unified Cellular and Ad-hoc Network Architecture. In *ACM MobiCom*, 2003.

[12] C. Peng, S.-B. Lee, S. Lu, H. Luo, and H. Li. Traffic-driven Power Saving in Operational 3G Cellular Networks. In *ACM MobiCom*, 2011.

[13] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation. In *IEEE ICNP*, 2010.

[14] B. Radunovic and A. Proutiere. On Downlink Capacity of Cellular Data Networks With WLAN/WPAN Relays. *IEEE/ACM Transactions on Networking*, 21(1):286–296, 2013.

[15] N. Ristanovic, J.-Y. Le Boudec, A. Chaintreau, and V. Erramilli. Energy Efficient Offloading of 3G Networks. In *IEEE MASS*, 2011.

[16] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan. Bartendr: A Practical Approach to Energy-aware Cellular Data Scheduling. In *ACM MobiCom*, 2010.

[17] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura. Serendipity: Enabling Remote Computing Among Intermittently Connected Mobile Devices. In *ACM MobiHoc*, 2012.

[18] C. Shi and H. Xie. Go This Way: Navigation for Better Access Quality in Mobile Networks. In *ACM MobiArch*, 2013.

[19] H. Soroush, P. Gilbert, N. Banerjee, M. D. Corner, B. N. Levine, and L. Cox. Spider: Improving Mobile Networking with Concurrent Wi-Fi Connections. In *ACM SIGCOMM*, 2011.

[20] Y. Wang and G. Noubir. Distributed Cooperation and Diversity for Hybrid Wireless Networks. *IEEE Transactions on Mobile Computing*, 12(3):596–608, 2013.

[21] S. Yi, S. Chun, Y. Lee, S. Park, and S. Jung. *Radio Protocols for LTE and LTE-Advanced*. Wiley, 2012.

[22] B. Zhao, W. Hu, Q. Zheng, and G. Cao. Energy-Aware Web Browsing on Smartphones. *IEEE Transactions on Parallel and Distributed Systems (TPDS), to appear*.

[23] B. Zhao, Q. Zheng, G. Cao, and S. Addepalli. Energy-Aware Web Browsing in 3G Based Smartphones. In *IEEE ICDCS*, 2013.

[24] X. Zhuo, W. Gao, G. Cao, and Y. Dai. Win-Coupon: An Incentive Framework for 3G Traffic Offloading. In *IEEE ICNP*, 2011.

[25] X. Zhuo, W. Gao, G. Cao, and S. Hua. An Incentive Framework for Cellular Traffic Offloading. *IEEE Transactions on Mobile Computing*, 13(3):541–555, 2014.