Enabling Edge Cooperation in Tactile Internet via 3C Resource Sharing

Ming Tang, *Student Member, IEEE*, Lin Gao, *Senior Member, IEEE*, and Jianwei Huang, *Fellow, IEEE*

***Abstract*— Tactile Internet often requires: 1) the ultra-reliable and ultra-responsive network connection and 2) the proactive and intelligent actuation at edge devices. A promising approach to address these requirements is to enable mobile edge devices to share their communication, computation, and caching (3C) resources via device-to-device connections. In this paper, we propose a general 3C resource sharing framework, which includes many existing 1C/2C sharing models in the literature as special cases. Comparing with the 1C/2C models, the proposed 3C framework can further improve the resource utilization efficiency by offering more flexibilities in terms of the device cooperation and resource scheduling. As a typical example, we focus on the energy utilization under the proposed 3C framework. Specifically, we formulate an energy consumption minimization problem, which is an integer non-convex optimization problem. To solve the problem, we first transform it into an equivalent integer linear programming problem that is much easier to solve. Then, we propose a heuristic algorithm based on linear programming, which can further reduce the computation time and produce an empirically close-to-optimal solution. Moreover, we evaluate the energy reduction due to the 3C sharing both analytically and numerically. Numerical results show that, comparing with the existing 1C/2C approaches, the proposed 3C sharing framework can reduce the total energy consumption by 83.8% when the D2D energy is negligible. The energy reduction is still 27.5% when the D2D transmission energy per unit time is twice as large as the cellular transmission energy per unit time.**

***Index Terms*— Tactile Internet, fog computing, D2D communication, edge resource sharing, resource optimization.**

I. Introduction

A. Background and Motivation

With the fast development of mobile communication and information technologies, Tactile Internet [2] has been recently proposed to support humans to control edge devices (e.g., robots, smart-phones, and virtual/augmented reality devices) remotely in real time. By enabling tactile sensations, Tactile Internet can enrich the human-to-machine interaction and revolutionize the interaction among edge devices. Hence, Tactile Internet has attracted various applications such as automation industrial, real-time gaming, and virtual/augmented reality.

Many applications of Tactile Internet have two main requirements [2]: (i) the ultra-reliable and ultra-responsive network connection, and (ii) the proactive and intelligent actuation at edges. To address these two requirements from the perspective of mobile edge devices, a promising approach is to enable device-to-device (D2D) based resource sharing among edge devices, where the resources include communication, computation, and caching (3C) resources. Such a resource sharing can be realized by D2D communication technologies [3], including the ad hoc mode of the IEEE 802.11 standards, WiFi direct, and Bluetooth. Some business instances also support such a sharing, such as Open Garden (https://www.opengarden.com/).

For example, to improve the reliability of network connections, edge devices can share their communication resources to better exploit the devices' heterogeneous network capacities to satisfy their quality-of-service (QoS) requirements. To promote the intelligent actuation, edge devices can also share their computation and caching resources, so as to efficiently utilize their resources to satisfy their intensive task requirements.

Many of the existing works focused on the sharing of one type of the 3C resources [4]–[9], which we call 1C sharing models. For example, the user-provided networking models in [4] and [5] focus on the sharing of communication resource, the ad hoc computation offloading models in [6] and [7] focus on the sharing of computation resource, and the ad hoc content sharing models in [8] and [9] focus on the sharing of caching resource. Some other recent works further considered the sharing of two types of the 3C resources, which we call 2C sharing models. Typical examples of 2C sharing include the distributed data analysis models in [10] and [11], which focus on the sharing of computation and caching resources.

Despite the success of the earlier 1C/2C resource sharing models, there are still significant potential benefits of exploiting the joint 3C resource sharing framework. Such a 3C sharing framework can further improve the resource utilization efficiency, by offering more flexibilities in terms of device cooperation and resource scheduling. Regarding the device cooperation, a joint 3C framework can enable devices performing different tasks to cooperate with each other, which leads to an increased number of participating devices and hence more cooperation opportunities. Regarding the resource scheduling, the joint optimization of 3C resources can lead to a more efficient resource allocation.

Manuscript received April 16, 2018; revised September 16, 2018; accepted September 21, 2018. Date of publication October 5, 2018; date of current version November 30, 2018. This work was supported in part by the General Research Funds established under the University Grant Committee of the Hong Kong Special Administrative Region, China, under Project CUHK 14206315 and Project CUHK 14219016, and in part by the National Natural Science Foundation of China under Grant 61771162. This paper was presented at the IEEE GLOBECOM [1]. *(Corresponding authors: Lin Gao; Jianwei Huang.)*

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 the Department of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen 518000, China (e-mail: gaol@hit.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSAC.2018.2874123

0733-8716 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

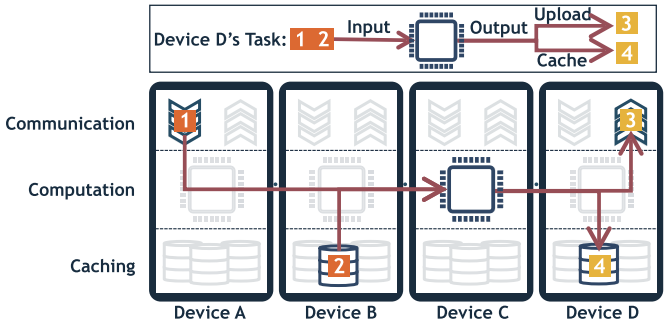

Fig. 1. An example of the general 3C framework.

B. Solution Approach and Contribution

In this paper, we present the first study regarding the general 3C resource sharing framework, which aims to generalize existing edge device resource sharing (1C/2C) models and provide additional network design and optimization flexibilities. A key feature of this new 3C sharing framework is that it is centered around the characterization of the resource requirements of the tasks initialized by mobile devices, i.e., "resource-centric", instead of emphasizing on the classification of these tasks, i.e., "task-centric". In other words, any of the tasks (e.g., content retrieving, data analysis, or uploading) is modeled by the resources that it requests, so that various types of tasks requesting some combinations of the 3C resources can coexist in the same framework.

Figure 1 illustrates a simple example of the proposed 3C framework, where four devices {A, B, C, D} connect with each other via D2D connections and share their 3C resources to complete tasks. In this example, device D initializes a task that involves the following procedures: (i) retrieving contents "1" and "2" (either downloading from the Internet or fetching from some devices' caches), (ii) performing computation, and (iii) outputting contents "3" (to the Internet) and "4" (to device D's local cache). With the 3C framework, devices can share their communication (downlink and uplink), computation (CPU), and caching resources. In this example, devices A and B are responsible for obtaining the inputting contents and delivering them to device C for computation, then device C performs the computation and sends the outputting contents to device D, and finally device D further uploads content "3" and caches content "4" in its local cache.

To show the benefits of the 3C framework concretely, we focus on the energy consumption of mobile edge devices, and solve an energy consumption minimization problem under the 3C framework. Note that the proposed methodology can also be applied to other system optimization objectives (e.g., delay minimization or QoS maximization). A common feature of these optimization problems under the 3C framework is the introduction of integer variables due to the proposed content-based framework (i.e, specifying the requested contents). The existence of the integer variables introduces difficulties in analyzing and solving the proposed problem. Moreover, tasks are often correlated with each other (e.g., due to the delays generated by the resource sharing), which further complicates the problem solving. We solve the energy minimization problem systematically and discuss the energy reduction due to the 3C sharing both analytically and numerically. Our key contributions are summarized as follows:

- *General 3C Resource Sharing Framework:* We propose a general 3C sharing framework and the corresponding "resource-centric" mathematical formulation. This framework generalizes many existing 1C/2C resource sharing models, and improves the resource utilization efficiency by encouraging more devices participating and more flexible resource scheduling.
- *Energy Efficiency Optimization:* We focus on the energy consumption of mobile edge devices under the 3C framework, and formulate and solve an energy consumption minimization problem. The problem is difficult as it is an integer non-convex optimization problem. We first transform it to an integer linear programming (ILP) problem, and then proposed a linear programming (LP) heuristic algorithm, whose output is empirically close to the optimal solution.
- *Theoretical Performance Analysis:* We analyze the energy consumption reduction due to the 3C resource sharing analytically. We show that if the 3C framework can double the number of cooperative devices (comparing with 1C models), it can reduce the energy by a maximum of about 20% of the energy consumed in noncooperation case (where devices do not cooperate).
- *Simulation and Performance Evaluation:* Comparing with existing 1C/2C sharing approaches, 3C sharing reduces the total energy by 83.8% when the D2D energy consumption is negligible, and the energy reduction is still 27.5% when the D2D energy per unit time is twice as large as the cellular energy per unit time.

The rest of this paper is organized as follows. Section II reviews the related work, and Section III presents the 3C framework. In Section IV, we present the energy minimization problem transformation and heuristic algorithm design. We analyze the energy reduction due to the 3C framework theoretically and numerically in Section V and Section VI, respectively. We conclude in Section VII.

II. Literature Review

There are extensive studies working on the 1C/2C sharing models. Due to the limited space, we only briefly discuss some representative works that are most closely related to this study.

Most of the existing works considered 1C models. For example, Iosifidis *et al.* [4] and Syrivelis *et al.* [5] proposed user-provided network models for communication resource sharing, where nearby devices share their Internet connectivity for cooperative downloading. Militano *et al.* [12] proposed an uploading resource sharing model, where devices form effective coalitions to share their uploading resources. Chen *et al.* [6] and Chi *et al.* [7] proposed ad hoc computation offloading models for computation resource sharing, where nearby mobile devices share their computation resources for data processing. Jiang *et al.* [8] and Chen *et al.* [9] proposed ad hoc content sharing models for cached content sharing, where devices share their cached contents through D2D connections.

Some recent works further considered 2C models. For example, Stojmenovic and Wen [10] and Destounis, *et al.* [11] considered distributed data analysis models, where some mobile devices share their cached data and other devices share their computation resources to process the shared data.

There are two limitations of the existing 1C/2C models: i) due to commonly adopted "task-centric" approach, devices with different types of tasks cannot cooperate (e.g., devices in user-provided network cannot cooperate with devices in ad hoc computation offloading), which restricts the pool of cooperative devices; and (ii) some tasks may request all of the 3C resources, which cannot be handled by these existing 1C/2C models. In comparison, our proposed 3C framework addresses the above two limitations and improve resource utilization efficiency by providing more device cooperation and resource scheduling flexibilities.

III. A General 3C Sharing Framework

A. System Model

We consider three key elements in the 3C framework: devices, tasks, and contents.

- **Device set** $\mathcal{N} = \{1, 2, ..., N\}$: The devices form a mesh network (through D2D connections) for cooperative task execution. For any device $n \in \mathcal{N}$, let $\mathcal{E}(n)$ denote the set of devices connected with device n through D2D connections. Note that $n \in \mathcal{E}(n)$.
- **Task set** $\mathcal{S} = \{1, 2, ..., S\}$: The devices initialize these tasks, where a device may initialize one or more tasks.
- **Content set** $\mathcal{K} = \{1, 2, ..., K\}$: The contents can be the inputting or outputting of the tasks. A content $k \in \mathcal{K}$ has a size of L_k (in bits).

Next we provide detailed explanations of devices and tasks.

1) Device Model: A device is characterized as follows, where the corresponding parameters are constants in a particular resource scheduling operation.[1]

Definition 1 (Device Model): Each device $n \in \mathcal{N}$ corresponds to a collection of tasks and resources, denoted by

$$\boldsymbol{Q}_n = (\boldsymbol{s}_n, Q_n^{down}, Q_n^{cpu}, Q_n^{up}, \boldsymbol{Q}_n^{ca}), \tag{1}$$

where each notation refers to a feature of the device:

- $\boldsymbol{s}_n$ - *the vector of its initializing tasks' indexes, where the dimension is the number of tasks it initializes,*
- Q_n^{down} - *downloading capacity (in bits per second),*
- Q_n^{cpu} - *computation capacity (in CPU cycles per second),*
- Q_n^{up} - *uploading capacity (in bits per second),*
- $\boldsymbol{Q}_n^{ca}$ - *the vector of cached contents of dimension K, where for any content $k \in \mathcal{K}$, $Q_{nk}^{ca} = 1$ if n has cached content k, and $Q_{nk}^{ca} = 0$ otherwise.*

Let c_n^{down}, c_n^{cpu}, and c_n^{up} denote the energy consumption of the downloading, computation, and uploading operations per unit second, respectively.

Next we define the model of the D2D connections.

[1]If the actual time for task execution is long, such constant capacities can be regarded as the average capacities over a relative long time period; if the actual time for task execution is short, such constant capacities can be regarded as the real-time capacities, which can be approximated as constants.

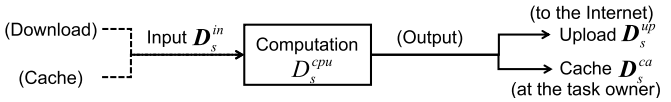

Fig. 2. The task model.

Definition 2 (Device-to-Device Model): For any two different devices $n, m \in \mathcal{N}$, let $Q_{n\to m}^{d2d}$ denote the D2D transmission capacity (bits per second) from device n to device m, and let $c_{n\to m}^{d2d}$ denote the D2D transmission energy per second.

2) Task Model: Each task $s \in \mathcal{S}$ is represented by the task model shown in Figure 2. Specifically, each task has a *computation* module (which can have a zero computation requirement if the task does not involve any computation). The computation module requests some inputting contents, which can be downloaded from the Internet or fetched from devices' caches. The computation module then produces some outputting contents, which can be uploaded to the Internet or cached at the task owner's device.

Definition 3 (Task Model): Each task $s \in \mathcal{S}$ is denoted by

$$\boldsymbol{D}_s = (u_s, \boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{D}_s^{up}, \boldsymbol{D}_s^{ca}), \tag{2}$$

where each notation refers to a feature of the task:

- u_s - *task owner (i.e., the device initializes this task),*
- $\boldsymbol{D}_s^{in}$ - *the vector of the inputting contents,*
- D_s^{cpu} - *computation requirement (in total CPU cycles),*
- $\boldsymbol{D}_s^{up}$ - *the vector of the uploading contents,*
- $\boldsymbol{D}_s^{ca}$ - *the vector of the caching contents.*

All three $\boldsymbol{D}_s^{in}$, $\boldsymbol{D}_s^{up}$, and $\boldsymbol{D}_s^{ca}$ have the same size of K. For any $k \in \mathcal{K}$, $D_{sk}^X = 1$ ($X \in in, up, ca$) if content k is requested by task s for inputting, uploading, or caching, respectively, and $D_{sk}^X = 0$ otherwise.

Each task can be divided into several subtasks:

Definition 4 (Subtask): Each task consists of three subtasks: inputting, computation, and uploading[2] subtasks.

Our proposed task model is content-based (i.e., specifying the requested contents) instead of data-based (i.e., only specifying the amount data needed). The content-based formulation makes the framework more flexible, as it does not limit where to obtain the requested contents. As a result, our task model can be used for modeling various applications whose tasks can be broken down to input contents, computation, and output contents (or a subset of these three types of subtasks). This is achieved by properly specifying the parameters in the task model, where the details are in Section III-C.

B. Problem Statement

To demonstrate the proposed 3C framework, we focus on an energy minimization problem, as mobile devices always concern about their energy consumption. Our mathematical formulation is also applicable to other optimization objectives.

We first introduce decision variables, constraints, and energy calculations. We then show the energy minimization problem.

[2]The contents to be cached, i.e., $\boldsymbol{D}^{ca}$, are cached at the task owner, so we do not regard it as a separate subtask.

1) Decision Variables: We consider a set of *binary* decision variables with possible values from $\{0, 1\}$ as follows. A variable equals 1 if its corresponding description is true, and equals 0 otherwise.

$x^{in}_{s,k\to n}$ - device n inputs task s' content k,
$x^{down}_{s,k\to n}$ - device n downloads task s' content k,
$x^{cpu}_{s\to n}$ - device n performs task s' computation,
$x^{up}_{s,k\to n}$ - device n uploads task s' content k,
$z^{in}_{s,k\to i,j}$- task s' inputting content k is delivered from i to j,
$z^{up}_{s,k\to i,j}$- task s' uploading content k is delivered from i to j,
$z^{ca}_{s,k\to i,j}$- task s' caching content k is delivered from i to j.

Here i and j refer to devices from set $\mathcal{N}$.

2) Constraints: The system constraints are as follows.

a) Allocation constraints: Any task s' computation subtask should be allocated to only one device:

$$\sum_{n\in\mathcal{N}} x^{cpu}_{s\to n} = 1, \quad \forall s \in \mathcal{S}. \tag{3}$$

The inputting of a requested content should be allocated to one device, i.e.,

$$\sum_{n\in\mathcal{N}} x^{in}_{s,k\to n} = D^{in}_{sk}, \quad \forall s \in \mathcal{S},\ k \in \mathcal{K}. \tag{4}$$

The uploading of a requested content should be allocated to one device, i.e.,

$$\sum_{n\in\mathcal{N}} x^{up}_{s,k\to n} = D^{up}_{sk}, \quad \forall s \in \mathcal{S},\ k \in \mathcal{K}. \tag{5}$$

b) Capacity constraints: The device that is responsible for inputting a content should either has the content in its local cache or will download it from the Internet:

$$x^{in}_{s,k\to n} \le Q^{ca}_{nk} + \sum_{s\in\mathcal{S}} x^{down}_{s,k\to n}, \quad \forall s \in \mathcal{S},\ n \in \mathcal{N},\ k \in \mathcal{K}. \tag{6}$$

c) Network flow balancing constraints: These constraints are related to the content delivery through multihop transmissions. For a content requested by a task, at any device, the incoming number of times that the device receives/generates the content should be the same as the number of times that the device transmits/consumes the content.

Taking the inputting content transmission variable $z^{in}_{s,k\to i,j}$ as an example. For any task $s \in \mathcal{S}$ and content $k \in \mathcal{K}$, the network flow balancing constraint at a device $i \in \mathcal{N}$ is

$$\sum_{j\in\mathcal{E}(i)} z^{in}_{s,k\to j,i} + x^{in}_{s,k\to i} D^{in}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{in}_{s,k\to i,j} + x^{cpu}_{s\to i} D^{in}_{sk}. \tag{7}$$

The left-hand side of (7) is the incoming number of times of task s' inputting content k, including the number of times that device i receives from its nearby devices and the number of times it generates for inputting (which equals one if $x^{in}_{s,k\to i} = 1$ and $D^{in}_{sk} = 1$). The right-hand side of (7) is the outgoing number of times of task s' inputting content k, including the number of times that device i transmits to its nearby devices and the number of times it consumes to perform computation (which equals one if $x^{cpu}_{s\to i} = 1$ and $D^{in}_{sk} = 1$).

Applying similar arguments to caching and uploading, we obtain the following constraints:

$$\sum_{j\in\mathcal{E}(i)} z^{ca}_{s,k\to j,i} + x^{cpu}_{s\to i} D^{ca}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{ca}_{s,k\to i,j} + \mathbf{1}_{i=u_s} D^{ca}_{sk}, \tag{8}$$

$$\sum_{j\in\mathcal{E}(i)} z^{up}_{s,k\to j,i} + x^{cpu}_{s\to i} D^{up}_{sk} = \sum_{j\in\mathcal{E}(i)} z^{up}_{s,k\to i,j} + x^{up}_{s,k\to i} D^{up}_{sk}. \tag{9}$$

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

d) Worst case delay constraints: The worst case delay is the maximum delay that may happen due to the resource sharing. We first explain the worst case delay constraints, and then compute the worst delays.

First, to execute a task s, the worst delay of downloading,[3] computation, and uploading subtasks, denoted by T^X_s ($X \in \{down, cpu, up\}$), should be smaller than the corresponding delay bounds, respectively:

$$T^{down}_s \le \bar{T}^{down}_s, T^{cpu}_s \le \bar{T}^{cpu}_s, T^{up}_s \le \bar{T}^{up}_s, \quad \forall s \in \mathcal{S}. \tag{10}$$

By using these separate delay constraints, we want to characterize the delay of each of the subtasks of a task. In addition, we ignore the D2D transmission delay for simplification.[4]

Then, we show how to calculate the worst case delays (T^{down}_s, T^{cpu}_s, and T^{up}_s). The first step is to calculate a device's time spending on completing all the subtasks allocated to it. Specifically, let τ^{down}_n, τ^{cpu}_n, and τ^{up}_n denote device n's total time spending on completing all the downloading, computation, and uploading subtasks allocated to it, respectively:

$$\tau^{down}_n = \frac{\sum_{s=1}^{S}\sum_{k=1}^{K} x^{down}_{s,k\to n} L_k}{Q^{down}_n}, \tag{11}$$

$$\tau^{up}_n = \frac{\sum_{s=1}^{S}\sum_{k=1}^{K} x^{up}_{s,k\to n} L_k}{Q^{up}_n}, \quad \tau^{cpu}_n = \frac{\sum_{s=1}^{S} x^{cpu}_{s\to n} D^{cpu}_s}{Q^{cpu}_n}. \tag{12}$$

The second step is to calculate the worst case delays. Using (11) as an example, we explain how to compute the worst downloading delay T^{down}_s. We will first discuss the maximum downloading time that a device n can impose on a task that it downloads for, and then discuss how a task s (requesting downloading from multiple devices) computes its worst downloading delay T^{down}_s.

For any device n, it may download contents for multiple tasks, and the multiple tasks may be ready for downloading at different times.[5] For simplicity, we assume that, if a device is

[3]Inputting contents may come from downloading from the Internet or fetching from caches. For simplicity, we assume that fetching from caches is instantaneous, so we can focus on the delay caused by downloading.

[4]In reality, such a delay is usually relatively small (e.g., Wi-Fi Direct has a transmission speed of up to 250Mbps [13]).

[5]The case of different ready times is more obvious for computation and uploading subtasks. Specifically, the computation of a task is ready for execution only when the corresponding downloading has finished, and this time could be different for different tasks. Similar for uploading subtasks.

downloading for multiple tasks at the same time, the device divides its total downloading capacity among the multiple tasks according to their total downloading volumes.[6] For example, a device n is downloading two contents (with sizes 1Mb and 2Mb, respectively) for task A, and one content (with a size 6Mb) for task B. Then, the device n allocates $(1+2)/(1+2+6) = 1/3$ and $6/(1+2+6) = 2/3$ of its downloading capacity to task A and task B, respectively. Under this, the maximum downloading time that device n can impose on a task is its total time spending on downloading, i.e., τ_n^{down}. This happens when all the tasks (allocated to device n) is ready for downloading at the same time, under which these tasks will share the device n's capacity during the entire downloading process.

For any task s, it can obtain multiple contents from different devices. The worst downloading delay that task s experiences is the maximum downloading time τ_n^{down} among the device set $\{n|\sum_{k=1}^{K} x_{s,k\to n}^{in}(1-Q_{nk}^{ca})>0\}$. This set refers to the set of devices that are responsible for task s' inputting but have not cached the contents yet. Formally,

$$T_s^{down} = \max_{\{n|\sum_{k=1}^{K} x_{s,k\to n}^{in}(1-Q_{nk}^{ca})>0\}} \tau_n^{down}. \tag{13}$$

A similar idea applies to the calculation of the worst computation and uploading delays. Specifically, the worst computation delay is the computation time of the device who performs task s' computation:

$$T_s^{cpu} = \sum_{n=1}^{N} x_{s\to n}^{cpu} \tau_n^{cpu}. \tag{14}$$

The worst uploading delay is the maximum uploading time τ_n^{up} among all the devices who perform task s' uploading:

$$T_s^{up} = \max_{\{n|\sum_{k=1}^{K} x_{s,k\to n}^{up}>0\}} \tau_n^{up}. \tag{15}$$

To clarify, these delay constraints are non-convex, since they contain quadratic forms that are not positive semidefinite, which makes it difficult to solve the energy minimization problem (to be presented in Section III-B.4).

3) Energy Calculations: The energy for executing a task s consists of the energy for downloading, computation, uploading, and D2D transmission. Formally,

$$E_s = E_s^{down} + E_s^{cpu} + E_s^{up} + E_s^{d2d}. \tag{16}$$

Each of these four terms is linear with the time that the devices spend on executing the corresponding operations:

$$E_s^{down} = \sum_{n=1}^{N} c_n^{down} \frac{\sum_{k=1}^{K} x_{s,k\to n}^{down} L_k}{Q_n^{down}}, \tag{17}$$

$$E_s^{cpu} = \sum_{n=1}^{N} c_n^{cpu} \frac{x_{s\to n}^{cpu} D_s^{cpu}}{Q_n^{cpu}},$$

$$E_s^{up} = \sum_{n=1}^{N} c_n^{up} \frac{\sum_{k=1}^{K} x_{s,k\to n}^{up} L_k}{Q_n^{up}}, \tag{18}$$

[6]We assume that when a subtask arrives at a device, it is executed without queuing, and it shares the capacity of the device with the other subtasks.

TABLE I
EXISTING MODELS THAT THE 3C FRAMEWORK CAN GENERALIZE

Shared Resource	Examples and Task Models $\boldsymbol{D}_s$
Downloading	(a) User-provided networks (e.g., [4], [5]) $(u_s, \boldsymbol{D}_s^{data}, 0, \boldsymbol{0}, \boldsymbol{D}_s^{data})$
Uploading	(b) Ad hoc content uploading (e.g., [12]) $(u_s, \boldsymbol{D}_s^{data}, 0, \boldsymbol{D}_s^{data}, \boldsymbol{0})$
Content	(c) Ad hoc content sharing (e.g., [8], [9]) $(u_s, \boldsymbol{D}_s^{data}, 0, \boldsymbol{0}, \boldsymbol{D}_s^{data})$
Computation	(d) Ad hoc computation offloading (e.g., [6], [7]) $(u_s, \boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{0}, \boldsymbol{D}_s^{out})$
Hybrid	(e) Distributed data analysis (e.g., [10], [11]) $(u_s, \boldsymbol{D}_s^{in}, D_s^{cpu}, \boldsymbol{0}, \boldsymbol{D}_s^{out})$

$$E_s^{d2d} = \sum_{i=1}^{N}\sum_{j=1}^{N} c_{i\to j}^{d2d} \times \frac{\sum_{k=1}^{K} (z_{s,k\to i,j}^{in} + z_{s,k\to i,j}^{up} + z_{s,k\to i,j}^{ca}) L_k}{Q_{i\to j}^{d2d}}. \tag{19}$$

For example, task s' downloading energy E_s^{down} is the sum of the energy consumed by various devices for downloading for task s, where each device's downloading energy equals to the product of its energy coefficient and the downloading time.

4) Problem Formulation: We minimize the energy consumption of the 3C framework under the proposed constraints:

$$\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{z}\in\{0,1\}}{\text{minimize}} \sum_{s\in\mathcal{S}} E_s \\ &\text{subject to (3)} \sim \text{(10)} \end{aligned} \tag{OPT}$$

Problem (OPT) is non-convex due to the delay constraints. In Section IV, we transform it into an ILP problem, which can be solved by standard optimizers. We further propose a heuristic algorithm with a lower computational complexity.

C. Generalization of Existing Models in the Literature

By properly specifying the task model $\boldsymbol{D}_s$, the proposed 3C framework can generalize many of the existing 1C and 2C models. Examples are illustrated in Table I. Among these models, the notation $\boldsymbol{D}_s^{data}$ (in (a), (b), and (c)) denotes the contents that are requested by the corresponding operations.

IV. ENERGY MINIMIZATION WITH 3C SHARING

In this section, we focus on the energy minimization problem (OPT), which is an integer non-convex optimization problem. To solve this problem, we first transform it into an ILP problem, which can be solved by standard optimizers. However, an ILP problem is an NP-complete problem [14]. Hence, we further propose a heuristic algorithm that solves a series of problems, each of which is a LP relaxation (relaxing integer variables to continuous ones) of the original problem (OPT). To clarify, the solutions in this section are based on a centralized scheduling, i.e., a centralized entity is needed for information collection and signaling.

A. Linear Transformation of Problem (OPT)

We first transform the integer non-convex problem (OPT) to an ILP problem. Since the non-convexity is mainly due to the delay constraints, the key focus will be how to transform the delay constraints to linear ones.

The key transforming idea is as follows. Consider a constraint in the form of $\tau \times y \leq \bar{T}$, where the continuous variable $\tau \geq 0$, the discrete variable $y \in \{0,1\}$, and the fixed parameter $\bar{T} \geq 0$. We can transform the constraint into an equivalent linear form $\tau - (1-y) \times M \leq \bar{T}$, where M is any number that satisfies $\tau - M \leq 0$. To see the equivalence of the two constraints, we consider two possible values of y. If $y = 1$, both the original and the transformed constraints are $\tau \leq \bar{T}$; if $y = 0$, both constraints are true for any value of τ. Hence, the two constraints are equivalent.

Next we use this key idea to explain the transformation of downloading delay constraints. The transformations of computation and uploading delay constraints are similar. Finally, we present the transformed problem.

1) Transforming Downloading Delay Constraints: We will transform the delay constraint $T_s^{down} \leq \bar{T}_s^{down}$ in (10) to a linear one. Specifically, this constraint can be written as

$$\tau_n^{down} y_{s\to n}^{down} \leq \bar{T}_s^{down}, \quad \forall s \in \mathcal{S},\ n \in \mathcal{N}, \tag{20}$$

where τ_n^{down} is a linear function of variables $x_{s,k\to n}^{down}$ as in (11). The variable $y_{s\to n}^{down} \in \{0,1\}$ indicates whether a task s' downloading is being allocated to device n, i.e.,

$$y_{s\to n}^{down} \leq \min\{\sum_{k=1}^{K} x_{s,k\to n}^{in}(1 - Q_{nk}^{ca}), 1\}, \tag{21}$$

$$y_{s\to n}^{down} \geq x_{s,k\to n}^{in}(1 - Q_{nk}^{ca}), \quad \forall k \in \mathcal{K}. \tag{22}$$

Specifically, when $x_{s,k\to n}^{in}(1 - Q_{nk}^{ca}) = 0$ for all k (i.e., device n does not download any content for task s), we have $y_{s\to n}^{down} = 0$; otherwise, when there exists a k such that $x_{s,k\to n}^{in}(1 - Q_{nk}^{ca}) = 1$ (i.e., device n downloads contents for task s), we have $y_{s\to n}^{down} = 1$.

Then, we transform constraints (20) based on the previously mentioned transforming idea. The constraint is given by

$$\tau_n^{down} - (1 - y_{s\to n}^{down}) M_n^{down} \leq \bar{T}_s^{down}, \quad \forall s \in \mathcal{S},\ n \in \mathcal{N}, \tag{23}$$

where M_n^{down} satisfies $\tau_n^{down} - M_n^{down} \leq 0$.[7]

2) Transforming Computation and Uploading Delay Constraints: Based on similar ideas, the equivalent computation delay constraint is given by

$$\tau_n^{cpu} - (1 - x_{s\to n}^{cpu}) M_n^{cpu} \leq \bar{T}_s^{cpu}, \quad \forall s \in \mathcal{S},\ n \in \mathcal{N}, \tag{24}$$

where M_n^{cpu} satisfies $\tau_n^{cpu} - M_n^{cpu} \leq 0$. The equivalent uploading delay constraint is given by

$$\tau_n^{up} - (1 - y_{s\to n}^{up}) M_n^{up} \leq \bar{T}_s^{up}, \quad \forall s \in \mathcal{S},\ n \in \mathcal{N}, \tag{25}$$

[7]Note that τ_n^{down} is a linear function of variables $x_{s,k\to n}^{down}$, which are unknown before solving the optimization problem. To ensure the inequality $\tau_n^{down} - M_n^{down} \leq 0$ holds, we can set $M_n^{down} = \max\{\bar{T}_s^{down}, \forall s\}$, $\forall n \in \mathcal{N}$. Similar ideas apply for the computation and uploading constraints.

where M_n^{up} satisfies $\tau_n^{up} - M_n^{up} \leq 0$, and $y_{s\to n}^{cpu}$ denotes whether device n uploads for task s, i.e.,

$$y_{s\to n}^{up} \leq \min\left\{\sum_{k=1}^{K} x_{s,k\to n}^{up}, 1\right\},$$
$$y_{s\to n}^{up} \geq x_{s,k\to n}^{up}, \quad \forall k \in \mathcal{K}. \tag{26}$$

3) The Linear Transformation of Problem (OPT): Once replacing the delay constraint (10) with (21)∼(26), we transform Problem (OPT) into the following equivalent problem:

$$\underset{\boldsymbol{x},\boldsymbol{z},\boldsymbol{y}\in\{0,1\}}{\text{minimize}} \sum_{s\in\mathcal{S}} E_s$$
$$\text{subject to } (3) \sim (9), (21) \sim (26) \quad \text{(OPT-LINEAR)}$$

Problem (OPT-LINEAR) is an ILP, which can be solved by standard optimizers, e.g., Gurobi (http://www.gurobi.com).

However, directly solving Problem (OPT-LINEAR) using standard optimizers works well when the network size (e.g., the number of devices and tasks) is reasonably small. As the system size increases, the computation time dramatically increases, as Problem (OPT-LINEAR) (an ILP) is NP-complete [14]. To address this complexity issue, we propose a heuristic algorithm based on the original Problem (OPT).

B. A Heuristic Algorithm of Solving Problem (OPT)

The key idea is to iteratively solve a series of modified versions of Problem (OPT), where we remove the delay constraints and relax the integer variables to continuous ones (i.e., LP relaxation [14], so that the modified problems are LP problems). At the end of each iteration, the algorithm will check whether the removed delay constraints are satisfied. If not, the algorithm will prevent some tasks from being allocated to certain devices (in order to address the violated delay constraints), and solve a new version of modified problem. The algorithm iterates until all the delay constraints are satisfied.

Next we first describe the modified problem, then we propose the heuristic algorithm.

1) A Modified Problem of Problem (OPT): Comparing with the original Problem (OPT), the modification involves removing delay constraints, relaxing integer variables, and adding control parameters that can prevent certain tasks from being allocated to certain devices. We first introduce the control parameters, then propose the modified problem.

In order to prevent particular subtasks from being allocated to certain devices, we introduce the following binary control parameters $\tilde{N}_{s,k\to n}^{in}$, $\tilde{N}_{s\to n}^{cpu}$, and $\tilde{N}_{s,k\to n}^{up}$:

$$x_{s,k\to n}^{in} \leq \tilde{N}_{s,k\to n}^{in},\ x_{s\to n}^{cpu} \leq \tilde{N}_{s\to n}^{cpu},\ x_{s,k\to n}^{up} \leq \tilde{N}_{s,k\to n}^{up}. \tag{27}$$

Take $\tilde{N}_{s,k\to n}^{in}$ as an example: if it equals zero, then (27) indicates that $x_{s,k\to n}^{in}$ can only be zero, so the content k of task s cannot be allocated to device n; if it equals one, then $x_{s,k\to n}^{in}$ can be either zero or one, so the allocation is not prevented. The same idea applies to $\tilde{N}_{s\to n}^{cpu}$ and $\tilde{N}_{s,k\to n}^{up}$.

Then, we introduce the modified problem of Problem (OPT) by removing delay constraints (10), relaxing integer variables

Algorithm 1 Heuristic Algorithm

1: **Initialization:** $\tilde{\boldsymbol{N}}^{in}, \tilde{\boldsymbol{N}}^{cpu}, \tilde{\boldsymbol{N}}^{up} \leftarrow 1$
2: $\boldsymbol{x}, \boldsymbol{z} \leftarrow$Solve Problem (OPT-RELAX) (e.g., using Simplex method [15])
3: Calculate $T_s^{down}, T_s^{cpu}, T_s^{up}$ using (13), (14), (15)
4: **while** delay constraints (10) are not fully satisfied **do**
5: **for** each task $s \in \mathcal{S}$ **do**
6: **if** $T_s^{down} > \bar{T}_s^{down}$ **then**
7: $\hat{n} \leftarrow \max_n\{\tau_n^{down} y_{s\to n}^{down}\}$ ▷ select a device
8: $\boldsymbol{I} \leftarrow \{s | \sum_{\hat{s}\in \boldsymbol{s}_{\hat{n}}} \sum_k x_{\hat{s},k\to\hat{n}}^{in} x_{s,k\to\hat{n}}^{in} \geq 1\}$ ▷ find device $\hat{n}$'s *non-preventable* task set
9: $\bar{s} \leftarrow \arg\min_s\{\bar{T}_s^{down} | y_{s\to\hat{n}}^{down} \neq 0, s \notin \boldsymbol{I}\}$ ▷ select a task $\bar{s}$ that is preventable
10: $\bar{\mathcal{K}} \leftarrow \{k | x_{\bar{s},k\to\hat{n}}^{in}(1 - \boldsymbol{Q}_{\hat{n}k}^{ca}) = 1\}$ ▷ select contents
11: $\tilde{N}_{s,k\to\hat{n}}^{in} \leftarrow 0, \forall\{s,k | x_{s,k\to\hat{n}}^{in} = 1, k \in \bar{\mathcal{K}}\}$ ▷ prevent the allocations
12: **end if**
13: **if** $T_s^{cpu} > \bar{T}_s^{cpu}$ **then**
14: $\hat{n} \leftarrow \max_n\{\tau_n^{cpu} y_{s\to n}^{cpu}\}$ ▷ select a device
15: $\bar{s} \leftarrow \arg\min_s\{\bar{T}_s^{cpu} | y_{s\to\hat{n}}^{cpu} \neq 0, s \notin \boldsymbol{s}_{\hat{n}}\}$ ▷ select a task
16: $\tilde{N}_{\bar{s}\to\hat{n}}^{cpu} \leftarrow 0$ ▷ prevent the allocation
17: **end if**
18: **if** $T_s^{up} > \bar{T}_s^{up}$ **then**
19: $\hat{n} \leftarrow \max_n\{\tau_n^{up} y_{s\to n}^{up}\}$ ▷ select a device
20: $\bar{s} \leftarrow \arg\min_s\{\bar{T}_s^{up} | y_{s\to\hat{n}}^{up} \neq 0, s \notin \boldsymbol{s}_{\hat{n}}\}$ ▷ select a task
21: $\tilde{N}_{\bar{s},k\to\hat{n}}^{up} \leftarrow 0, \forall k$ ▷ prevent the allocations
22: **end if**
23: **end for**
24: $\boldsymbol{x}, \boldsymbol{z} \leftarrow$Solve Problem (OPT-RELAX) (e.g., using Simplex method [15])
25: Calculate $T_s^{down}, T_s^{cpu}, T_s^{up}$ using (13), (14), (15)
26: **end while**
27: **return** $\boldsymbol{x}, \boldsymbol{z}$

(i.e., relaxing $\boldsymbol{x}, \boldsymbol{z} \in \{0,1\}$ to be $\boldsymbol{x}, \boldsymbol{z} \in [0,1]$), and adding control constraints (27):

$$\underset{\boldsymbol{x},\boldsymbol{z}\in[0,1]}{\text{minimize}} \sum_{s\in\mathcal{S}} E_s$$
$$\text{subject to (3)} \sim \text{(9), (27)} \quad \text{(OPT-RELAX)}$$

To clarify, there are many versions of Problem (OPT-RELAX), each of which corresponds to a set of parameter choices of $\tilde{N}_{s,k\to n}^{in}$, $\tilde{N}_{s\to n}^{cpu}$, and $\tilde{N}_{s,k\to n}^{up}$. Moreover, Problem (OPT-RELAX) is an LP problem, which can be solved using various methods such as Simplex method [15].

2) A Heuristic Algorithm to Solve Problem (OPT): The heuristic algorithm will iteratively solve multiple versions of Problem (OPT-RELAX) as follows. At the beginning, no allocation is prevented, i.e., $\tilde{N}_{s,k\to n}^{in} = \tilde{N}_{s\to n}^{cpu} = \tilde{N}_{s,k\to n}^{up} = 1$ for all s, k, and n. We first optimize the corresponding Problem (OPT-RELAX), and check whether the optimal solution satisfies the delay constraints in (10). If yes, then the obtained optimal solution of Problem (OPT-RELAX) is the optimal solution of Problem (OPT); if not, we need to revise the control parameters in Problem (OPT-RELAX) (i.e., setting some $\tilde{N}_{s,k\to n}^{in}$, $\tilde{N}_{s\to n}^{cpu}$, or $\tilde{N}_{s,k\to n}^{up}$ to be zeros), with details discussed in the next paragraph. We optimize Problem (OPT-RELAX) iteratively until obtaining a solution that satisfies the delay constraints in (10). The algorithm is given in Algorithm 1.

We now discuss how the algorithm chooses the proper version of Problem (OPT-RELAX) to solve by setting $\tilde{N}_{s,k\to n}^{in}$, $\tilde{N}_{s\to n}^{cpu}$, or $\tilde{N}_{s,k\to n}^{up}$ for inputting (in lines 6-12 of Algorithm 1),[8] computation (in lines 13-17), and uploading (in lines 18-22) subtasks, respectively. We first introduce the general idea, then explain a special setting of the inputting subtask.

The general idea of preventing some allocations for the inputting, computation, and uploading subtasks is as follows. For a particular subtask of a task s, if its corresponding delay (after solving a version of Problem (OPT-RELAX)) is larger than the delay bound, then the algorithm will (i) find the device $\hat{n}$ that induces the maximum delay, (ii) at device $\hat{n}$, find the task $\bar{s}$ with the tightest delay bound among all the tasks that are allocated to device $\hat{n}$ (excluding device $\hat{n}$'s tasks), (iii) prevent task $\bar{s}$ from being allocated to device $\hat{n}$ by setting the corresponding $\tilde{N}_{s,k\to n}^{in}$, $\tilde{N}_{s\to n}^{cpu}$, or $\tilde{N}_{s,k\to n}^{up}$ to be zeros.

Next we discuss a special setting of the inputting (downloading) subtask. Specifically, device $\hat{n}$ may download the same content for both itself and other devices, but it should not prevent the content downloading of its own tasks. So we define a *non-preventable* set $\boldsymbol{I}$ (in line 8), which contains the tasks that request a same content as device $\hat{n}$ does. Only the tasks outside the non-preventable set can be prevented (in line 9). In addition, only the contents that have not be cached (have to be downloaded) are prevented (in lines 10 and 11), because cached contents do not induce delays.

3) Properties of Algorithm 1: We first make an assumption that is often satisfied in practice, and then characterize several properties of the proposed heuristic algorithm.

Assumption 1 (Feasible Noncooperation Case): Noncooperation (i.e., each of the devices performs its tasks on its own) is within the feasible region of Problem (OPT).

This assumption implies that each device is capable of executing its tasks on its own. If Assumption 1 is violated, some tasks may become infeasible to complete, as cooperation is not always guaranteed in practice.

Under Assumption 1, Algorithm 1 is guaranteed to converge and output an integer feasible solution of Problem (OPT).

Proposition 1 (Guarantee of Feasible Output): Algorithm 1 is guaranteed to converge and produce an integer solution that is within the feasible region of Problem (OPT).

The proof is given in the online report [16]. Specifically, to prove the convergence, we have to show that the noncooperation solution will never be excluded from the feasible region of Problem (OPT-RELAX), so the algorithm will definitely

[8]The inputting subtask prevention corresponds to downloading delays, because the downloading is the operation inducing inputting delays.

converge when reaching the noncooperation solution (if it has not converged earlier). To prove the integer feasible solution, we have to prove that the optimal solution of any version of Problem (OPT-RELAX) is always integer (as its matrix of constraint coefficients is totally unimodular [17]) and within the feasible region of Problem (OPT).

We now show the performance guarantee of Algorithm 1.

Proposition 2 (Performance Guarantee): The energy consumption of the heuristic algorithm output is no larger than that of the noncooperation case. When there is no delay constraint, the heuristic algorithm output is an optimal solution of the original problem (OPT).

The proof is given in the online report [16]. We will further evaluate the performance of this heuristic algorithm under the settings with delay constraints in Section VI-A.

Regarding the complexity of Algorithm 1, its maximum iteration time is as follows:

Proposition 3 (Maximum Iteration Time): The maximum iteration time of this heuristic algorithm is $S \times (N-1)$, where S is the task number and N is device number.

The proof is given in the online report [16]. Recall that Problem (OPT-LINEAR) is NP-complete, which cannot be solved in polynomial time in general. In comparison, Proposition 3 shows that the heuristic algorithm has a maximum of $S \times (N-1)$ iterations, each of which solves an LP problem (OPT-RELAX) that is a P-complete problem. This implies that the heuristic algorithm can terminate in polynomial time.

V. Energy Reduction Due to 3C Sharing

The proposed 3C framework is "resource-centric" instead of "task-centric", so that it provides additional flexibilities in terms of device cooperation. More specifically, it promotes cooperation opportunities through enabling devices performing different tasks to cooperate. In this section, we study how much a 3C framework can reduce the energy consumption through a specific problem setting, comparing with 1C models. We first introduce system settings, then discuss the energy reduction due to the 3C framework.

A. System Settings

In order to derive the closed-form solutions of the energy reduction, we consider specific device and task models as follows. We consider a random graph model $G(N,p)$ [18], where there are N devices in the graph and every two devices are connected randomly and independently with a probability p. Suppose that the network is large and sparse, so that N approaches infinite with Np being a constant [18]. These devices initialize a set of tasks. Since we focus on the comparison between 1C models and 3C framework, we assume that each task only needs one of the 3C resources.

The devices are heterogeneous in terms of their owned resources. Specifically, each device n owns some resources Q_n^{down}, Q_n^{cpu}, and Q_n^{up}. The capacities Q_n^X ($X \in \{down, cpu, up\}$) is independent and identically distributed (i.i.d.) with the cumulative distribution function $F_Q^X(x)$ and the probability density function $f_Q^X(x)$. The support of capacity Q_n^X is $(\underline{Q}^X, \overline{Q}^X)$, hence $F_Q^X(\underline{Q}^X) = 0$ and $F_Q^X(\overline{Q}^X) = 1$. For the convenience of analysis, we assume that the energy coefficients of the devices are homogeneous, i.e., $c_n^X = c^X, X \in \{down, cpu, up\}$, $\forall n$. In addition, each device n uniformly and randomly caches M^{ca} contents in its cache, i.e., $\sum_{k=1}^{K} Q_{nk}^{ca} = M^{ca}$ for all n. For simplification, we assume that all the contents have the same size that is normalized to one, i.e., $L_k = 1$, for each k, similar as in existing caching studies on performance analysis (e.g., [19]).

We aim to study the system under the general distribution function, which is quite challenging to do. Hence, we further make the following simplifying assumption for the rest of Section V. A more realistic case (with these assumptions relaxed) is evaluated empirically in Section VI.

Assumption 2: 1) The D2D transmission energy is relatively small and can be ignored; 2) there is no delay constraint; 3) devices can only cooperate with their one-hop neighbors.

Under Assumption 2, for any device m, the optimal allocation of any of its tasks $s \in \boldsymbol{s}_m$ is as follows. Regarding the inputting subtask, for a content requested by task s, if any device $n \in \mathcal{E}(m)$ has cached it, then the content will be inputted from device n. If none of the devices in set $\mathcal{E}(m)$ has cached it, then the device who has the highest downloading capacity among set $\mathcal{E}(m)$ downloads the content. Regarding the computation and uploading subtasks, they will be allocated to the devices with the highest computation and uploading capacities among the devices $\mathcal{E}(m)$, respectively.

B. Energy Reduction Due to the 3C Framework

In this subsection, we study how much a 3C framework can reduce the energy consumption through providing more cooperation opportunities. Since that each of the tasks only requests one kinds of the 3C resources, we can analyze the tasks requesting each of the 3C resources separately.

Next we will compute the energy reduction of the tasks requesting each of the 3C resources one by one. We will first discuss the tasks requesting communication/computation resource (both of which are capacity-based resources), and then discuss the tasks requesting caching resource.

1) Communication/Computation: In the following analysis, we focus on the tasks requesting a particular resource (i.e., downloading, uploading, or computation). Hence, for presentation simplicity, the term "resource" in Section V-B.1 only refers to the particular resource, and we omit the corresponding resource-specific super-scripts and sub-scripts. Without the loss of generality, we normalize the energy coefficient of the particular resource to be one, i.e., $c = 1$.

We will first formulate the expected energy consumption, then introduce the analysis idea. In the random graph $G(N,p)$, suppose each device joins the cooperative system with a probability $\alpha \in [0,1]$. Under these, each task requesting the particular resource will have an expected energy consumption denoted by $W(\alpha, Np)$.[9] Under the 1C model, let us denote the probability that each device joins the corresponding 1C model (that shares the particular resource) as $\alpha^{1C} \in [0,1]$; under

[9]Under the homogeneous distribution settings in Section V-A, all the tasks will have the same expected energy consumption, so we only need to study the expected energy consumption of a task.

the 3C framework, the corresponding probability is $\alpha^{3C} = \min\{r\alpha^{1C}, 1\}$, where $r \geq 1$ is a coefficient reflecting the ratio of the increased cooperation opportunities. We will compute the energy reduction $\Delta W(r, \alpha^{1C}, Np) \triangleq W(\alpha^{1C}, Np) - W(\alpha^{3C}, Np)$ for $r \geq 1$.

First, we calculate the expected energy, i.e., $W(\alpha, Np)$, under particular α and Np. As we have explained, under Assumption 2, any device's task will be allocated to its neighbor who has the highest capacity. By using the order statistic result [20], the probability density function of the highest capacity among a total of n devices is given by

$$f_{(n)}(x) = n(F(x))^{n-1} f(x). \tag{28}$$

In the random graph, for a device with a degree m, the probability that $\hat{N}$ of its neighbors join in the cooperative system is $P(\hat{N}|m) = C_m^{\hat{N}} \alpha^{\hat{N}} (1-\alpha)^{m-\hat{N}}$, and the corresponding distribution of the highest capacity among these $\hat{N}$ devices and itself is $f_{(\hat{N}+1)}(x)$. Taking the expectation over $\hat{N} = \{0, ..., m\}$, the expected energy consumption of this device's task is given by

$$\hat{W}_m(\alpha, Np) = \sum_{\hat{N}=0}^{m} P(\hat{N}|m) \int_{\underline{Q}}^{\overline{Q}} \frac{1}{x} f_{(\hat{N}+1)}(x) dx. \tag{29}$$

Taking the expectation of $\hat{W}_m(\alpha, Np)$ over all degrees $m = \{0, ..., \infty\}$ [18], the expected energy of a task is

$$W(\alpha, Np) = \frac{1}{\overline{Q}} + \int_{\underline{Q}}^{\overline{Q}} e^{Np(F(x)-1)\alpha} F(x) x^{-2} dx, \tag{30}$$

with the detailed proof given in [16].

Then, we discuss how much the 3C framework can reduce the energy consumption under a coefficient $r \geq 1$. We are interested in the best (the maximum energy reduction) that the 3C framework can achieve for any α^{1C} and p under an r, i.e., $\max_{\alpha^{1C}, p} \Delta W(r, \alpha^{1C}, Np)$. The maximum energy reduction that is caused by the 3C framework is as follows:

Theorem 1 (Maximum Energy Reduction of Communication/Computation): Under a coefficient $r \geq 1$, the maximum energy reduction due to the 3C framework is given by

$$\max_{\alpha^{1C}, p} \Delta W(r, \alpha^{1C}, p) = \int_{\underline{Q}}^{\overline{Q}} \left(e^{\frac{N\tilde{p}(F(x)-1)}{r}} - e^{N\tilde{p}(F(x)-1)} \right) F(x) x^{-2} dx, \tag{31}$$

where $\tilde{p}$ satisfies

$$\int_{\underline{Q}}^{\overline{Q}} (F(x)-1) F(x) \left(e^{\frac{N\tilde{p}(F(x)-1)}{r}} - r e^{N\tilde{p}(F(x)-1)} \right) dx = 0. \tag{32}$$

The proof is given in [16]. The key idea is to show that the non-concave energy reduction $\Delta W(r, \alpha^{1C}, Np)$ has a unique maximizer, which satisfies the first order condition.

Theorem 1 shows the maximum energy reduction under a general capacity distribution $F(x)$. In order to reveal practical insights, we show a concrete example.

Example 1: Let us consider the truncated normal distribution $F(x) = F(x; \mu, \sigma, a, b)$, which can be regarded as a normal distribution $N(\mu, \sigma^2)$ that lies within the interval $[a, b]$ (please refer to [16] and [21] for details). Figure 3 shows the maximum energy reduction (normalized by the energy consumption in the noncooperation case, $W(0,0)$). From Figure 3, we conclude as follows. (i) The energy reduction is higher when the variance σ is larger. Intuitively, when the devices and tasks are more heterogeneous, the framework benefits more from exploiting the devices' and tasks' heterogeneities. (ii) Under a large variance σ (e.g., $\sigma = 10$), doubling the sharing devices fraction (i.e., $r = 2$) leads to a maximum energy reduction of around 20% of the energy consumed in noncooperation.

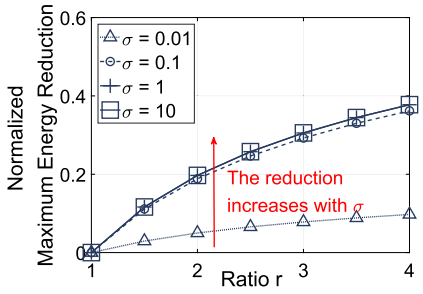

Fig. 3. Normalized maximum energy reduction ($\mu = 2$).

2) Caching: The analysis for caching is similar as that for the communication/computation, where the details and proofs are given in the online report [16]. The expected energy reduction of a content $Z(\alpha, Np)$ is as follows:

$$Z(\alpha, Np) = (1 - \frac{M^{ca}}{K}) e^{-\alpha Np \frac{M^{ca}}{K}}. \tag{33}$$

Under this, the energy reduction due to 3C framework is $\Delta Z(r, \alpha^{1C}, Np) \triangleq Z(\alpha^{1C}, Np) - Z(\alpha^{3C}, Np)$. Then, the normalized maximum energy reduction is given as follows.

Theorem 2 (Maximum Energy Reduction of Caching): Under a coefficient $r \geq 1$, the normalized maximum energy reduction (i.e., normalized by the energy consumption in the noncooperation case $Z(0,0)$) due to the 3C framework is given by

$$\frac{\max_{\alpha^{1C}, p} \Delta Z(r, \alpha^{1C}, Np)}{Z(0,0)} = e^{-\frac{\ln r}{(r-1)}} - e^{-\frac{r \ln r}{(r-1)}}. \tag{34}$$

Based on Theorem 2, we conclude as follows. (i) The normalized maximum energy reduction is independent of the caching ratio $M^{ca}/K > 0$. This means that no matter how many contents that devices have cached, the normalized maximum energy reduction is fixed. (ii) Doubling the sharing device fraction (i.e., $r = 2$) leads to a maximum energy reduction of around 25% of the energy consumed in noncooperation.

VI. Simulation and Performance

We compare the energy consumption between optimal and heuristic solutions. And we evaluate the energy reduction due to 3C framework under different D2D transmission energy and different devices' and tasks' heterogeneities. To emphasize, we relax Assumption 2 in this section.

We consider a scenario with a set of N devices, who form pair-wise connections with a probability $p = 0.3$. Each

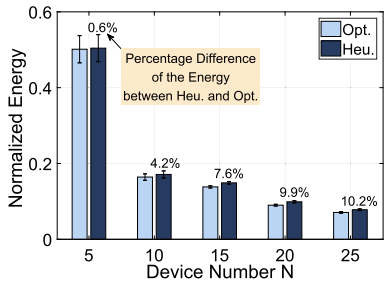

Fig. 4. Comparison between optimal and heuristic solutions.

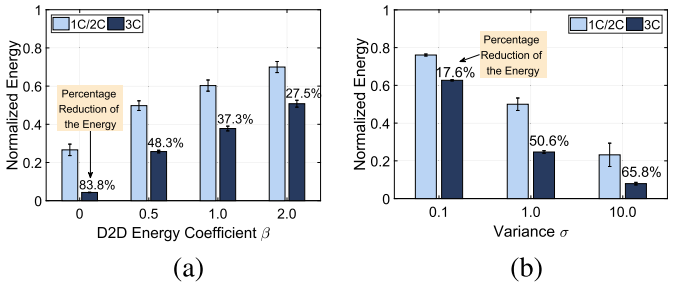

Fig. 5. Impact of D2D energy β (under $\sigma = 1.0$) and variance σ (under $\beta = 0.5$) on the normalized energy consumption.

device has one task to execute. For each simulation setting, we perform 100 rounds and show the average results. For each simulation round, we randomly generate the parameters of the device and task models, which are randomly generated based on truncated normal distributions [21], with an identical variance σ (which will be evaluated later) and different means. The detailed settings are shown in the online report [16].

A. Comparison: Optimal and Heuristic Solutions

We show how the device number N affects the energy consumption of the optimal (named as "Opt.") and the heuristic algorithm (named as "Heu."). Figure 4 shows the energy comparison between "Opt." and "Heu.". The energy is normalized by the energy consumed in the noncooperation case (i.e., each device executes its task by itself). When $N = 5$, the normalized percentage difference of the energy between "Heu." and "Opt." is only 0.6%. As N increases, the energy gap between "Opt." and "Heu." slightly increases.

B. Comparison: 1C/2C Models and 3C Framework

We let each device randomly selects a task among downloading, content sharing, and distributed data analysis. Then, we perform simulations in two cooperation settings: (i) "1C/2C", where only the devices selecting the same kinds of tasks cooperate; (ii) "3C", where all the devices cooperate.

In Figure 5, we compare the energy of the two cooperation settings under different D2D energy coefficients β (the ratio of the D2D energy per unit time to the downloading energy per unit time) and variances σ (the variance for generating tasks and devices). Note that the energy consumption is normalized by the energy consumed in the noncooperation case. The percentage reduction in the figure is the energy difference between "1C/2C" and "3C", normalized by the energy consumed in "1C/2C".

In Figure 5 (a), "3C" can reduce the energy consumption by 83.8% when $\beta = 0$, i.e., no additional energy consumption due to D2D transmission. Such an energy reduction decreases in β, but still achieves a value of 27.5% when $\beta = 2.0$, i.e., the D2D energy per unit time is twice as large as the downloading energy per unit time. In Figure 5 (b), as the heterogeneity of devices and tasks (measured by the variance σ) increases, the energy reduction caused by 3C framework increases. Intuitively, a higher heterogeneity can provide more opportunities for the devices to share resources and help each other. Hence, implementing the 3C framework is more beneficial when devices and tasks are more heterogeneous.

VII. Conclusion

In this paper, we propose a general 3C framework that enables the joint 3C resource sharing among mobile edge devices, which potentially enhances the reliability and intelligence of Tactile Internet. This "resource-centric" framework generalizes existing D2D resource sharing models, and provides a structure for future D2D resource sharing analysis. We theoretically and numerically show that the 3C framework can further exploit resource sharing potentials and improve resource utilization efficiency significantly. For future work, it is interesting to design a distributed algorithm for the 3C framework, where the information collection and allocation scheduling are operated in a distributed fashion.

References

[1] M. Tang, L. Gao, and J. Huang, "A general framework for crowdsourcing mobile communication, computation, and caching," in *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.

[2] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "5G-enabled tactile Internet," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 460–473, Mar. 2016.

[3] M. R. Palattella *et al.*, "Internet of Things in the 5G era: Enablers, architecture, and business models," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 510–527, Mar. 2016.

[4] G. Iosifidis, L. Gao, J. Huang, and L. Tassiulas, "Efficient and fair collaborative mobile Internet access," *IEEE/ACM Trans. Netw.*, vol. 25, no. 3, pp. 1386–1400, Jun. 2017.

[5] D. Syrivelis, G. Iosifidis, D. Delimpasis, K. Chounos, T. Korakis, and L. Tassiulas, "Bits and coins: Supporting collaborative consumption of mobile Internet," in *Proc. IEEE INFOCOM*, Apr./May 2015, pp. 2146–2154.

[6] M. Chen, Y. Hao, Y. Li, C.-F. Lai, and D. Wu, "On the computation offloading at ad hoc cloudlet: Architecture and service modes," *IEEE Commun. Mag.*, vol. 53, no. 6, pp. 18–24, Jun. 2015.

[7] F. Chi, X. Wang, W. Cai, and V. C. M. Leung, "Ad hoc cloudlet based cooperative cloud gaming," in *Proc. IEEE 6th Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2015, pp. 190–197.

[8] J. Jiang, S. Zhang, B. Li, and B. Li, "Maximized cellular traffic offloading via device-to-device content sharing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 82–91, Jan. 2016.

[9] Z. Chen, Y. Liu, B. Zhou, and M. Tao, "Caching incentive design in wireless D2D networks: A Stackelberg game approach," in *Proc. IEEE ICC*, May 2016, pp. 1–6.

[10] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Sep. 2014, pp. 1–8.

[11] A. Destounis, G. S. Paschos, and I. Koutsopoulos, "Streaming big data meets backpressure in distributed network computation," in *Proc. INFOCOM*, Apr. 2016, pp. 1–9.

[12] L. Militano, A. Orsino, G. Araniti, A. Molinaro, and A. Iera, "A constrained coalition formation game for multihop D2D content uploading," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 2012–2024, Mar. 2016.
[13] *Wi-Fi Direct Alliance*. Accessed: Oct. 10, 2018. [Online]. Available: https://www.wi-fi.org/knowledge-center/faq/how-fast-is-wi-fi-direct
[14] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1998.
[15] G. B. Dantzig, "Origins of the simplex method," in *A History of Scientific Computing*. New York, NY, USA: ACM Press, 1990, pp. 141–151.
[16] *Online Technical Report*. Accessed: Oct. 10, 2018. [Online]. Available: http://jianwei.ie.cuhk.edu.hk/publication/TechnicalReport_3CSharing_2018.pdf
[17] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1998.
[18] A.-L. Barabási, *Network Science*. Cambridge, U.K.: Cambridge Univ. Press, 2016.
[19] M. Ji, G. Caire, and A. F. Molisch, "Wireless device-to-device caching networks: Basic principles and system performance," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 1, pp. 176–189, Jan. 2016.
[20] B. C. Arnold, N. Balakrishnan, and H. N. Nagaraja, *A First Course in Order Statistics*. Philadelphia, PA, USA: SIAM, 2008.
[21] D. R. Barr and E. T. Sherrill, "Mean and variance of truncated normal distributions," *Amer. Statist.*, vol. 53, no. 4, pp. 357–361, 1999.

Ming Tang (S'16) received the Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2018. She was a Visiting Student with the Department of Management Science and Engineering, Stanford University, from 2017 to 2018. Her research interests include wireless communications and network economics, with a particular emphasis on user-provided networks and fog computing.

Lin Gao (SM'16) received the Ph.D. degree in electronic engineering from Shanghai Jiao Tong University in 2010. He is currently an Associate Professor with the School of Electronic and Information Engineering, Harbin Institute of Technology, Shenzhen, China. His main research interests are in the area of network economics and games, with applications in wireless communications and networking. He received the IEEE ComSoc Asia–Pacific Outstanding Young Researcher Award in 2016.

Jianwei Huang (F'16) is currently a Professor with the Department of Information Engineering, The Chinese University of Hong Kong. He has co-authored six books, including the textbook on *Wireless Network Pricing*. He was a recipient of nine Best Paper Awards, including the IEEE Marconi Prize Paper Award in Wireless Communications 2011. He has served as the Chair of the IEEE Technical Committee on Cognitive Networks and the Technical Committee on Multimedia Communications. He is an IEEE ComSoc Distinguished Lecturer and a Clarivate Analytics Highly Cited Researcher. More detailed information can be found at http://jianwei.ie.cuhks.edu.hk/.