Energy-Traffic Tradeoff Cooperative Offloading for Mobile Cloud Computing

Jian Song*, Yong Cui*, Minming Li†, Jiezhong Qiu* and Rajkumar Buyya‡
*Tsinghua University, Beijing, China
Email: song-j12@mails.tsinghua.edu.cn, cuiyong@tsinghua.edu.cn, qjz12@mails.tsinghua.edu.cn
†Department of Computer Science, City University of Hong Kong, Hong Kong, China
Email: minmli@cs.cityu.edu.hk
‡Department of Computing and Information Systems, The University of Melbourne, Australia
Email: rbuyya@unimelb.edu.au

***Abstract*—This paper presents a quantitative study on the energy-traffic tradeoff problem from the perspective of entire Wireless Local Area Network (WLAN). We propose a novel Energy-Efficient Cooperative Offloading Model (E2COM) for energy-traffic tradeoff, which can ensure the fairness of energy consumption of mobile devices and reduce the computation repetition and eliminate the Internet data traffic redundancy through cooperative execution and sharing computation results. We design an Online Task Scheduling Algorithm (OTS) based on a pricing mechanism and Lyapunov optimization to address the problem without predicting future information on task arrivals, transmission rates and so on. OTS can achieve a desirable trade-off between the energy consumption and Internet data traffic by appropriately setting the tradeoff coefficient. Simulation results demonstrate that E2COM is more efficient than no offloading and cloud offloading for a variety of typical mobile devices, applications and link qualities in WLAN.**

I. Introduction

Mobile devices (e.g. smart phones) have become increasingly popular in our daily lives, whereas the capacity of mobile devices is severely constrained by the restricted battery power. An efficient way to reduce the computation overhead is to offload computing tasks to powerful machines or to the cloud. Mobile devices can save energy and reduce execution delay of applications through offloading tasks to the cloud. Several solutions have been proposed for computation offloading, such as MAUI [1], Clonecloud [2], SmartDiet [3]. However, these research efforts of offloading technology mainly focus on optimizing energy consumption of a single device.

On the other hand, though the coverage ratio of 3G networks is much higher than WiFi networks [4], there exist some challenges for offloading tasks to remote cloud through 3G [5]. For example, 3G provides Internet services with lower bandwidth, higher communication latency and higher energy consumption compared with WLAN [6]. Moreover, the growth rate of 3G network capacity cannot catch up with the demand of mobile Internet data traffic [7]. WLAN is considered as a solution to ease the traffic pressure on 3G. However, many mobile communications access the Internet through the same Access Controller (AC) or Access Points (AP) simultaneously, which causes serious congestion and lower available bandwidth in large-scale WLAN. Conserving network-wide energy consumption and controlling the Internet data traffic are becoming a major concern for network operators [8].

In this paper, we propose an Energy-Efficient Cooperative Offloading Model (E2COM) in one-hop, low-latency and large-scale WLAN, which aims to minimize the long-term energy consumption of mobile devices with Internet data traffic constraint from the entire WLAN perspective. E2COM Controller is arranged on AC to schedule tasks for every mobile user based on energy consumption and Internet data traffic. Mobile devices can execute tasks locally, offload tasks to other devices or to the cloud in accordance with the decision-making of E2COM Controller. Many types of tasks can benefit from E2COM. Examples of these tasks consist of location information acquisition, optical character recognition (OCR), image processing, and so on. Moreover, we can use fingerprints proposed in [9], [10] to describe the similarity of tasks. Mobile devices can share computation results of similar tasks with each other to reduce the computation repetition and eliminate the Internet data traffic redundancy.

In order to reduce the Internet data traffic, we design a non-competitive pricing mechanism which involves a financial deficit to record the amount of offloading services that each device has already received. A device cannot gain more offloading services, if its bill backlog is larger than a threshold. On the other hand, a device can reduce its bill backlog by providing services for others. The reduction amount of the bill backlog depends on the data size of the tasks.

To address the task allocation and execution problem, we design an online task scheduling algorithm based on Lyapunov optimization [11] to minimize the network-wide long-term average energy consumption while stabilizing the deficit queues of all devices in the WLAN. Moreover, the algorithm can achieve the energy consumption and the Internet data traffic tradeoff by adjusting the tradeoff coefficient V.

The rest of this paper is organized as follows. Section II is the description for the E2COM framework and the model of computation and transmission. In Section III, we design OTS to solve the optimization problem. Simulation results are presented in Section IV. Section V concludes this paper.

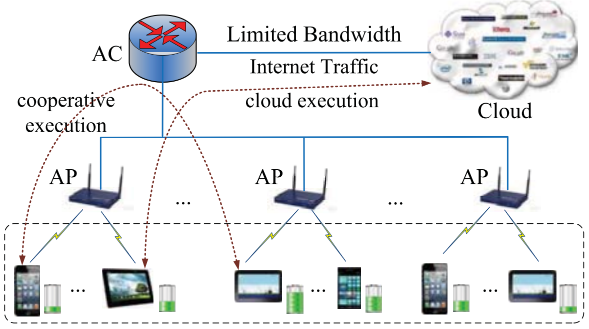


Fig. 1. The Scenario of E2COM

II. Framework and Modeling

A. System Framework

E2COM is arranged in a large-scale WLAN, as shown in Fig. 1. In order to reduce the extra cost of equipments, E2COM Controller is arranged on AC and records profiles of every mobile device, e.g., CPU power, transmission rate and so on.

Because of the limited storage capacity of AC, E2COM Controller does not store task execution results, but stores the fingerprints [9] of tasks executed recently. When a device has a task to execute, it makes a request to the E2COM Controller. The E2COM Controller schedules the task based on energy consumption and bill backlog of the device. Since the data size of the request is very small, we can ignore the energy consumption of sending the request to E2COM Controller.

Since the tasks generated by different mobile users may be similar (e.g. query the same keyword through search engine or request for the same video, location information), devices can share computation results with each other to achieve higher performance and lower energy consumption. If a task generated by device A has been completed by device B, A can directly request the result of the task from B. This can reduce the repetition of computation and Internet data traffic. Intuitively, when the repetition rate of tasks is high, the proposed model can save more energy by reducing the repetitive execution. On the other hand, if the repetition rate is low, the proposed model can also achieve network-wide energy efficiency by offloading the tasks to the cloud.

Assume that there are n devices in the WLAN. They have m tasks to execute in a long time period T, denoted by set $\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$. Every task belongs to a device. We use C_h^i to represent task C_h which belongs to device i, where $h = 1, 2, \cdots, m$. We further represent the profile of a task which belongs to device i as $C_h^i(ID, D_h^{in}, D_h^{out})$, where ID, D_h^{in} and D_h^{out} denote the type of C_h^i, the data size of the input and output respectively.

B. Energy Consumption of Computation

CPU is the dominant energy consumer on a device for executing a task. The energy consumption is determined by CPU workload, CPU clock frequency, device type and so on. It is difficult to module the energy consumption of CPU. [12] depicts the CPU energy consumption from computing efficiency perspective, i.e., a measure for the amount of computation that can be performed with given energy (in cycles per joule). It shows that dynamic voltage and frequency scaling (DVFS) does affect the energy efficiency of computing but not radically. The number of CPU cycles depends on the input data size and the type of the task [13].

The relationship between the input data size of the task and the number of CPU cycles needed by the task is related to the task type ([13], [12]). We assume that device i needs N_h CPU cycles to execute computation task C_h^i. It can be derived from [13] as follows,

$$N_h = f_X(D_h^{in}), \tag{1}$$

where the function $f_X(\cdot)$ is related to the application type X, which is determined by the task ID.

For some popular applications (e.g., deflate compression algorithm, $x264$ video encoder and so on), the CPU cycles needed by a computation task can be expressed as a linear function of the input data size as $N_h = X \cdot D_h^{in}$ [12], where the complexity coefficient X is the ratio of CPU cycles and the input data size which is related to the application type.

When device j executes task C_h^i, the computation energy consumption of device j can be defined formally as follows.

Definition 1. Energy Consumption of Computation *is given by [14]*

$$\mathcal{E}_C^j(C_h^i) = (\rho_0^j + \rho_1^j f_j^3)\frac{N_h}{f_j}, \tag{2}$$

where ρ_0^j, ρ_1^j represent the static power and the dynamic power coefficients respectively of j's CPU, f_j is the CPU clock frequency of device j.

If device j has already executed a task and stored the computation result, the energy consumption for executing the same task (with the same ID) is close to 0. For example, if device i nearby requires position information, it can request j to complete the positioning task C_h^i instead of turning on the GPS and computing the position by itself. If device j has the most recent location information, it can send the position information to i directly. Then, the energy consumption of computation that j completes task C_h^i is 0.

C. Energy Consumption of Offloading

The WLAN can be modeled by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where $\mathcal{V}$ and $\mathcal{L}$ are the sets of nodes and directional edges. Each node $i \in \mathcal{V}$ corresponds to a device in the network. An edge $(i, j) \in \mathcal{L}$ in the graph represents a wireless link from node i to node j. Each edge (i, j) is associated with three non-negative real number weights $(\rho_T^{ij}, \rho_R^{ij}, \phi_T^{ij})$, where ρ_T^{ij}, ρ_R^{ij} are transmitting and receiving power of i to and from j, and ϕ_T^{ij} is the transmission rate on link (i, j) at the current time. The energy consumption of transmission from i to j can be formulated as $\rho_T^{ij} \cdot (D_h^{in}/\phi_T^{ij})$ [15]. We set $\rho_T^{ii} = 0$ and $\phi_T^{ii} = \infty$. We assume that the E2COM Controller can collect and store these information in real time.

If device i requests device j to execute a task, we define i as the client and j as the server. In order to characterize the energy consumption of each device in the task offloading process clearly, we divide the energy consumption of offloading into client energy consumption and server energy consumption.

Definition 2. Client Energy Consumption of Offloading *is the total energy consumption of client i when task C_h^i is offloaded from client i to server j, including transmission energy consumption of sending input data and receiving output data of the task, i.e.,*

$$\mathcal{E}_L^{ij}(C_h^i) = \rho_T^{ij} \cdot \frac{D_h^{in}}{\phi_T^{ij}} + \rho_R^{ij} \cdot \frac{D_h^{out}}{\phi_T^{ji}}. \qquad (3)$$

Note that $\mathcal{E}_L^{ii}(C_h^i) = 0$, i.e., local execution will not generate transmission energy consumption.

Definition 3. Server Energy Consumption of Offloading *is the sum of transmission and computation energy consumption of server j when a task C_h^i is offloaded from client i to server j, and the transmission energy consumption includes transmission energy consumption of receiving input data and sending output data of the task, i.e.,*

$$\mathcal{E}_R^{ij}(C_h^i) = \rho_R^{ji} \cdot \frac{D_h^{in}}{\phi_T^{ij}} + \rho_T^{ji} \cdot \frac{D_h^{out}}{\phi_T^{ji}} + \mathcal{E}_C^{j}(C_h^i). \qquad (4)$$

If a same task as C_h^i has been executed by server j, the computation energy consumption can be defined as 0. Moreover, when $i = j$, equation (4) is consistent with equation (2), i.e., $\mathcal{E}_R^{ii}(C_h^i) = \mathcal{E}_C^{i}(C_h^i)$.

In order to measure the network-wide energy consumption of executing tasks, we define the energy consumption of offloading, which includes both client and server energy consumption. We consider the cloud as a special server denoted as $n+1$. The energy consumption of the cloud is not included in the network-wide energy consumption.

Definition 4. Energy Consumption of Offloading *is the energy consumption of mobile devices when task C_h^i is executed remotely, which can be formulated as,*

$$\mathcal{E}_O^{ij}(C_h^i) = \begin{cases} \mathcal{E}_L^{ij}(C_h^i) + \mathcal{E}_R^{ij}(C_h^i), & j = 1, \dots, n; \\ \mathcal{E}_L^{ij}(C_h^i), & j = n+1. \end{cases}$$

III. Task Scheduling Mechanism

In the large-scale WLAN, the E2COM Controller needs to find a proper task allocation scheme. The task allocation can be formalized as an optimization problem. The network-wide energy consumption is the optimization objective, and the Internet data traffic is used as the constraint.

A. Traffic-aware Energy Optimization Problem Formulation

In order to achieve network-wide energy consumption and data traffic optimization, the E2COM Controller needs to select the appropriate server (another device, the cloud or the device itself) for every task C_h^i, where $C_h^i \in \mathbf{C}$.

E2COM is operated in a discrete time manner. Each time slot matches the timescale at which the offloading decision is made for one task. In each time slot t_h, E2COM makes a decision on a vector $\vec{\alpha}(C_h^i) = (\alpha_1(C_h^i), \cdots, \alpha_{n+1}(C_h^i))$ as,

$$\alpha_j(C_h^i) = \begin{cases} 1, & j \text{ executes task } C_h^i; \\ 0, & \text{otherwise,} \end{cases}$$

where $j = 1, 2, \cdots, n, n+1$. Especially, $\alpha_{n+1}(C_h^i) = 1$ denotes that task C_h^i will be offloaded to the cloud. The energy consumption of executing task C_h^i can be denoted as,

$$\mathcal{E}(C_h^i) = \sum_{j=1}^{n+1} \alpha_j(C_h^i) \cdot \mathcal{E}_O^{ij}(C_h^i).$$

If task C_h^i is offloaded to the other device or cloud, the produced Internet data traffic can be approximately expressed as $D_T(C_h^i) = D_h^{in} + D_h^{out}$. Since local execution and cooperative execution between devices will not generate Internet data traffic, the Internet data traffic caused by the execution of task C_h^i can be denoted as $\mathcal{D}(C_h^i) = \alpha_{n+1}(C_h^i) \cdot D_T(C_h^i)$.

The optimization objective is to minimize the *Network-Wide Energy Consumption* with the Internet data traffic constraints (denoted by Ψ) for every device, i.e.,

$$\min \sum_{h=1}^{m} \mathcal{E}(C_h^i) \qquad (5)$$

subject to:

$$\begin{aligned} &\sum_{h=1}^{m} \mathcal{D}(C_h^i) < \Psi && i = 1, 2, \cdots, n \\ &\alpha_j(C_h^i) \in \{0, 1\}, && h = 1, 2, \cdots, m \\ & && j = 1, 2, \cdots, n+1 \\ &\sum_{j=1}^{n+1} \alpha_j(C_h^i) = 1, && h = 1, 2, \cdots, m. \end{aligned}$$

With a bound on the Internet data traffic, the offline energy minimization problem in (5) can be proved NP-hard by a reduction from the knapsack problem.

B. Online Task Scheduling Algorithm

In order to solve the problem in (5) effectively, we design the Online Task Scheduling Algorithm, which is based on Lyapunov optimization and makes a tradeoff between the energy consumption and Internet data traffic.

1) Service Pricing: E2COM includes a non-competitive pricing approach to encourage cooperation between mobile devices. If a device's bill backlog is larger than a threshold, it will not be allowed to receive more offloading services from E2COM. A device can pay its bill by executing tasks for others. To enjoy offloading services continually, devices bill backlogs cannot increase unboundedly.

We treat cloud as a special device in the WLAN. It can increase other devices' bill backlogs by providing offloading services, but cannot consume other devices' bill backlogs. The Internet data traffic can be reduced, if devices offloads tasks to other devices instead of cloud. In order to achieve network-wide (client devices and server devices) cost balance, we design an asymmetric charging mode according to the contributions for reducing the Internet data traffic.

Definition 5. Service Charging Function *can be denoted as follows. The bill backlog of client i will increase by* $b_i(t_h) = f(D_T(C_h^i))$*, when client i has a task* C_h^i *to execute at time slot* t_h*; if* C_h^i *is executed by itself, the bill backlog of i will decrease by* $d_i(t_h) = f(D_T(C_h^i))$*; if* C_h^i *is offloaded from i to j, the bill backlog of server j will decrease by* $d_j(t_h) = f(D_T(C_h^i))$.

The increased bill backlog $b_i(t_h)$ $(i = 1, \cdots n)$ is to tackle the randomness of incoming tasks. If we set $f(D_T(C_h^i)) = D_T(C_h^i)$, we can get an informative conclusion. Intuitively, when task C_h^i is offloaded from i to j, the bill backlog of device j decreases by $d_j(t_h)$ which is proportional to the Internet data traffic saved for the network, and the bill backlog of i increases by $b_i(t_h)$ which is exactly related to the data traffic if the task is offloaded to be executed.

We denote the n devices' total amount of bills at time slot t_h as $\mathbf{Q}(t_h) \triangleq (Q_1(t_h), \cdots, Q_n(t_h))$, where t_h denotes the time slot at which the task C_h^i is executed. For each device i, $Q_i(t_h)$ represents its bill backlog at the beginning of time slot t_h.

The service bills of all devices generated in every time slot t_h are denoted as $\mathbf{b}(t_h) \triangleq (b_1(t_h), \cdots, b_n(t_h))$, which are added in their corresponding queues $\mathbf{Q}(t_h)$. Because of executing tasks for others, devices' bills are paid partially, which are denoted as $\mathbf{d}(t_h) \triangleq (d_1(t_h), \cdots, d_n(t_h))$. So the bill backlog evolves as $Q_i(t_{h+1}) = \max[Q_i(t_h) - d_i(t_h) + b_i(t_h), 0]$ with an initially empty bill (i.e., $Q_i(t_0) = 0$).

In order to limit the bill backlog of every device, we require all the bills to be stable in the time average sense, i.e.,

$$\overline{\mathbf{Q}} = \limsup_{m\to\infty} \frac{1}{m} \sum_{h=1}^{m} \sum_{i=1}^{n} \mathbb{E}\{|Q_i(t_h)|\} < \infty. \tag{6}$$

2) Online Schedule: The network-wide average energy consumption of executing the task set $\mathbf{C}$ in a long time period T can be expressed as,

$$\mathcal{E} = \limsup_{m\to\infty} \frac{1}{m} \sum_{h=1}^{m} \mathbb{E}\{|\mathcal{E}(C_h^i)|\}. \tag{7}$$

Then, we obtain a new optimization problem, i.e.,

$$\min \mathcal{E} \tag{8}$$

subject to:

$$\begin{aligned} &\overline{\mathbf{Q}} < \infty \\ &\alpha_j(C_h^i) \in \{0,1\}, \quad h = 1, 2, \cdots, m \\ &\qquad\qquad\qquad\qquad j = 1, 2, \cdots, n+1 \\ &\sum_{j=1}^{n+1} \alpha_j(C_h^i) = 1, \quad h = 1, 2, \cdots, m \end{aligned}$$

Let $\mathcal{E}^*$ be the target value of the optimization problem defined in (8). In each time slot t_h, E2COM makes an online offloading decision, with the objective of minimizing the time average energy consumption under bill backlog constraints for all devices.

Based on Lyapunov optimization, we first present our *Lyapunov function*, $L(\mathbf{Q}(t_h))$, which is a scalar measure of the total bill backlogs of all the devices in the WLAN, defined as,

$$L(\mathbf{Q}(t_h)) \triangleq \frac{1}{2} \sum_{i=1}^{n} Q_i^2(t_h). \tag{9}$$

A larger value of $L(\mathbf{Q}(t_h))$ implies that at least one bill backlog is large. In order to ensure that the bill backlog of each device is smaller than the threshold, we need to keep the *Lyapunov function* at a small value. We next introduce the *Lyapunov drift* $\triangle(\mathbf{Q}(t_h))$ as,

$$\triangle(\mathbf{Q}(t_h)) \triangleq \mathbb{E}\{L(\mathbf{Q}(t_{h+1})) - L(\mathbf{Q}(t_h)) \mid \mathbf{Q}(t_h)\}, \tag{10}$$

which is the expected change in the *Lyapunov function* over one time slot. Following the Lyapunov optimization approach [11], we then add the expected energy consumption over one time slot to both sides of (10), and then lead to the definition of *drift-plus-penalty* term.

Definition 6. Drift-Plus-Penalty *can be calculated as follows,*

$$DPP(t_h) = \triangle(\boldsymbol{Q}(t_h)) + V \cdot \mathbb{E}\{\mathcal{E}(C_h^i) \mid \boldsymbol{Q}(t_h)\}.$$

Here V is a non-negative tradeoff coefficient that is chosen to adjust the performance tradeoff, i.e., how much we care about the energy consumption compared to the bill backlog. The key derivation step is to obtain an upper bound on the *drift-plus-penalty*. Given any possible bill backlogs $\mathbf{Q}(t_h)$, arrival rates $b_i(t_h)$ and service rates $d_i(t_h)$ at each bill, under any possible decision $\vec{\alpha}(t_h)$, the constraint is satisfied as follows,

$$\begin{aligned} DPP(t_h) \leq\, & B + V \cdot \mathbb{E}\{\mathcal{E}(C_h^i) \mid \mathbf{Q}(t_h)\} + \\ & \sum_{i=1}^{n} \mathbb{E}\{Q_i(t_h)(b_i(t_h) - d_i(t_h)) \mid \mathbf{Q}(t_h)\}. \end{aligned} \tag{11}$$

Following the design principle of *Lyapunov framework*, the objective of our optimal offloading decision vector $\vec{\alpha}(C_h^i)$ is to minimize the upper bound of the *drift-plus-penalty* term, i.e., we need to minimize the right hand side of (11) in every time slot t_h. Since only the terms $\mathcal{E}(C_h^i)$ and $Q_i(t_h)(b_i(t_h) - d_i(t_h))$ depend on the decision vector $\vec{\alpha}(C_h^i)$, we can minimize the bound of the right hand side of (11) by minimizing these terms. Thus, our algorithm finally minimizes the simplified term as,

$$DPP_{bound} = V \cdot \mathcal{E}(C_h^i) + \sum_{i=1}^{n} Q_i(t_h)(b_i(t_h) - d_i(t_h)). \tag{12}$$

Based on (12), we design the Online Task Scheduling Algorithm to allocate tasks among devices. When device i requests the result of a task, the algorithm is triggered. The bill backlog of i is updated in line 1. Lines 3-6 compute DPP of every device j according to (12). Lines 7-10 record the smallest DPP_j and determine the offloading decision vector $\alpha(C_h)$. According to the *Lyapunov Optimization Approach*, we will allocate the task to the device with the smallest DPP value. Lines 13-15 update the bill backlogs of corresponding devices. Since cloud execution will do nothing to the bill backlog, the algorithm will not do anything when cloud execution.

```
Online Task Scheduling Algorithm
Input: $\{\rho_{i1}^T, \cdots, \rho_{in}^T\}$, $\{\rho_{i1}^R, \cdots, \rho_{in}^R\}$, $\{\phi_{i1}, \cdots, \phi_{in}\}$,
    $C_h^i(D_h^{in}, D_h^{out})$, $\{Q_1(t_h), \cdots, Q_n(t_h)\}$
Output: $\{\alpha_1(C_h^i), \cdots, \alpha_{n+1}(C_h^i)\}$,
    $\{Q_1(t_{h+1}), \cdots, Q_n(t_{h+1})\}$
 1: $Q_i(t_{h+1}) \leftarrow Q_i(t_h) + b_i(t_h)$
 2: for $j \leftarrow 1$ to $n+1$ do
 3:     $DPP_j \leftarrow V \cdot \mathcal{E}(C_h^i)$
 4:     for $k \leftarrow 1$ to $n$ do
 5:         $DPP_j \leftarrow DPP_j + Q_k(t_h)(b_k(t_h) - d_k(t_h))$
 6:     end for
 7:     if $DPP_j < DPP_{bound}$ then
 8:         $DPP_{bound} \leftarrow DPP_j$
 9:         $\vec{\alpha}(C_h^i) \leftarrow e_j$ //where $e_j$ denotes the vector with a $1$
            in the $j$th coordinate and $0'$s elsewhere
10:     end if
11: end for
12: $j \leftarrow$ the dimension which $\alpha_j(C_h^i) = 1$
13: if $j \neq n+1$ then
14:     $Q_j(t_{h+1}) \leftarrow Q_j(t_h) - d_j(t_h)$
15: end if
16: return
    $\{\alpha_1(C_h^i), \cdots, \alpha_{n+1}(C_h^i)\}$, $\{Q_1(t_{h+1}), \cdots, Q_n(t_{h+1})\}$
```

Note that there will be at most two devices related to each task. When computing DPP of every device j in order to find the smallest one, we only need to calculate the related DPP, i.e., device i and device j. In this way, the time complexity of OTS is $O(n)$, where n is the number of devices in the WLAN.

Substituting constants ϵ, $\mathcal{E}^*$ into inequality (11), and calculating the time average bill size for every slot $t_h > 0$, we can obtain the performance bounds of OTS. Given a tradeoff coefficient V, the time average energy deviates by $O(1/V)$ from optimality at most, while the bill backlog is bounded by $O(V)$. We can use an arbitrarily large V to make the time average energy cost $\mathcal{E}$ close to optimum $\mathcal{E}^*$.

IV. Performance Evaluation

We evaluate the performance of E2COM by simulations on energy consumption and Internet data traffic. No Offload (all the tasks are executed on the mobile devices locally), Cloud Offload (all the tasks are executed on the cloud remotely) and an Energy Greedy Algorithm (GA) are taken as the performance reference for comparison on the same topology in simulations. GA ignores the bound on the Internet data traffic in (5), and simply selects the server with the smallest energy consumption in each time slot.

A. Evaluation Setup and Methodologies

We consider a WLAN with 50 mobile devices and 5000 tasks which belong to these devices to be executed in a long time period T. The CPU static power and the dynamic power coefficients of every device are set from 0.5 to 1 randomly, and the CPU clock frequency is set from $1GHz$ to $1.5GHz$ randomly. The transmission rate on link (i, j) is a random value between $15Mbps$ and $20Mbps$ [6], $\forall i, j \in \mathcal{V}$. The transmission rate between i and the cloud via E2COM Controller is a random value between $1Mbps$ and $2Mbps$. The transmission power of every device is set between $0.5W$ and $1W$ determined by the transmission rate [6].

B. Simulation Results

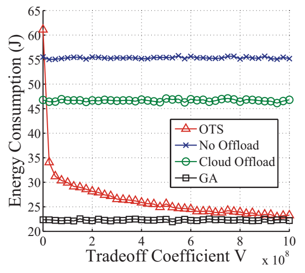
1) Performance with Different V: This simulation illustrates how the parameter V controls the energy consumption and bill backlog tradeoff. We set $X = 1000$, the repetition rate of tasks 40% and V increasing from 0 to 10^9 in step 2.5×10^7. We present the average energy consumption, average bill backlog and Internet data traffic of executing 5000 tasks in Fig. 2a, Fig. 2b and Fig. 2d respectively.

When V goes from 0 to 10^9, the average energy consumption drops from $61J$ to $23J$, and the average bill backlog grows from 0.6 to 30.3 in OTS. A notable phenomenon is that the average energy consumption falls quickly when V is small and then decreases slowly, but the average bill backlog grows linearly with V increasing. This result confirms the $[O(1/V), O(V)]$ tradeoff between energy consumption and bill backlog. Users can select the appropriate V to achieve the best energy-efficiency, according to their budget for the Internet data traffic.

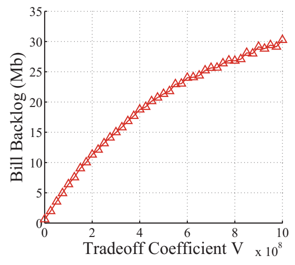
Moreover, from Fig. 2a and Fig. 2d, we can see that the energy consumption and Internet data traffic of OTS get closer to the Energy Greedy Algorithm as V increases. According to (12), the Energy Greedy Algorithm can be understood as a limiting case of OTS ($V \to \infty$). Our algorithm can minimize the energy consumption arbitrarily close to the Energy Greedy Algorithm and control the Internet data traffic by appropriately setting the tradeoff coefficient V. The energy consumption changes with V increasing because the number of tasks executed by itself, the cloud or other devices is changing with different V, as shown in Fig. 2c.

In E2COM, a task can be executed by the device itself (Local), the cloud (Cloud) or other devices, which may have executed the same task before (Share), or not (Cooperation). Fig. 2c shows the number of tasks executed by itself, the cloud or other devices with different V. Most tasks are executed by other devices when $V = 0$. In this case, as bill backlog is the only optimization objective, OTS will select the server with the longest bill backlog. As V increases, the number of tasks sharing results stays around 2500. It is 50% of the total number of tasks and slightly higher than the repetition rate (40%). This is because some task results have been shared several times. Because the importance of energy optimization grows as V increases, the number of Cloud Execution increases and the number of Local Execution decreases slowly.

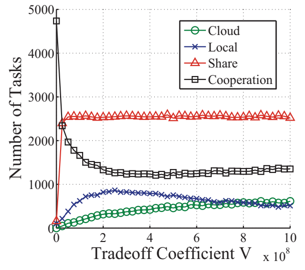
2) Performance with Different Repetition Rates of Tasks: In this part, we study the performance of OTS under different



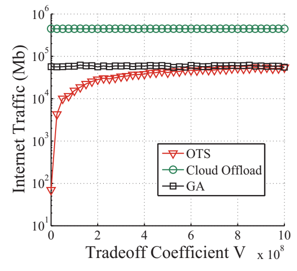
(a) The Average Energy Consumption



(b) The Average Bill Backlog

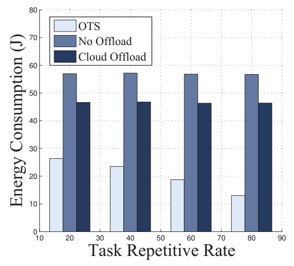


(c) The Number of Tasks Executed Locally or Remotely in E2COM

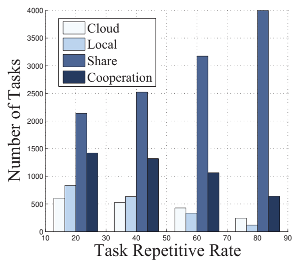


(d) The Internet Data Traffic

Fig. 2. The Performance of E2COM with Different V



(a) The Average Energy Consumption



(b) The Number of Tasks

Fig. 3. The Network-wide Average Energy Consumption and Number of Tasks Executed Locally or Remotely with Different Repetition Rates of Tasks

repetition rates of tasks as 20%, 40%, 60% and 80%. Parameter V is also set to 8×10^7. Fig. 3a illustrates the average energy consumption of OTS. Fig. 3b shows the number of tasks executed locally or remotely.

The average energy consumption of OTS is lower than the Cloud Offload all the time when we set the appropriate value of V. Moreover, the higher the repetition rate of tasks is, the more energy is saved by OTS than Cloud Offload. This is because most of the tasks are offloaded to the cloud to save energy, when the repetition rate of tasks is low. But in order to control the bill backlogs of devices, some tasks select Local Execution or Collaborative Execution. This is not the most energy efficient way, but can limit the Internet data traffic. When the repetition rate of tasks is high, mobile users can share more results of tasks with each other to save energy, as the energy consumption of computation approximates 0 for executing the same task once more.

V. Conclusions And Future Work

In this paper, we propose E2COM in large-scale WLAN to save energy and reduce the Internet data traffic of the WLAN, and design a non-competitive pricing mechanism to encourage cooperation among mobile users. Based on the pricing mechanism and Lyapunov optimization, we design OTS, which can minimize the network-wide long-term energy consumption and limit the Internet data traffic by reducing the repetitive computation. OTS does not rely on any prediction for future information on tasks arrivals, transmission rate and so on, which makes the solution more practical. As future work, we will evaluate the performance of E2COM based on real traffic traces and extend our model to consider implications of the QoS constraints of different type applications.

Acknowledgement

This work is supported by National Natural Science Foundation of China (no. 61120106008, 61225012, 61161140454), National 863 project (no. 2013AA010401), CNGI project (CNGI-12-03-003).

References

[1] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *ACM MobiSys*, 2010.

[2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *EuroSys*, 2011.

[3] A. Saarinen, M. Siekkinen, Y. Xiao, J. Nurminen, M. Kemppainen, and P. Hui, "Offloadable apps using smartdiet: towards an analysis toolkit for mobile application developers," *CoRR, abs/1111.3806*, 2011.

[4] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3g using wifi," in *ACM MobiSys*, 2010.

[5] M. Sharifi, S. Kafaie, and O. Kashefi, "A survey and taxonomy of cyber foraging of mobile devices," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, 2012.

[6] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *ACM SIGCOMM IMC*, 2009.

[7] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: how much can wifi deliver?" in *Proceedings of the 6th International Conference. Co-NEXT '10*. ACM, 2010.

[8] C. Jiang, Y. Shi, Y. T. Hou, and W. Lou, "Cherish every joule: Maximizing throughput with an eye on network-wide energy consumption," in *IEEE INFOCOM*, 2012.

[9] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *ACM SIGCOMM*, 2000.

[10] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: the implications of universal redundant traffic elimination," in *ACM SIGCOMM*, 2008.

[11] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.

[12] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, 2010.

[13] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones," in *IEEE INFOCOM*, 2012.

[14] R. Xu, D. Zhu, C. Rusu, R. Melhem, and D. Mossé, "Energy-efficient policies for embedded clusters," in *ACM SIGPLAN Notices*, vol. 40, no. 7. ACM, 2005, pp. 1–10.

[15] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.