Energy Optimization Through Traffic Aggregation in Wireless Networks

Wenjie Hu and Guohong Cao
Department of Computer Science and Engineering
The Pennsylvania State University
E-mail: {wwh5068, gcao}@cse.psu.edu

***Abstract*—Cellular networks can provide pervasive data access for smartphones, but also consume lots of energy, because the cellular interface has to stay in high power state for a long time (called long tail problem) after a data transmission. In this paper, we propose to reduce the tail energy by aggregating the data traffic of multiple nodes using their P2P interfaces. This traffic aggregation problem is formalized as finding the best task schedule to minimize energy. We first propose an $A^\star$ search algorithm, which can reduce the search space for finding the optimal schedule offline, and then introduce an online traffic aggregation algorithm. We have implemented the online traffic aggregation algorithm on Android smartphones, and have built a small testbed. Trace-driven simulations and Experimental results show that our traffic aggregation algorithm can significantly reduce the energy and delay.**

I. Introduction

Smartphones have become the essential components of our daily life; however, we are also frustrated with their short battery life. One major source of the power consumption comes from the cellular interface which is used for supporting mobile data. In UMTS 3G network or 4G (HSPA+) network, multiple timers are used to control the cellular interface, and the timeout value for releasing the radio resource can be more than 15 seconds [20]. Thus, it is possible that the cellular interface continues to consume a large amount of energy (also referred to as the *long tail problem*) before the timer expires, even when there is no network traffic. For example, recent research [6], [9] showed that the energy wasted in the long tail in 3G networks could be more than that of the real data transmission in many applications, and this becomes worse in 4G network due to its higher tail power and longer tail time.

There has been some research [20], [6] on reducing the tail energy of the cellular interface. Most of them try to adaptively adjust the tail time using techniques similar to *fast dormancy* [1]; i.e., adjusting the timer and putting the cellular interface into low power mode quickly. However, only few smartphones support fast dormancy due to the limitation of OS and implementation. Furthermore, keeping the long tail can reduce the latency of future data transmission that arrives before the timer expires, because the connection between the smartphone and the backbone network is still available. Otherwise, the backbone network has to allocate the radio resource again, which will consume more time (called the *promotion delay*) and energy. Thus, fast dormancy may waste power and introduce long delay if the next data request comes quickly.

We propose to reduce the tail energy and delay by aggregating the data traffic of multiple nodes. Consider a group of people on a bus or train, they may access various webpages using smartphones. Although these small data requests (tasks) can be quickly served by the cellular interface, all of them have to pay for the tail energy and the promotion delay. If we can aggregate the tasks whose intervals are shorter than the tail time to one smartphone (called the *proxy*), the total tail energy can be reduced. Other smartphones can use their peer to peer (P2P) interfaces such as Bluetooth or WiFi direct to communicate with the proxy and get the data. Since the P2P interfaces consume much less energy compared to the cellular interfaces, this approach can save energy as a group. The delay can also be reduced since the smartphones will send their data request immediately to the proxy which is already using the cellular interface. As a result, it can save the delay (e.g. about one second) of starting the cellular interface at the requesting smartphones.

Since scheduling a task to other smartphones (nodes) has a tradeoff between saving the cellular tail energy and bringing extra P2P energy, we ask the following question: *Given a group of nodes and a number of tasks, and each task can be scheduled either locally or to other nodes, what is the best schedule to minimize energy?* As a node can schedule any task to anyone, it is hard to find the optimal solution using traditional algorithms.

To solve this problem, we first divide it into sub problems, and then propose an $A^\star$ search algorithm to reduce the search space for finding the optimal schedule offline. Since this solution requires future knowledge, which may not be possible in many cases, we further introduce an online traffic aggregation algorithm. The contributions of this paper are as follows.

- We reduce the tail energy and delay by aggregating the data traffic of multiple nodes using their P2P interfaces.
- We design an offline traffic aggregation algorithm to minimize energy, and introduce an online traffic aggregation algorithm.
- We have implemented the online traffic aggregation

This work was supported in part by the US National Science Foundation (NSF) under grant number CNS-1218597, and by Network Science CTA under grant W911NF-09-2-0053.

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

algorithm on Android smartphones, and have built a small testbed. Experimental results show that our traffic aggregation algorithm can save 30% of energy and delay. Trace-driven simulations show that the proposed traffic aggregation algorithm is more effective with more tasks.

The rest of this paper is organized as follows. Section II discusses related work. Section III introduces the energy models of various wireless interfaces. Section IV presents the optimal traffic aggregation algorithm. We introduce the online traffic aggregation algorithm in Section V and the testbed in Section VI. Section VII evaluates the performance of our algorithm, and Section VIII concludes the paper.

II. Related Work

Energy optimization for smartphones has attracted considerable attention in recent years, and lots of research focuses on the energy consumption of the wireless interface. Mittal *et al.* [13] have introduced an energy emulation tool to estimate the energy consumption of smartphones, and have showed that the cellular interface consumes 30-50% of the total energy. Other researchers [6], [9] also point out that the cellular interface will stay in high power state for a long time (tail time) after a data transmission, and the energy consumed during the tail time accounts for almost 60% of the total energy for a typical 3G data transmission [6].

In order to save energy during data transmission in cellular networks, one method is to reduce the tail time using protocols like fast dormancy [1]. However, this requires support from both mobile devices and cellular carriers. Furthermore, it cannot know when the next data transmission will happen. As explained in the introduction, if the next data transmission happens quickly, fast dormancy may waste energy on switching the smartphone back to high power state. Another feasible method is to aggregate network tasks together to save tail energy. Zhao *et al.* [20] proposed to combine data transfers when opening a webpage to save the tail energy and reduce the delay for web browsing. Other researchers [14] propose techniques to combine network tasks by deferring some tasks to amortize the tail energy, but at the cost of introducing long delay for the deferred tasks. There is also research on reducing the usage of cellular interface by offloading the traffic through opportunistic communications [15], [11], [21], but these can only be used for delay tolerant applications.

Since mobile devices have multiple wireless interfaces, properly combining them can save energy [5], or improve network throughput [16], [17], [18]. Our work is related to the research [4], [10], [12] that leverages the coexistence of cellular interfaces and P2P interfaces. In [4], a system was built to let users form a group via their P2P interfaces and then use cellular interfaces to collaboratively download data. MicroCast [12] uses multiple phones to download video clips from their cellular interfaces and then broadcast them via the WiFi interfaces to increase the downloading speed. However, in these works multiple phones use their cellular interfaces simultaneously, and thus everyone has to pay for the tail energy. Different from them, we aggregate traffic from multiple smartphones to save the tail energy.

TABLE I
The power consumption of various wireless interfaces

State	Power (mW)	Duration (s)	T'put (Mbps)
Idle (NS)	548.1 ± 30.0	-	-
Idle (GS3)	614.7 ± 35.2	-	-
3G Promotion	1102.2 ± 40.4	0.9 ± 0.1	-
3G Data	1432.6 ± 105.7	-	0.9 ± 0.3
3G Tail	1414.8 ± 163.4	8.0 ± 0.1	-
4G* Promotion	1422.2 ± 34.1	2.3 ± 0.5	-
4G Data	1990.9 ± 44.2	-	4.5 ± 1.3
4G Tail	1622.6 ± 39.6	11.4 ± 1.4	-
WiFi direct Data	1323.6 ± 13.9	-	29.5 ± 1.2
Bluetooth Data	945.9 ± 0.9	-	1.2 ± 0.02

*: 4G indicates the HSPA+ network.
The power consumption is measured when the screen is on.

III. Energy Model

In this section, we introduce the energy model. We consider two kinds of interfaces: cellular interfaces such as 3G and 4G, and P2P interfaces such as Bluetooth and WiFi direct.

A. Energy Measurement

As different cellular and P2P networks require different hardware support, we use Google Nexus S (NS) to measure the 3G and Bluetooth power consumption, and use Samsung Galaxy S III (GS3) to measure the 4G (HSPA+) and WiFi direct power consumption. We ported *iperf* [2] to these smartphones and extended it to support Bluetooth and WiFi direct, so we can adjust the data transmission patterns as needed. To measure the power consumption, similar to [19], [8], we use Labview on a laptop to configure the Agilent E3631A power supply to provide the current with constant voltage to the smartphones instead of using battery. All measured values are summarized in Table I, where the value includes the power consumption of the whole phone when the screen is on. Each value is measured 10 times.

B. Power Model of the Cellular Interface

Figure 1 shows the power state of smartphone when using the 3G interface to transmit data. In 3G networks, the radio interface can work at three states: *DCH, FACH,* and *IDLE*. Initially, the smartphone stays at the IDLE state, which consumes very little power. At IDLE state, the phone does not have any signaling connection with the backbone network, and hence it cannot transmit user data.

To send or receive data, the smartphone should be at the DCH state (from $t2$ to $t3$). At DCH, the backbone network allocates dedicated transmission channels (uplink and downlink) to the smartphone to transmit user data and signaling information at high speed. This requires high level of power as shown in Fig. 1. It takes some time for the smartphone to promote from the IDLE state to the DCH state. This delay (from $t1$ to $t2$) is called the *promotion delay* (t_{pro}). In the DCH state, a timer will be triggered after the data transmission completes. When the timer expires, the smartphone will move

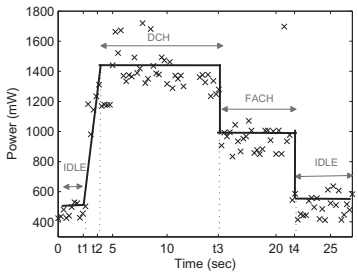


Fig. 1. The power level of using 3G cellular interface

to the FACH state (from $t3$ to $t4$). and then to IDLE state after another timer expires. If there are new data requests coming during the FACH period, the phone will promote to the DCH state immediately.

4G (HSPA+) network introduces High Speed Downlink/Uplink Packet Access (HSDPA/HSUPA) to UMTS 3G network to improve the downlink and uplink data transmission rate. The power state of 4G interface is similar to that of 3G. Therefore, we can generalize the power consumption of 3G/4G cellular interface into three states: *promotion, data transmission* and *tail*, and the power of these states are denoted as P_{pro}, P_{cell} and P_{tail}, respectively. The energy consumption of a data request (task) in cellular network can be modeled as follows. Suppose task T_i arrives at t_i and lasts for d_i time, and the most recent task on the same node is T_j. The interval between task T_i and T_j is $\Delta t = t_i - t_j - d_j$. There are three cases to compute T_i's energy 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 some 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. In summary, the energy of T_i in cellular network is computed as Eq. 1.

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

C. Power Model of the P2P Interface

We consider two types of P2P interfaces: Bluetooth and WiFi direct. Bluetooth can be found on almost all smartphones. In order to realize data transmission via Bluetooth, two users need to pair (bond) with each other first. The Bluetooth communication on Android is based on RFCOMM protocol, which provides a simple and reliable data stream to users, like TCP. To create a connection, the Bluetooth server needs to register a service with a special identity UUID, and then the client can connect to the server using its MAC address and the same UUID.

WiFi direct is a new standard to support P2P connection, and can be considered as the Ad Hoc mode of the WiFi interface. It is supported by Android 4.0 and higher. WiFi direct has much higher speed and larger transmission range (up to 60 meters) than its Bluetooth counterpart. To initiate a WiFi direct connection, two phones need to be connected first. After that, the data transmission works the same as TCP. The server listens at a port and the client connects to the server according to its IP address and port number.

For Bluetooth and WiFi direct, the promotion and tail energy are negligible. The energy consumption of task T_i is mainly related to the data size s_i and the data transmission rate r, as shown in Eq. 2.

$$E^i_{P2P} = P_{P2P} \times \frac{s_i}{r} \quad (2)$$

IV. Optimal Traffic Aggregation

In this section, we define the problem of traffic aggregation and find its optimal solution.

A. Problem Statement

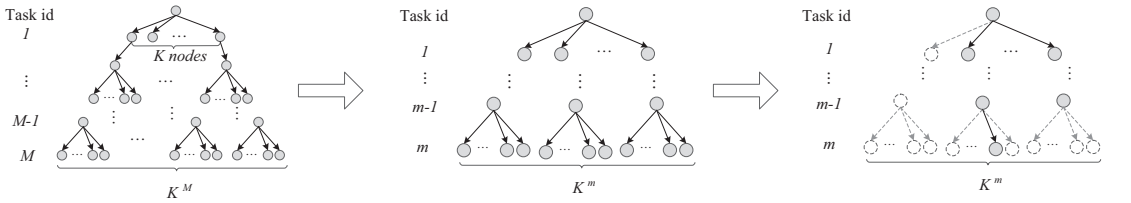
Suppose K smartphones (nodes) are close to each other and have been paired (connected) with each other via Bluetooth or WiFi direct. These K nodes have M tasks and each task involves some data transfer through the Internet. Task T_i, with a data size s_i, is generated at time t_i and should be scheduled at t_i either locally or to some other nodes. The result of traffic aggregation is a scheduling sequence $\mathcal{S}$ with M elements, where the ith element is a node which processes T_i with its cellular interface. Given a scheduling sequence, the energy consumption of all tasks can be calculated using Eq. 1 and Eq. 2. The *traffic aggregation problem* is to find a schedule sequence $\mathcal{S}$ with minimum energy consumption.

As any task in M can be executed on any of the K nodes, the solution space of this problem is $O(K^M)$, as shown in Fig. 2(a), and then it is hard to find the optimal solution. To solve this problem, we first introduce the concept of burst group to divide the original problem into subproblems with m tasks, where m is far smaller than M, as shown in Fig. 2(b). Then, we introduce an $A^\star$ search algorithm [7] to search for the optimal solution more efficiently by reducing the search space, as shown in Fig. 2(c). Next, we discuss these two steps in the following subsections.

B. Burst Group

Suppose there is a set of network tasks in a node. If two network tasks in a node have an interval smaller than the tail time t_{tail}, the cellular interface will stay on high power state during the whole period of these two tasks; i.e., these two tasks belong to the same burst from the energy point of view. A burst B is defined as a group of tasks $\{T_1, T_2 \cdots T_b\}$, where $t_i - t_{i-1} < t_{tail}, 1 < i \leq b$. The duration of a burst is $[t_1, t_b + t_{tail}]$, since it will keep the cellular interface on high power state (non-IDLE states) during the whole period. For example, in Fig. 3, the first two tasks in node 1 belong to one burst, while the third one belongs to another.

For a group of nodes, as long as any of their bursts have overlap, there are opportunities to aggregate their tasks



(a) original problem, solution space O(K^M) (b) divide to sub problems using Burst Group, m<< M (c) improve speed using A*, search space << O(K^m)

Fig. 2. Solutions to the traffic aggregation problem

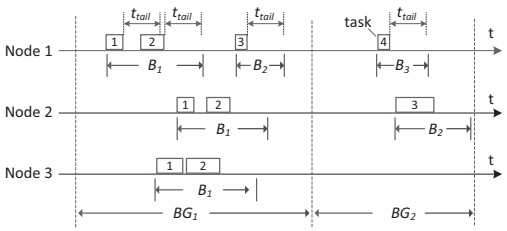


Fig. 3. Burst and burst group. The length of each box indicates the task duration. The duration of a burst is from the time of the first task to the time of the last task plus t_{tail}.

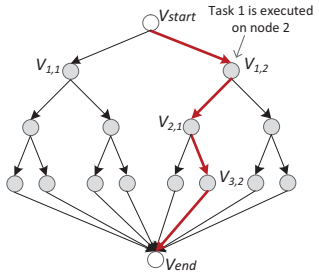


Fig. 4. Mapping the traffic aggregation problem to the shortest path problem

together to save the tail energy. These bursts will be merged into one burst group. For example, the first two bursts in node 1, the first bursts in node 2 and the first burst in node 3 in Fig. 3 form one burst group BG_1. Traffic aggregation (schedule) in different burst groups is independent, because there is no overlap to reduce the tail energy. By combining the optimal schedule sequence of each burst group, we can obtain the global optimal schedule sequence. Thus, we can focus on how to schedule tasks in one burst group.

C. The Optimal Traffic Aggregation Algorithm

Suppose there are m tasks and K nodes in a burst group. We aim to find an optimal schedule sequence to minimize energy. This problem can be mapped to the shortest path problem as follows. First we create a directed graph as a full K-ary tree. The root is a virtual node V_{start} with depth $h = 0$. Each virtual node at depth $h = i-1 (1 \le i \le m)$ then expands K children at depth $h = i$, where the kth child corresponds to task i being executed in node k. For simplicity, we also refer a virtual node in the graph as $V_{i,k}$, which means task i is scheduled in node k. Finally we add another destination virtual node V_{end} and create a link from each virtual node at depth m to it. In such a graph, every path from V_{start} to V_{end} will map to one and only one schedule sequence, and vice verse. An example of such a graph with 3 tasks and 2 nodes is shown in Fig. 4. The bold path corresponds to scheduling tasks 1, 2, 3 in nodes 2, 1, 2, respectively.

1) Link and Virtual Node Weight: In the predefined graph like Fig. 4, the link weight from a virtual node $V_{i,k}$ to its child $V_{i+1,k'}$ represents the additional energy consumed by T_{i+1} if it is scheduled in node k'. The computation of this weight is not straightforward, since the energy of T_{i+1} on the cellular interface is related with the previous task in node k', which is determined by the schedule history of all previous tasks. To solve this problem we add a K dimension task vector $\overrightarrow{C}_{V_{i,k}}$ to each virtual node $V_{i,k}$, where the jth value records the last task on user j. The task vector on V_{start} is initialized to $\overrightarrow{0}$. With $\overrightarrow{C}_{V_{i,k}}$, we can compute the cellular interface energy of T_{i+1} using Eq. 1. Additionally, if T_{i+1} is not executed on its generating node, there will be some additional P2P energy to transfer the data between nodes. Adding them together, we can compute the link weight using Eq. 3 for all links from $V_{i,k}$ to $V_{i+1,k'} (0 \le i < m)$. Here $1_{k',i+1}$ is 1 if T_{i+1} is not generated in node k' and 0 otherwise. After that, $\overrightarrow{C}_{V_{i+1,k'}}$ is setting to $\overrightarrow{C}_{V_{i,k}}$ but the k'th value is updated to T_{i+1}. The link weight from $V_{m,k}$ to V_{end} is set to 0.

$$W_{V_{i,k}}^{V_{i+1,k'}} = E_{cell}^{i+1}(\overrightarrow{C}_{V_{i,k}}[k']) + 1_{k',i+1} \times E_{P2P}^{i+1} \quad (3)$$

In the $A^\star$ search Algorithm, each virtual node $V_{i,k}$ also has a weight $\mathbb{F}(V_{i,k})$ to estimate the cost of the shortest path that may pass through it, which contains two parts, as defined in Eq. 4.

$$\mathbb{F}(V_{i,k}) = \mathbb{G}(V_{i,k}) + \mathbb{H}(V_{i,k}) \quad (4)$$

$\mathbb{G}(V_{i,k})$ is the cost from the start point to $V_{i,k}$, i.e., the sum of the link weight from the V_{start} to $V_{i,k}$.

$$\mathbb{G}(V_{i,k}) = \sum_{link_{V_{j,u}}^{V_{j+1,u'}} \in path_{V_{start}}^{V_{i,k}}} W_{V_{j,u}}^{V_{j+1,u'}} \quad (5)$$

$\mathbb{H}(V_{i,k})$ is the estimation of the cost from $V_{i,k}$ to the destination, and $\mathbb{H}(V_{i,k})$ should never be bigger than the actual minimum cost from $V_{i,k}$ to the destination so that $A^\star$ can find the optimal solution. The more accurate this estimation is, the faster $A^\star$ runs. To satisfy the requirement, $\mathbb{H}(V_{i,k})$ is set to be the cellular energy consumed by all the remaining tasks if they are all scheduled on node k, as Eq. 6. After $V_{i,k}$, node k will have the latest cellular activity time, and then aggregating all

remaining tasks in node k will consume the minimum cellular energy when compared with other nodes. Also, we ignore the P2P power consumption when predicting the future energy, so $\mathbb{H}(V_{i,k})$ will always be smaller than the actual minimum cost.

$$\mathbb{H}(V_{i,k}) = \sum_{j=i+1}^{m} E_{cell}^{j}(T_{j-1}) \tag{6}$$

For one virtual node $V_{i,k}$, if $\mathbb{F}(V_{i,k}) + \mathbb{H}(V_{i,k})$ is bigger than the cost of the shortest path, then $V_{i,k}$ and all its children will not be searched by the $A^\star$ algorithm, and thus we can quickly find the shortest path (optimal schedule sequence).

2) Search for the Optimal Schedule: The $A^\star$ algorithm can be used to find the optimal schedule, as shown in Algorithm 1. The key idea of the $A^\star$ algorithm is the best-first search, which explores a graph by expanding the most promising node chosen according to a specified rule. More specifically, it keeps two virtual node sets: an open set that keeps the virtual nodes that are about to be searched in the next round, and a closed set that contains the virtual nodes that have been searched. The open set and close set are initialized as V_{start} and empty, respectively. $A^\star$ picks a virtual node with minimum $\mathbb{F}$ value from the open set and expands its neighbor virtual nodes into the open set. Then, this virtual node is moved to the closed set. Each virtual node also keeps a pointer to its parent. Finally, when V_{end} is picked from the open set, $A^\star$ can back track to V_{start} and find out the optimal schedule, as shown in Algorithm 2.

Algorithm 1: Search for the optimal schedule using the $A^\star$ algorithm

```
Input: Task set, V_start, V_end
Initialization:
Open set Open ← V_start
Closed set Close ← ∅
F(V_start) ← H(V_start)
while Open ≠ ∅ do
    pick virtual node V_{i,k} ∈ Open with min F value
    if V_{i,k} == V_end then
        return Reconstruct optimal schedule(V_{i,k})
    end
    Close ← Close ∪ V_{i,k}
    Open ← Open \ V_{i,k}
    for k' from 1 to K do
        create virtual node V_{i+1,k'}
        C⃗_{V_{i+1,k'}} ← C⃗_{V_{i,k}}
        F(V_{i+1,k'}) ← Eq.4
        C⃗_{V_{i+1,k'}}[k'] ← T_{i+1}
        V_{i+1,k'}.parent ← V_{i,k}
        Open ← Open ∪ V_{i+1,k'}
    end
end
```

Algorithm 2: Reconstruct the optimal schedule

```
Input: Task set, last virtual node V_{i,k}
Initialization:
Schedule sequence S ← ∅
for j from m to 1 do
    S[j] ← V_{i,k}
    V_{i,k} ← V_{i,k}.parent
end
```

V. Online Traffic Aggregation

Although the optimal traffic aggregation algorithm can minimize the energy consumption, it requires the complete knowledge of future tasks such as the task interval (i.e., the time interval between two consecutive tasks) and the requested data size. In this section, we relax these restrictions and present an online traffic aggregation algorithm.

A. Problem Statement

Suppose a group of nodes connect to each other via Bluetooth or WiFi direct, and one of them works as the scheduler. If there is no scheduler, the node with the maximum remaining energy will be elected as a scheduler. The scheduler maintains all group nodes' information, including the remaining energy, ip address and listening port. If a scheduler does not want to serve others, it picks another node based on the energy information or any other user-defined criteria, and then forwards the group information to the new scheduler. The new scheduler will broadcast its existence to other nodes, which will then keep the information of the scheduler. When a node wants to join or leave an existing group, it notifies the scheduler. If a scheduler fails, a new one will be elected.

The scheduler will aggregate the tasks in the group and schedule them to proper nodes (proxies) to use the cellular interface. Note that the scheduler and the proxy may be the same node. The scheduler may become a bottleneck if all data requests have to go through it. To reduce the number of requests to the scheduler, a node needs to determine whether to send its requests to the scheduler, based on the energy cost of the P2P interface and the energy saved in the cellular interface. The decision is easier for offline scheduling, but much harder for online schedule due to the following reasons. First, the data size is unknown before downloading, so it is hard to compute the accurate power consumption of the P2P interface. Second, the status of the cellular interface at other nodes is unknown, so it is hard to estimate the saved tail energy.

To address these problems, we first describe our data set, based on which we develop techniques to predict the data size and the task arrival time. Then, we introduce the online schedule algorithm.

B. Data Set

We distributed five rooted Nexus S smartphones to users with *tcpdump* pre-installed, and then collected the network trace for one month. From the traces, we extract all TCP and UDP flows using five-tuples (source/destination address, source/destination port, protocol). For TCP, All consecutive packets from the first *SYN* packet to the last *ACK* with the same five-tuple are treated as one flow. For UDP, all consecutive packets with the same five-tuple are treated as one flow. The sum of the packet size in a flow is the data size. In total, we collect 5176 tasks (data requests), and most of these tasks are related to web browsing and email access.

C. Data Size and Task Arrival Interval

1) Data Size: The requested data size affects the energy consumption of the P2P interface and then the decision on task schedule. To get a better understanding of the requested data

size distribution, we draw a histogram of the collected data size in Fig. 5(a). As can be seen, the data size roughly follows a normal distribution with a mean value of 48KB. Thus, we get a global estimation of data size s_i by sampling from the normal distribution, $s_g = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$. On the other hand, nodes tend to have similar type of tasks in a short period of time, so we also have a local estimation of data size based on the average data size of the previous Q tasks in the same node.

To improve accuracy, we consider both global data size distribution and individual data access patterns. By combining them together, the estimated data size is calculated according to Eq. 7. We have tried various Q and α, to measure the error between the estimated data size and the real value. The result is shown in Fig. 6. Here, all errors are normalized based on the minimum value. We found that setting Q to 10 and α to 0.6 can achieve the minimum error rate, and thus they are used as the default values in our algorithms.

$$\overline{s_i} = (1-\alpha) \times \frac{1}{Q} \sum_{j=i-Q}^{i-1} s_j + \alpha \times s_g \tag{7}$$

2) Task Arrival Interval: We collect the task interval of each node and draw the CCDF (Complementary CDF) of all task arrival intervals in Fig. 5(b). As can be seen, it follows power law distribution, which means that many task arrival intervals are short and few are long. The probability for the task arrival interval to be T is $p(t=T) = \frac{aT^b+c}{\sum_{x=0}^{t_{max}}(ax^b+c)\,dx}$, where a, b and c are three constant parameters and t_{max} is the 90% percentile of all intervals, which is 847 seconds.

D. Scheduling Decision

Nodes only send task requests to the scheduler if energy can be saved based on their predictions. For task T_i, if it is scheduled on some other node, the estimated energy consumed by the P2P interface is: $P_{p2p} \times \overline{s_i}/r$. To save energy, it should only be scheduled to a proxy when the saved tail energy is bigger than this value, i.e., the interval between t_i and the proxy's most recent task is smaller than a threshold $\beta_i = P_{P2P}/P_{tail} \times \overline{s_i}/r$.

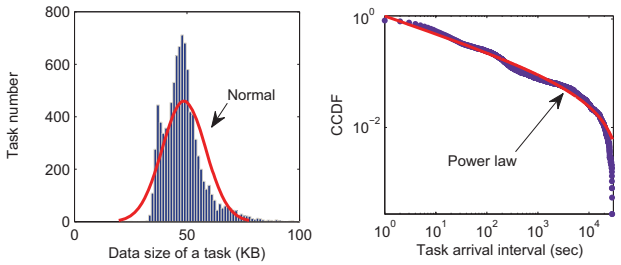
For a single node, the probability that there is a task within β_i time from T_i is:

$$P(t < \beta_i) = \frac{\int_0^{\beta} p(t)\,dt}{\int_0^{t_{max}} p(t)\,dt}$$

Assume the distributions of the task arrival interval are independent. For a group of K nodes, the probability that at least one node has a task within β_i time from T_i is computed as Eq. 8.

$$\mathbb{P}(t < \beta_i) = 1 - \Pi_{k=1}^{K} \left(1 - P(t < \beta_i)\right) \tag{8}$$

Considering delay, the maximum time a node can save by scheduling a task to the proxy is the promotion delay t_{pro}. The extra delay to transmit T_i on the P2P interface is $t_{p2p} = \overline{s_i}/r$. Combining energy and delay consideration together, a node will send T_i's request to the scheduler if the probability to save energy $\mathbb{P}(t < \beta_i)$ is bigger than 80% and it is possible to reduce delay ($t_{p2p} < t_{pro}$); otherwise, the node will schedule T_i locally. In our experiments, changing the energy saving probability from 70% to 90% does not change the result too much.



(a) Data size distribution (b) Task arrival interval distribution

Fig. 5. The distribution of data size and task arrival interval. Data size follows normal distribution, and task arrival interval follows power law distribution.

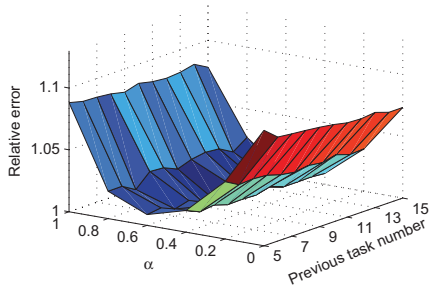


Fig. 6. Predicting the data size

From Eq. 8, we can see that that increasing the number of nodes in a group and reducing the data size will enlarge $\mathbb{P}(t < \beta_i)$ and decrease t_{p2p}, and thus increase the number of tasks scheduled on the scheduler. As a result, more tasks are aggregated and more energy can be saved. The evaluation results also confirm this hypothesis.

E. Online Traffic Aggregation Algorithm

If a node decides to send the task request to the scheduler to save energy, it also provides its local address and listening port. The scheduler will schedule the request to the proper proxy, based on an online scheduling solution.

The online schedule uses the $A^\star$ algorithm to search for the optimal schedule based on local knowledge. The main difficulty is to compute the node weight $\mathbb{F}(V_{i,k})$. In online schedule, the previous tasks before T_i are already scheduled and the decision cannot be changed, so we ignore the previous path cost and set $\mathbb{G}(V_{i,k})$ to the link weight based on Eq. 3. Since future tasks are unpredictable, we simply set $\mathbb{H}(V_{i,k})$ to 0. Then, the $A^\star$ algorithm is reduced to a greedy algorithm.

As the computation of $\mathbb{F}$ is very simple, the scheduler can quickly find out the node with the minimum $\mathbb{F}$ value as the proxy. The scheduler will forward the task request to the proxy, along with the requester's local address and port number. When the data is downloaded, the proxy will connect to the requesting node and send the data back via the P2P interface.

VI. Testbed Development and Evaluation

To evaluate the performance of the traffic aggregation algorithm, we have implemented the algorithm and measured its performance on a small testbed.

A. Testbed Development

We have implemented the online traffic aggregation algorithm on Android smartphones to schedule web browsing tasks. We modified the Zirco web browser [3], an open source mobile web browser, to record the timestamp information and then extended it to support traffic aggregation. The modified Zirco browser is installed on three Galaxy S III smartphones with 4G data plans. At the beginning of the test, all phones are paired/connected with each other via Bluetooth/WiFi direct manually. The WiFi interface (infrastructure mode) is turned off so that all traffic will go through the cellular interface or P2P interface (Bluetooth or WiFi direct). Each phone is powered by the Agilent E3631A Power Supply to record the accurate power consumption.

Three users have used our phones to surf webpages in two stages. First, they turn off the traffic aggregation module and surf webpages using the cellular interface following their normal browsing patterns for 10 minutes. The timestamp of the start and end of opening webpages are recorded. In the second stage, we set a timer for each user which beeps at the start time of his previous tasks based on the record. Users turn on the aggregation module and select Bluetooth as the P2P interface. Then they surf the previous webpages following the timer's guide. The second stage is repeated three times to reduce errors, and the web caches are cleared at the beginning of each run.

B. Experimental Results

Figure 7 shows the experimental results. We use "`Original`" to indicate the method that uses only cellular interface to download data. Users access almost the same number of webpages and their consumed energy is comparable, as shown in Fig. 7(a). In the traffic aggregation algorithm ("`Ours`"), User 1 is selected as the scheduler and it also works as the proxy sometimes. As User 1 listens to incoming requests from others and schedules the data requests (tasks), it consumes more energy than that in the original method. However, considering User 2 and User 3, they access many webpages via Bluetooth interfaces, and thus save lots of energy. In total, the energy consumed by the three users is reduced by 30.4%.

Normally, it takes users longer than the tail time to read webpages, so there is always a promotion delay (one or two seconds) for the cellular interface to access the webpage. However, in our traffic aggregation algorithm, such promotion delay can be avoided as the scheduler can schedule the task to a proxy whose cellular interface is still in the high power state. We compare the delay of both methods and the result is shown in Fig. 7(b). As can be seen, although our method has an extra Bluetooth delay for User 2 and User 3, the delay is small since the webpage size is small. More importantly,

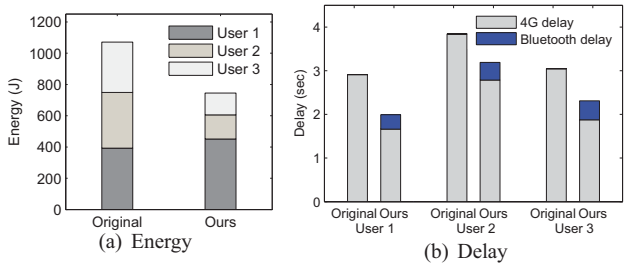


Fig. 7. Energy and delay comparison of algorithms with/without traffic aggregation. In the online traffic aggregation algorithm, User 1 works as the scheduler.

our method can avoid the promotion delay most of time. In average, our method can reduce the delay by about 31%.

Discussions: As can be seen from Figure 7, although the total energy consumed by the nodes in our algorithm can be saved by 30%, the scheduler (User 1) in our algorithm consumes a little more energy; i.e., about 8% more than the original method. This is because the cellular interface of the scheduler will stay in the high power state longer to serve other nodes. As a result, User 2 and User 3 can significantly reduce the energy consumption in our algorithm. As mentioned before, the role of the scheduler can be rotated inside the group, or taken by a node which has more energy or does not have energy limitation (e.g., can be recharged soon). If these nodes belong to the same organization, e.g., a group of soldiers or a group of fire fighters for the same mission, selecting the scheduler should be easy. For more general applications, credit-based approaches can be used to provide incentive for nodes to be schedulers. Since this is not the focus of this paper, we will not further discuss it.

VII. Performance Evaluations

A. Trace-Driven Simulation

Based on the collected trace as described in Section V-B, we evaluate the performance of our traffic aggregation algorithms. In the evaluations, we use "`Original`" to indicate the method that uses only cellular interface to download data. For our algorithms, we use "`BT`" and "`WiFi`" to indicate the traffic aggregation method that uses Bluetooth and WiFi direct as the P2P interfaces, respectively. The prefix of "`on-`" before a method is the online version of that method. For example, "on-WiFi" means that the online traffic aggregation algorithm using WiFi direct as the P2P interface.

In each test, we schedule the tasks to nodes with different methods and then compute the power consumption and delay using the models described in Section III. We assume each node (phone) turns on the P2P interface before it begins to access data and then turns it off if it predicts there is no data transmission in the next 10 minutes.

B. Comparisons on Energy Consumption and Delay

The power consumption and delay of different methods in 3G and 4G networks are shown in Fig. 8. The energy saving rate and delay reduction rate compared to the original method are shown on the top of each bar. Our method can save at

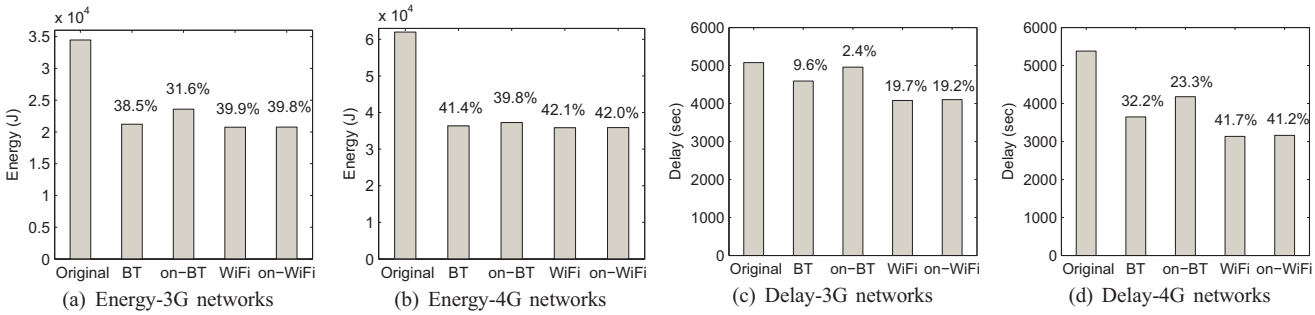


Fig. 8. Performance improvement of the traffic aggregation algorithms. The energy saving and delay reduction rates are marked on top of each bar.

least 32% of energy in 3G and 40% in 4G. By comparing Fig. 8(a) and Fig. 8(b), we have three observations. First, 4G consumes more energy than 3G because 4G has higher power consumption during data transmission and tail period, as shown in Table I. Second, WiFi direct saves more energy than Bluetooth, since it has higher energy efficiency during data transmission. Third, the online traffic aggregation algorithm can achieve similar energy saving rate as the optimal solution, although not as good as the optimal one. This validates the effectiveness of implementing online traffic aggregation on smartphones.

Considering delay reduction, WiFi direct can save 19% and 41% delay in 3G and 4G networks, as shown in Fig. 8(c) and Fig. 8(d), respectively. The large delay reduction in 4G comapred to 3G is mainly due to the promotion delay, i.e., about two seconds in 4G and one second in 3G as shown in Table I. For delay, Bluetooth underperforms WiFi direct because Bluetooth is much slower than WiFi direct.

C. Impact of Parameters

In this section, we evaluate the impact of three important parameters on the performance of traffic aggregation; i.e., the number of nodes, the task arrival interval, and the data size. We use energy saving rate and delay reduction rate as the metrics.

1) Impact of Node Number: The number of nodes affects the opportunity to reduce the tail time by traffic aggregation. In the extreme case, e.g., there is only one node, our method falls back to the original method.

We pick 2 to 5 nodes from the collected trace to form a group and evaluate the energy saving using Bluetooth and WiFi direct. The results are shown in Fig. 9(a) and Fig. 9(b). As can be seen, the energy saving rate increases linearly with the number of nodes, since more nodes will bring more opportunities to aggregate the tasks. The average energy saving rate in 4G increases 18% when the number of nodes increases from 2 to 5, and it increases 16% in 3G. Also, in all cases, WiFi direct outperforms Bluetooth, so we should use WiFi direct as the P2P interface if possible. For delay, as shown in Fig. 9(c) and Fig. 9(d), WiFi direct also outperforms Bluetooth.

2) Impact of Task Arrival Interval: The task arrival interval also affects the opportunity of traffic aggregation. For example, if each node generates one task per hour, it is hard to aggregate traffic even if there are 10 nodes in a group. In order to have more tasks to aggregate, we reduce the task arrival interval in the trace. Given a task arrival interval T and a ratio $\eta \in (0, 1]$, the compressed task arrival interval will be ηT.

The energy saving rates of different methods when the task arrival interval changes are shown in Fig. 10(a) and Fig. 10(b). If the task arrival interval is shorter, more tasks will have opportunities to be aggregated and more energy will be saved. When the task arrival interval is reduced to 40%, the energy saving rate increases 6% in 3G network and 8% in 4G networks. This improvement is not as much as changing the group size. The reason is that reducing the task arrival interval also makes the original method more energy efficient since more tail energy will be saved even if tasks are only scheduled locally.

Besides saving energy, our algorithm reduces delay when the task arrival interval is reduced, as shown in Fig. 10(c) and Fig. 10(d), except the online Bluetooth algorithm in 3G network, as it has fewer chances to schedule tasks according to Section V-D .

3) Impact of Data Size: The energy consumed by the P2P interface is mainly affected by the data size. As a result, the data size affects the energy saving of traffic aggregation. The average data size in the original trace is around 50KB. In this experiment, we increase the data size by 5 to 30 times, and show the results in Fig. 11(a). As can be seen, with the increase of data size, the energy saving rate drops quickly for Bluetooth, but maintains relative high for WiFi direct. To further explain the result, Fig. 11(b) shows the number of task requests sent to the proxy. As can be seen, Bluetooth sends fewer requests to the proxy when the data size increases, and hence has less chance of saving power. This is because, as a node receives a data request, it determines if it should send the data request to the scheduler based on the data size. With Bluetooth, when the data size is too large, the power consumption on the P2P interface will be too high, and hence the request will not be sent to the scheduler.

To summarize, traffic aggregation can save energy and reduce delay, especially when the number of nodes is high or when the task arrival interval is short. Bluetooth works well only when the average data size is smaller than hundreds of kilobytes. However, WiFi direct does not have such limitations, and it generally outperforms Bluetooth.

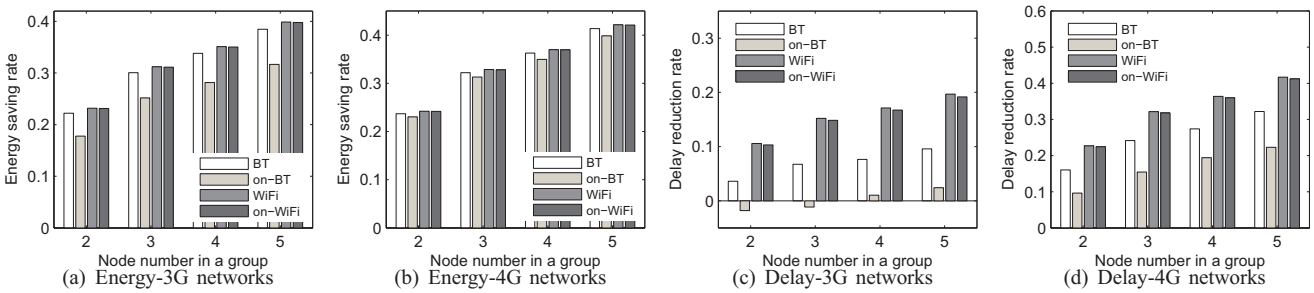


Fig. 9. Performance of traffic aggregation algorithms with different number of nodes in a group

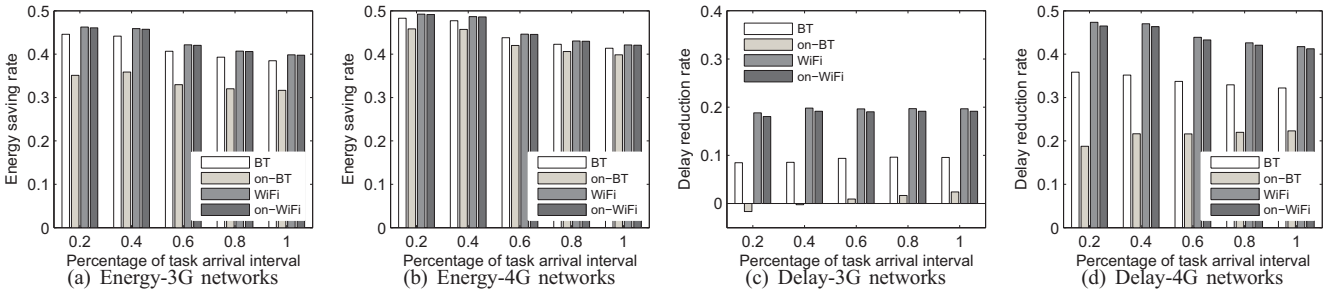


Fig. 10. Performance of traffic aggregation algorithms when task arrival interval changes

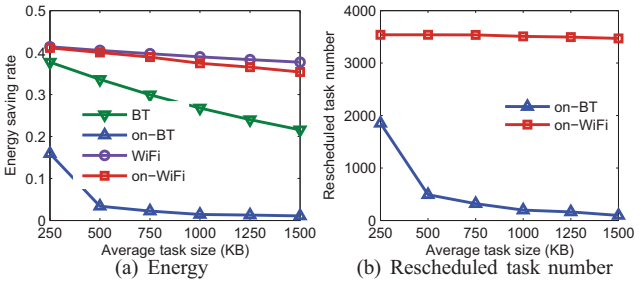


Fig. 11. Energy saving of traffic aggregation when data size changes in 4G networks

VIII. Conclusions

In this paper, we proposed to reduce the tail energy and data access delay by aggregating the data traffic of multiple nodes using their P2P interfaces. This traffic aggregation problem was formalized as finding the best task schedule to minimize energy. We first proposed an A^* algorithm to reduce the search space for finding the optimal schedule, and then introduced an online traffic aggregation algorithm. We have implemented the online traffic aggregation on Android smartphones, and built a small testbed. Experimental results show that our traffic aggregation algorithm can save 30% of energy and delay. Trace-driven simulations show that the proposed traffic aggregation algorithm is more effective when there are more nodes in the group or when the P2P interface is based on WiFi direct.

References

[1] "Configuration of fast dormancy. rel. 8," http://www.3gpp.org/.
[2] "iperf," http://iperf.fr/.
[3] "Zirco browser," https://code.google.com/p/zirco-browser/.
[4] G. Ananthanarayanan, V. N. Padmanabhan, L. Ravindranath, and C. A. Thekkath, "COMBINE: Leveraging the Power of Wireless Peers through Collaborative Downloading," in *ACM MobiSys, 2007*.
[5] G. Ananthanarayanan and I. Stoica, "Blue-Fi: Enhancing Wi-Fi Performance Using Bluetooth Signals," in *ACM MobiSys, 2009*.
[6] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *ACM IMC, 2009*.
[7] R. Dechter and J. Pearl, "Generalized Best-first Search Strategies and the Optimality of A*," *Journal of the ACM*, 32(3):505–536, July 1985.
[8] 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*.
[9] 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*.
[10] A. Iera, L. Militano, L. Romeo, and F. Scarcello, "Fair Cost Allocation in Cellular-Bluetooth Cooperation Scenarios," *IEEE Transactions on Wireless Communications*, 10(8):2566–2576, 2011.
[11] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and Scalable Distribution of Content Updates over a Mobile Social Network," in *IEEE INFOCOM, 2009*.
[12] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "MicroCast: Cooperative Video Streaming on Smartphones," in *ACM MobiSys, 2012*.
[13] R. Mittal, A. Kansal, and R. Chandra, "Empowering Developers to Estimate App Energy Consumption," in *ACM MobiCom, 2012*.
[14] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Periodic Transfers in Mobile Applications: Network-wide Origin, Impact, and Optimization," in *ACM WWW, 2012*.
[15] N. Ristanovic, J.-Y. Le Boudec, A. Chaintreau, and V. Erramilli, "Energy Efficient Offloading of 3G Networks," in *IEEE MASS, 2011*.
[16] 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*.
[17] C.-L. Tsao and R. Sivakumar, "On Effectively Exploiting Multiple Wireless Interfaces in Mobile Hosts," in *ACM CoNEXT, 2009*.
[18] X. Xing, S. Mishra, and X. Liu, "ARBOR: Hang Together Rather Than Hang Separately in 802.11 WiFi Networks," in *IEEE INFOCOM, 2010*.
[19] B. Zhao, B. C. Tak, and G. Cao, "Reducing the Delay and Power Consumption of Web Browsing on Smartphones in 3G Networks," in *IEEE ICDCS, 2011*.
[20] B. Zhao, Q. Zheng, G. Cao, and S. Addepalli, "Energy-Aware Web Browsing in 3G Based Smartphones," in *IEEE ICDCS, 2013*.
[21] X. Zhuo, W. Gao, G. Cao, and Y. Dai, "Win-Coupon: An Incentive Framework for 3G Traffic Offloading," in *IEEE ICNP, 2011*.