A General Framework for Crowdsourcing Mobile Communication, Computation, and Caching

Ming Tang, Lin Gao, and Jianwei Huang

***Abstract*—Today's mobile devices are capable of tackling various complicated tasks that may require a large amount of communication, computation, and caching (3C) resources. Due to users' heterogeneous resources and service requirements, it is challenging for each user to always accomplish his task satisfactorily. To alleviate this issue, mobile users can exploit the heterogeneity and crowdsource their resources to enhance the task execution performance. In this paper, we propose a general 3C framework that enables mobile users to share all three types of resources through device-to-device connections. Such a framework generalizes many existing 1C/2C resource sharing models (that only shares one or two types of resources among users). To quantify the benefit of the proposed framework, we focus on an energy minimization problem, and show that the 3C framework always achieves a smaller total energy consumption, comparing with other 1C/2C models. Furthermore, we show that the energy reduction is maximized, when user connection probability and content caching ratio are neither too large nor too small. Our numerical results show that, when ignoring device-to-device transmission energy, the general 3C framework can reduce the total energy consumption by 82.98%, comparing with the 1C/2C models.**

I. Introduction

A. Background and Motivation

We have been increasingly using mobile devices to tackle various complicated tasks, such as online gaming, data processing, and augmented reality. These tasks may request a large amount of *communication* resources (for data downloading and uploading), *computation* resources (for data processing), and *caching* resources (for data storage and retrieval), named *3C resources*. Due to users' heterogeneous resources and service requirements, it is challenging for each user to always accomplish his task satisfactorily.

To resolve the issue of potential mismatch of users' requirements and resources, some recent research have focused on the approach of mobile cloud offloading [1], where mobile devices offload their data processing tasks or data storage to the cloud. However, such an offloading approach is most suitable when the tasks request a large amount of computation or storage resources but a small amount of communication resources, in which case the overhead induced by the additional communications between users and the cloud does not cause significant energy and latency concerns [2].

Another line of studies focused on mobile user resource sharing, where nearby mobile devices share their resources

This work is supported by the General Research Fund (Project Number CUHK 14206315 and 14219016) established under the University Grant Committee of the Hong Kong Special Administrative Region, China. M. Tang and J. Huang are with the Chinese University of Hong Kong, Hong Kong, E-mail: tm014@ie.cuhk.edu.hk, jwhuang@ie.cuhk.edu.hk; L. Gao is with Harbin Institute of Technology (Shenzhen), China, E-mail: gaol@hit.edu.cn.

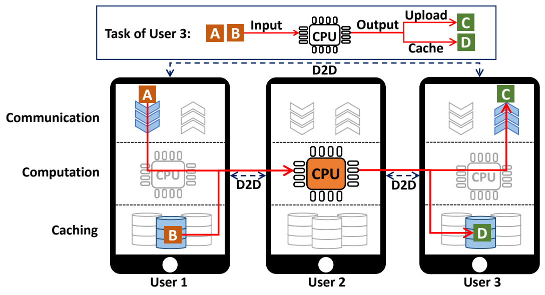


Fig. 1. An example of the general 3C framework.

for cooperative task executions through local device-to-device (D2D) connections (e.g., Bluetooth and WiFi Direct). Such resource sharing can effectively pool the mobile users' heterogeneous resources, and improve the overall network performance.

There have been several end-user resource sharing models that involve the sharing of one type of the 3C resources, named 1C sharing models. For example, papers [3] and [4] proposed user-provided networks, where nearby mobile users share their Internet connectivity for cooperative downloading. Papers [5] and [6] proposed ad hoc computation offloading models, which enables the computation resources sharing among nearby mobile users for data processing. Papers [7] and [8] proposed ad hoc content sharing models, where nearby mobile users share their cached contents. Some recent work also considered 2C resource sharing models (i.e., sharing two types of resources), such as distributed data analysis models [9], [10] that enables the sharing of caching and computation for data collection and data analysis.

However, these earlier excellent studies did not exploit the potential benefit of jointly optimizing the 3C resources. There are several benefits of studying a general framework that enables the joint 3C resources sharing among mobile users. First, the general framework allows users who are interested in different applications to cooperate with each other, hence may open up new resource sharing possibilities. Second, comparing with 1C/2C resource sharing, the joint 3C resource sharing can further exploit the diversity of the mobile devices and achieve a more effective resource allocation. However, fully characterizing the benefit of such a general 3C sharing under different network scenarios is a challenging and long-term task.

B. Solution Approach and Contribution

In this paper, we present the first study regarding the general 3C framework, which aims to generalize many of the existing end-user resource sharing models and provide additional network design and optimization flexibility. Instead of modeling the network based on the tasks or applications that users want to accomplish, this framework is centered around the characterization of the resource requirements of each user task and the proper sharing and allocation of the resources.

Figure 1 shows an example of the general 3C framework. User 3 has a task with the following components: retrieving contents "A" and "B" (either from the Internet through downloading or directly from some user's cache), performing computation, and outputing contents "C" (to the Internet) and "D" (to user 3's local cache). With the general 3C framework, three users can form a group with D2D connections and share their communication (downlink and uplink), computation (CPU), and caching resources. In this example, user 1 is responsible for obtaining the input contents and delivering them to user 2 for computation, user 2 performs the computation and sends the output contents to user 3, and user 3 further uploads content "C" and stores content "D" in his local cache.

To demonstrate the concrete benefits of the general 3C framework, we will focus on an energy minimization problem, and show how much the 3C framework can reduce the total energy consumption, comparing with the 1C/2C models. We characterize the significance of such energy reduction analytically and numerically. Our key contributions are as follows:

- *A General 3C Resource Sharing Framework:* We propose a general framework for the joint 3C resources sharing, which generalizes many of the existing end-user resource sharing models.
- *Energy Efficiency Optimization:* We focus on an energy minimization problem, and show that the proposed 3C framework always achieves a lower overall energy consumption, comparing with the 1C/2C models. In addition, we show that the energy reduction is maximized, when user connection probability and content caching ratio are neither too large nor too small.
- *Experiments and Performances:* Numerical results show that, when ignoring the D2D transmission energy, the general 3C framework can reduce the total energy consumption by 82.98%, comparing with the 1C/2C models.

The rest of this paper is organized as follows. We propose the general framework in Section II. Focusing on the energy minimization problem, we analyze the energy reduction due to the framework theoretically and numerically in Sections III and IV, respectively. In Section V, we conclude our work.

II. A General 3C Sharing Framework

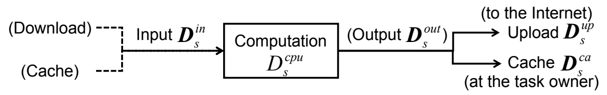
A. System Model

We consider a set of mobile users $\mathcal{N} = \{1, 2, ..., N\}$, some of which have tasks to execute. These users form a mesh network for cooperative task execution, where $\mathcal{E}(n)$ denotes the set of users who have D2D connections with user n.

Let $\mathcal{K} = \{1, 2, ..., K\}$ denote the set of all contents possibly involved in the system, where $k \in \mathcal{K}$ has a size L_k. These contents can be downloaded from the Internet, retrieved from users' caches, or fetched from the output of tasks.

User Model: Each user $n \in \mathcal{N}$ owns different kinds of resources, represented by a tuple:

$$\boldsymbol{Q}_n = (Q_n^{down}, Q_n^{up}, Q_n^{cpu}, \boldsymbol{Q}_n^{ca}).$$



(Download)
(Cache)
Input $\boldsymbol{D}_s^{in}$
Computation D_s^{cpu}
(Output $\boldsymbol{D}_s^{out}$)
(to the Internet)
Upload $\boldsymbol{D}_s^{up}$
Cache $\boldsymbol{D}_s^{ca}$
(at the task owner)

Fig. 2. General task model.

Here, Q_n^{down}, Q_n^{cpu}, and Q_n^{up} denote user n's capacities of downloading (bits per second), computation (CPU cycles per second), and uploading (bits per second), respectively. The vector $\boldsymbol{Q}_n^{ca} = \{Q_{n1}^{ca}, ..., Q_{nK}^{ca}\}$ denotes user n's cached contents, i.e., $Q_{nk}^{ca} = 1$ if user n has cached content k, and $Q_{nk}^{ca} = 0$ otherwise. Let $Q_{n\to m}^{d2d}$ be the D2D transmission capacity (bits per second) from user n to user m. Let c_n^{down}, c_n^{cpu}, c_n^{up}, and $c_{n\to m}^{d2d}$ denote the energy consumption of the corresponding operations per second, respectively.

Task Model: Each user may or may not initiate a task. Let $\mathcal{S} = \{1, 2, .., S\}$ denote the set of all tasks initiated by users, where each task is represented by the task model in Figure 2. Namely, each task has a main *computation* module[1] (e.g., data processing), which requests some input contents (from downloading or users' caches) and generates some output contents (for uploading or storing in the task owner's cache).

Based on Figure 2, each task can be denoted by a tuple:

$$\boldsymbol{D}_s = (u_s, \boldsymbol{A}_s^{cpu}, (\boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{D}_s^{out}), \boldsymbol{D}_s^{up}, \boldsymbol{D}_s^{ca}).$$

Parameter u_s denotes the task owner ID, i.e., the user who requests the task s. Vector $\boldsymbol{A}_s^{cpu} = \{A_{s1}^{cpu}, ..., A_{sN}^{cpu}\}$ indicates each user's capability of performing the computation, i.e., $A_{sn}^{cpu} = 1$ if user n is capable of performing the computation of task s. Parameter D_s^{cpu} is the total CPU cycles requested by task s. The K-dimensional vectors $\boldsymbol{D}_s^{in}$, $\boldsymbol{D}_s^{out}$, $\boldsymbol{D}_s^{up}$, and $\boldsymbol{D}^{ca}$ refer to the contents that requested for the corresponding operations, i.e., $D_{sk}^{X} = 1$ ($X = in, out, up, ca$) if content k is requested by task s for input, output, uploading, or cache, respectively.

Based on above discussions, we can divide each task into three *subtasks* — the input subtask, the computation subtask, and the uploading subtask. We consider a general 3C framework, where different subtasks (of a same task) can be potentially performed by different users. Moreover, the downloading or the uploading of different contents (of a subtask) can also be performed by different users, leading to the maximum flexibility for resource sharing and system performance optimization.

B. Problem Statement

The 3C framework allows us to model various network resource sharing scenarios with different system objectives, e.g., energy minimization or task number maximization. We first explain the decision variables and system constraints, and then propose an energy minimization problem.

We consider the following *binary* decision variables. Let $x_{s\to n}^{cpu}$ denote whether user n performs the computation subtask of task s. Let $x_{s,k:\to n}^{in}$ or $x_{s,k:\to n}^{up}$ denote whether the input or the uploading of task s's content k is performed by user n. Let $x_{s,k:\to n}^{down}$ denote whether an input content k of task s is

[1]If computation is not requested, the requested computation amount is zero.

downloaded user n. For each of the above variables, $x = 1$ represents that the subtask (or the content of the subtask) is allocated to the corresponding user. To model the multi-hop content delivery, we define variables $z^{in}_{s,k:i\to j}$, $z^{up}_{s,k:i\to j}$, and $z^{ca}_{s,k:i\to j}$, where $z^{X}_{s,k:\to n} = 1$ $(X = in, up, ca)$ if task s's (input, uploading, and cache) content k is delivered from user i to j, respectively.

*1) **Constraints**:* The task allocation decisions should satisfy the following constraints.

Allocation constraints: When executing a task, its computation subtask should be allocated to exactly one user (even if D^{cpu} is zero[2]):

$$\sum_{n\in\mathcal{N}} x^{cpu}_{s\to n} = 1,\ \forall s \in \mathcal{S}. \tag{1}$$

The input and the uploading of a requested content should be allocated to one user, respectively:

$$\sum_{n\in\mathcal{N}} x^{in}_{s,k:\to n} = D^{in}_{sk},\ \forall s \in \mathcal{S}, k \in \mathcal{K}. \tag{2}$$

$$\sum_{n\in\mathcal{N}} x^{up}_{s,k:\to n} = D^{up}_{sk},\ \forall s \in \mathcal{S}, k \in \mathcal{K}. \tag{3}$$

Capacity constraints: The user who is assigned to perform the computation subtask should be capable of performing it:

$$x^{cpu}_{s\to n} \le A^{cpu}_{sn},\ \forall s \in \mathcal{S}, n \in \mathcal{N}. \tag{4}$$

The input of a requested content is executed by a user who either has the content in his local cache or is going to download it from the Internet (for any task $r \in \mathcal{S}$):

$$x^{in}_{s,k:\to n} \le Q^{ca}_{nk} + \sum_{r\in\mathcal{S}} x^{down}_{r,k:\to n},\ \forall s \in \mathcal{S}, n \in \mathcal{N}, k \in \mathcal{K} \tag{5}$$

Network flow balancing constraints: These constraints are applicable when the content delivery is through multi-hop transmissions. For any content at any user, the incoming number of the contents (i.e., the number of the contents that received by the user) should equal the outgoing number (i.e., the number of the contents that are transmitted from the user).

Taking the content input ($z^{in}_{s,k:i\to j}$) as an example. For any task $s \in \mathcal{S}$ and content $k \in \mathcal{K}$, the flow constraint at user i is

$$\sum_{j\in\mathcal{E}(i)} z^{in}_{s,k:j\to i} + x^{in}_{s,k:\to i} D^{in}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{in}_{s,k:i\to j} + x^{cpu}_{s\to i} D^{in}_{sk}. \tag{6}$$

The left-hand side is the incoming number of task s's content k, including the number of the content that user i receives from his nearby users and the number of the content he is responsible for input (one if $x^{in}_{s,k:\to i} = 1$ and $D^{in}_{sk} = 1$). The right-hand side is the outgoing number of task s's content k, including the number of the content that user i transmits to his nearby users and the number of the content he uses to perform computation (one if $x^{cpu}_{s\to i} = 1$ and $D^{in}_{sk} = 1$).

We can apply the similar argument to the caching and uploading and obtain the following constraints,

$$\sum_{j\in\mathcal{E}(i)} z^{ca}_{s,k:j\to i} + x^{cpu}_{s\to i} D^{ca}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{ca}_{s,k:i\to j} + \mathbf{1}_{i=u_s} D^{ca}_{sk}; \tag{7}$$

$$\sum_{j\in\mathcal{E}(i)} z^{up}_{s,k:j\to i} + x^{cpu}_{s\to i} D^{up}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{up}_{s,k:i\to j} + x^{up}_{s,k:\to i} D^{up}_{sk}. \tag{8}$$

Operator $\mathbf{1}_{i=u_s} = 1$ if $i = u_s$, and $\mathbf{1}_{i=u_s} = 0$ if $i \ne u_s$.

Worst Delay Constraints: We first show a worst case delay constraint, and then explain how the worst delay is calculated.

For executing a task s, the *worst* (maximum) delay, denoted by T_s, should be smaller than a delay restriction $\bar{T}_s$:

$$T_s \le \bar{T}_s, \forall s \in \mathcal{S}. \tag{9}$$

The worst delay is the maximum delay that may happen due to the resource sharing. Constraint (9) ensures that the delay constraint $\bar{T}_s$ is always satisfied in the worst case.

The worst delay of a task comprises the worst delays of the downloading, computation, uploading, and D2D transmission[3]:

$$T_s = T^{down}_s + T^{cpu}_s + T^{up}_s + T^{d2d}_s.$$

Let τ^{down}_n, τ^{cpu}_n, and τ^{up}_n be user n's total performing time of completing all the allocated downloading, computation, and uploading subtasks, and let $\tau^{d2d}_{n\to m}$ be the total D2D transmission time from user n to user m of completing all the allocated transmitting contents. Formally,

$$\tau^{down}_n = \frac{\sum_{r=1}^{S}\sum_{k=1}^{K} x^{down}_{r,k:\to n} L_k}{Q^{down}_n},$$

$$\tau^{cpu}_n = \frac{\sum_{r=1}^{S} x^{cpu}_{r\to n} D^{cpu}_r}{Q^{cpu}_n};\ \ \tau^{up}_n = \frac{\sum_{r=1}^{S}\sum_{k=1}^{K} x^{up}_{r,k:\to n} L_k}{Q^{up}_n}.$$

$$\tau^{d2d}_{n\to m} = \frac{\sum_{r=1}^{S}\sum_{k=1}^{K}(z^{in}_{s,k:n\to m} + z^{up}_{s,k:n\to m} + z^{ca}_{s,k:n\to m}) L_k}{Q^{d2d}_{n\to m}}.$$

The task s's worst downloading delay is derived as follows. Suppose any user n divides his total downloading capacity to multiple tasks (allocated to him) according to the *total size* of each task's downloading amount. When all the downloading requests allocated to the user n arrive simultaneously, these tasks will share the user n's capacity during the entire downloading process, which leads to the maximum downloading time τ^{down}_n for each of these tasks. As a task s can obtain multiple contents from different users, the worst downloading delay that task s experiences is the maximum downloading time τ^{down}_n among all the users who are responsible for downloading any of the task s's requiring contents:

$$T^{down}_s = \max_{\{n \mid \sum_{k=1}^{K} x^{in}_{s,k:\to n}(1-Q^{ca}_{nk})>0\}} \tau^{down}_n.$$

A similar idea applies to the formulation of the worst computation and uploading delays:

$$T^{cpu}_s = \sum_{n=1}^{N} x^{cpu}_{s\to n}\tau^{cpu}_n,\quad T^{up}_s = \max_{\{n \mid \sum_{k=1}^{K} x^{up}_{s,k:\to n}>0\}} \tau^{up}_n.$$

The worst D2D transmission delay is approximately the maximum delay among all the D2D connections that transmit task s's contents[4]:

$$T^{d2d}_s = \max_{\{n,m \mid \sum_{k=1}^{K}(z^{in}_{s,k:n\to m} + z^{up}_{s,k:n\to m} + z^{ca}_{s,k:n\to m})>0\}} \tau^{d2d}_{n\to m}.$$

[2]When $D^{cpu} = 0$, the computation subtask will be allocated to one of the content input users, the uploading users, or the task owner (depending on the resource allocation objectives), so that it can be regarded as a content gathering without introducing additional energy consumption.

[3]The content input from a user's cache (without downloading) or the content output to the task owner's cache will not induce any delay.

[4]Due to the complexity of modeling multi-hop transmission delays, we consider the worst delay approximation, which will be close to the actual worst delay when a few connections incur delays much larger than the others.

TABLE I
EXISTING MODELS THAT THE 3C FRAMEWORK CAN GENERALIZE.

Shared Resource	Examples and Task Models $\boldsymbol{D}_s$
Downloading	(1) User-provided networks [3], [4] $(u_s, \mathbf{1}, (\boldsymbol{D}_s^{data}, 0, \boldsymbol{D}_s^{data}), \mathbf{0}, \boldsymbol{D}_s^{data})$
Computation	(2) Ad hoc computation offloading [5], [6] $(u_s, \boldsymbol{A}_s^{cpu}, (\boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{D}_s^{out}), \mathbf{0}, \boldsymbol{D}_s^{out})$
Uploading	(3) Ad hoc content uploading [11] $(u_s, \mathbf{1}, (\boldsymbol{D}_s^{data}, 0, \boldsymbol{D}_s^{data}), \boldsymbol{D}_s^{data}, \mathbf{0})$
Content	(4) Ad hoc content sharing [7], [8] $(u_s, \mathbf{1}, (\boldsymbol{D}_s^{data}, 0, \boldsymbol{D}_s^{data}), \mathbf{0}, \boldsymbol{D}_s^{data})$
Hybrid	(5) Distributed data analysis [9], [10] $(u_s, \boldsymbol{A}_s^{cpu}, (\boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{D}_s^{out}), \mathbf{0}, \boldsymbol{D}_s^{out})$
Device-to-device connections	(6) Ad hoc data forwarding [12] $(u_s, \boldsymbol{A}_s^{rec}, (\boldsymbol{D}_s^{data}, 0, \mathbf{0}), \mathbf{0}, \mathbf{0})$

2) ***Energy****:* Now we define the energy for executing a task s, which consists of the energy for downloading, computation, uploading, and D2D transmission. Formally

$$E_s = E_s^{down} + E_s^{cpu} + E_s^{up} + E_s^{d2d}.$$

We assume that the energy consumptions are linear with the performing time [1]:

$$E_s^{down} = \sum_{n=1}^{N} \frac{c_n^{down} \sum_{k=1}^{K} x_{s,k:\to n}^{down} L_k}{Q_n^{down}};$$

$$E_s^{up} = \sum_{n=1}^{N} \frac{c_n^{up} \sum\limits_{k=1}^{K} x_{s,k:\to n}^{up} L_k}{Q_n^{up}}; E_s^{cpu} = \sum_{n=1}^{N} \frac{c_n^{cpu} x_{s\to n}^{cpu} D_s^{cpu}}{Q_n^{cpu}};$$

$$E_s^{d2d} = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{c_{i\to j}^{d2d} \sum\limits_{k=1}^{K} (z_{s,k:i\to j}^{in} + z_{s,k:i\to j}^{up} + z_{s,k:i\to j}^{ca}) L_k}{Q_{i\to j}^{d2d}}.$$

3) ***Problem Formulation****:* We are interested in minimizing the overall system energy consumption, i.e.,

$$\begin{aligned} &\text{minimize} && \sum_{s\in\mathcal{S}} E_s \\ &\text{subject to} && (1) \sim (9) \end{aligned} \qquad (10)$$

The problem can be converted into an integer linear optimization problems, and can be solved by standard optimization solvers, such as CVX Gurobi. In practice, users may be widely distributed within a large area, hence a distributed algorithm might be needed for computing the resource allocation. We will discuss this further in Section III.

C. Generalization of Existing Models in the Literature

Through properly specifying various parameters, our proposed 3C framework can generalize the existing 1C and 2C resource sharing models in the literature, as illustrated in Table I. In the table, vectors $\boldsymbol{D}_s^{data}$ denote the contents that are requested for the corresponding operations. Vector $\boldsymbol{A}_s^{rec}$ denotes the receiver set of the forwarded data. In addition, although the ad hoc computation offloading and the distributed data analysis have the same task model, the modeling of users who have the input contents are different, which corresponds to task owners and any subset of users, respectively.

III. ENERGY MINIMIZATION IN 3C SHARING

Comparing with the 1C/2C models, the 3C framework improves the system performance from two aspects: enlarging the cooperative group and enabling the multi-resource cooperation. In this section, we demonstrate the advantages of the two aspects by analyzing the energy minimization problem (10). To simplify the analysis, we ignore the D2D transmission energy, the impact of which is examined numerically in Section IV.

A. System Settings

We consider a random graph model $G(N, p)$ [13] to model the mobile users and their D2D connections, where N users in the graph and every two users are connected randomly and independently with the same probability p. Suppose that the network is large and sparse, so that N approaches infinite with Np being a constant.[5] These users are heterogeneous in terms of their own resources. Specifically, each user n owns some resources Q_n^{down}, Q_n^{cpu}, Q_n^{up}, and $\boldsymbol{Q}_n^{ca}$. The capacities Q_n^X $(X = \{down, cpu, up\})$ is independent and identically distributed (i.i.d.) with the cumulative distribution function (cdf) $F_Q^X(x)$ and the probability density function (pdf) $f_Q^X(x)$, and the capacities are distributed within range $(\underline{Q}^X, \overline{Q}^X)$, i.e.,

$$F_Q^X(\underline{Q}^X) = 0, \; F_Q^X(\overline{Q}^X) = 1.$$

Different kinds of capacities will follow different distributions. In addition, each user n uniformly and randomly caches M^{ca} contents in $\boldsymbol{Q}_n^{ca}$ before the task allocation[6], i.e., $\sum_{k=1}^{K} Q_{nk}^{ca} = M^{ca}$. For the convenience of analysis, we assume that all the contents have the same size that is normalized to one, i.e., $L_k = 1$ for all k, and the users are homogeneous in terms of energy coefficients, i.e., $c_n^X = c^X$, $X = \{down, cpu, up\}$.

As mobile users may be distributed within a large area, it may be difficult to optimize the network performance using a centralized algorithm that sometimes require huge information gathering across the entire network. Hence, in the following analysis, we consider a distributed cooperation algorithm: each user can choose to allocate his tasks to his neighbors only (i.e., the users who are connected with him) to minimize the energy consumption. This algorithm provides a suboptimal solution of the original problem (10). The performance gap between the suboptimal distributed algorithm and the optimal centralized algorithm (10) will be quantified in Section IV.

B. Energy Reduction Due to the Enlarged Cooperative Group

In this subsection, we focus on a particular resource (e.g., downloading, uploading, or computation), and show how the number of the users who contribute resources affects the expected energy consumption on that resource. For presentation simplicity, the term "resource" in Section III-B only refers to one particular resource, so we omit the resource-specific superscripts and sub-scripts. Without loss of generality, we set the energy coefficient of the resource to be one.

[5]This assumption is reasonable as most real networks that are sparse [13].

[6]The uniformly and randomly caching is a widely used benchmark in proactive caching, leading to the lower bound of the system performance [14].

Suppose that each user is willing to contribute his resource to other users with a probability α (hence, on average α fraction of users contributes resources). Under these network homogeneity assumptions, each task will have the same expected energy consumption. We will try to understand how the fraction α affects the expected energy of each task.

We first characterize the expected lower bound of the energy consumption of each task under a particular α (i.e., the best that the system can achieve when α fraction of users contributes), and then show how the fraction α affects such a lower bound.

1) Expected Lower Bound Energy Consumption: Under the general framework, we want to allocate each task to minimize the energy under the delay constraints. When the delay constraint is large enough, each task can be allocated to the neighbors (of the task owner) who has the highest capacity, and this leads to the lower bound energy consumption. By using the order statistic result [15], the distribution of the highest capacity among total n users is given by

$$f_{(n)}(x) = n(F(x))^{n-1} f(x). \tag{11}$$

In the random graph $G(N, p)$, any user's degree (i.e., the number of the user's neighbors) is distributed as follows [13]:

$$P(degree = m) = \frac{(Np)^m e^{-Np}}{m!}. \tag{12}$$

For a user with a degree m, the probability that $\hat{N}$ of his neighbors share their resources is $P(\hat{N}|m) = C_m^{\hat{N}} \alpha^{\hat{N}} (1-\alpha)^{m-\hat{N}}$. Taking the expectation over $\hat{N} = \{0, 1, ..., m\}$, the expected lower bound energy consumption of this user's task is

$$\hat{W}(\alpha, m) = \sum_{\hat{N}=0}^{m} P(\hat{N}|m) \int_{\underline{Q}}^{\overline{Q}} \frac{1}{x} f_{(\hat{N}+1)}(x) dx, \tag{13}$$

where $f_{(\hat{N}+1)}(x)$ is the distribution of the highest capacity among the $\hat{N}$ neighbors and the user himself. Then, taking the expectation of $\hat{W}(\alpha, m)$ over all degrees $m = \{0, 1, ..., \infty\}$, we obtain the expected lower bound energy of each task:

Proposition 1 (Lower Bound Energy). *In the random graph $G(N, p)$, if α fraction of users contributes their resources, the expected lower bounded energy consumption of the resource on each task is*

$$W(\alpha) = \frac{1}{\overline{Q}} + \int_{\underline{Q}}^{\overline{Q}} e^{Np(F(x)-1)\alpha} F(x) x^{-2} dx. \tag{14}$$

2) Impact of Contributing User Fraction α: Based on Proposition 1, $W(\alpha)$ satisfies the following properties.

Corollary 1 (Non-Increasing Energy in α). *The expected lower bound energy $W(\alpha)$ is non-increasing and convex in α.*

Intuitively, as the fraction α increases, the total number of contributing user increases, so the expected lower bound energy decreases. Furthermore, we observe the diminishing marginal return, i.e., the additional increase of contributing users induces less energy reduction.

Corollary 2 (Existence of Optimal p). *For any contributing user fraction $\alpha \in (0, 1]$ and distribution function $f(x)$, there always exists an unique optimal $p^*(\alpha, f(x))$ that maximizes the marginal decrease of the expected lower bound energy consumption $|W'(\alpha)|$, where $Np^*(\alpha, f(x)) > 1$.*

Intuitively, when p is small (i.e., the users are sparse), the increasing contributing users may not have connections with the requiring users, so the marginal energy reduction is small; when p is large (i.e., the users are dense), most users may already connect with some high capacity neighbors, so additional contributing users make small differences.

Hence, in practice, for a particular $f(x)$ and α, the system can reduce the energy consumption through increasing the fraction of contributing users α, especially when $p = p^*(\alpha, f(x))$ (under which a large number of users form a giant component, and the others form numerous isolated components [13]).

C. Energy Reduction Due to the Multi-Resource Cooperations

We take user-provided network [3] as an example to demonstrate the energy reduction from exploiting multi-resource cooperations in the 3C framework. In the user-provided network, users share their downloading capacities only, without sharing their cached contents. In the 3C framework, users can share both their downloading capacities and their cached contents, which leads to a lower energy consumption.

We first describe users' content requesting model. Then, we compare the lower bound energy consumption between the user-provided networks and the 3C framework. Without loss of generality, we set $\alpha = 1$ (full cooperation among users) and normalize the downloading energy coefficient to be one.

1) Content Requesting Model: Suppose each user requests one content input task, which involves uniformly and randomly requesting M^{req} contents from the content set $\mathcal{K}$, i.e., $\sum_{k=1}^{K} D_{nk}^{in} = M^{req}$. Considering a user with m neighbors, and let Z_m denote the number of requested contents (of the user) that have to be downloaded. The expectation of Z_m is

$$\mathbb{E}[Z_m] = M^{req} \left(1 - \frac{M^{ca}}{K}\right)^{m+1}, \forall m = 0, ..., \infty. \tag{15}$$

Similar as in Section III-B, we consider the energy consumption of one task due to the network homogeneity assumptions.

2) Lower Bound Energy Comparison: In the user-provided networks, the users share their downloading resources. Based on Section III-B, the expected lower bound energy is given by

$$W^U = \sum_{m=0}^{\infty} P(degree = m) \mathbb{E}[Z_0] \hat{W}(1, m), \tag{16}$$

where $\mathbb{E}[Z_0]$ is the expected number of the contents that have to be downloaded (without considering the possibility of retrieving content from neighbors' caches), and $\hat{W}(1, m)$ is the lower bound energy for downloading one content (under m neighbors), as in (13).

In the 3C framework, the requested contents can be either retrieved from users' cache or downloaded from the Internet. Through considering both the caching and the communication dimensions, the expected lower bound energy is given by

$$W^G = \sum_{m=0}^{\infty} P(degree = m) \mathbb{E}[Z_m] \hat{W}(1, m), \tag{17}$$

where $\mathbb{E}[Z_m]$ is the expected number of the contents that have to be downloaded under m neighbors.

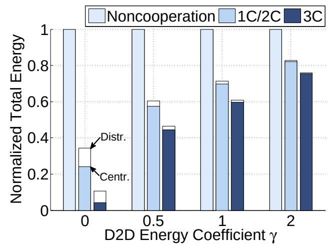


Fig. 3. Energy consumption comparisons.

The gap between W^U in (16) and W^G in (17) is as follows:

Proposition 2 (Energy Comparisons)**.** *The expected lower bound energy gap between W^U and W^G is as follows:*

$$W^U - W^G = M^{req}\left(1-\frac{M^{ca}}{K}\right) \times \Bigg[\left(1-e^{-Np\frac{M^{ca}}{K}}\right)\frac{1}{\overline{Q}} + \int_{\underline{Q}}^{\overline{Q}} \frac{F(x)}{x^2}\left(e^{Np(F(x)-1)} - e^{Np((1-\frac{M^{ca}}{K})F(x)-1)}\right)dx \Bigg] \qquad (18)$$

From Proposition 2, the 3C framework reduces the energy consumption, comparing with the user-provided models, because $W^U - W^G \geq 0$. As the requested number of contents M^{req} increases, the gap $W^U - W^G$ increases (i.e., the more the users request, the more the energy reduces). Moreover,

Corollary 3 (Impact of M^{ca})**.** *There exists a unique optimal caching ratio $M^{ca}/K = R^*(f(x), Np)$ that maximizes $W^U - W^G$, where the optimal ratio satisfies $R^*(f(x), Np) > \frac{(Np+2)-\sqrt{(Np)^2+4}}{2Np}$ for any $f(x)$.*

This corollary shows that, when each user caches $M^{ca} = R^*(f(x), Np) \cdot K$ contents, the content sharing due to the general 3C framework leads to the most significant energy consumption reduction. Intuitively, when each user caches a large number of contents, there is a large probability that the requested content has already been cached by the content requester himself; in this case, further content sharing may not help with the energy reduction. When each user caches a small number of contents, the content sharing may not provide much energy reduction due to the limited cached contents from the content requester's neighbors. Moreover, as Np increases, the lower bound of the optimal ratio $R^*(f(x), Np)$ (i.e., $((Np+2)-\sqrt{(Np)^2+4})/(2Np)$) decreases. Intuitively, when the connection probability p is larger, each user has more neighbors (hence is available to access more cached contents), so a smaller caching ratio will be sufficient to achieve the maximum energy reduction (due to the 3C framework).

IV. Experiment and Performance

We consider a scenario of 20 users, who form pair-wise connections with a probability $p = 0.3$. Each user has a task to execute. We perform 100 simulations and show the average results. In each simulation, we randomly generate the parameters of the user and task models. Each user randomly selects a mobile application among the user-provided network, the content sharing, and the distributed data analysis. Moreover, we set the average value of the D2D transmission energy to be γ and normalize the average value of all the other energy to be one (to analyze the impact of the D2D transmission energy).[7]

We perform experiments in three cooperation settings: (i) noncooperation benchmark; (ii) the 1C/2C models, where only the users selecting the same application can cooperate; (iii) the 3C framework, where all the users will cooperate.

Figure 3 shows the normalized total energy with respective to the total energy consumption in (i). Note that, for each bar, we show the energy consumptions under both the centralized and distributed algorithms. The rest discussions only focus on the results of the centralized algorithm.

Comparison between (i) and (iii): When ignoring the D2D energy (i.e., $\gamma = 0$), the 3C framework reduces the energy by 95.88%. As γ increases, the reduction decreases because of the increasing cooperation cost. However, when $\gamma = 2$ (i.e., the D2D energy coefficient is two times as large as the other energy coefficients), the reduction is still more than 24.51%.

Comparison between (ii) and (iii): When ignoring the D2D energy (i.e., $\gamma = 0$), the 3C framework reduces the energy by 82.98%. As γ increases, this energy reduction decreases due to the increasing D2D transmission cost. When $\gamma = 2$, the 3C framework still reduces the energy by 7.98%.

V. Conclusion

In this paper, we propose a general 3C framework, which generalizes many of the existing studies on end-user resource sharing. We show that the 3C framework significantly reduces the energy consumption (comparing with the 1C/2C models), especially when user connection probability and content caching ratio are neither too large nor too small. This work serves as the first step for studying the joint 3C resource sharing. For future work, incentive mechanisms have to be designed to motivate the sharing among mobile users.

References

[1] K. Kumar, J. Liu, Y.H. Lu, and B. Bhargava "A survey of computation offloading for mobile systems," *Mobile Netw. and Appl.*, 2013, 18(1): 129-140.

[2] K. Kumar and Y.H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?," *Comput. J.*, 2010, 43(4): 51-56.

[3] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas,"Efficient and Fair Collaborative Mobile Internet Access", *IEEE/ACM Trans. on Netw.*, 2017.

[4] D. Syrivelis, *et al.*, "Bits and coins: Supporting collaborative consumption of mobile internet," *IEEE INFOCOM*. 2015.

[5] F. Chi, X. Wang, W. Cai, and V. Leung, "Ad Hoc Cloudlet Based Cooperative Cloud Gaming," *IEEE CloudCom*, 2014.

[6] M. Chen, *et al.*, "On the computation offloading at ad hoc cloudlet: architecture and service modes," *IEEE Commun. Mag.*, 2015.

[7] J. Jiang, Y. Zhu, B. Li, and B Li, "Rally: Device-to-Device Content Sharing in LTE Networks as a Game," *IEEE MASS*, 2015.

[8] Z. Chen, Y. Liu, B. Zhou, and M. Tao, "Caching Incentive Design in Wireless D2D Networks: A Stackelberg Game Approach," *IEEE ICC*, 2016.

[9] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," *ComSIS*, 2014.

[10] A. Destounis, G.S. Paschos, and I. Koutsopoulos, "Streaming Big Data meets Backpressure in Distributed Network Computation," *INFOCOM*, 2016.

[11] L. Militano, *et al.*, "A constrained coalition formation game for multihop D2D content uploading," *IEEE Trans. on Wireless Commun.*, 2016.

[12] G.D. Caro, F. Ducatelle, and L.M. Gambardella, "AntHocNet: an ant-based hybrid routing algorithm for mobile ad hoc networks," *PPSN*, 2004: 461-470.

[13] A.L. Barabsi and M. Psfai, "Network Science," *Cambridge University Press*, 2016.

[14] E. Bastug, M. Bennis, and M. Debbah, "Living on the edge: The role of proactive caching in 5G wireless networks," *IEEE Commun. Mag.*, 2014, 52(8): 82-89.

[15] B.C. Arnold, N. Balakrishnan, and H.N. Nagaraja, "A first course in order statistics," *SIAM*, 2008.

[16] G.P. Perrucci, F.H.P. Fitzek, and J. Widmer, "Survey on energy consumption entities on the smartphone platform," *VTC*, 2011.

[7]In reality, coefficient $\gamma \leq 1$ usually holds [16].