Energy Efficiency of Green Social Internet of Things in Mobile Edge Networks

Chih-Hang Wang†, Jian-Jhih Kuo$^{\sharp}$, De-Nian Yang$^{\sharp}$, and Wen-Tsuen Chen$^{\dagger\sharp}$
†Department of Computer Science, National Tsing Hua University, Hsin-Chu 300, Taiwan
$^{\sharp}$Institute of Information Science, Academia Sinica, Nankang, Taipei 115, Taiwan
E-mail: s100062591@m100.nthu.edu.tw, and {lajacky, dnyang, chenwt}@iis.sinica.edu.tw

***Abstract*—With the development of AI on chips, social Internet of Things (SIoT) has drawn increasing attention because a group of devices with social relationships can collaborate to directly identify and handle local events without the help of servers. On the other hand, Mobile Edge Computing (MEC) has been regarded as a promising way for locally managing IoT data to reduce data transmission and computation in the networks. Nevertheless, for emerging Green Social Internet of Things with MEC (GSIoT-MEC), new research challenges arise due to communication and computation trade-off, cross-layer design in an SIoT, and forwarding and aggregation trade-off in MEC networks. In this paper, therefore, we formulate the IoT Collaboration Group and Device Selection Problem (ICGDSP) to minimize the total energy consumption in GSIoT-MEC. We prove that the problem is NP-hard and design an approximation algorithm that explores SIoT community, and selects IoT collaboration groups and IoT devices to handle user requests, where the SIoT and physical locations of devices all need to be carefully examined. Furthermore, we design an approach to distributionally implement the proposed algorithm to offload the computation from the cloud server to MEC servers. Also, we extend the proposed algorithm to support dynamic user requests. Simulations on two real networks and a real multi-camera map manifest that the energy consumption of GSIoT-MEC can be effectively reduced by more than 50%.**

***Index Terms*—SIoT, energy consumption, MEC, approximation algorithm.**

I. Introduction

Internet of Things (IoT) has been recognized as one of the most important paradigms in 5G networks, where IoT services, such as urban monitoring and computing, usually involve a large number of devices. In recent years, new mobile computing paradigms, such as Mobile Cloud Computing (MCC) [1] and Mobile Edge Computing (MEC) [2],[1] emerge to alleviate the network loads with computation and traffic offloading. Especially, MEC has received increasing attention in both academia and industry, such as the car-to-car system from Nokia [3], [4], because it can directly collect and process local IoT data in the MEC servers collocated with base stations or aggregating points to effectively reduce the latency and jitter [5], [6]. On the other hand, smart IoT devices with Artificial Intelligence (AI) on chips[2] have appeared recently to communicate and collaboratively process local data [7]–[9]. For example, multiple smart cameras can jointly identify the traffic congestion region [10], where each smart camera has a specific perspective on the surroundings, and all information is combined to form a more comprehensive understanding of *why* and *how* the traffic is on a road [7]. A group of smart home devices and robots can also collaborate for home surveillance without transmitting sensitive private data to the servers.[3] To foster the above scenarios, Social IoT (SIoT) has been proposed as a promising way to create and maintain the collaborative relations among smart IoT devices. The idea of SIoT is to build social collaborative networking for smart IoT devices to achieve local and distributed data processing according to the rules set by owners [11]–[15]. In addition to academic research, many SIoT projects have also been carried out [16]–[18]. These projects proposed novel infrastructures and platforms to deliver future IoT solutions based on SIoT paradigms. To bring amenities for the people, some governments have recently dedicated to realizing smart city by building SIoT systems. For example, the AI traffic system in Pittsburgh [10], where the cameras for traffic monitoring can create social links according to their *co-location* relationships since they are on the same region and collaboratively deal with traffic events.

Traditionally, large-scale IoT services usually rely on cloud computing to manage and analyze heterogeneous big IoT data for various services (e.g., ambient monitoring, surveillance, remote health caring, etc.) [19]–[21]. However, the massive amount of data are inclined to introduce a tremendous load on both the networks and servers. To address the above critical issue, MEC and SIoT recently are deemed as promising ways for network and computation offloading. Specifically, an MEC server in the network edge aggregates local IoT flows to reduce the data transmitted in the networks and reaching the servers [2], [5], [6]. On the other hand, smart IoT devices autonomously establish social collaborative groups to jointly process local data for a comprehensive understanding of the environment without the help of servers [7]–[9]. Meanwhile, minimizing the energy consumption becomes increasingly important due to the explosive growth of Information and Communication Technology (ICT) [22], especially for pervasive IoT demands [23].[4] Nevertheless, little attention has been given to optimizing the overall energy efficiency by simulta-

[1]ADLINK and Saguna introduce MEC solutions for extreme outdoor environment: https://goo.gl/A6MHZ4.

[2]Apple Inc.: https://goo.gl/ZoxV6U, ARM: https://goo.gl/PRi4wX, Intel: https://goo.gl/t4xg1W

[3]Home robots: https://goo.gl/WfNSGq, https://goo.gl/9nbMmF

[4]Companies use AI and IoT to go green: https://goo.gl/VgpBLq

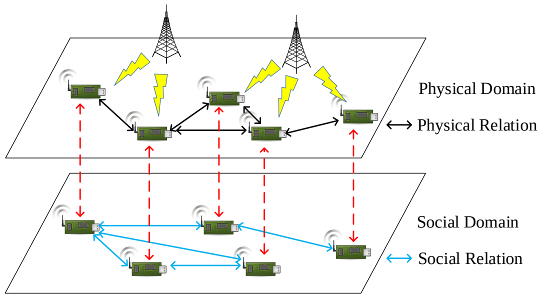


Fig. 1: The relation between the social and physical domains of SIoT devices.

neously considering the computation energy for collaborative data processing in IoT devices and big data processing in MEC servers (i.e., data aggregation) as well as the communication energy in networks.

In this paper, therefore, we make the first attempt to explore new research issues on Green Social Internet-of-Things with Mobile Edge Computing (GSIoT-MEC). The first one is *communication and computation trade-off*. An IoT device with good centrality in a device social network can collaborate with more devices to effectively reduce the transmission energy consumption to cloud servers by processing local data jointly. Nevertheless, information exchange among those devices and AI computation incurs additional energy consumption. The second issue is *cross-layer design in SIoT*. Nearby IoT devices in the physical world may not always be connected in their social networks, and collaboration of IoT friends that are close in social networks but distant in the physical world may require additional multi-hop wireless and wired transmissions. An example is shown in Fig. 1, where the neighboring devices may not be friends since they are designed for different purposes and unable to collaboratively deal with an event.[5] In this case, the selection of devices with social relations to provide the desired services becomes more complicated. The third one is the *forwarding and aggregation trade-off*. Although an MEC server aggregates multiple flows to reduce the total flow size, the total energy consumption is not always minimized by flow aggregation since the shortest path from an IoT device to the remote cloud server does not always include an MEC server. Therefore, a longer path passing through an MEC server is required to be employed and incurs additional transmission energy. To address the above challenges, we formulate a new optimization problem, named IoT Collaboration Group and Device Selection Problem (ICGDSP), to minimize the total communication and computation energy, which includes the total transmission energy in wireless and wired networks and the total computation energy in MEC servers and an SIoT.[6] Specifically, given 1) a set of devices forming an SIoT, 2) a set of MEC servers, 3) a set of user requests for IoT services,[7] 4) the energy consumption model to forward and process data, 5) a set of monitored locations, ICGDSP aims to find 1) a set of IoT devices returning desired information, 2) a set of collaboration groups constructed by IoT devices with social relationships, and 3) a set of MEC servers for data processing to ensure the *request guarantee*, *collaborative computation*, and *responding* constraints. The request guarantee constraint states that all the corresponding locations of each request are monitored by at least one IoT device or one collaboration group. The collaborative computation constraint ensures that 1) the devices selected to construct a collaboration group for a request are connected in an SIoT,[8] and 2) the corresponding locations of the request must be monitored by at least one device in the collaboration group. If a collaboration group is selected, the responding constraint states that one of the group member must be elected as the representative device to transmit the computation result. Also, the flows of selected devices can be directly transmitted to the cloud server or first aggregated with some other flows on an MEC server. We prove that ICGDSP is NP-hard and investigate ICGDSP under different SIoT scenarios to find the optimal collaborative group for a request and the intrinsic properties of collaborative group construction. Then, we devise an approximation algorithm, called Energy Efficient IoT and Collaboration Group Selection (EICGS). EICGS minimizes the overall energy consumption by exploring different SIoT communities and finding proper transmission paths for individual devices and SIoT communities. Afterward, we present the distributed implementation of EICGS for large-scale networks with collaborative MEC and extend EICGS to support dynamic requests. Finally, we evaluate the proposed algorithm on two real networks, and the results show that the energy consumption in GSIoT-MEC can be effectively reduced by more than 50%.

The rest of this paper is organized as follows. Section II summarizes the related works. We introduce the system model, formulate the ICGDSP, and present the hardness result in Section IV. Section V investigates different SIoT scenarios to find the optimal collaborative group for a request and the intrinsic characteristics of ICGDSP. Section VI presents the proposed approximation algorithm EICGS and Section VII discusses the distributed implementation of EICGS and extends EICGS to support dynamic requests. Section VIII compares different algorithms in two real networks. Conclusions are drawn in Section IX.

II. Related Work

Data aggregation with MEC has been regarded as a promising way to effectively reduce the amount of transmitted data [5], [25], [26]. Salman et al. [5] proposed a new architecture that assigns an MEC server as an IoT gateway to aggregate and process IoT data. Shan et al. [25] considered the initial

[5]For example, a fire alarm system consists of smoke, heat, carbon monoxide, and carbon dioxide detectors. If one of them is missing, these devices can only provide services on their own even if they are physically neighboring.

[6]Note that our formulation can also support optimization for other objectives, such as network bandwidth or latency minimization by replacing the coefficients in the energy model with the ones in the bandwidth consumption or delay model.

[7]In this paper, we apply the point-based model in [24], where each request corresponds to a set of locations with the desired information.

[8]The devices are connected in an SIoT when those devices have social relationships (i.e., connected in a social network) and thereby can contact to and share information via multi-hop transmissions [11].

energy of sensors and built shortest-path aggregation trees to maximize the network lifetime. Wu et al. [26] proved that finding the optimal aggregation is NP-complete and presented an approximation algorithm. Nevertheless, none of the above work targeted on SIoT, explored the cross-layer design in SIoT, and addressed the communication and computation trade-offs in SIoT and MEC networks.

SIoT has grasped increasing attention with the recent development of smart devices [11]–[14]. Atzori et al. [11] identified the policies for the establishment and management of social relationships between devices. Ortiz et al. [12] designed a general SIoT architecture, including actors, intelligent system, interface, and the Internet. Asl et al. [13] proposed to dynamically adjust IoT social networks according to different activities. Huang et al. [14] explored the SIoT for group touring. To manage big IoT data, centralized cloud-based IoT has recently drawn increasing attention [19], [20], [27]–[29]. Nasic et al. [19] presented a framework for on-demand provisioning of the software-defined IoT cloud. Kovatch et al. [20] proposed an architecture for scalable IoT cloud services based on the constrained application protocol. Liu et al. [27] proposed an software-defined IoT architecture for urban sensing applications. Jararweh et al. [28] presented a new software-defined IoT model, including networks, storage, and security. Wu et al. [29] focused on heterogeneous IoT services and proposed UbiFlow for flow control and mobility management. However, none of the above work considered MEC and addressed the forwarding and aggregation trade-off in MEC networks as well as the communication and computation trade-off in an SIoT.

III. Social Internet of Things

The idea of SIoT is to utilize social networking elements in the IoT to allow devices to autonomously establish their social relationships. In the past few years, many SIoT models have been proposed for various application scenarios [11], [13]–[15]. In this paper, for ease of understanding, we refer to the proposed models in [11], [15]. First, IoT devices produced in the same period by the same manufacturer can have *parental object relationship*. Next, devices can build *co-location object relationship* and *co-work object relationship* if they are neighboring to the others and designed for detecting or managing similar events. Moreover, the relationship of devices belonging to the same owner is defined as *ownership object relationship*. In the following, we present two scenarios to explain the above relationships in detail.

The first SIoT example is about traffic monitoring in smart city, where each smart camera has a specific perspective on the surroundings and these information are combined together to form a more comprehensive understanding of *why* and *how* the traffic is on a raod [7]. The details are as follows. A user wishes to know why the traffic of this road is so bad and sends a request to the cloud server. Then, the server informs the cameras on that road of the requests, where each camera is able to locally analyze the traffic status and share its information with other cameras to collaboratively make a decision [10]. Finally, the cameras come out that the traffic jam here is due to a crash three miles ahead and one of these cameras propagates the result to the user via the cloud server. The above case shows that the cameras have co-location and co-work relationships since they are on the same road and collaboratively deal with an event. Moreover, if these cameras belong to the government and are manufactured by the same factory, they have ownership and parental object relationships.

Another example is about the smart closet at smart home, where the information from multiple sensors are combined to make the decision for choosing clothes [11]. The details are shown as follows. A user is preparing for his next travel and wants to acquire the list of clothes to be used for a comfortable travel. Therefore, the controller of his closet needs to acquire the status of environments from multiple IoT devices such as temperature, crowdedness of the ambient, humidity level, and other parameters about the weather. All these devices may exchange the information with each other and the controller of the closet to give some appropriate advice to the user for choosing clothes. Obviously, the sensors have co-work and co-location relationships since they are co-related to detect events in a specific area for the same goal (i.e., make decision for choosing appropriate clothes).

IV. Problem Description

In this section, we introduce our system model, formulate ICGDSP, and show the hardness of ICGDSP.

A. Problem Formulation

Given a GSIoT-MEC, we formulate a new problem, named IoT Collaboration Group and Device Selection Problem (ICGDSP), as follows. Let $G^s = (V^s, E^s)$ denote an SIoT network, where V^s is the set of IoT devices and E^s is the set of social ties. Each device $n \in V^s$ with a set of covering locations $C_n \subseteq \mathbb{K}$ is associated with a base station (BS), where $\bigcup_n C_n = \mathbb{K}$ and $\mathbb{K}$ is the universal set of locations. A link $e_{n,m} \in E^s$ with weight $w_{n,m}$ (detailed later) exists if the devices $n, m \in V^s$ have social relationships, such as co-work, co-location, or parental object social relationships [11], [15]. The co-location relationships can be built if the devices are within a sensing range threshold, whereas the co-work or parental object relationships can be established if the devices belong to the same type or the same owner (e.g., the government or company). Let $\mathbb{Q} = \{q_1, q_2, ..., q_Q\}$ be the set of user requests (e.g., a user wonders where and why the traffic is congested and sends a request accordingly) with each request $q \in \mathbb{Q}$ corresponding to a set of locations L_q with the desired information [24], [30], where $\bigcup_{q \in \mathbb{Q}} L_q = \mathbb{K}$. To ensure the *request guarantee* constraint, each request q must be processed by a cloud server after collecting the desired information from individual devices or directly handled by a group of SIoT devices, where the *collaborative computation* constraint needs to be ensured as follows.

Definition 1. A request q can be handled by an SIoT group if 1) each location in L_q is covered by at least one device in the group, and 2) those devices are connected in their social networks.

The first criterion ensures that the required information in L_q is collected for further processing [7], and the second criterion states that correlated devices with the relation specified by a social network (e.g., devices with co-location, co-work, or parental object relationships) can collaboratively deal with the request [12]. After a request is addressed by an SIoT collaboration group, *responding* constraint states that one of the group members (e.g., the representative device) needs to transmit the result to the cloud server (e.g., the web server receives the request from a user [24]). Therefore, the energy cost for an SIoT collaboration group g is as follows.

$$\zeta_g = \phi_m + \gamma \cdot \sum_{u,v \in S_g, e_{u,v} \in E^s} w_{u,v}, \tag{1}$$

where γ is a weighted collaboration parameter according to the system design [31] or the type of request (scenario),[9] S_g is the selected devices for building group g, ϕ_m is the energy cost of the representative device $m \in S_g$ to transmit the result to the server, and $w_{u,v}$ summarizes the total communication and computation costs for the cooperation of devices to deal with a user request.[10] In this paper, the requests are assumed to be independent, and each device n can participate in multiple collaboration groups for different requests [24].[11] The communication and computation costs of $w_{u,v}$ for each query can vary in different links and need to be summed up to find the total energy consumption [32].

In contrast, if device n is only in charge of collecting the information for a location and then returning it to the cloud server, the information only needs to be sent back once and can be employed by similar requests. Note that although leveraging IoT devices for collaborative data processing may incur additional energy consumption from them, the service delay can be effectively reduced since big IoT data usually create heavy loads in networks and servers [2]. Also, if the device energy is limited for collaborative processing, the energy parameter α in (2) can be set to a larger value such that the processing is more inclined to be facilitated on the cloud server.

Moreover, let $G = (V, E)$ denote the network topology, where V is the set of nodes including a cloud server D, R MEC servers $\mathbb{R} = \{r_1, r_2, ..., r_R\}$, and multiple BSs and switches. E is the set of links. The cloud server D receives and assigns user requests, whereas each MEC server $r \in \mathbb{R}$ can aggregate sensory flows by compressing them with a ratio β [33]. That is, the flow size $b(n)$ originally sent from the device n becomes $b(n) \cdot (1 - \beta)$ sent from an MEC server to D. Each flow can also be directly transmitted to D.

In this paper, the energy consumption consists of the following four parts: 1) IoT devices, 2) the cloud server, 3) MEC servers, and 4) switches. The energy consumption of fundamental IoT sensing is correlated to the transmitted data size [34].

$$\alpha \cdot x_n \cdot c(n), \tag{2}$$

where α is the energy parameter representing the importance of IoT device energy (which is usually large) since the energy of IoT devices is more valuable due to the limited battery [25], [26]. x_n represents if device n is selected, and $c(n)$ is the transmission energy of n, which is calculated based on the data size [34].

For the cloud server, we consider the energy consumption for processing collected data, where the energy cost is proportional to the computational load of CPU. Therefore, the energy model of the server is as follows [35].

$$\delta = e_{idle}^{server} + e_{unit}^{server} \cdot \sum_{n \in \mathbb{N}} y_n \cdot b(n), \tag{3}$$

where e_{idle}^{server} is the idle power of the cloud server, and e_{unit}^{server} is the unit energy to process the data of a device. $\mathbb{N}$ denotes the set of devices, while y_n represents whether the device n is processed by the cloud server.

For an MEC server $r \in \mathbb{R}$, the energy consumption is proportional to the computation load of CPU as follows [35].

$$\sigma_r = e_{idle}^{mec} + e_{unit}^{mec} \cdot \sum_{n \in \mathbb{N}} z_{n,r} \cdot b(n), \tag{4}$$

where e_{idle}^{mec} is the idle power of an MEC server, e_{unit}^{mec} is the unit energy consumption to aggregate a unit flow, and $z_{n,r}$ denotes if the flow of n is aggregated on the MEC server r.

According to [36], the energy consumption of a switch v to forward an IoT flow is a linear function of flow rate.

$$\rho_v = e_{idle}^{switch} + e_{unit}^{switch} \cdot \sum_{i=1}^{M} d_{i,v}, \tag{5}$$

where e_{idle}^{switch} is the idle power of a switch, $d_{i,v}$ is the data rate of the port i on the switch v, M is the number of ports, and e_{unit}^{switch} is the unit energy to forward a flow.

Equipped with the above energy models (2)–(5), we formulate ICGDSP as follows.

Problem: IoT Collaboration Group and Device Selection Problem (ICGDSP)
Instance: A set of IoT devices $\mathbb{N} = \{n_1, n_2, ..., n_N\}$ with N coverage sets $\mathbb{C} = \{C_1, C_2, ..., C_N\}$, a set of user requests $\mathbb{Q} = \{q_1, q_2, ..., q_Q\}$ with the corresponding location sets $\mathbb{L} = \{L_1, L_2, ..., L_Q\}$, the energy consumption model, an SIoT $G^s = (V^s, E^s)$, and an MEC network $G = (V, E)$.
Task: To select 1) a subset of IoT devices $\mathbb{S} \subseteq \mathbb{N}$, 2) a subset of collaboration groups constructed by devices with social relationships, and 3) a set of MEC servers for data processing,

[9] In general, a more complicated situation needs a larger γ since devices consume more energy in computation and message exchange. For example, for the traffic monitoring depicted in Section I, each camera first processes a huge amount of video and then shares the information extracted from these images for collaborative processing. On the other hand, for the smart home, each device just exchanges small sensing results such as temperature or humidity level, and parameter γ can be lowered in this case.

[10] IoT communication and computation occur simultaneously in popular commercial platforms [32]. In [7], adjacent cameras communicate for collaborative user identification to build a global map of inhabitants' locations in real time.

[11] IoT devices in smart cities provide various independent services like public safety, transportation, and healthcare: https://goo.gl/5NzxTW.

where the flows of selected devices can be directly transmitted to D or first aggregated with some other flows on an MEC server $r \in \mathbb{R}$, such that 1) the total energy consumption in GSIoT-MEC is minimized and 2) each request q is handled by D or a collaboration group of devices with social relationships. The energy consumption of GSIoT-MEC includes 1) the transmission energy of devices, 2) the processing energy of the cloud and MEC servers, 3) the forwarding energy of switches, and 4) the communication and computation energy of collaboration SIoT groups as follows.[12]

$$\alpha \sum_{n \in \mathbb{N}} x_n \cdot c(n) + \delta + \sum_{r \in \mathbb{R}} \sigma_r + \sum_{v \in V \backslash \{D\}} \rho_v + \sum_{i=1}^{Q} \zeta_{g_i}. \quad (6)$$

Remark that since ICGDSP is designed for IoT services, instead of the general traffic, the objective of ICGDSP is to minimize the energy consumption in GSIoT-MEC. Moreover, the energy consumption in GSIoT-MEC is dependent on the amount of IoT flows. Therefore, minimizing the energy consumption can reduce the amount of IoT flows such that the throughput and utilization of general traffic can be maximized.

B. Hardness Result

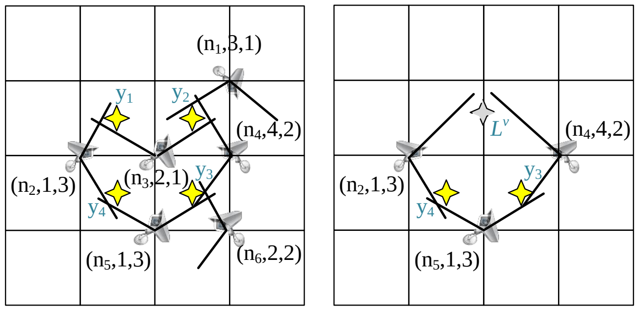
In the following, we prove that ICGDSP is NP-hard with the reduction from Group Steiner Tree Problem (GSTP).

Theorem 1. *ICGDSP is NP-hard and inapproximable within* $\log^2 |\mathbb{N}|$.

Proof. We first identify the special case of ICGDSP by theoretically defining 1) the energy consumption of MEC servers is zero, 2) the size of sensory data is large enough (i.e., $\gamma \times \max_{e_{n,m} \in E^s}\{w_{n,m}\} \times N$), 3) the size of collaboration result data is zero, 4) the compression ratio β is zero, and 5) the number of requests is 1. It is noteworthy that the transmission energy is related to the data size according to (2). Therefore, the problem becomes selecting a collaboration group of IoT devices to cover all the locations corresponding to the request such that the energy consumption of a collaboration group is minimized and the selected devices are connected (see Definition 1). In the following, we prove that this special case of ICGDSP is NP-hard with the reduction from GSTP.

Let $G = (V, E)$ be the graph of GSTP, where each edge $e \in E$ is associated with a weight w_e. Let $\mathbb{S} = \{s_1, s_2, \cdots, s_k\}$ be a collection of vertex sets, where $s_1, s_2, \cdots, s_k \subset V$. The problem is to find a minimum cost subtree T of G such that at least one vertex in s_i is also in T for all $i \in \{1, ..., k\}$. For each set of GSTP, we construct an instance of ICGDSP as follows. First, for each vertex $v \in V$, we construct an IoT device n and an associated BS, and the BS directly connects to the cloud server in one hop. Next, we create a corresponding observed location in L_q for each vertex set in $\mathbb{S}$. Then, an observed location k is added into the coverage of the device n if the corresponding vertex $v \in V$ of n is in the corresponding set $s \in \mathbb{S}$ of k. Finally, we construct a social tie with a

[12]We use the index of group same as the one of the requests since each request is handled by at most one group.



(a) Original deployment (b) Transformed deployment

Fig. 2: An example of OCGS with six cameras and four observed locations on the Manhattan street map.

weight $w_{n,m}$ in E^s for any two devices n, m in $\mathbb{N}$ if their corresponding vertices $u, v \in V$ have a link with a weight $w_{u,v}$ in E.

For the GSTP, if there exists a subtree T of G such that at least one vertex in s_i is in T for all $i \in \{1, ..., k\}$, we can find the corresponding subset $\mathbb{N}^* \subseteq \mathbb{N}$ such that the union of the observed locations covered by the devices in $\mathbb{N}^*$ is L_q and the devices in $\mathbb{N}^*$ are connected. Conversely, if there exists a subset $\mathbb{N}^* \subseteq \mathbb{N}$ such that the union of covered locations in $\mathbb{N}^*$ is L_q and the devices in $\mathbb{N}^*$ are connected, we can find the corresponding subtree T of G such that at least one vertex in s_i is in T for all $i \in \{1, ..., k\}$. Note that the total cost also remains the same after the transformation, because each social tie in E^s with weight $w_{n,m}$ corresponds to an edge in E with weight w_e. Hence, the special case of ICGDSP is as hard as GSTP, and therefore ICGDSP is NP-hard. Also, Halperin and Krauthgamer [37] proved that GSTP is inapproximable within $\log^{2-\epsilon} n$ for every fixed $\epsilon > 0$, where n here is the number of vertices. Therefore, the theorem follows. ☐

V. Collaborative Groups in Different SIoT Scenarios

In this section, we first investigate ICGDSP under different SIoT scenarios to find the optimal collaborative group for a request and the intrinsic properties of ICGDSP. Then, the general solution will be provided in the next section. Firstly, we explore the grid deployment of SIoT devices with directional sensing (e.g., smart cameras) being deployed at the intersections in a city such as the Manhattan street map, where the SIoT devices include co-location relationship. Next, we explore the scenario without IoT device selection (i.e., all the devices are selected), and we focus on building collaborative groups with all-pair argumentation, which lead to the consensus in the groups.[13]

A. Grid Deployment

In this section, we explore the grid deployment of SIoT devices with directional sensing (e.g., smart cameras) as shown

[13]Since arguments among devices vary according to different SIoT scenarios [38], we consider all-pair argumentation for all possible message transmission between devices.

in Fig. 2(a). The devices are deployed on some intersections, while the observed locations are located inside some blocks. Each intersection includes only one device, and each block contains only one location, which in turn therefore shows that each device monitors at most two locations in the grid map. In the following, we propose an algorithm, Optimal Collaborative Group Selection (OCGS), to find the optimal collaborative group such that the energy cost of the group is minimized.

Recall that one of the selected devices must be elected as a representative device to transmit a collaborative result to the cloud server. To deal with this issue, our idea is to try every device $n \in \mathbb{N}$ as a candidate, find solutions with different candidates, and extract the one with minimum energy cost in SIoT. Since a device may collaborate with its neighbors, we consider all possible communications and define the cost of a device n in SIoT as follows.

$$c^{SIoT}(n) = \sum_{e_{n,m} \in E^s} w_{n,m}. \tag{7}$$

Specifically, OCGS first selects a candidate device $n \in \mathbb{N}$, and then modifies the original deployment as follows. If the selected candidate n monitors two locations (i.e., $|C_n| = 2$), these two locations are merged into a virtual location L^v and L^v is monitored by the devices except n that cover these two locations. On the other hand, if n covers only one location, the deployment remains the same. Moreover, it is noteworthy that any device $\hat{n} \in \mathbb{N}$ covering only one location can be removed because at least one of neighboring devices of $\hat{n}$ that cover two locations must be selected in OCGS. Otherwise, the selected devices cannot form a collaborative group since they are not connected in the SIoT graph. An example is shown in Fig. 2, where the device n_3 is selected as a candidate device, and the observed locations y_1 and y_2 are merged into the virtual location L^v covering by n_2 and n_4.

After transforming the deployment, OCGS finds the optimal collaborative group with the specific candidate device n on the transformed geography. First, OCGS considers a random location l, and selects the device into $\mathbb{S}$ that covers l and includes the minimum cost in the SIoT defined in (7), where $\mathbb{S}$ is the set of selected devices. Then, OCGS chooses the device with minimum cost that monitors one location covered by $\mathbb{S}$ and one uncovered location. As the above step, OCGS repeatedly adds devices into $\mathbb{S}$ until all targets are monitored by at least one SIoT device. After that, OCGS examines the next candidate representative device with the above procedure to find a collaborative group. After extracting all groups with different candidate devices, OCGS selects the one with minimum cost. The pseudocode of OCGS is provided in Algorithm 1.

Algorithm 1 Optimal Collaborative Group Selection (OCGS)

```
Input: Set of IoT devices $\mathbb{N}$ with $N$ coverage sets $\mathbb{C}$, set of observed
    locations $\mathbb{K}$, an SIoT $G^s = (V^s, E^s)$, and an MEC network
    $G = (V, E)$
Output: Selected devices for a collaborative group
 1: for all IoT device $n$ in $\mathbb{N}$ do
 2:   Initialize: $\mathbb{Y} = \mathbb{K}$, $\mathbb{S} = \emptyset$
 3:   Select $n$ as a candidate;
 4:   if $|C_n| = 2$ then
 5:     Merge the locations in $C_n$ into a virtual location;
 6:   end if
 7:   Remove any device covering only one location;
 8:   Start from a random location $l$ and select the device $m$ with
      minimum cost covering $l$;
 9:   $\mathbb{Y} \setminus C_m$; $\mathbb{S} \cup \{m\}$;
10:   while $\mathbb{Y} \neq \emptyset$ do
11:     Select the device $m$ that monitors one location covered by $\mathbb{S}$
        and one uncovered location;
12:     $\mathbb{Y} \setminus C_m$; $\mathbb{S} \cup \{m\}$;
13:   end while
14:   Record the collaborative group with the candidate $n$
15: end for
16: Select the group with minimum cost;
```

Example. Fig. 2 shows an example with six cameras and four observed locations in a grid map, where the tuple beside each camera represents (*ID*, *SIoT cost*, *transmission cost*). First, OCGS selects n_3 as the representative device, and merges the observed locations y_1 and y_2 into L^v which is covered by n_2 and n_4. Then, OCGS removes n_1 and n_6 since they cover only one location. The ultimately transformed deployment is shown in Fig. 2(b). After transforming the deployment, OCGS selects a collaborative group on the transformed deployment. Assume that OCGS starts from the location L^v, OCGS first selects the camera n_2 since it covers L^v with the minimum SIoT cost (i.e., 1). Next, OCGS chooses n_5 since it includes the minimum SIoT cost (i.e., 1) among the unselected cameras that monitor one covered and one uncovered locations. Finally, the selected cameras are n_2, n_3, and n_5, where n_3 is the candidate representative camera, and the total cost is $1+2+1+1 = 5$. With the same process, OCGS tries the next candidate representative device and selects the corresponding collaborative group. In this example, the optimal solution is n_2, n_3, and n_5 with n_3 being the representative device, where the total cost is 5.

Time complexity. Since OCGS considers each device as a candidate, it takes at most $O(N)$ time to transform the deployment. Then, OCGS takes at most $O(N^2)$ time to build a collaborative group since there are at most N devices can be selected and each covered location set can be monitored by at most N devices that contain one uncovered location in their coverage. Therefore, OCGS takes $O(N^3)$ time to build an optimal collaborative group.

In the following, we prove that OCGS can find the optimal collaborative group.

Theorem 2. *OCGS produces an optimal collaborative group.*

Proof. We first prove that OCGS can find the optimal collaborative group with a specific representative device n on the corresponding transformed deployment. Let $A_{OCGS}(n)$ and $OPT(n)$ respectively be the solution obtained by OCGS and the optimal solution with a specific n. If $A_{OCGS}(n)$ is equal to $OPT(n)$, then $A_{OCGS}(n)$ is optimal. Otherwise, let $m(l_i, l_j)$ be the first device covering the locations l_i and l_j, which is chosen at the k^{th} iteration of OCGS, and not selected in $OPT(n)$. Let S_c be the set of connected devices covering the continuously neighboring locations from

l_i to l_j in $OPT(n)$, and $\hat{m}$ be the device in the set S_c with the coverage including one covered location l_i and one uncovered location at the $(k-1)^{th}$ iteration. Since OCGS iteratively selects the device with minimum cost that monitors one covered and an uncovered locations, we obtain that the cost of $\hat{m}$ is larger than $m(l_i, l_j)$. Therefore, we substitute $\hat{m}$ with $m(l_i, l_j)$ and obtain another solution with a lower total cost and one more common selected device with $A_{OCGS}(n)$ than $OPT(n)$, while all the locations are still covered. The above process can be repeated until $OPT(n)$ is equal to $A_{OCGS}(n)$. Since each substitution generates a solution with lower cost than $OPT(n)$ which contradicts the optimality of $OPT(n)$, $A_{OCGS}(n)$ is the optimal solution with a specific representative device n.

Now, we show that OCGS generates the optimal solution to ICGDSP with grid deployment. For each specific representative device n, two possible cases exists. The first case is that n is included in the optimal solution with n being the representative device, and it is obvious that OCGS generates the optimal solution due to the previous proof. The other case is that the optimal solution includes the representative device other than n. Since OCGS chooses the solution containing the representative device with minimum total cost, OCGS obtains the optimal solution among all possible solutions with different representative devices. Therefore, the theorem follows. □

B. All-Pair Argumentation

In this section, we assume that all the devices are selected, and focus on building the communication topology of a collaborative group by considering all-pair argumentation among devices (i.e., all possible transmission way between devices). It is worth noting that since the argumentation is changing according to different SIoT scenarios [38] and some messages may be simultaneously transmitted to multiple devices, considering all-pair argumentation and forming a connected topology of devices can provide the worst-case cost in a collaborative group.

In the following, we propose an algorithm, Approximated Collaborative Group (ACG), to form an approximated collaborative group with all-pair argumentation. Since we aim to minimize the total cost of all-pair transmission, ACG first selects the device with minimum total transmission cost to all devices as a *communication center*. Specifically, the center m minimizes the cost function $f(m) = \sum_{n \in G^s} c^T(n, m)$, where $c^T(n, m)$ is the cost of the minimum-cost transmission path between n and m which can be found by a shortest path algorithm.[14] Then, ACG builds a minimum-cost connected topology rooted at the center m by selecting the links that are included in the transmission path with minimum cost between each device $n \in V^s$ and the center m. The pseudocode of ACG is provided in Algorithm 2.

Algorithm 2 Approximated Collaborative Group (ACG)

```
Input: Set of IoT devices $\mathbb{N}$, and an SIoT $G^s = (V^s, E^s)$
Output: Connected topology of IoT devices
1: Select the device $m$ minimizing $\sum_{n \in G^s} c^T(u, m)$;
2: for all $n \in V^s$ do
3:     Build the minimum-cost transmission path between $n$ and $m$ in $G^s$, and select those links on the path;
4: end for
```

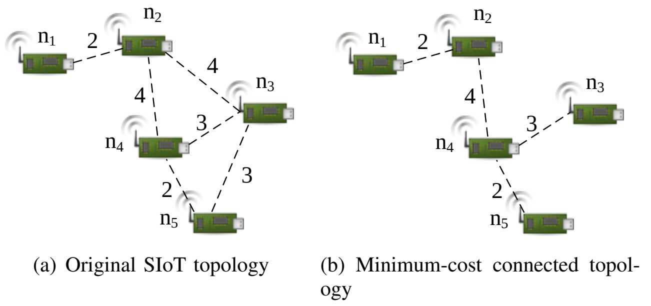

n1
2
n2
4
n3
4
3
n4
3
2
n5
(a) Original SIoT topology
n1
2
n2
4
n3
3
n4
2
n5
(b) Minimum-cost connected topology

Fig. 3: An example of ACG with five SIoT devices.

Example. Fig. 3 shows an example of ACG with five SIoT devices, where the number beside each dotted line indicates the energy cost between the two devices. First, ACG calculates the transmission path with minimum cost for each device pairs and selects the device m minimizing $f(m) = \sum_{n \in G^s} c^T(n, m)$ as the center. To simplify the presentation, we take the device n_4 as an example and have $f(n_4) = c^T(n_1, n_4) + c^T(n_2, n_4) + c^T(n_3, n_4) + c^T(n_4, n_4) + c^T(n_5, n_4) = 6 + 4 + 3 + 0 + 2 = 15$. The other devices n_1, n_2, n_3, and n_5 incur costs 22, 16, 19, and 19, respectively. Therefore, ACG selects n_4 as the center and then builds the minimum-cost connected topology by selecting the links included in the minimum-cost transmission paths from n_4 to all the other devices. The result shows in Fig. 3(b).

Time complexity. First ACG finds the center by finding the transmission path with minimum cost for N devices, where these paths can be found by Dijkstra's algorithm taking at most $O(N^3)$. Then, ACG selects the device with minimum total transmission cost to all devices as the center and builds a minimum-cost connected topology rooted at the center in $O(1)$ time. This is because the transmission path with minimum cost has been found by the previous step. Therefore, ACG takes at most $O(N^3)$ time.

In the following, we prove that ACG is a 2-approximation algorithm to find a collaborative group.

Theorem 3. *ACG is a 2-approximation algorithm.*

Proof. Let w be the center of the SIoT $G^s = (V^s, E^s)$ and A be the minimum-cost connected topology rooted at the center w, which is found by ACG. It is noteworthy that the triangle inequality holds for the transmission cost between any pair of devices since the transmission energy is correlated to the physical distances between devices [39]. By the triangle inequality, we have $c^T_A(n, m) \leq c^T_A(n, w) + c^T_A(w, m)$ for any devices n and m, where $c^T_A(n, m)$ denotes the minimum transmission cost between the devices n and m on the topology A. To calculate the cost of all-pair argumentation on A, we sum up the transmission costs between all-pair devices and

[14]Note that $c^T(n, m) = 0$ if $n = m$.

obtain

$$\begin{aligned}\Gamma(A) &= \sum_{n\in V^s}\sum_{m\in V^s} c_A^T(n,m) \\ &\le |V^s|\sum_{n\in V^s} c_A^T(n,w) + |V^s|\sum_{m\in V^s} c_A^T(w,m) \\ &= 2|V^s|\sum_{n\in V^s} c_A^T(n,w),\end{aligned} \tag{8}$$

where the inequality holds due to the triangle inequality. Since w is the center of the SIoT, we have $\sum_{n\in V^s} c^T(n,w) \le \sum_{n\in V^s} c^T(n,m)$ for any m, and it follows that

$$\sum_{n\in V^s} c^T(n,w) \le \frac{1}{|V^s|}\sum_{n\in V^s}\sum_{m\in V^s} c^T(n,m). \tag{9}$$

Note that in a minimum-cost connected topology, the transmission path from the center to any device is a path with minimum cost in the SIoT. Therefore, we have $c_A^T(n,w) = c^T(n,w)$ for any device n, and consequently obtain

$$\begin{aligned}\Gamma(A) &\le 2|V^s|\sum_{n\in V^s} c_A^T(n,w) \\ &= 2|V^s|\sum_{n\in V^s} c^T(n,w) \\ &\le 2\sum_{n\in V^s}\sum_{m\in V^s} c^T(n,m),\end{aligned} \tag{10}$$

where the inequalities hold due to (8) and (9). respectively. Finally, let A^* be the optimal minimum-cost connected topology, and it is obvious that $c^T(n,m) \le c_{A^*}^T(n,m)$ since $c^T(n,m)$ is the minimum transmission cost between the devices n and m in the SIoT. Therefore, by (10), we obtain $\Gamma(A) \le 2\sum_{n\in V^s}\sum_{m\in V^s} c^T(n,m) \le 2\sum_{n\in V^s}\sum_{m\in V^s} c_{A^*}^T(n,m) = 2\Gamma(A^*)$, and the theorem follows. □

VI. Algorithm Design for EICGS

To effectively solve the ICGDSP, we design an algorithm, called Energy Efficient IoT and Collaboration Group Selection (EICGS), which consists of four phases: 1) Energy Cost Assignment (ECA), 2) Collaborative Group Construction (CGC), 3) Coverage Transformation (CT) and 4) Device and Group Selection (DGS). It minimizes the overall energy consumption by carefully exploring good chances for data aggregation on MEC servers and group collaboration for SIoT. More specifically, ECA first optimizes the energy efficiency by extracting the paths with the minimum energy consumption based on the shortest path scheme employed in the current Internet [40], where the energy includes the transmission energy of IoT devices, the computation energy of MEC servers for aggregation, and the forwarding energy of switches. Then, CGC finds the collaboration group with the minimum energy cost for each request by considering every possible representative device connecting to the other members of the group. Next, CT generates *assist locations* for the physical locations such that all the requests can be satisfied by covering the *assist locations*. Finally, DGS selects a set of individual devices or a collaboration group of devices with the minimum cost-coverage ratio (detailed later) to effectively minimize the total energy consumption. Due to the space constraint, the pseudocode and the detailed approximation ratio analysis for EICGS are presented in [41].

A. Algorithm Design

In the following, we describe the details of each phase.

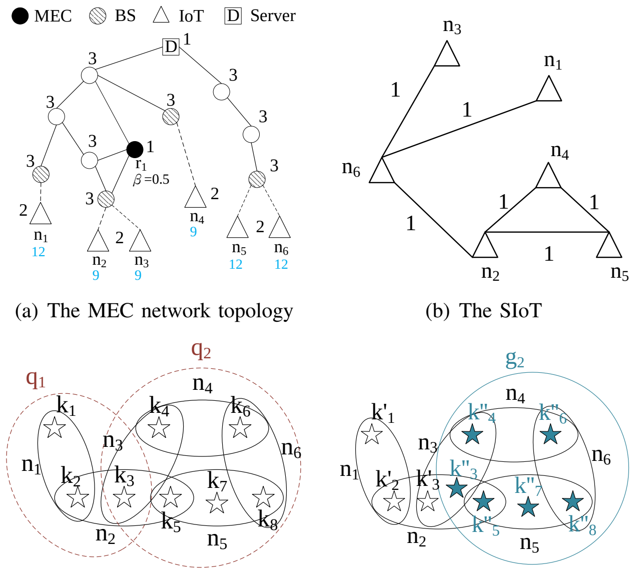
1) Energy Cost Assignment (ECA)

ECA first extracts the path with the minimum energy consumption for each device $n \in \mathbb{N}$ and sets the dissipated energy on the path to ϕ_n. Recall that the energy consumption of a path for each device includes the transmission energy, forwarding energy, aggregation energy (if the data is aggregated), and processing energy on the cloud server. For the non-aggregation path of the flow f_n, ECA extracts the shortest path from s_n to D to minimize the forwarding energy according to (5), where s_n is the serving BS of n. Therefore, the total energy consumption of a non-aggregation path is $\alpha \cdot c(n) + e_{unit}^{switch} \cdot b(n) \cdot \sum_{u\in P_{s_n,D}^{st}} 1 + e_{unit}^{server} \cdot b(n)$, where $P_{s_n,D}^{st}$ denotes the shortest path from s_n to D.

On the other hand, for the aggregation path of the flow f_n, since the flow size becomes smaller after traversing an MEC server, ECA first calculates the energy consumption of the aggregation path from BS s_n to the cloud server D. The total transmission energy of the paths from s_n to the MEC server $v \in \mathbb{R}$ and from v to D is $e_{unit}^{switch} \cdot b(n) \cdot \sum_{u\in P_{s_n,v}^{st}} 1 + e_{unit}^{switch} \cdot (b(n)\cdot(1-\beta)) \cdot \sum_{u\in P_{v,D}^{st}} 1$, where $P_{s_n,v}^{st}$ and $P_{v,D}^{st}$ are the shortest paths from s_n to v and v to D, respectively. According to (4), each device n consumes $e_{unit}^{mec}\cdot b(n)$ energy for data aggregation in an MEC server. Therefore, the total energy consumption of an aggregation path is $\alpha\cdot c(n) + e_{unit}^{switch} \cdot b(n) \cdot \sum_{u\in P_{s_n,v}^{st}} 1 + e_{unit}^{switch} \cdot (b(n)\cdot(1-\beta)) \cdot \sum_{u\in P_{v,D}^{st}} 1 + e_{unit}^{mec}\cdot b(n) + e_{unit}^{server}\cdot(b(n)\cdot(1-\beta))$. After finding the energy consumption of aggregation and non-aggregation paths of each device n, ECA iteratively selects the path with the minimum energy consumption for n and sets its energy cost ϕ_n. The flows from multiple devices are compressed and aggregated to a smaller one if they select the aggregation paths with the same MEC server.

Example. Fig. 4 presents an illustrative example of EICGS with the compression ratio $\beta = 50\%$ and the transmission energy of IoT device as 2. The data size of devices is 1 and is assumed to be identical. The energy consumption of the switches/BSs, the cloud server, and the MEC server are respectively set to 3, 1, and 1 in this example, where the energy parameter α, collaboration parameter γ, and the energy costs in the SIoT are all set to 1.

In the ECA phase, the energy consumption of non-aggregation path for the device n_1 is $3\times3+2+1 = 12$ because the shortest path from the serving BS of n_1 to D consists of 3 hops. The transmission energy of n_1 is 2, and the computation energy of D is 1. On the other hand, the energy consumption on the aggregation path is $3\times3+2\times1.5+2+1+1 = 16$ since the forwarding energy of switches becomes half (i.e., 1.5) after traversing through the MEC server r_1, and the aggregation



(a) The MEC network topology (b) The SIoT

(c) The coverage of IoT devices (d) The result of CT

Fig. 4: An illustrative example of EICGS.

energy on r_1 is 1. Therefore, ECA selects the non-aggregation path for n_1 since it consumes less energy and sets the cost of n_1 to 12. Similarly, ECA selects the aggregation path for n_2 since it consumes the minimum energy (i.e., 9). The final energy cost of $(n_1, n_2, n_3, n_4, n_5, n_6)$ is (12,9,9,9,12,12), as shown in Fig. 4(a).

2) Collaboration Group Construction (CGC)

CGC finds the collaboration group with the minimum energy cost for each request to minimize the total energy consumption. According to Definition 1 and the energy cost in (1), CGC chooses the collaboration group according to the following rules: 1) the devices in the group are connected in an SIoT, 2) the corresponding locations of a request are covered by the devices in the group, and 3) the communication cost of the group is small. To achieve the above goal, our idea is to exploit the notion of Group Steiner Tree (GST) for a graph as a subroutine, where each device and location is regarded as a vertex and a group in the graph, respectively. A vertex is in a group if the corresponding device covers the location corresponding to that group, and an edge connects two vertices if the corresponding devices have social relationships.

Definition 2. Given a graph $G = (V, E)$ and groups of vertices $g_1, g_2, ..., g_k \subset V$, where each edge $e \in E$ is associated with a weight w_e, the objective is to find the minimum cost subtree T of G such that $V(T) \cap g_i \neq \emptyset$ for all $i \in \{1, ..., k\}$.

According to the above definition, we construct a collaboration group such that 1) at least one device (vertex) of each location is selected, 2) the selected devices are connected, and 3) the total weight of the edges in an SIoT between the selected devices is minimized. However, the energy cost for the representative IoT device to return the result to the cloud server is not incorporated. To address this issue, CGC examines every device as a candidate representative device, adds it to the solution, and then finds the solutions with different candidates to extract the one with the minimum energy. More specifically, we design a *rooted* algorithm as follows. For every candidate representative device n, we generate an *augmented location* that is only covered by n. Therefore, the device n must be included in the extracted collaboration group by GST since each location corresponding to the request needs to be covered, and the energy cost can be derived according to (1). Afterward, we extract the solution with the minimum energy. Note that the proposed *rooted* algorithm can benefit the distributed implementation since a collaboration group can be locally managed by considering only a part of devices.

Example. In the CGC phase, for the request q_2 in Fig. 4(c), we exploit the proposed *rooted* algorithm to select the collaboration group with the minimum energy by trying every possible representative device.[15] The group with the minimum energy thereby includes the devices $\{n_2, n_4, n_5\}$, and the energy cost of the group is $1 \times 3 + (9 - 1) = 11$. The subtraction of 1 is because the cloud server does not need to process the result transmitted from the group. Note that a collaboration group cannot be found for the request q_1 since the devices (i.e., n_1, n_2, and n_3) covering the corresponding locations L_{q_1} do not have social relationships as shown in Fig. 4(b).

3) Coverage Transformation (CT)

CT creates a set of *assist locations* for each physical location and replaces the physical locations with their corresponding *assist locations* such that the requests can be satisfied by ensuring the coverage of *assist locations*. Recall that the requests are independent and each IoT device can participate in multiple collaboration groups. To sum up the energy of a device consumed in different groups, an assist location is required for each request such that a collaboration group can only cover the corresponding assist location set. Therefore, once a location required by multiple requests is covered (i.e., all the corresponding assist locations are covered), the energy of a device can be derived by selecting the proper collaboration groups. More specifically, for each location $k \in \mathbb{K}$, CT generates an assist location a_q^k if k is in L_q. The locations are then replaced by the corresponding assist locations, and each collaboration group g dealing with q covers only the corresponding assist locations (i.e., $\hat{C}_g = \{a_q^k | \forall k \in L_q\}$). This is because the information of a location may be required by different requests and therefore needs to be covered by multiple collaboration groups such that the energy of a device participating in multiple groups can be summed up. On the other hand, the physical device n covers all the corresponding assist locations in its coverage range (i.e., $\hat{C}_n = \{a_q^k | \forall k \in C_n, q \in \mathbb{Q}\}$). It is because after the sensory data from a device is transmitted to the cloud server, other related requests can be handled by the server even though their corresponding collaboration groups are not selected. With the above transformation, the requests can be satisfied if all

[15]Note that the energy cost of each device has been obtained in ECA.

the assist locations are covered by at least one IoT device or collaboration group.

Example. In the CT phase, since the location k_3 in Fig. 4(c) is requested by q_1 and q_2, we generate two *assist locations* k_3' and k_3'' accordingly. For the other locations, CT generates $\{k_1', k_2'\}$ and $\{k_4'', k_5'', k_6'', k_7'', k_8''\}$ for $\{k_1, k_2\}$ and $\{k_4, k_5, k_6, k_7, k_8\}$, respectively, since they are only covered by one of the requests q_1 and q_2. Then, CT replaces the locations by the corresponding *assist locations* and the result is shown in Fig. 4(d).

4) Device and Group Selection (DGS)

DGS iteratively extracts the IoT device or the collaboration group built in CGC with the minimum cost-coverage ratio until all the assist locations are covered by at least one device or collaboration group. Recall that by CT, the coverage of the collaboration group g corresponding to the request q is $\hat{C}_g = \{a_q^k | \forall k \in L_q\}$. Therefore, the cost-coverage ratio of the collaboration group g (device n) is defined as

$$\frac{\phi_g}{|\hat{C}_g \cap \hat{\mathbb{Y}})|}\left(\frac{\phi_n}{|\hat{C}_n \cap \hat{\mathbb{Y}})|}\right), \tag{11}$$

where ϕ_g (ϕ_n) is the energy cost of the collaboration group g (device n), and $\hat{\mathbb{Y}}$ is the set of uncovered assist locations. Specifically, DGS iteratively selects the device n or collaboration group g with the minimum cost-coverage ratio and updates the uncovered assist location set $\hat{\mathbb{Y}}$ by removing the locations in $\hat{C}_g$ or $\hat{C}_n$. DGS stops until all the assist locations are covered; that is, each request is handled by the cloud server or a collaboration group.

Example. In the DGS phase, the collaboration group g_2 in Fig. 4(d) is first selected since its cost-coverage ratio (i.e., $\frac{11}{6}$) is the minimum one in the first iteration, and the uncovered locations are $\{k_1', k_2', k_3'\}$ after g_2 is selected. Through the same process, DGS then sequentially selects the devices n_2 and n_1 to cover the remaining locations. The total energy cost is thereby $11 + 9 + 12 = 32$ which is optimal in this example. It is worth noting that the optimal energy cost is 42 by current IoT sensing (i.e.,$\{n_1, n_3, n_4, n_5\}$). Therefore, the collaboration in the SIoT can effectively reduce the total energy consumption.

Time Complexity. In the ECA phase, EICGS first determines the path with the minimum energy consumption for each IoT device $n \in \mathbb{N}$ by considering at most $R + 1$ paths, which includes R aggregation paths and one non-aggregation path. Therefore, it takes at most $O(NR)$ to find $R+1$ shortest path via different MEC servers for N devices. Then, EICGS takes $O(N)$ to set the energy cost for each device, and therefore it takes $O(NR)$ in the ECA phase. Next, CGC takes at most $N \cdot Q \cdot T_{GST}$ to build collaboration groups for Q requests with at most N roots in CGC, where T_{GST} is the time complexity of GST. Then, EICGS takes $O(KQ)$ to create at most Q assist locations for each location in CT. Finally, in the DGS phase, EICGS takes $O(N + Q)$ iterations to guarantee the coverage since there are at most $N + Q$ IoT devices and groups. Moreover, the minimum cost-coverage ratio can be found in $O(\log(N + Q))$ by using a priority queue in each iteration. Therefore, the time complexity of DGS is $O((N + Q)\log(N + Q))$. The overall time complexity of EICGS is $O(NR) + N \cdot Q \cdot T_{GST} + O(KQ) + O((N+Q)\log(N+Q))$. In practice, it is worth noting that the time complexity is dominated by DGS since only a subset of devices considered in CGC and the number of locations K is usually smaller than the number of devices N.

VII. Discussion

In this section, we discuss the distributed implementation of EICGS and extend EICGS to support dynamic user requests.

A. Distributed Implementation of EICGS

For large-scaled networks, it is obliged to employ distributed computation architecture like [42] where each MEC server is in charge of managing a subset of networks. In the following, we describe the distributed implementation of EICGS, namely D-EICGS, for large-scale networks, where each MEC server is able to share the information via backhaul links. It is noteworthy that D-EICGS leads to the same result as EICGS since we offload and distribute the computation on the cloud server to MEC servers without changing the computation steps. In our design, we assume that the cloud server is responsible for progress tracking and phase switching. D-EICGS consists of four phases: 1) D-ECA, 2) D-CGC, 3) D-CT, 4) D-DGS, which are the distributed implementation of each phase in EICGS.

1) D-ECA

In this phase, for aggregation paths, D-ECA distributes the energy consumption calculation from the cloud server to MEC servers. Recall that we employ the shortest path scheme currently employed in the Internet [40]. For each MEC server, therefore, it is able to obtain the energy consumption of its shortest path to the cloud server with the path provided by the Internet Service Provider (ISP). Also, the energy consumption of the shortest path from each IoT device $n \in \mathbb{N}$ to r can be acquired by r. Afterward, each MEC server r obtains the energy consumed on the aggregation path of each n aggregated on r by summing up 1) the transmission energy of n, 2) the computation energy on the cloud and MEC servers, and 3) the energy consumption of the shortest path from n to r and from r to the cloud server. Finally, the cloud and MEC servers propagate the energy consumption of each n to the MEC server managing n and the MEC server then sets the energy cost ϕ_n to n's minimum energy consumption. After each MEC server obtains the energy cost of all its managed IoT devices, D-EICGS goes to the D-CGC phase.

2) D-CGC

D-CGC forms a collaboration group for each request, where the members in a group need to be connected in the SIoT and the communication cost of the group is minimized. In this phase, a collaboration group for a request q is locally constructed by an MEC server, which IoT devices cover some locations of q. More specifically, the cloud server first broadcasts the information of each request $q \in \mathbb{Q}$ to all the MEC servers. Then, each MEC server checks if the managed IoT devices cover the corresponding locations of the request

q and propagates the result to the cloud server. Next, the cloud server notifies the MEC servers, which manage the devices related to the request q, of their identifications. For the sake of load balance, the notified MEC servers compare their computation loads and elect the one with the minimal loads as the *manager* to build a collaboration group for the request q. On the other hand, the notified MEC servers other than the manager send the SIoT, energy cost, and coverage information of devices covering L_q to the manager. Afterward, the manager selects devices to construct a collaboration group for the request q by the proposed rooted algorithm in Section VI. Each MEC server notifies the cloud server after finishing its computation and D-EICGS goes to the D-CT phase.

3) D-CT

In this phase, the cloud server generates a set of assist locations for each physical location since it is in charge of receiving user requests, while each MEC server locally determines the coverage on assist locations for collaboration groups and IoT devices. More specifically, for each location $k \in \mathbb{K}$, the cloud server creates an assist location a_q^k if k is related to the request q. Then, the cloud server broadcasts the assist location set $\mathbb{A} = \{a_q^k | q \in \mathbb{Q}, k \in \mathbb{K}\}$ to all MEC servers. Afterward, each MEC server locally manages the coverage of the devices and collaboration groups that it creates in the D-CGC phase. For each collaboration group g dealing with the request q, the MEC server creating the group g sets the coverage of g as $\hat{C}_g = \{a_q^k | \forall k \in L_q\}$, while the coverage of each device n is set as $\hat{C}_n = \{a_q^k | \forall k \in C_n, q \in \mathbb{Q}\}$. D-EICGS goes to the D-DGS phase after the coverage of each device and each collaboration group is determined.

4) D-DGS

In this phase, MEC servers collaboratively select IoT devices and collaboration groups to cover the assist locations generated in the D-CT phase such that the total energy consumption in GSIoT-MEC can be minimized. More specifically, each MEC server first calculates the cost-coverage ratio of its managed collaboration groups and IoT devices based on (11). Then, each MEC server broadcasts the cost-coverage ratio of the managed groups and devices to the other MEC servers, and therefore every MEC server knows the one with the minimum cost-coverage ratio (assume that it is the device n). Next, the MEC server managing the device n broadcasts the coverage of n to the other MEC servers and each one updates the uncovered assist location set accordingly. Note that all the MEC servers have received the information of each assist location corresponding to the requests from the cloud server in D-CT. The above steps repeat until all the assist locations are covered.

B. Dynamic User Request

In EICGS, we select IoT devices and collaboration groups to fulfill static user requests by ensuring that all the assist locations are covered. In practice, some new requests may come and some existing requests may stop after being serviced. In the following, we further extend EICGS to deal with dynamic requests (i.e., arriving requests and serviced requests).

1) Arriving request

For arriving requests, EICGS first checks each of them by comparing corresponding locations with existing requests and categorizes these arriving requests into three cases: 1) the requests that are the same as some of existing requests, 2) the requests with the locations overlapped with some existing requests, 3) the completely new requests.

For the first case, EICGS does not further select any individual IoT device and collaboration group since the requests can be fulfilled by the existing device and collaboration group selection. For the second case, EICGS first finds a collaboration group for each arriving request by CGC. Then, EICGS identifies the uncovered locations corresponding to these requests (i.e., the locations that are not covered by the currently selected individual devices). Next, EICGS runs the CT phase to generate a set of assist locations for each uncovered location and replace them with their corresponding assist locations.[16] Finally, EICGS runs DGS for these unsatisfied requests until all assist locations are covered. For the last case, a similar process to the second case is applied except that EICGS does not identify the uncovered locations before CT phase since all the locations corresponding to the new requests are not covered.

2) Serviced request

Since our goal is to minimize the total energy consumption, EICGS iteratively removes the individual IoT device or collaboration group with maximum energy cost for serviced requests, while the coverage requirements of non-serviced requests still need to be satisfied. Specifically, EICGS first identifies the IoT devices and collaboration groups, which is in charge of the serviced requests, and sorts them according to their energy costs from large to small. Then, EICGS starts from the device (collaboration group) with the maximum energy cost and sequentially checks whether the coverage requirements of other non-serviced requests are still satisfied if the considered device or group is removed. If any non-serviced request is unsatisfied, EICGS skips the considered device (collaboration group) and goes to the next one; otherwise, EICGS removes it. EICGS stops when all the devices and groups related to the serviced requests have been checked and none of them can be removed anymore.

VIII. Simulation

A. Simulation Setup

We conduct extensive simulations implemented in Java to evaluate the performance of EICGS in two real networks: INS IXC Services and PalmettoNet [43]. INS IXC Services includes 33 switches and 41 links, while PalmettoNet contains 45 switches and 70 links. To construct the GSIoT-MEC topology, the cloud server is randomly associated to a switch. Next, we randomly choose some other switches and associate one BS to each of them [44], where the default number of BSs is 19. The MEC servers are generated and connected to some randomly selected BSs or their nearby switches,

[16]Note that we only generate assist locations for arriving requests here.

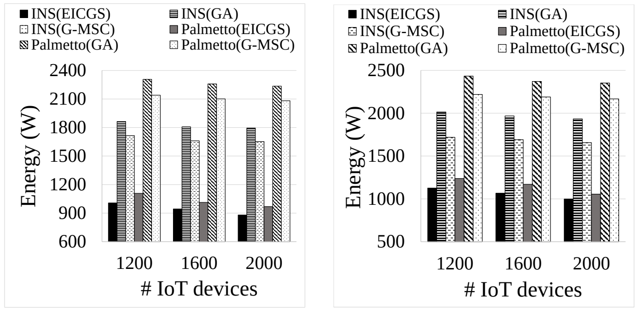
where the maximum distance between an MEC server and the nearest BS is at most 2 hops [2], [45]. To ensure that each location is covered by at least one IoT device, a device is first deployed for each spatial location of a real camera map of Baltimore [46], and other devices are then randomly distributed over the whole area [47]. The default number of independent user requests is set as 200, and each request is randomly associated to an area of locations. Without loss of generality, each request contains at most 15 locations, and each location is included in at most 10 requests. For the energy consumption, the line rate of switches is set as 1 Gbps, where the idle power and transmission power are 67.7 mW and 913 mW [48], respectively. The transmission energy of general IoT devices and cameras are set as 200 mW [49] and 460 mW [50], respectively. The energy parameter α is set as 5, while the idle power and unit computational power of servers, including the cloud and MEC servers, are set as 22 W and 200 mW, respectively [51].

We consider two SIoT scenarios: 1) traffic monitoring and 2) smart home, which examples are described in Section I. For the traffic monitoring, the collaboration parameter γ is set to 1 due to more complicated collaboration between devices (i.e., cameras extract and share video information with each other) [7], [10], while γ is set to 0.02 for the smart home since sensory data is more simple than video for processing (i.e., the values of temperature, humidity level, or etc.) [11]. The social relationship of the traffic monitoring is established based on the locations of devices (i.e., cameras on the same street with co-location relationship), while the relationship in the smart home is more complicated due to a variety of devices (e.g., temperature, humidity, or etc.) with various social relationships (e.g., co-location or co-work relationships) [11]. Without loss of generality, the average default node degree in the SIoT is set to 30 and ranges from 30 to 60 for the smart home.

Since there is no related work exploring GSIoT-MEC, we compare EICGS with Greedy Allocation (GA) [52] and Greedy-Maximum Set Cover (G-MSC) [53] to select IoT devices, and the shortest path routing is employed to connect the cloud server and the selected devices in GA and G-MSC. To evaluate EICGS, we vary the following parameters: 1) number of devices, 2) number of MEC servers, 3) compression ratio β, 4) collaboration parameter γ, and 5) edge density of an SIoT. The number of devices varies from 1200 to 2000, while the number of MEC servers ranges from 2 to 8. The default compression ratio is set to 50% and varies from 40% to 70% [54]. We measure the following three performance metrics: 1) energy consumption, 2) number of collaboration groups, 3) service latency, and 4) running time. Each simulation result is averaged over 200 samples.

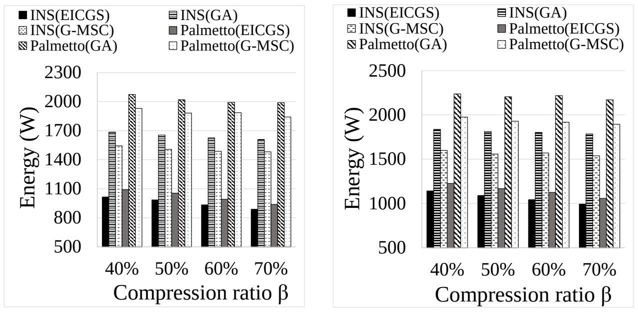
B. Simulation Result

Fig. 5 investigates the impacts of different numbers of IoT devices on energy consumption under different SIoT scenarios. The numbers of BSs, MEC servers, locations, and requests are 19, 16, 1500, and 200, respectively. EICGS outperforms GA and G-MSC in both networks since more devices with



(a) Smart home (b) Traffic monitoring

Fig. 5: Performance of different algorithms for various numbers of IoT devices under different SIoT scenarios.



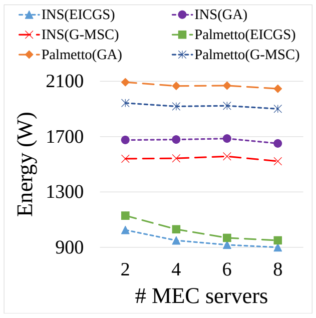
(a) Smart home (b) Traffic monitoring

Fig. 6: Performance of different algorithms for various compression ratios under different SIoT scenarios.

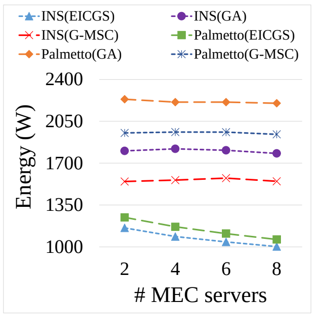
better cost-coverage ratios are being selected. On the other hand, GA and G-MSC do not jointly consider the energy consumption of data delivery and processing in the cloud server, MEC servers and SIoT, and therefore consume much more energy. Moreover, the opportunity to reduce the energy consumption via collaboration groups is also missing in GA and G-MSC. Comparing Fig. 5(a) with Fig. 5(b), the energy is consumed more in the traffic monitoring scenario due to 1) the larger transmission energy of cameras and 2) more complicated collaboration between cameras in the SIoT.

Fig. 6 compares the energy consumption with different compression ratios. The numbers of IoT devices, locations, and requests are fixed at 1200, 1000, and 200, respectively. When the compression ratio increases, the energy consumption decreases significantly. This is because EICGS exploits the data processing on MEC servers to considerably reduce the amount of data forwarding in the networks. On the other hand, the energy consumption of GA and G-MSC remains high since the algorithms ignore the opportunity of data processing and simply employ the shortest path routing, where an MEC server might not be included on the path. From Figs. 5 and 6, EICGS effectively reduces about 44% to 60% energy consumption in both networks.

Fig. 7(a) shows the impact of different numbers of MEC servers on the energy consumption. G-MSC only slightly outperforms GA since G-MSC selects fewer IoT devices to

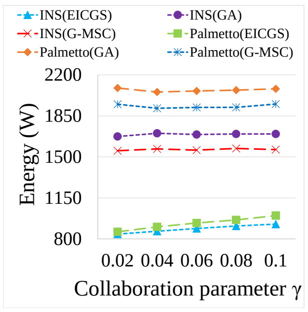


(a) Smart home

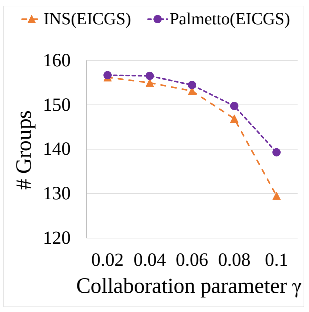


(b) Traffic monitoring

Fig. 7: The impact of various numbers of MEC servers on energy consumption under different SIoT scenarios.



(a) Energy consumption

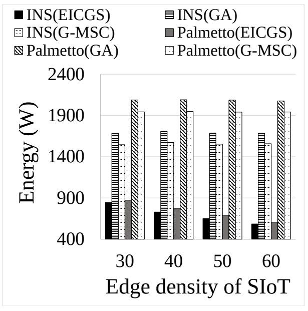


(b) Number of collaboration groups

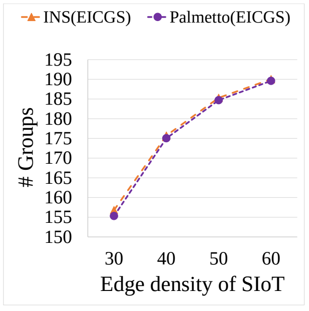
Fig. 8: The impact of collaboration parameter γ on energy consumption and the number of collaboration groups at smart home.

ensure the coverage, and the total transmission energy is reduced. Moreover, since GA and G-MSC do not consider the data processing in both MEC servers and SIoT devices, the energy consumption remains high and only slightly decreases when the number of MEC servers increases. On the other hand, EICGS effectively reduces the energy consumption by more than 50% in PalmettoNet and 40% in INS IXC Services. It implies that data aggregation on MEC servers is more crucial when the end-to-end routing paths become longer.

Fig. 8 shows the impact of different collaboration parameters on energy consumption and the number of collaboration



(a) Energy consumption



(b) Number of collaboration groups

Fig. 9: The impact of edge density of SIoT on energy consumption and the number of collaboration groups at smart home.

TABLE I

Latency and Running Time (unit: ms)

	100	200	300	400
Latency (EICGS-INS)	64.8	65.6	64.1	64.2
Latency (GA-INS)	355.6	365.7	358.6	356.9
Latency (G-MSC-INS)	350.8	360.1	344.1	360
Latency (EICGS-Palmetto)	74.2	71.7	71.7	73.3
Latency (GA-Palmetto)	448.8	449.3	443.8	454.3
Latency (G-MSC-Palmetto)	449.2	446.5	444.2	448.3
Running time (EICGS-INS)	163.2	326.8	498.6	698.7
Running time (EICGS-Palmetto)	169.6	328.5	509.9	714.6

groups at smart home. When the collaboration parameter γ increases, fewer groups are selected because EICGS tends to encourage data processing in the cloud server when the energy cost from collaboration groups increases. However, the energy consumption only slightly increases due to more data transmitted to the cloud server, which shows that the proposed algorithm can effectively reduce the energy even though the collaborative energy consumption between smart devices is high. Fig. 9 investigates the impact of edge density of SIoT on energy consumption and the number of collaboration groups for EICGS. As the figure shows, EICGS significantly outperforms both GA and G-MSC since EICGS carefully examines the social community and provides data aggregation to reduce the total energy consumption in GSIoT-MEC. When the density of SIoT increases, EICGS has more opportunity to benefit from social collaboration of IoT devices since each device can find more members with smaller weights. Figs. 8 and 9 point out that the energy consumption considerably decreases with more collaboration groups in the SIoT. In general, EICGS effectively reduces more than 50% energy and the results manifest that SIoT and MEC play important roles in GSIoT-MEC.

C. Performance of D-EICGS

To evaluate the IoT latency and running time of D-EICGS, we conduct the simulation on a server with four Intel Xeon E7-8870 v4 2.1GHz CPUs and 1080G memory. D-EICGS is implemented in Java with multi-thread, where each thread is used to simulate the calculation on an MEC server for distributed computation. Based on the statistic data shown in [55], we set the link delay from 10 ms to 100 ms. The numbers of IoT devices, locations, and MEC servers are fixed at 1200, 1000, and 16, respectively. We define the latency of a request as follows. For each request, if it is handled by the cloud server, the latency is dominated by the IoT device, which covers the corresponding locations of the request, with the largest transmission delay. On the other hand, if it is handled by a collaboration group, the latency is the transmission delay of the representative device. In Table I, we evaluate the IoT latency and the running time of D-EICGS under different numbers of independent requests from 100 to 400. The result manifests that D-EICGS guarantees acceptable latency (i.e., less than 300 ms for general IoT traffic [56]) in both networks with slightly increasing running time in a larger network. On the other hand, GA and G-MSC cause more latency since they do not exploit SIoT computation and the latency is always dominated by the device with the largest delay.

IX. Conclusion

With the development of AI on chips, SIoT has recently grasped further attention since a group of smart devices can collaboratively identify and handle local events without the help of cloud servers. In addition, minimizing the energy consumption is crucial since ICTs are expected to consume about 14.57% of the world's total electricity in 2020. To the best of our knowledge, this paper makes the first attempt to minimize the energy consumption in GSIoT-MEC by jointly considering traffic engineering, big data processing, and the collaboration of devices. We formulate the ICGDSP and prove that the problem is NP-hard. To efficiently solve the problem, we devise an approximation algorithm that selects a set of collaboration groups and a set of individual devices transmitting the information required for handling user requests by carefully examining the SIoT community and data processing on MEC servers. Simulation results demonstrate that the energy consumption can be reduced by more than 50% in both real networks. In the future, we will investigate the emphasized problem by considering different SIoT interaction models and the dynamic social relationships, where the social relationships between each IoT device may evolve.

References

[1] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches." *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[2] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1628–1656, third quarter 2017.

[3] Nokia: multi-access edge computing. [Online]. Available: https://networks.nokia.com/solutions/multi-access-edge-computing

[4] Nokia: mobile edge computing. [Online]. Available: http://resources.alcatel-lucent.com/asset/200546

[5] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab, "Edge computing enabling the internet of things," in *IEEE World Forum on Internet of Things (WF-IoT)*, Dec 2015.

[6] X. Sun and N. Ansari, "EdgeIoT: mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, Dec 2016.

[7] M. Lippi, M. Mamei, S. Mariani, and F. Zambonelli, "Coordinating distributed speaking objects," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, Dec 2017.

[8] M. Endler, J. P. Briot, F. S. E. Silva, V. P. d. Almeida, and E. H. Haeusler, "An approach for real-time stream reasoning for the internet of things," in *IEEE International Conference on Semantic Computing (ICSC)*, Jan 2017.

[9] H. Agrawal, S.-w. Leigh, and P. Maes, "L'evolved: autonomous and ubiquitous utilities as smart agents," in *ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, Sept 2015.

[10] AI traffic system in Pittsburgh. [Online]. Available: https://goo.gl/ybHqB3

[11] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (SIoT) - when social networks meet the internet of things: concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594–3608, Nov 2012.

[12] A. M. Ortiz, D. Hussein, S. Park, S. N. Han, and N. Crespi, "The cluster between internet of things and social networks: Review and research challenges," *IEEE Internet of Things Journal*, vol. 1, no. 3, pp. 206–215, June 2014.

[13] H. Z. Asl, A. Iera, L. Atzori, and G. Morabito, "How often social objects meet each other? analysis of the properties of a social network of iot devices based on real data," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2013.

[14] C.-M. Huang, C.-H. Shao, S.-Z. Xu, and H. Zhou, "The social internet of thing (S-IOT)-based mobile group handoff architecture and schemes for proximity service," *IEEE Transactions on Emerging Topics in Computing*, vol. 5, no. 3, pp. 425–437, July-Sept 2017.

[15] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social internet of things," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1253–1266, May 2014.

[16] INPUT. [Online]. Available: https://goo.gl/AtZQJa

[17] DOMUS - technological district on home automation. [Online]. Available: https://goo.gl/r3atR3

[18] Demanes. [Online]. Available: http://www.demanes.eu/

[19] S. Nastic, S. Sehic, D. H. Le, H. L. Truong, and S. Dustdar, "Provisioning software-defined IoT cloud systems," in *IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2014.

[20] M. Kovatsch, M. Lanter, and Z. Shelby, "Californium: scalable cloud services for the internet of things with CoAP," in *IEEE International Conference on the Internet of Things (IOT)*, Oct 2014.

[21] M. S. Khan, M. S. Islam, and H. Deng, "Design of a reconfigurable RFID sensing tag as a generic sensing platform toward the future internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 300–310, Aug 2014.

[22] W. Vereecken, W. V. Heddeghem, D. Colle, M. Pickavet, and P. Demeester, "Overall ICT footprint and green communication technologies," in *IEEE International Symposium on Communications, Control and Signal Processing (ISCCSP)*, Mar 2010.

[23] F. K. Shaikh, S. Zeadally, and E. Exposito, "Enabling technologies for green internet of things," *IEEE Systems Journal*, vol. 11, no. 2, pp. 983–994, June 2017.

[24] C. Perera, A. Zaslavsky, C. H. Liu, M. Compton, P. Christen, and D. Georgakopoulos, "Sensor search techniques for sensing as a service architecture for the internet of things," *IEEE Sensors Journal*, vol. 14, no. 2, pp. 406–420, Feb 2014.

[25] M. Shan, G. Chen, D. Luo, X. Zhu, and X. Wu, "Building maximum lifetime shortest path data aggregation trees in wireless sensor networks," *ACM Transactions on Sensor Networks*, vol. 11, no. 1, pp. 11:1–11:24, July 2014.

[26] Y. Wu, S. Fahmy, and N. B. Shroff, "On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-completeness and approximation algorithm," in *IEEE Conference on Computer Communications (INFOCOM)*, Apr 2008.

[27] J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin, "Software-defined internet of things for smart urban sensing," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 55–63, Sept 2015.

[28] Y. Jararweh, M. Al-Ayyoub, A. Darabseh, E. Benkhelifa, M. Vouk, and A. Rindos, "SDIoT: a software defined based internet of things framework," *Journal of Ambient Intelligence and Humanized Computing*, vol. 6, no. 4, pp. 453–461, June 2015.

[29] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "Ubiflow: mobility management in urban-scale software defined IoT," in *IEEE Conference on Computer Communications (INFOCOM)*, Apr 2015.

[30] D. Hussein, S. Park, S. N. Han, and N. Crespi, "Dynamic social structure of things: a contextual approach in CPSS," *IEEE Internet Computing*, vol. 19, no. 3, pp. 12–20, May 2015.

[31] T. Allard and S. Shekh, "Hierarchical multi-agent distribution planning," in *ACM Australasian Joint Conference on Advances in Artificial Intelligence (AI)*, Dec 2012.

[32] M. Tahir and R. Farrell, "Optimal communication-computation tradeoff for wireless multimedia sensor network lifetime maximization," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr 2009.

[33] S. J. Baek, G. de Veciana, and X. Su, "Minimizing energy consumption in large-scale sensor networks through distributed data compression and hierarchical aggregation," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 6, pp. 1130–1140, Aug 2004.

[34] J. Ammer and J. Rabacy, "The energy-per-useful-bit metric for evaluating and optimizing sensor network physical layers," in *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Sept 2006.

[35] C. Mbius, W. Dargie, and A. Schill, "Power consumption estimation models for processors, virtual machines, and servers," *IEEE Transactions*

on Parallel and Distributed Systems, vol. 25, no. 6, pp. 1600–1614, June 2014.

[36] P. Mahadevan, P. Sharma, S. Banerjee, and P. Ranganathan, “A power benchmarking framework for network devices,” in *ACM International IFIP-TC 6 Networking Conference*, May 2009.

[37] E. Halperin and R. Krauthgamer, “Polylogarithmic inapproximability,” in *ACM Symposium on Theory of Computing (STOC)*, June 2003.

[38] M. Lippi, M. Mamei, S. Mariani, and F. Zambonelli, “An argumentation-based perspective over the social iot,” *IEEE Internet of Things Journal*, vol. PP, no. 99, pp. 1–1, 2017.

[39] Q. Wang, M. Hempstead, and W. Yang, “A realistic power consumption model for wireless sensor network devices,” in *IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Sept 2006.

[40] J. Moy, *OSPF Version 2*, RFC 2178, Internet Engineering Task Force, July 1998.

[41] C.-H. Wang, J.-J. Kuo, D.-N. Yang, and W.-T. Chen, “Green social internet of things with data processing in mobile edge networks (full version),” *NTHU, Technical Report*, Dec 2017. [Online]. Available: http://mnet.cs.nthu.edu.tw/GSIoT-MEC-TechRep2017.pdf

[42] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, “Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges,” *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, Apr 2017.

[43] The internet topology zoo. [Online]. Available: http://www.topology-zoo.org/dataset.html

[44] C. S. Chen, V. M. Nguyen, and L. Thomas, “On small cell network deployment: a comparative study of random and grid topologies,” in *IEEE Vehicular Technology Conference (VTC Fall)*, Sept 2012.

[45] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, “A survey on mobile edge networks: convergence of computing, caching and communications,” *IEEE Access*, vol. 5, no. 99, pp. 6757–6779, Mar 2017.

[46] Camera map in baltimore. [Online]. Available: https://catalog.data.gov/dataset/cctv-cameras

[47] C. H. Wang, D. N. Yang, and W. T. Chen, “Scheduling for multi-camera surveillance in lte networks,” in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2015.

[48] T. M. Nam, N. H. Thanh, N. Q. Thu, H. T. Hieu, and S. Covaci, “Energy-aware routing based on power profile of devices in data center networks using SDN,” in *IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, June 2015.

[49] J. Markkula and J. Haapola, “LTE and hybrid sensor-LTE network performances in smart grid demand response scenarios,” in *IEEE International Conference on Smart Grid Communications (SmartGridComm)*, Oct 2013.

[50] Y. Wang, D. Wang, X. Zhang, J. Chen, and Y. Li, “Energy-efficient image compressive transmission for wireless camera networks,” *IEEE Sensors Journal*, vol. 16, no. 10, pp. 3875–3886, May 2016.

[51] A. Krioukov, P. Mohan, S. Alspaugh, L. Keys, D. Culler, and R. Katz, “Napsac: design and implementation of a power-proportional web cluster,” *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 1, pp. 102–108, Jan 2011.

[52] S. He, D. H. Shin, J. Zhang, J. Chen, and Y. Sun, “Full-view area coverage in camera sensor networks: dimension reduction and near-optimal solutions,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 9, pp. 7448–7461, Sept 2016.

[53] M. Cardei, M. T. Thai, Y. Li, and W. Wu, “Energy-efficient target coverage in wireless sensor networks,” in *IEEE Conference on Computer Communications (INFOCOM)*, Mar 2005.

[54] A. Reinhardt, M. Hollick, and R. Steinmetz, “Stream-oriented lossless packet compression in wireless sensor networks,” in *IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2009.

[55] Y. Xu, C. Yu, J. Li, and Y. Liu, “Video telephony for end-consumers: measurement study of google+, ichat, and skype,” *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 826–839, June 2014.

[56] J. M. Liang, J. J. Chen, H. H. Cheng, and Y. C. Tseng, “An energy-efficient sleep scheduling with QoS consideration in 3GPP LTE-advanced networks for internet of things,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 1, pp. 13–22, Mar 2013.