Software Defined Cooperative Offloading for Mobile Cloudlets

Yong Cui, Jian Song, Kui Ren, *Fellow, IEEE, Member, ACM*, Minming Li, Zongpeng Li, Qingmei Ren, and Yangjun Zhang

***Abstract*— Device to Device communication enables the deployment of mobile cloudlets in LTE-advanced networks. The distributed nature of mobile users and dynamic task arrivals makes it challenging to schedule tasks fairly among multiple devices. Leveraging the idea of software defined networking, we propose a software defined cooperative offloading model (SDCOM), where the SDCOM controller is deployed at the PDN gateway and schedules tasks in a centralized manner to save the energy of mobile devices and reduce the traffic on access links. We formulate the minimum-energy task scheduling problem as a 0-1 knapsack problem and prove its NP-hardness. To compute the optimal solution as a benchmark, we design the conditioned optimal algorithm based on the aggregated analysis of energy consumption. The greedy algorithm with a polynominal-time complexity is proposed to solve large-scale problems efficiently. To address the problem without predicting future information on task arrivals, we further design an online task scheduling algorithm (OTS). It can minimize the energy consumption arbitrarily close to the optimal solution by appropriately setting the tradeoff coefficient. Moreover, we extend OTS to design a proportional fair online task scheduling algorithm to achieve the fair energy consumption among mobile devices. Extensive trace-based simulations demonstrate the effectiveness of SDCOM for a variety of typical mobile devices and applications.**

***Index Terms*— Software defined networking, mobile cloudlet, energy-efficiency, offloading.**

I. Introduction

MOBILE applications are witnessing rapid developments, calling for enhanced storage and computational capacity on mobile devices. However, the capacity of a mobile device is fundamentally limited by its physical size. A series of recent offloading solutions [1], [2], MAUI [3] and Clonecloud [4] aim to augment the capacities of mobile devices. Offloading tasks to a remote cloud through cellular networks is costly [1] due to the lower bandwidth, higher WAN (Wide Area Network) latency and higher energy consumption [5]. Moreover, the growth rate of current cellular system capacity cannot catch up with the demand of mobile data traffic, such as online video streaming and multimedia file sharing [6]. Limiting the traffic usage of access links is becoming a major concern of network operators [7].

To address these challenges, *mobile cloudlet* [8] was proposed. A group of nearby mobile devices are connected wirelessly, e.g., using WiFi or Bluetooth. Mobile devices can be providers as well as clients of computing service. Potential application scenarios of mobile cloudlet include multimedia sharing at an event, location information acquisition and language translation for a group of tourists. Device to Device (D2D) communication is specified by 3GPP in LTE-Advanced, which brings *mobile cloudlet* to reality. However, due to the distributed nature of mobile users and dynamic task arrivals, effectively scheduling tasks to appropriate devices and the cloud remains a challenging network-wide optimization problem. Such optimization should further ensure energy fairness among mobile devices in the process of cooperation.

The recent Software Defined Networking (SDN) paradigm ([9]–[11]) enables logically centralized control over the distributed information among mobile devices [12]. Leveraging SDN, we design a Software Defined Cooperative Offloading Model (SDCOM) for mobile cloudlets. An SDCOM Controller deployed at the LTE Gateway periodically collects information of mobile devices, forming a global view of network states. It schedules tasks for every mobile device based on energy consumption and external traffic usage (traffic departing local D2D network). Mobile devices can execute tasks locally, cooperate with other devices or offload tasks to the cloud in accordance with the decision of the SDCOM Controller. We can use fingerprints proposed in [13], [14] or the hash indexes derived from the input of tasks to describe the similarity of tasks. Mobile devices can share computation results of tasks with each other to eliminate redundancy in computation and external traffic usage.

We formulate such task scheduling as a 0-1 knapsack problem, and prove its NP-hardness. The scheduling aims to minimize the energy consumption of mobile devices under the external traffic constraint from the perspective of the entire network. To compute the optimal solution as a benchmark,

Manuscript received March 25, 2016; revised October 24, 2016; accepted December 20, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor P.-J. Wan. Date of publication February 2, 2017; date of current version June 14, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61422206, in part by the National High Technology Research and Development Program 863 Program of China under Grant 2015AA016101, and in part by the Tsinghua National Laboratory for Information Science and Technology and the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant City U 117913.

Y. Cui, J. Song, Q. Ren, and Y. Zhang are with Tsinghua University, Beijing 100084, China (e-mail: cuiyong@tsinghua.edu.cn; song-j12@mails.tsinghua.edu.cn; rqm15@mails.tsinghua.edu.cn; zhangyangjun13@mails.tsinghua.edu.cn).

K. Ren is with the Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY 14260 USA (e-mail: kuiren@buffalo.edu).

M. Li is with the Department of Computer Science, City University of Hong Kong, Hong Kong, China (e-mail: minming.li@cityu.edu.hk).

Z. Li is with the Department of Computer Science, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: zongpeng@ucalgary.ca).

Digital Object Identifier 10.1109/TNET.2017.2650964

we design an offline algorithm, the Conditioned Optimal Algorithm (COA), by approaching the optimization as a set covering problem. Our offline solution is optimal when computation dominates offloading in energy consumption. Due to its exponential time complexity, COA is practical only for a moderate number of task types. We further propose an efficient approximation algorithm, the Energy-saving Greedy Algorithm (EGA), which has a polynomial-time complexity and achieves a good performance to the optimal solution.

The offline algorithms require full knowledge of all tasks, and therefore are sensitive to the accurate prediction of future tasks (task type, time of occurrence, etc.). We further design a traffic queuing mechanism based on the external traffic usage to encourage cooperation between mobile devices and solve the task scheduling problem in the online setting. The traffic queuing mechanism employs a traffic queue to record the amount of offloading services that each device has already received. It can indicate the gap between received and offered offloading services. A device can reduce its queue backlog by providing services for others. If the traffic queue (*i.e.*, the queue backlog) is larger than a threshold, then the mobile device cannot offload more tasks to other devices.

Based on the proposed traffic queuing mechanism and Lyapunov optimization [15], we design an Online Task Scheduling Algorithm (OTS) to address the problem without predicting future tasks. OTS aims to minimize network-wide long-term average energy consumption while stabilizing the traffic queues of all devices in the LTE-Advanced network. It can not only minimize the energy consumption arbitrarily close to the optimal solution, but can also enable a flexible tradeoff between the energy consumption and external traffic usage by adjusting a tradeoff coefficient. Moreover, we extend OTS to design a Proportional Fair Online Task Scheduling Algorithm (PF-OTS) based on the concept of Relative Energy Consumption. It not only exhibits similar performance to OTS, but also ensures the fairness of energy consumption among mobile devices, such that the energy efficiency of one device will not hurt the energy performance of others. We conduct extensive trace-based simulations to show the effectiveness of SDCOM under various network conditions.

II. Related Work

The existing research efforts on offloading in wireless networks and mobile cloud computing can be divided into two categories: computation offloading and traffic offloading.

Computation Offloading: The concept of Cyber Foraging [16] was proposed in 2001 with the aim of augmenting the computing capability of mobile devices, offloading computation tasks of mobile devices to static idle computers for remote execution. In recent years, with the emergence of Mobile Cloud Computing (MCC), several similar solutions have been proposed. MAUI [3] and ThinkAir [17] provide method-level computation offloading, and do not require special support from the operating system. However, they still need the programmer to access the source code and partition the application manually. Clonecloud [4] and Cloudlets [18] use virtual machines to establish execution environment for mobile devices on the cloud or a nearby device that is rich in resource. SociableSense [19] shows how society related applications can benefit from cloud offloading. COMET [20] allows threads to be migrated to other machines without modification of the program code. These schemes focus on how to partition tasks statically or dynamically, realizing method-level or application-level migration, and when to offload applications from the single device perspective. Some other works [21]–[25] focus on evaluating the feasibility and cost of computation offloading in terms of bandwidth, energy consumption of CPU and network interfaces on the device.

Furthermore, in order to overcome the high access latencies of cloud services through cellular networks and to mitigate traffic pressure on cellular networks, mobile cloudlet [8] is proposed, following a peer-to-peer model for Mobile Cloud Computing. In mobile cloudlet, a group of nearby mobile devices connected wirelessly can execute tasks cooperatively. Mobile devices can be computing service providers as well as clients of the service. Li *et al.* investigate the impact of cloudlet size, cloudlet node's lifetime and reachable time on the feasibility and performance of mobile cloudlet [8]. They derive upper and lower bounds on computing capacity and computing speed for users to decide whether to offload tasks to remote clouds or local mobile cloudlets for better mobile application services.

However, the research on optimal policies for scheduling multiple tasks is limited. How to gather the distributed information of mobile devices and schedule tasks to appropriate devices or to the cloud are the main challenges in designing the mobile cloudlet model. Departing from existing solutions, our schemes not only focus on minimizing the energy consumption of mobile devices, but also aim to guarantee the energy fairness of mobile devices in the process of cooperation and reduce the traffic volume at access links.

Traffic Offloading: A number of proposals were put forward towards the challenges of explosive cellular network traffic. Lee *et al.* suggest that in certain metropolitan areas, some of the mobile traffic has already been offloaded to WiFi [6]. Ha *et al.* present the architecture and implementation of a time-dependent pricing system, which aims to manage the growing demand on cellular networks via dynamic pricing [26]. Wiffler [27] designs a model to predict WiFi connectivity, and uses these predictions to offload delay-tolerable data to WiFi. Zhuo *et al.* consider utilizing WiFi and delay tolerant networks for offloading traffic of cellular networks, and provide an incentive framework to motivate users to leverage their delay tolerance for cellular traffic offloading [28].

These previous studies focus on offloading mobile traffic to WiFi. With the advent of D2D communication, our sight turns to offloading mobile traffic through D2D since cellular networks have larger coverage and work at licensed band that can guarantee a high QoS service compared with other wireless networks. It is rather typical for a cellular user to stay within a single coverage zone throughout a communication session. Habak *et al.* consider using a collection of co-located devices to provide a dynamic mobile cloud [29]. Xie *et al.* take advantage of energy efficient cooperative communication to minimize the total power consumption of the system while

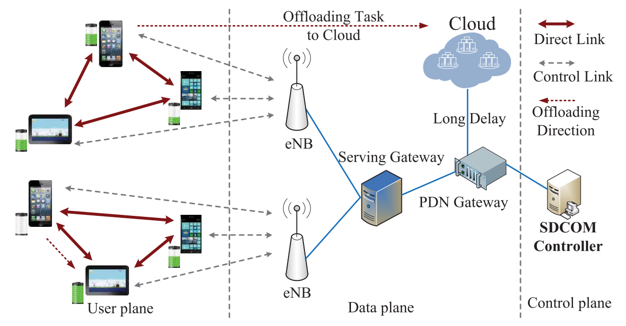


Fig. 1. An overview of SDCOM.

guaranteeing transmission reliability [30]. Chen *et al.* design a streaming system to enable nearby mobile users to share downloaded data [31]. It can reduce the usage of cellular links and avoid redundant transmissions of cellular networks. These findings indicate burgeoning attempt of D2D communication and D2D-based offloading.

However, the cooperative issues between multiple users need to be addressed. Extending these methods, we design SDCOM, which enables mobile devices to execute tasks cooperatively and share the results with each other. It can not only save the energy consumption of mobile devices by eliminating the computation redundancy, but can also reduce the external traffic usage of access links caused by accessing cloud services. To the best of our knowledge, this paper is the first to solve the task scheduling problem by leveraging the idea of SDN framework in the mobile cloudlet scenario.

III. Problem Formulation and Modeling

In this section, we overview the scenario of SDCOM, then define the scheduling problem and prove its NP-hardness.

A. Scenario Description

SDCOM resides in the LTE-Advanced network, as shown in Fig. 1. SDCOM Controller is arranged on the PDN Gateway, and provides data management services. A centralized control is realized by deploying small cell agents at each Serving Gateway and communicating with the Controller like traditional SDN. We utilize the DC-OC [32] (direct D2D communication with operator controlled link establishment) as the mode for device to device communication. The connection is established between these two devices with the help of the controller and data between devices is then transmitted directly. In SDCOM, the control plane is completely decoupled from the data plane, as though it were a centralized application, rather than a distributed system. The Controller knows the global network conditions and records profiles of every mobile device. Because of the limited storage capacity, the SDCOM Controller does not store the results of tasks, but stores the fingerprints [13] of tasks recently executed. The Controller has a Device Information Table and a Task Information Table.

The Device Information Table records profiles of mobile devices in the network, *e.g.*, the residual energy (remaining battery capacity), the CPU clock frequency, the interface bandwidth. The CPU frequency, as a constant profile, is reported once only upon UE registration in the network. The Task Information Table records the task information, *e.g.*, which devices have executed the task and cached the result, the fingerprint derived from the task. In a LTE network, a UE periodically sends the measurement report to the serving BS in a heartbeat style [33]. Task information and device information except the CPU frequency are periodically reported along with the measurement report. Then the BS integrates multiple users' heartbeats together and periodically transmits to the Controller in bursts. Besides, variable profiles are also updated when the UE makes a request. Based on the information in these tables, SDCOM Controller schedules tasks to the appropriate provider (another local device or the cloud) by solving an optimization problem in a discrete time manner.

SDCOM operation includes the following phases. A device who has a task to execute sends a request to the BS; then BS forwards the corresponding request to the SDCOM Controller. The Controller updates all its information tables, discovers whether there are available peers who have already executed the same task, and then executes a task scheduling algorithm. According to the solution, the Controller assigns the task to the appropriate provider. Since the size of the offloading request is very small, we can ignore the energy consumption of sending the request to SDCOM Controller. The transmission delay from the device to the Controller is a few milliseconds, which is shorter than the transmission delay on the Internet by one or two orders of magnitude.

Since tasks generated by different mobile users may be similar (for example, querying the same keyword through a search engine or requesting the same video), 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 network traffic. Intuitively, when the repetition rate of tasks is high, the proposed model can save more energy by reducing computation redundancy. When the repetition rate is low, SDCOM can also improve the network-wide energy efficiency by offloading tasks to the cloud.

B. Notation

Assume there are n devices in the LTE-Advanced network. They have m tasks to execute over a long time period, denoted by set $\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$. Every task is requested by one of the devices. We use C_h^i to represent task C_h that belongs to device i, where $h = 1, 2, \cdots, m$. Mobile devices can execute tasks locally, offload tasks to other devices for cooperative execution or to the cloud in accordance with the decision-making of the SDCOM Controller, as shown in Fig. 2. Energy consumption for executing tasks is defined as *energy consumption of computation*. Energy consumption of offloading tasks to other devices or to the cloud is defined as *energy consumption of offloading*.

We further denote the profile of a task who belongs to device i as $C_h^i(Type, D_h^{in}, D_h^{out})$. $Type$ is the type of task C_h^i, D_h^{in} is the data size of the input of the task and D_h^{out} is the data size of the output of the task. The symbols used in this paper is summarized in TABLE I.

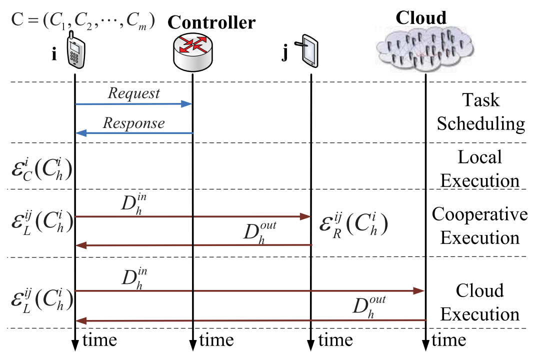


Fig. 2. The procedure of offloading.

C. Problem Formulation

1) Energy Consumption of Computation: During task execution, CPU dominates in energy consumption of a mobile device. CPU energy consumption depends on the CPU type, workload, and clock frequency, whose precise characterization is challenging. Miettinen et al. [34] depict CPU energy consumption from the perspective of computing efficiency: the amount of computation accomplished with a unit 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 size and the $Type$ of a task [21], [34]. We assume that device i needs N_h CPU cycles to execute computation task C_h^i. From reference [21] we can derive:

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

where the function $f_X(\cdot)$ is determined by the task $Type$.

For some popular applications, the CPU cycles needed by a computation task can be expressed as a linear function of the input data size as $N_h = X \cdot D_h^{in}$ [34], where the complexity coefficient X is the ratio of CPU cycles and the input data size, dependant on the task $Type$. A table about specific value of X is given in table II.

While modeling the energy consumption of CPU is an active area of research and beyond the scope of this paper, we simply define the computation energy consumption of device j for executing task C_h^i as,

$$\mathcal{E}_C^j(C_h^i) = F_j(N_h), \tag{2}$$

where $F_j(\cdot)$ is the function of power coefficient of j's CPU.

If device j has already executed the task and cached the result, the energy consumption of j for executing the task with the same $Type$ again is close to 0. For example, if a nearby device i requires location 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 that to i directly. Hence the energy consumption of computation that j completes task C_h^i is 0.

2) Energy Consumption of Offloading: The LTE-Advanced network 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 directed edges, respectively. 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 weights $(\rho_T^{ij}, \rho_R^{ij}, \phi_T^{ij})$, where ρ_T^{ij}, ρ_R^{ij} are transmitting (resp. receiving) power of i to (resp. 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 $\rho_T^{ij} \cdot (D_h^{in}/\phi_T^{ij})$ [35]. We set $\rho_T^{ii} = 0$ and $\phi_T^{ii} = \infty$. We assume that the SDCOM Controller can collect and store these information at real time. To characterize the energy consumption of each device in the task offloading process clearly, we define i as the customer and j as the provider when device i requests device j to execute a task.

TABLE I
INDEX OF SYMBOLS

C_h^i	A task belonging to device i
ρ_0^i, ρ_1^i	The static and dynamic power coefficients of i's CPU
f_i	The CPU clock frequency of device i
ρ_T^{ij}	The transmitting power of i to j
ρ_R^{ij}	The receiving power of i from j
ϕ_T^{ij}	The transmission rate on link (i, j) at the current time
$\mathcal{E}_C^j(C_h^i)$	The energy consumption of computation that device j used to execute task C_h^i
$\mathcal{E}_O^{ij}(C_h^i)$	The energy consumption of device i and j when task C_h^i is offloaded from device i to j
$\mathcal{E}_L^{ij}(C_h^i)$	The energy consumption of device i when task C_h^i is offloaded from device i to j
$\mathcal{E}_R^{ij}(C_h^i)$	The energy consumption of device j when task C_h^i is offloaded from device i to j
$\mathcal{D}(C_h^i)$	The external traffic for offloading task C_h^i
$\mathcal{E}$	The network-wide average energy consumption for executing the tasks set **C**
t_h	The time slot for executing task C_h^i
$Q_i(t_h)$	Device i's queue backlog at the beginning of time slot t_h
$b_i(t_h)$	The increase of device i's queue backlog for offloading task C_h^i to other devices/the cloud at time slot t_h
$d_i(t_h)$	The decrease of device i's queue backlog for executing task for other devices at time slot t_h
X	the ratio of CPU cycles and the input data size, dependant on the task Type
V	a non-negative tradeoff coefficient to adjust the performance tradeoff
$\mathcal{E}^*$	the minimum network-wide average energy consumption of the optimization problem in (15)

Definition 1: Customer Energy Consumption of Offloading *is the total energy consumption of client i when task C_h^i is offloaded from customer i to provider j, including transmission energy consumption of sending input data and receiving output data of the task, i.e.,*

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

TABLE II
COMPUTATION TO DATA RATIO FOR VARIOUS TYPE TASKS

Task Type	Cycles/byte
gzip ASCII compress	330
x264 CBR encode	1900
html2text en.wikipedia.org	5900
pdf2text E72 data sheet	8900

Note that $\mathcal{E}_L^{ii}(C_h^i) = 0$, *i.e.*, local execution incurs zero transmission energy.

Definition 2: Provider Energy Consumption of Offloading *is the sum of transmission and computation energy consumption of provider j when a task C_h^i is offloaded from customer i to provider j, and the transmission energy consumption includes transmission energy consumption of receiving input data and sending output data of the task, i.e.,*

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

Specifically, when $i = j$, Equation (4) is consistent with Equation (2), *i.e.*, $\mathcal{E}_R^{ii}(C_h^i) = \mathcal{E}_C^i(C_h^i)$. Moreover, if provider j has executed the same task C_h^i and cached the result, the computation energy consumption can be considered as 0.

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

Definition 3: Energy Consumption of Offloading *is the energy consumption of mobile devices when task C_h^i is executed remotely, which can be formulated as,*

$$\mathcal{E}_O^{ij}(C_h^i) = \begin{cases} \mathcal{E}_L^{ij}(C_h^i) + \mathcal{E}_R^{ij}(C_h^i), & j = 1, \ldots, n; \\ \mathcal{E}_L^{ij}(C_h^i), & j = n+1. \end{cases} \quad (5)$$

D. Traffic-Aware Energy Optimization Problem

In the LTE-Advanced network, the SDCOM Controller needs to find a proper task allocation scheme, *i.e.*, finding an optimal provider (another device, the cloud or the device itself) for every task C_h^i, where $C_h^i \in \mathbf{C}$. The task allocation can be formalized as an optimization problem. The network-wide energy consumption is the optimization objective, and the external traffic usage is used as the constraint. Because the latency of D2D communication is smaller than WAN latency by two orders of magnitude [18], we ignore the time constraints in this work.

SDCOM is operated in a discrete time manner. The length of a slot matches the timescale at which the offloading decision is made for one task. In each time slot t_h, SDCOM 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 $j = 1, 2, \cdots, n, n+1$. In particular, we have $\alpha_{n+1}(C_h^i) = 1$ when task C_h^i is offloaded to the cloud.

The energy consumption of executing task C_h^i is:

$$\mathcal{E}(C_h^i) = \sum_{j=1}^{n+1} \alpha_j(C_h^i) \cdot \mathcal{E}_O^{ij}(C_h^i). \quad (6)$$

If C_h^i is offloaded to another device or cloud, the produced traffic can be expressed as $D_T(C_h^i) = D_h^{in} + D_h^{out}$. Since local execution and cooperative execution between devices do not generate external traffic, the external traffic caused by the execution of task C_h^i can be computed 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 external traffic constraints (denoted by Ψ), i.e.,

$$\min \sum_{h=1}^{m} \mathcal{E}(C_h^i) \quad (7)$$

$$\text{subject to:} \sum_{h=1}^{m} \mathcal{D}(C_h^i) \le \Psi, \quad i = 1, 2, \cdots, n, \quad (8)$$

$$\sum_{j=1}^{n+1} \alpha_j(C_h^i) = 1, \quad h = 1, 2, \cdots, m, \quad (9)$$

$$\alpha_j(C_h^i) \in \{0, 1\}, \quad h = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n+1. \quad (10)$$

Constraints (8) model the external traffic constraint at the access link. We can control the external traffic usage proportion by adjusting the coefficient Ψ. Constraints (9) ensure every request from node i is satisfied by only one provider. Finally, Constraints (10) enforce the non-negativity and integrality of the decision variables. With a constraint on the external traffic, the energy minimization problem in (7) can be proved NP-hard by a reduction from the knapsack problem.

Theorem 1: Solving the optimization problem presented in (7) is NP-hard.

Proof: We transform 0-1 knapsack problem which is NP-hard [36] into our problem (7) as follows. Given a set of m items denoted by set $\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$ and a knapsack. Denote the profit and weight of one item as $\mathcal{U}(C_h^i)$ and $\mathcal{D}(C_h^i)$ respective. The capacity of the knapsack is Ψ. The problem is to select a set $\mathbf{C}$ to maximize $\sum_{h=1}^{m} \mathcal{U}(C_h^i)$, satisfying the capacity constraint $\sum_{h=1}^{m} \mathcal{D}(C_h^i) < \Psi$. We define the external traffic caused by the execution of task C_h^i as the weight, $\mathcal{D}(C_h^i)$, and the saved energy as profit, $\mathcal{U}(C_h^i) = -\mathcal{E}(C_h^i)$. The 0-1 knapsack problem is equivalent to the problem presented in (7). This completes the proof. ■

IV. OFFLINE TASK SCHEDULING

There are n devices in the LTE-Advanced network. They have m tasks to execute in a long time period. In SDCOM, devices can execute tasks by itself, fetch results from other devices who have executed the tasks, or offload to other devices or the cloud. If traversal is directly used to handle the task scheduling problem, each state has $n+1$ successors.

So the time complexity is $O(n^m)$. Since traversal has an exponential growth on time and memory with the increase of problem size, comprising between the efficiency and the performance of the offline algorithm design becomes a pragmatic necessity.

We formulate the task scheduling as a specific set covering problem, and propose the Conditioned Optimal Algorithm (COA), which can achieve the optimal solution when the energy consumption of computation is far more than the energy consumption of offloading. Due to its exponential time complexity, this method is suitable for a moderate number of task types. To solve large scale problems efficiently, we further design an Energy-saving Greedy Algorithm (EGA), which computes an approximate optimal solution.

A. Conditioned Optimal Algorithm

Note that the scheduling of one $Type$ of tasks does not affect another $Types$ when external traffic quota is sufficiently high, thus we can make scheduling decisions separately for each $Type$ of tasks. Since offline algorithms have access to complete information including task arrivals and transmission rates, we can divide the task set $\mathbf{C} = \{C_1, C_2, \cdots, C_m\}$ into M subsets based on the $Type$ of tasks. Every subset $Set(T)$ consists all the tasks of the same $Type\ T$, and customers of these tasks form the subset $\mathcal{V}_T \subseteq \mathcal{V}$.

The energy consumption of computation and transmission of the cloud is not included in the *Network-Wide Energy Consumption*, so offloading tasks to the cloud consumes less energy than offloading to other devices for execution. At the same time, obtaining results from other devices is more efficient than from the cloud. Therefore, if $\mathcal{V}_T$ can use the cloud to execute tasks in $Set(T)$, it must use the cloud only once for the first instance. Customers of subsequent tasks can obtain the result from customers who have cached the result. Deciding which tasks to offload to the cloud under the constraints of the external traffic is a 0-1 knapsack problem as shown in Theorem 1. When the number of types is not large, we can enumerate all possibilities about which tasks should be offloaded to the cloud for saving more energy according to the constraints of the external traffic.

If $\mathcal{V}_T$ does not use the cloud to execute tasks in $Set(T)$, it must find one (maybe itself) or more providers to execute tasks in $Set(T)$. The problem of finding providers for every subset $Set(T)$ can be formulated as a set covering problem (SCP), which is also NP-hard as shown in Theorem 2.

Theorem 2: Solving the optimization problem presented in (7) without constraints (8) is NP-hard.

Proof: The set cover problem (SCP) is a classical combinatorial optimization problem that is proven NP-hard in [37]. Given a set of elements $U = \{1, 2, \cdots, m\}$ (called the universe) and a set S of n sets whose union equals the universe, SCP is to identify the smallest subset of S whose union equals the universe.

We transform SCP into our problem presented in Theorem 2 as follows. Given a set of devices denoted by set $\mathcal{V}_T = \{1, 2, \cdots, m_T\}$ (called the universe). Every device i has a task C_T to be executed. Device i (as customer) can execute

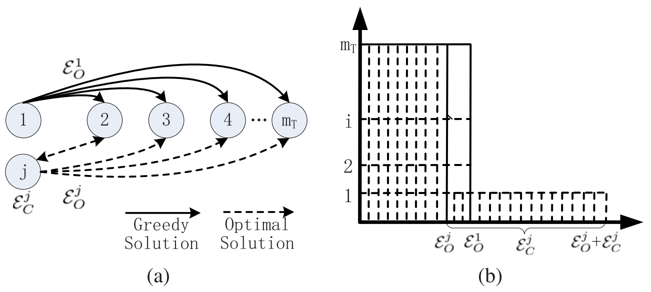


Fig. 3. Performance analysis of offline algorithm. (a) The scheduling decision. (b) The energy consumption gap.

C_T by itself with the Energy Consumption of Computation $\mathcal{E}_C^i$, or fetch the result of C_T from another device j (as provider) with the Energy Consumption of Offloading $\mathcal{E}_O^{ij}$. Customers which fetch the result from the same provider j constitute a set $v_j \in S$. Obviously, S's union equals the universe.

Solving the optimization problem presented in (7) without constraints (8), is to minimize the total energy consumption of every device set $\mathcal{V}_T$. We select devices with the less Energy Consumption of Computation $\mathcal{E}_C^j$ as providers, which execute task for themselves, and share result with other devices. For example, customers in a subset $v_j = \{1, 2, \cdots, k\}$ fetch the result of C_T from provider j, and the total energy consumption can be formulated as $E_j = \mathcal{E}_C^j + \sum_{i=1}^{k} \mathcal{E}_O^{ij}$.

Because the Energy Consumption of Offloading among some devices are larger, we cannot select only one provider to cover the universe. For example, if $\mathcal{E}_O^{hj} > \mathcal{E}_C^h$, selecting other provider with less energy consumption or executing by itself will be more energy efficient for device h. Due to the high cost, we need to select the smallest number of providers to decrease the Energy Consumption of Computation, i.e., cover all of the customers with the smallest number of subsets in S. The set cover problem is equivalent to the problem presented in Theorem 2. This completes the proof. ■

However, we can efficiently compute the optimal solution of the problem in some special cases. The worst case that $\mathcal{V}_T$ can select only one provider is shown in Fig. 3. The energy consumption of offloading task C_T is $\mathcal{E}_O^1$ between device 1 and device i, and $\mathcal{E}_O^j$ between device j and device i, where $i \in \mathcal{V}_T$. Provider j's energy consumption of executing task C_T is $\mathcal{E}_C^j$. We set $\mathcal{E}_O^j < \mathcal{E}_O^1$, but $\mathcal{E}_O^j + \mathcal{E}_C^j > \mathcal{E}_O^1$, Device 1 offloads C_T to the cloud and caches the result. Device j does not cache the result.

As shown in Fig. 3a, device i can fetch the result of C_T from device 1, and the total energy consumption is $(m_T - 1)\mathcal{E}_O^1$. However, if $(m_T - 1)\mathcal{E}_O^j + \mathcal{E}_C^j < (m_T - 1)\mathcal{E}_O^1$, $\mathcal{V}_T$ can also offload C_T to device j, *i.e.*, $\mathcal{V}_T$ can find more than one providers to execute C_T. We can show that $\mathcal{V}_T$ can find no more than one provider to execute C_T if the energy consumption satisfies the condition below.

$$\mathcal{E}_C^j > (m_T - 1)(\mathcal{E}_O^1 - \mathcal{E}_O^j), \tag{11}$$

which means that a subset can select only one provider to achieve the optimal solution, when the energy consumption of computation is far more than the energy consumption

Algorithm 1 Conditioned Optimal Algorithm

```
Input: $\{\rho_{i1}^T, \cdots, \rho_{in}^T\}$, $\{\rho_{i1}^R, \cdots, \rho_{in}^R\}$, $\{\phi_{i1}, \cdots, \phi_{in}\}$, $\{C_h^i(Type, D_h^{in}, D_h^{out})\}$
Output: $E$, $\{\vec{\alpha}\}$
1: for all $Type\ T$ do
2:   $E_{local}^T \leftarrow \infty$
3:   Sorting the tasks in $Set(T)$ by arriving order, and rename the task No. to be $\{1', 2', \ldots, |Set(T)|'\}$
4:   for $j \leftarrow 1$ to $n$ do
5:     $\mathcal{E}^T \leftarrow \mathcal{E}_O^{ij}(C_{1'}^i)$
6:     for $h = 2' \rightarrow |Set(T)|'$ do
7:       Selecting provider $k^*$ with $\min_{k\in(1,n)} \mathcal{E}_S^{ik}(C_h^i)$
8:       $\mathcal{E}^T \leftarrow \mathcal{E}^T + \mathcal{E}_S^{ik^*}(C_h^i)$
9:     end for
10:    if $E_{local}^T > \mathcal{E}^T$ then
11:      $E_{local}^T \leftarrow \mathcal{E}^T$
12:    end if
13:  end for
14:  $E_{cloud}^T \leftarrow \mathcal{E}_O^{i(n+1)}(C_1'^i)$
15:  for $h = 2' \rightarrow |Set(T)|'$ do
16:    Selecting provider $k^*$ with $\min_{k\in(1,n)} \mathcal{E}_S^{ik}(C_h^i)$
17:    $E_{cloud}^T \leftarrow E_{cloud}^T + \mathcal{E}_S^{ik^*}(C_h^i)$
18:  end for
19: end for
20: Enumerate all possible solutions $E_{local}^T, E_{cloud}^T$ under the constraints $D < \Psi$.
21: Set $\vec{\alpha}(Set(T))$ for all $Type\ T$
22: return $\{\vec{\alpha}\}$
```

of offloading. Based on the above analysis, we design a **C**onditioned **O**ptimal **A**lgorithm (COA), shown in Algorithm 1, which can achieve the optimal solution when the energy consumption satisfies (11).

In lines 2-12, if the first task of $Set(T)$ is not offloaded to the cloud, we enumerate providers to execute the first task, find the best provider in $\mathcal{V}_T$ for subsequent tasks in $Set(T)$ to fetch the result, and record the minimum energy consumption with E_{local}^T. The time complexity of this step is $O(|Set(T)| * n^2)$. In lines 13-17, if the first task of $Set(T)$ is offloaded to the cloud, we select the best provider in $\mathcal{V}_T$ for subsequent tasks in $Set(T)$. The time complexity of this step is $O(|Set(T)| * n)$. We enumerate all possible solutions E_{local}^T, E_{cloud}^T of all $Type$s to find the minimum sum of energy consumption under the constraints $D < \Psi$. The time complexity of this step is $O(2^M)$, where M is the number of $Type$s. COA can compute the optimal solution, that can be used as a benchmark to measure performance of other algorithms. However, due to its exponential time complexity, this method is suitable only for the case where the number of $Type$s is not very large.

B. Energy-Saving Greedy Algorithm

To improve efficiency, we further design an **E**nergy-saving **G**reedy **A**lgorithm (EGA) in Algorithm 2, working with linear complexity in the offline setting.

Algorithm 2 Energy-saving Greedy Algorithm

```
Input: $\{\rho_{i1}^T, \cdots, \rho_{in}^T\}$, $\{\rho_{i1}^R, \cdots, \rho_{in}^R\}$, $\{\phi_{i1}, \cdots, \phi_{in}\}$, $\{C_h^i(Type, D_h^{in}, D_h^{out})\}$
Output: $E$, $\{\vec{\alpha}\}$
1: Sorting the first tasks of every $Set(T)$ by non-decreasing Energy Saving of Unit Traffic $\frac{\mathcal{E}_C^i(C_1^i)-\mathcal{E}_O^{i(n+1)}(C_1^i)}{D_T(C_1^i)}$.
2: Sequentially selecting tasks from the head to offload to the cloud under the constraints $D < \Psi$
3: for all $Type\ T$ do
4:   Sorting the tasks in $Set(T)$ by arriving order, and rename the task No. to be $\{1', 2', \ldots, |Set(T)|'\}$
5:   if $C_1'^i$ is offloaded to the cloud then
6:     $\alpha_{n+1}(C_1'^i) \leftarrow 1$
7:     $E = E + \mathcal{E}_O^{i(n+1)}(C_1'^i)$
8:     for $h = 2' \rightarrow |Set(T)|'$ do
9:       Selecting provider $k^*$ with $\min_{k\in(1,n)} \mathcal{E}_O^{ik}(C_h^i)$
10:      $\alpha_{k^*}(C_h^i) \leftarrow 1$
11:      $E \leftarrow E + \mathcal{E}_O^{ik^*}(C_h^i)$
12:    end for
13:  else
14:    for $h = 1' \rightarrow |Set(T)|'$ do
15:      Selecting provider $k^*$ with $\min_{k\in(1,n)} \mathcal{E}_O^{ik}(C_h^i)$
16:      $\alpha_{k^*}(C_h^i) \leftarrow 1$
17:      $E \leftarrow E + \mathcal{E}_O^{ik^*}(C_h^i)$
18:    end for
19:  end if
20: end for
21: return $E$, $\{\vec{\alpha}\}$
```

In line 1, EGA constructs a subset by gathering tasks of the same $Type\ T$ together, and sorting tasks in every subset $Set(T)$ in chronological order within time complexity $O(|Set(T)| \log |Set(T)|)$. We allocate the first task of every subset $Set(T)$ to offload to the cloud according to the constraints of the external traffic in line 2. In lines 3-11, we select the best provider for subsequent tasks of subsets, whose first instance is offloaded to the cloud. Meanwhile, we record the scheduling scheme and the value of energy consumption. In lines 12-19, we select the best provider for tasks of other subsets. The time complexity of this algorithm is $O(m * n)$.

Compared with the COA, EGA differs in two aspects, which results in the performance differences. Consider each subset as a whole, while deciding the provider for the first tasks, EGA chooses the subsets offloading to the cloud by normalized energy consumption per unit traffic, which is equivalent to a greedy solution for 0-1 knapsack problem. The greedy solution avoids the $O(2^M)$ time complexity of the searching operations, but leads to some performance loss. Nonetheless, when we take the subset as a whole, EGA no longer traverses every possible n provider for the first task of each subset. Instead, EGA just chooses to offload to the cloud or not, decreases the time complexity to $O(|Set(T)| * n^2)$. The performance loss depends on the difference between the maximum energy and minimum energy consumption of local D2D offloading, which is $M * (E_{max} - E_{min})$. In a network with good signal, E_{max} approximately equals E_{min} and the loss can be ignored.

TABLE III
TRANSMISSION RATE AND POWER OF DEVICES

Uplink(Mbps)	P_T(W)	Downlink(Mbps)	P_R(W)
4	3.041	20	2.327
3.5	2.822	16	2.119
3	2.603	12	1.911
2	2.16	9	1.756

V. ONLINE TASK SCHEDULING

The offline algorithms need full knowledge of all tasks, which may not be available in practice. We further propose the **O**nline **T**ask **S**cheduling Algorithm (OTS), which removes the necessity of future information on task arrivals. Moreover, we extend OTS to design a Proportional Fair Online Task Scheduling Algorithm (PF-OTS). It can not only achieve performance similar to OTS, but also ensure the fairness of energy consumption of mobile devices.

A. Traffic Queuing

We propose a traffic queuing mechanism to encourage cooperation between mobile devices and control the usage of external traffic. Traditional charging policy is that all the customers pay the bills towards the ISP. Simple applying the policy, the providers earn nothing but the energy consumption. Moreover, our BitTorrent-like mechanism has been proved to possess a Nash equilibrium [38]. Thus the traffic queuing mechanism is necessary and effective for cooperative offloading. If a device's queue backlog is larger than a threshold, it is not allowed to offload more tasks to other devices. A device can reduce its queue backlog by executing tasks for others. Each time the D2D connection closes, BS would count the amount of offloading traffic to update the device's traffic queue. In order to offload tasks continually, queue backlogs of devices cannot increase unboundedly.

We treat the cloud as a special device. It can increase other devices' queue backlogs by providing offloading services, but cannot consume other devices' queue backlogs. The external traffic usage can be reduced if devices offload tasks to other devices instead of the cloud. To achieve network-wide (customer devices and provider devices) backlog balance, we design an asymmetric queuing model according to the contributions for reducing the external traffic usage.

Definition 4: Traffic Queuing Function. *The queue backlog of customer i will increase by $b_i(t_h) = f(D_T(C_h^i))$, when customer i has a task C_h^i to execute at time slot t_h; if C_h^i is executed by itself, the queue 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 queue backlog of provider j will decrease by $d_j(t_h) = f(D_T(C_h^i))$.*

The increased queue 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 queue backlog of device j decreases by $d_j(t_h)$ which is proportional to the external traffic saved for the network, and the queue backlog of i increases by $b_i(t_h)$ which is exactly related to the traffic if the task is offloaded to be executed.

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

The service queues 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' queues are paid partially, which are denoted as $\mathbf{d}(t_h) \triangleq (d_1(t_h), \cdots, d_n(t_h))$. So the queue backlog evolves according to the following dynamics

$$Q_i(t_{h+1}) = \max\left[Q_i(t_h) - d_i(t_h) + b_i(t_h), 0\right], \qquad (12)$$

with an initially empty queue ($Q_i(t_0) = 0$).

To bound the queue backlog of every device, we require all the queues to be stable in the time average sense:

$$\overline{\mathbf{Q}} = \limsup_{m\to\infty} \frac{1}{m} \sum_{h=1}^{m} \sum_{i=1}^{n} \mathbb{E}\{|Q_i(t_h)|\} < \infty. \qquad (13)$$

B. Online Task Scheduling Algorithm

The network-wide average energy consumption of executing the task set $\mathbf{C}$ in a long time period T is:

$$\mathcal{E} = \limsup_{m\to\infty} \frac{1}{m} \sum_{h=1}^{m} \mathbb{E}\{|\mathcal{E}(C_h^i)|\}. \qquad (14)$$

We obtain a new optimization problem:

$$\begin{aligned} &\min \mathcal{E} \\ &\text{subject to: } \overline{\mathbf{Q}} < \infty \\ &\qquad \alpha_j(C_h^i) \in \{0,1\}, \quad h = 1,2,\cdots,m \\ &\qquad j = 1,2,\cdots,n+1 \\ &\qquad \sum_{j=1}^{n+1} \alpha_j(C_h^i) = 1, \quad h = 1,2,\cdots,m \end{aligned} \qquad (15)$$

Let $\mathcal{E}^*$ be the target value of the optimization problem defined in (15). In each time slot t_h, SDCOM makes an online offloading decision, with the objective of minimizing the time average energy consumption under queue backlog constraints for all devices. Applying Lyapunov optimization, we define our *Lyapunov function* for each slot $L(\mathbf{Q}(t_h))$ as follow:

$$L(\mathbf{Q}(t_h)) \triangleq \frac{1}{2} \sum_{i=1}^{n} Q_i^2(t_h). \qquad (16)$$

$L(\mathbf{Q}(t_h))$ can be obtained by adding up squares of the queue backlog of all devices in the LTE-Advanced network, which is an effective measure used for vectors like $\mathbf{Q}(t_h)$. A large value of $L(\mathbf{Q}(t_h))$ implies that at least one queue backlog is large. To ensure that the queue backlog of each device is below the threshold, we need to keep the *Lyapunov function* small. We next introduce the *Lyapunov drift* $\triangle(\mathbf{Q}(t_h))$:

$$\triangle(\mathbf{Q}(t_h)) \triangleq \mathbb{E}\{L(\mathbf{Q}(t_{h+1})) - L(\mathbf{Q}(t_h)) \mid \mathbf{Q}(t_h)\}, \qquad (17)$$

which represents the expected variation of the *Lyapunov function* during one time slot. Following the Lyapunov optimization approach [15], we then add the expected energy consumption over one time slot to both sides of (17), leading to the following *drift-plus-penalty* term.

Definition 5: The Drift-Plus-Penalty *is computed as:*

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

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 queue backlog. The key derivation step is to obtain an upper bound on the *drift-plus-penalty*. The following lemma establishes such an upper bound.

Lemma 1: Given any possible queue backlogs $\boldsymbol{Q}(t_h)$, arrival rates $b_i(t_h)$ and service rates $d_i(t_h)$ at each queue, under any possible decision $\vec{\alpha}(t_h)$, we have:

$$\begin{aligned} DPP(t_h) \le B + V \cdot \mathbb{E}\{\mathcal{E}(C_h^i) \mid \boldsymbol{Q}(t_h)\} \\ + \sum_{i=1}^{n} \mathbb{E}\{Q_i(t_h)(b_i(t_h) - d_i(t_h)) \mid \boldsymbol{Q}(t_h)\}. \end{aligned} \quad (18)$$

Proof: Based on equation (12), we can prove an upper-bound of the *Lyapuonv drift*,

$$Q_i^2(t_{h+1}) \le [Q_i(t_h) - d_i(t_h) + b_i(t_h)]^2.$$

Moving all the squares of the queue to the left side, summing all the i together and dividing by 2, taking conditional expectations at both sides, we can obtain the bound on Lyapunov drift as follows,

$$\begin{aligned} \triangle(\mathbf{Q}(t_h)) \le \mathbb{E}[B(t_h)|\mathbf{Q}(t_h)] \\ + \sum_{i=1}^{n} Q_i(t_h)\mathbb{E}[(b_i(t_h) - d_i(t_h))|\mathbf{Q}(t_h)]. \end{aligned} \quad (19)$$

where

$$B(t_h) = \frac{1}{2}\sum_{i=1}^{n}[b_i(t_h)^2 + d_i(t_h)^2 - 2b_i(t_h)d_i(t_h)].$$

Because the arrival and service pattern of tasks can be controlled and adjusted, there must exist some real number $\epsilon > 0$ such that the expectation of the difference between arrivals and services of each queue is smaller than $-\epsilon$,

$$\mathbb{E}[(b_i(t_h) - d_i(t_h))|\mathbf{Q}(t_h)] \le -\epsilon, \quad (20)$$

and we can find a proper finite constant $B > 0$ such that the following inequality holds for all the different t_h and all possible queue vectors $\mathbf{Q}(t_h)$:

$$B \ge \mathbb{E}[B(t_h)|\mathbf{Q}(t_h)].$$

Adding $V \cdot \mathbb{E}\{\mathcal{E}(t_h) \mid \mathbf{Q}(t_h)\}$ on both sides of inequality (19), we can obtain an upper bound as required in (18). ■

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 (18) 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 (18) by minimizing these terms:

$$DPP_{bound} = V \cdot \mathcal{E}(C_h^i) + \sum_{i=1}^{n} Q_i(t_h)(b_i(t_h) - d_i(t_h)). \quad (21)$$

Algorithm 3 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
    ${\alpha_1(C_h^i), \cdots, \alpha_{n+1}(C_h^i)}$, ${Q_1(t_{h+1}), \cdots, Q_n(t_{h+1})}$
```

Based on the above lemma, we design the Online Task Scheduling Algorithm to allocate tasks. When device i requests the result of a task, the algorithm is triggered. The queue backlog of i is updated in line 1. Lines 3-6 compute DPP of every device j according to (21). 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. Since we have computed the DPP_{bound} and the offloading decision vector before, lines 13-15 update the queue backlogs of corresponding devices. The algorithm does not change the queue backlogs when a task is offloaded to the cloud.

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 LTE-Advanced network.

C. Performance Analysis

Substituting constants ϵ, $\mathcal{E}^*$ into inequality (18), and calculating the time average queue size for every slot $t_h > 0$, we can obtain the performance bounds of OTS:

Theorem 3: Assume the difference between arrivals and services at each queue satisfies property (20), the time average energy consumption and queue backlog can be bounded by:

$$\mathcal{E} \le \mathcal{E}^* + \frac{B}{V}, \quad (22)$$

$$\overline{\boldsymbol{Q}} \le \frac{B + V\mathcal{E}^*}{\epsilon}. \quad (23)$$

Proof: Since the network can be controlled, there exists at least one stationary and randomized allocation policy that can stabilize the queue backlog as follows,

$$\mathbb{E}\{\mathcal{E}(C_h^i)\} = \mathcal{E}^*. \tag{24}$$

By applying (24) and (20) to (18), we obtain:

$$\begin{aligned}&\triangle\left(\mathbf{Q}(t_h)\right) + V \cdot \mathbb{E}\{\mathcal{E}(C_h^i)\}\\&\quad \le B + V \cdot \mathcal{E}^* - \epsilon \cdot \sum_{i=1}^{n} \mathbb{E}\{Q_i(t_h)\}.\end{aligned} \tag{25}$$

Summing the above inequality over all time slots t_h, where $h = 0, 1, 2, \ldots, m-1$, we get:

$$\begin{aligned}&\sum_{h=0}^{m-1} \triangle\left(\mathbf{Q}(t_h)\right) + V \cdot \sum_{h=0}^{m-1} \mathbb{E}\{\mathcal{E}(C_h^i)\}\\&\quad \le m \cdot (B + V \cdot \mathcal{E}^*) - \epsilon \cdot \sum_{h=0}^{m-1}\sum_{i=1}^{n} \mathbb{E}\{Q_i(t_h)\}.\end{aligned} \tag{26}$$

Simplifying (26) with (17), we get:

$$\begin{aligned}&\frac{\mathbb{E}\{L(\mathbf{Q}(t_m)) - L(\mathbf{Q}(t_0))\}}{m \cdot V} + \frac{1}{m}\sum_{h=0}^{m-1} \mathbb{E}\{\mathcal{E}(C_h^i)\}\\&\quad \le \frac{B}{V} + \mathcal{E}^* - \frac{\epsilon}{m \cdot V}\sum_{h=0}^{m-1}\sum_{i=1}^{n} \mathbb{E}\{Q_i(t_h)\}.\end{aligned} \tag{27}$$

Since *Lyapunov function* is non-negative, so is $\mathbb{E}\{Q_i(t_h)\}$. Furthermore, as we have defined $L(\mathbf{Q}(t_0))$ to be 0, we obtain:

$$\frac{1}{m}\sum_{h=0}^{m-1} \mathbb{E}\{\mathcal{E}(C_h^i)\} \le \frac{B}{V} + \mathcal{E}^*. \tag{28}$$

Let $m \to \infty$ in (28):

$$\limsup_{m\to\infty} \frac{1}{m}\sum_{h=0}^{m-1} \mathbb{E}\{\mathcal{E}(C_h^i)\} \le \frac{B}{V} + \mathcal{E}^*. \tag{29}$$

Since the left side of (29) is the definition of $\mathcal{E}$, we prove (22) as required. By a similar method, we can prove (23). ■

Theorem 3 shows that, given a tradeoff coefficient V, the time average energy deviates by $O(1/V)$ from optimality at most, while the queue backlog is bounded by $O(V)$. A large V can help the time average energy cost $\mathcal{E}$ approach $\mathcal{E}^*$.

D. Extension of Online Task Scheduling

In order to guarantee the fairness of energy consumption of network-wide mobile devices, we define Relative Energy Consumption as our optimization objective. According to the definition of Relative Energy Consumption, if device i is connected to the external power, the $R_i(C_h^i)$ can tend to infinity. If terminal i has already executed task C_h^i and stored the computation result, the $\mathcal{E}_C^i(C_h^i)$ which it consumes to execute task C_h^i again is close to 0. In both cases, the Relative Energy Consumption is close to 0.

Definition 6: Relative Energy Consumption *is equal to the ratio of energy consumption and residual energy of mobile devices to execute the task C_h^i, which can be formulated as,*

$$\widetilde{\mathcal{E}_O^{ij}}(C_h^i) = \begin{cases} \dfrac{\mathcal{E}_L^{ij}(C_h^i)}{R_i(C_h^i)} + \dfrac{\mathcal{E}_R^{ij}(C_h^i)}{R_j(C_h^i)}, & j = 1, \ldots, n;\\ \dfrac{\mathcal{E}_L^{ij}(C_h^i)}{R_i(C_h^i)}, & j = n+1,\end{cases}$$

where $R_i(C_h^i)$ is the residual energy of i at the beginning of time slot for executing C_h^i.

We substitute Relative Energy Consumption $\widetilde{\mathcal{E}}(C_h^i)$ for Energy Consumption $\mathcal{E}(C_h^i)$ in OTS, and propose the **P**roportional **F**air **O**nline **T**ask **S**cheduling Algorithm (PF-OTS), which can ensure the fairness of energy consumption of mobile devices, *i.e.*, the energy efficiency of one device will not hurt the energy performance of others. Moreover, we use Relative Energy Consumption to make scheduling decisions, which can balance the energy consumption of devices. For example, if several devices have the result of a task that device j requires, device j prefers to request the result from the device with most residual energy available.

VI. Performance Evaluation

We evaluate the performance of SDCOM by simulating the energy consumption and the external traffic usage. No Offload (all tasks are executed on the mobile devices locally) and Cloud Offload (all tasks are executed on the cloud remotely) are taken as the performance reference for comparison on the same topology with the same parameter setting.

A. Evaluation Setup and Methodologies

We develop a protocol independent simulator, which can support self-defined protocols and simulate the decision-making of SDCOM Controller and interactions between mobile devices. Our evaluation leverages real traces collected at border routers of a campus network. We filter the task requests from the collected hourly traces and calculate the distribution as the request distribution for different type tasks. We consider a fully connected LTE-Advanced network with 20 mobile devices, with randomly generated distances among devices. These devices can communicate with each other and access the cloud through SDCOM Controller. There are 1000 tasks that belong to these devices to be executed in a long time period T. These tasks fall into 20 $Type$s, for example, the location query and the sentence translation. We refer to the throughput-power model for data transfer over LTE-Advanced network measured in a recent measurement study [39], as the standard of the transmission rate on link (i, j) and the transmission power, as shown in TABLE III. The uplink and downlink throughput towards the cloud is set to be 2 Mbps and 9 Mbps respectively.

In practice, a wireless network witnesses not only a normal operation but also abnormalities such as link failure during data transfer. When the task results are removed from the provider's cache, offloading tasks will fail. These abnormal situations lead to task reassignments and increase the overhead. To verify that our algorithms can deal with all the

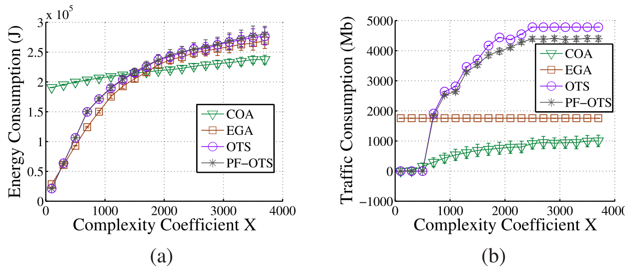


Fig. 4. The energy consumption and external traffic with different complexity coefficient X. (a) The energy consumption. (b) The external traffic.

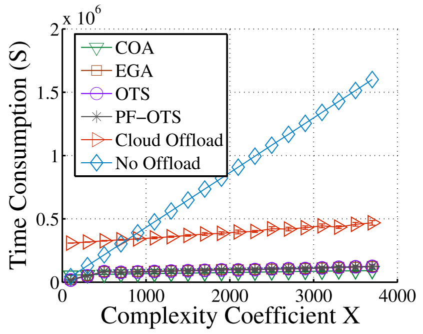


Fig. 5. The time consumption with different X.

cases effectively, we set the possibility of offloading failure to 10%. To obtain a reasonable result on average, we run 10 or 20 groups of test in the following evaluation. We use the mean and the standard deviation to plot an error-bar figure.

In SDCOM, 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). If the remote execution is not successful, e.g., in a case of a link failure, the device would choose to execute the task itself to ensure that the task is dealt with as soon as possible. Note that the energy consumption of computation is related to parameter X described in (1) which is decided by the $Type$ of tasks. Beyond that, the tradeoff coefficient V and the number of $Type$s will also affect the decision-making.

B. Simulation Results

1) Impact of Complexity Coefficient X: To validate SDCOM for different types of tasks, we present the energy consumption, external traffic usage of our algorithms for different values of X in Fig. 4 and time consumption in Fig. 5. We vary X from 100 to 3700, which includes most of the typical cases of mobile applications [34]. Tradeoff coefficients of OTS and PF-OTS are set to 120 and 10^7 respectively. Tests are executed ten times and results are averaged.

The energy consumption of No Offload and Cloud Offload increases linearly with the increase of X, as shown in TABLE IV. Moreover, Cloud Offload always consumes $91084Mb$ network traffic. Apparently in the case of local execution (No Offload), energy consumption doubles as X doubles. When $X < 1200$, No Offload consumes less energy than Cloud Offload for the same tasks. This indicates that local execution is more efficient than offloading for data intensive

TABLE IV
ENERGY CONSUMPTION OF NO OFFLOAD AND CLOUD OFFLOAD

Complexity X	500	1000	1500	2500
No Offload(J)	294918	589836	884754	1474590
Cloud Offload(J)	685493	717094	742188	809460

tasks (*e.g.*, gzip compression with $X = 330$), consistent with the conclusion drawn in [34]. The energy consumption of Cloud Offload is slightly more than 3.2 times of our algorithms and is not affected by X.

On the other hand, since our algorithms can select an appropriate provider for every task according to the application type and transmission rate, they can always achieve better performance than No Offload and Cloud Offload. Note that COA consumes less energy than EGA for the same tasks when $X > 1700$, which means that tasks with $X > 1700$ satisfy (11), *i.e.*, COA can achieve the optimal solution for computation intensive tasks. Another notable phenomenon is that the energy consumption of our algorithms rises quickly when $X < 1300$ and then increases slowly. Because both local executing and cooperative executing are more expensive than caching and sharing the results or cloud offload when $X > 1300$, more tasks are executed by the cloud or by sharing the results and more external traffic are consumed, the energy consumption of OTS tends to stabilize. Moreover, sharing the task results is more efficient than cloud execution, thus the energy consumption of OTS is lower than Cloud Offload.

Furthermore, regardless of the value of X, EGA always consumes less energy than the two online algorithms OTS and PF-OTS. For the first task of each type, EGA selects either local or cloud execution depending on which method has the lower energy consumption. While for the following tasks, the optimal provider that has already executed the task is selected.

The time consumption presents a similar trend to the energy. Time consumption increases linearly with X for Cloud Offload and No Offload, which always consume much more time than our strategies. Because our algorithms always choose an energy efficient offloading method, which take the time consumption of devices into consideration. For these tasks which have already been calculated by other devices (available offloading peers), our algorithms prefer to choose a nearby peer, because it gets rid of the computation delay. Furthermore, considering the case where there are no available offloading peers, our algorithms prefer the cloud offloading more with X increasing, which is much energy-efficient. Besides, our strategies EGA, OTS and PF-OTS all overlap, and COA consumes 20000 less seconds than the three others.

2) Performance of Online Algorithms with Different Tradeoff Coefficient V: We next study how tradeoff coefficient controls the energy consumption and external traffic usage of OTS and PF-OTS. We let V increase from 0 to 150 with the step size 10 for OTS and V increase from 0 to 10^7 for PF-OTS.

We present the energy consumption, the external traffic usage and the sum of queue backlog with a specified value of V for OTS in Fig. 6 and Fig. 7a. A notable observation is that the energy consumption falls quickly as V increases when V is small and then decreases slowly. At the same time, the

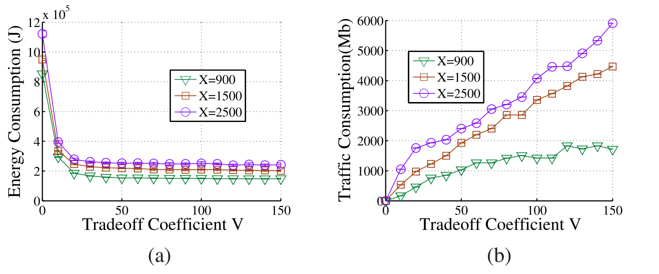


Fig. 6. The energy consumption and external traffic with different tradeoff coefficient V. (a) The energy consumption of OTS. (b) The external traffic of OTS.

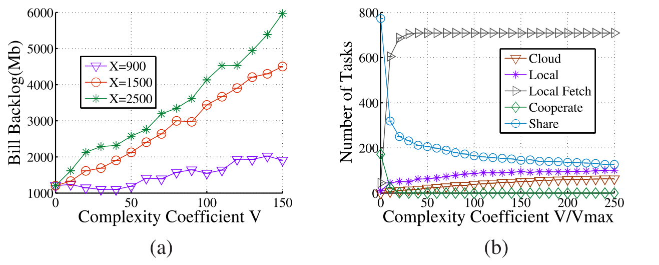


Fig. 7. The task statistics of OTS with different V. (a) The average queue backlog. (b) The number of tasks executed locally or remotely.

average queue backlog grows linearly with V increasing. This result confirms the $[O(1/V), O(V)]$ tradeoff between energy consumption and queue backlog as captured in (22) and (23).

Energy consumption of online algorithms is similar for different values of X, but the external traffic usage varies widely because online algorithms can select appropriate providers for tasks for different X. When the value of X is small, the energy cost of the local execution is low, therefore tasks are mostly executed locally, which in turn leads to low traffic consumption, and vice versa. Users can select the appropriate V to achieve the best energy-efficiency according to their budget for external traffic.

The energy consumption changes with V because the number of tasks executed by the device itself, the cloud, or other devices are changing with different V. Fig. 7b shows the number of tasks executed by the device itself, the cloud, or other devices with different V in our online algorithms. Almost all the tasks are executed by other devices when $V = 0$. With the queue backlog being the only optimization objective, OTS will select the provider with the longest queue backlog. As V increases, the number of cooperating tasks decreases to 0. The cooperation method is most energy-consuming, and it only happens at special cases where the cloud execution consumes high energy and the local device has a low energy consumption. Furthermore, since energy optimization takes a more important role in the optimization as V increases, the number of Cloud Execution and Local Fetch tasks increases. The growth of the Cloud Execution leads to the increase of the traffic, which is consistent with the result in Fig. 6b, and the number of Local Fetch reaches the maximum because only these tasks are repetitive at the same device.

Fig. 8 uses PF-OTS as an example to present the impact of V and X on energy consumption. As the figure shows, the energy consumption does vary for different V and X. But we

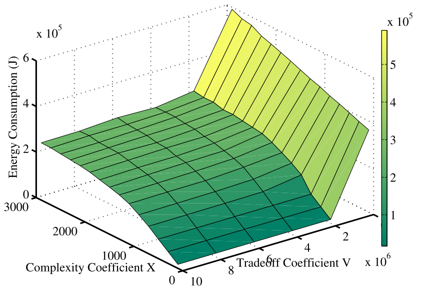


Fig. 8. The energy consumption with different V and X.

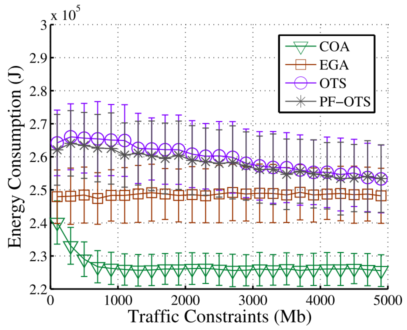


Fig. 9. The energy consumption with different external traffic constraints.

also can see that, for different values of X, when V increases, the energy consumption exhibits a similar decreasing tendency. When the tradeoff coefficient V increases from 10^6 to 10^7, take $X = 1100$ as the example. At first the energy consumption rapidly falls to 20346 from 44368, then it stabilizes at around 17600. Different X does affect the choice of V, but to a less radical extent. In this paper, to achieve a good and stable performance, we choose a relatively large value for V, 10^7.

3) Energy Consumption with Different External Traffic Usage: We next study the tradeoff between energy consumption and external traffic usage of our algorithms. We set $X = 2500$ for these four algorithms. Based on traffic requirements of algorithms shown in Fig. 4b, we plot Fig. 9 by adjusting the external traffic usage constraints of the four algorithms from $0Mb$ to $5000Mb$. The figure depicts the energy consumption of our algorithms, and we can see COA outperforms the other three because COA searches the optimal provider for each task type. With the same amount of traffic, EGA consumes less energy than the two online algorithms, as can be seen in Fig. 4a. When traffic is above $2000Mb$, the energy consumption of these two offline algorithms is no longer decreasing because the maximum traffic requirement of offline algorithms is around $2000Mb$. Note that the performance of PF-OTS is similar to that of OTS, and PF-OTS is somehow superior to OTS. As the traffic volume increases, the energy consumption of online algorithms exhibits a downward trend all the way. We can decrease the energy consumption of online algorithms to approach the offline by loosening traffic constraints.

4) Performance with Different Number of Task Types: We next set the number of task types to 40, 80, 120 and 160 in order to study the performance of our online algorithms under different task patterns. Generated patterns depend on

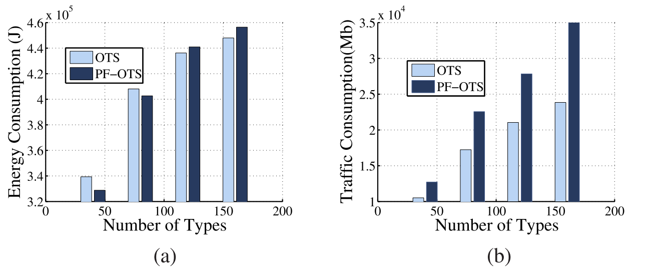


Fig. 10. The energy consumption and external traffic with different number of task types. (a) The energy consumption. (b) The external traffic.

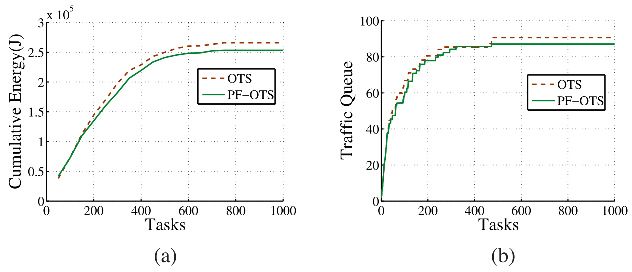


Fig. 11. The energy consumption and queue backlog over time with constant parameters. (a) The cumulative energy. (b) The queue backlog.

the number of types and are distributed according to Zipf distribution, as observed in real access traces [40]. The $k-th$ most popular content has $p_j(f_k) = \frac{k^{-\alpha}}{\sum_{i=1}^{m} i^{-\alpha}}$, where α is the Zipf Distribution Coefficient. A larger Zipf Distribution Coefficient means more requests of the most popular content, *i.e.*, the requests of the most popular content occupy a larger proportion of all requests. We set $X = 2500$, and V to 120 for OTS and 10^7 for PF-OTS to run this simulation.

We present the energy consumption of our online algorithms in Fig. 10a and the external traffic is shown in Fig. 10b. Increasing the number of task types increases the energy consumption and traffic. As generated task patterns become scattered, new types of tasks decrease the probability of the Collaborative Execution and Local Fetch. Therefore more tasks are executed either locally or on the cloud.

5) Performance with Constant Parameters: To explore the effectiveness of our online algorithms further, we examine the change of energy consumption and the external traffic usage during the running process with constant parameters. We set the tradeoff coefficient V to 120 for OTS and 10^7 for PF-OTS, X to 1500, the number of types to 20. Fig. 11a depicts the cumulative energy consumption of the execution of 1000 tasks. The average queue backlog of mobile devices is presented in Fig. 11b. At the beginning, there are not many results of tasks stored by devices. Thus, most tasks are offloaded to the cloud or calculated locally. The energy consumption of our online algorithms is increasing almost linearly with the number of tasks. Meanwhile, the queue backlog increases quickly due to usage of the Cloud Execution. Later the queue backlog tends to stabilize and the energy consumption of each task becomes smaller, since users can share more and more results of tasks with each other. This reflects the effectiveness of our online algorithms to the energy conservation and queue backlog control.

To show the effectiveness of PF-OTS for achieving the fairness of energy consumption in mobile devices, we sort 20 mobile devices in descending order by initial residual energy, as shown in Fig. 12, which omits the initial energy curve for clear display. The initial energy of the 20 mobile devices follows a uniform distribution over $[11000, 30000]J$. Note that in OTS, nearly half of the devices consumed more energy than PF-OTS. Obviously, PF-OTS can balance the energy consumption of mobile devices — it consumes more energy in devices with a higher battery capacity and less energy in devices with a lower battery capacity.

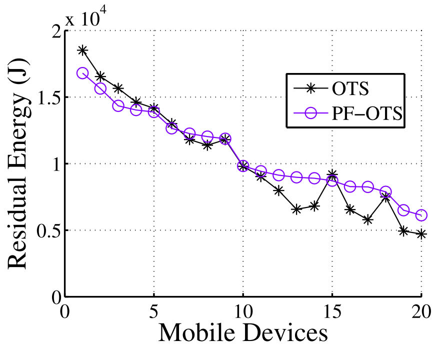


Fig. 12. The residual energy and consumed energy.

VII. Conclusions

Towards efficient offloading cooperation among mobile users, we leverage SDN to design our framework SDCOM, which aims to save energy of mobile devices and reduce traffic on LTE-Advanced access links. Based on distributed device information, a Controller makes task scheduling decisions periodically. We formulate the minimum-energy task scheduling problem as a 0-1 knapsack problem, and design offline algorithms to compute the optimal solution as a performance benchmark. We further propose an Online Task Scheduling Algorithm for realtime decision making, and extend it into a Proportional Fair Online Task Scheduling Algorithm. A thorough evaluation of SDCOM demonstrates that our cooperative offloading models can reduce the energy and traffic dramatically and guarantee the fairness among mobile devices.

References

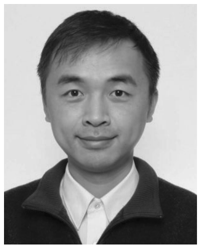
[1] M. Sharifi, S. Kafaie, and O. Kashefi, “A survey and taxonomy of cyber foraging of mobile devices,” *IEEE Commun. Surveys Tuts.*, vol. 14, no. 4, pp. 1232–1243, 4th Quart., 2012.
[2] H. Flores *et al.*, “Mobile code offloading: From concept to practice and beyond,” *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.
[3] E. Cuervo *et al.*, “Maui: Making smartphones last longer with code offload,” in *Proc. ACM MobiSys*, 2010, pp. 49–62.
[4] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “CloneCloud: Elastic execution between mobile device and cloud,” in *Proc. EuroSys*, 2011, pp. 301–314.
[5] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, “Energy consumption in mobile phones: A measurement study and implications for network applications,” in *Proc. ACM SIGCOMM IMC*, 2009, pp. 280–293.
[6] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, “Mobile data offloading: How much can wifi deliver?” in *Proc. ACM Co-NEXT*, 2010, Art. no. 23.
[7] C. Jiang, Y. Shi, Y. T. Hou, and W. Lou, “Cherish every joule: Maximizing throughput with an eye on network-wide energy consumption,” in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1934–1941.

[8] Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 1060–1068.
[9] M. Casado *et al.*, "Ethane: Taking control of the enterprise," in *Proc. ACM SIGCOMM*, 2007, pp. 1–12.
[10] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," in *Proc. ACM SIGCOMM*, 2008, pp. 69–74.
[11] R. Buyya, R. N. Calheiros, J. Son, A. V. Dastjerdi, and Y. Yoon, "Software-defined cloud computing: Architectural elements and open challenges," in *Proc. 3rd Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Sep. 2014, pp. 1–12.
[12] D. Levin, A. Wundsam, B. Heller, N. Handigol, and A. Feldmann, "Logically centralized?: State distribution trade-offs in software defined networks," in *Proc. ACM SIGCOMM HOT SDN*, 2012, pp. 1–6.
[13] N. T. Spring and D. Wetherall, "A protocol-independent technique for eliminating redundant network traffic," in *Proc. ACM SIGCOMM*, 2000, pp. 87–95.
[14] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker, "Packet caches on routers: The implications of universal redundant traffic elimination," in *Proc. ACM SIGCOMM*, 2008, pp. 219–230.
[15] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.
[16] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Pers. Commun.*, vol. 8, no. 4, pp. 10–17, Aug. 2001.
[17] 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 *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.
[18] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.
[19] K. K. Rachuri, C. Mascolo, M. Musolesi, and P. J. Rentfrow, "SociableSense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing," in *Proc. ACM Mobicom*, 2011, pp. 73–84.
[20] M. S. Gordon, D. A. Jamshidi, S. Mahlke, Z. M. Mao, and X. Chen, "COMET: Code offload by migrating execution transparently," in *Proc. OSDI*, 2012, pp. 93–106.
[21] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 2716–2720.
[22] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1285–1293.
[23] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 2685–2700, Dec. 2013.
[24] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 190–194.
[25] L. Xiang, S. Ye, Y. Feng, B. Li, and B. Li, "Ready, set, go: Coalesced offloading from mobile devices to the cloud," in *Proc. IEEE INFOCOM*, Apr. 2014, pp. 2373–2381.
[26] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "TUBE: Time-dependent pricing for mobile data," in *Proc. ACM SIGCOMM*, 2012, pp. 247–258.
[27] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *Proc. ACM MobiSys*, 2010, pp. 209–222.
[28] X. Zhuo, W. Gao, G. Cao, and S. Hua, "An incentive framework for cellular traffic offloading," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 541–555, Mar. 2014.
[29] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *Proc. 8th IEEE Int. Conf. Cloud Comput. (CLOUD)*, Jun. 2015, pp. 9–16.
[30] K. Xie, J. Cao, X. Wang, and J. Wen, "Optimal resource allocation for reliable and energy efficient cooperative communications," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 4994–5007, Oct. 2013.
[31] M.-H. Chen *et al.*, "Towards energy-efficient streaming system for mobile hotspots," in *Proc. ACM SIGCOMM*, 2011, pp. 450–451.
[32] M. N. Tehrani, M. Uysal, and H. Yanikomeroglu, "Device-to-device communication in 5G cellular networks: Challenges, solutions, and future directions," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 86–92, May 2014.
[33] A. Ulvan, R. Bestak, and M. Ulvan, "The study of handover procedure in lte-based femtocell network," in *Proc. 3rd Joint IFIP Wireless Mobile Netw. Conf. (WMNC)*, Oct. 2010, pp. 1–6.
[34] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, p. 4.
[35] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[36] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *Eur. J. Oper. Res.*, vol. 155, no. 1, pp. 1–21, 2004.
[37] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms*. Berlin, Germany: Springer, 2000.
[38] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like peer-to-peer networks," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 367–378, 2004.
[39] J. Huang *et al.*, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 225–238.
[40] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zsipf-like distributions: Evidence and implications," in *Proc. IEEE INFOCOM*, vol. 1. Mar. 1999, pp. 126–134.

Yong Cui received the B.E. degree and the Ph.D. degree in computer science and engineering from Tsinghua University, China, in 1999 and 2004, respectively. He is currently a Full Professor with Tsinghua University and the Co-Chair of IETF IPv6 Transition WG Softwire. He has published over 100 papers in refereed journals and conferences. He was a recipient of the National Award for Technological Invention in 2013 and the Influential Invention Award of China Information Industry in 2012 and 2004. He serves on the Editorial Board of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and the IEEE TRANSACTIONS ON CLOUD COMPUTING. His major research interests include mobile wireless Internet and computer network architecture.

Jian Song received the B.E. and M.E. degrees from the Department of Computer Science and Technology, Information Engineering University, Zhengzhou, China, in 2003 and 2006, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include the areas of wireless networking and mobile cloud computing

Kui Ren (M'07–SM'11–F'16) received the Ph.D. degree from Worcester Polytechnic Institute. He is currently an Associate Professor of computer science and engineering and the Director of the UbiSeC Laboratory with the State University of New York at Buffalo. His current research interests include cloud and outsourcing security, wireless and wearable system security, and human-centered computing. He received the U.S. National Science Foundation CAREER Award in 2011 and the Sigma XI/IIT Research Excellence Award in 2012. He has received several Best Paper Awards, including the IEEE ICNP 2011. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, *Wireless Communications*, the IEEE INTERNET OF THINGS JOURNAL, the IEEE TRANSACTIONS ON SMART GRID, *Pervasive and Mobile Computing*, and *The Computer Journal*. He is a member of the ACM



Minming Li received the B.E. and Ph.D. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2002 and 2006, respectively. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong. His research interests include wireless ad hoc networks, algorithm design and analysis, and combinatorial optimization.

Zongpeng Li received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, in 1999, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2005. Since 2005, he has been with the Department of Computer Science, University of Calgary, Calgary, AB, Canada. His research interest includes computer networks, network coding, and cloud computing.

Qingmei Ren received the B.E. degree from the Department of Communication Engineering, Beijing University of Posts and Telecommunication, Beijing, China, in 2015. She is currently pursuing the master's degree with the Department of Computer Science and Technology, Tsinghua University, Beijing. Her research interests include the areas of wireless networking and mobile cloud computing.

Yangjun Zhang received the B.E. degree from the Department of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the master's degree with the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include the areas of wireless networking and mobile cloud computing.