Competition-Based Participant Recruitment for Delay-Sensitive Crowdsourcing Applications in D2D Networks

Yanyan Han, *Student Member, IEEE*, Tie Luo, *Member, IEEE*, Deshi Li, and Hongyi Wu, *Member, IEEE*

Abstract—Device-to-Device (D2D) networks impose a significant challenge on delay-sensitive crowdsourcing due to the highly non-deterministic and intermittent network connectivity. Under this setting, the paper investigates a participant recruitment problem in which an initial set of recruited nodes, which we call *seeds*, need to make an optimal decision on what other nodes to recruit to perform the crowdsourcing task. These seeds face the dilemma that recruiting more nodes increases their own payment but on the other hand also increases the risk of being excluded from the crowdsourcing task. As a first attack to this problem, we propose a dynamic programming algorithm. However, it is a centralized solution and hence the practicality is compromised. Therefore, we introduce two distributed alternatives. One is based on the *divide-and-conquer* paradigm by first partitioning a network into a set of opportunistic Voronoi cells and then running an optimization algorithm in each cell. The other is a *task-splitting scheme* which recursively delegates the recruitment task to newly joined nodes. We implemented our proposed solutions on an Android-based prototype and built a testbed using 25 Dell Streak tablets. Our experiments which lasted for 24 days demonstrate that the distributed schemes approximate the theoretical optimum with affordable complexity. Moreover, we conducted simulations with a much larger scale and more diverse settings. The simulation results corroborate the experimental data and confirm that our proposed distributed solutions closely approach the performance of the centralized solution while satisfying the optimization goal under different network configurations.

Index Terms—Delay-constrained, single-copy multi-path, optimization, prototype, opportunistic network

1 Introduction

Recent years we have witnessed the remarkable proliferation of intelligent wireless devices and the rapid growth of mobile-broadband services such as ultra-high-resolution video streaming, data sharing and synchronization, and virtual and augmented reality that continue driving the demand for higher consumer data rates [1]. At the same time, the vast majority of today's wireless communications systems operate in the microwave spectrum below 3 GHz, which is experiencing severe shortage and has become a crowded and limited resource. Therefore, the millimeter wave (mmWave) band, operating at frequencies between 20 and 300 GHz, has been proposed for next-generation (5G) cellular systems. The massive underutilized mmWave spectrum provides great potential to support multiple gigabit-per-second user data rates and thousand-fold increase in total mobile broadband data. However, the use of mmWave band brings a new set of technical challenges including low diffraction, weak reflection, limited Non-Line-of-Sight distance, and coverage holes.

- *Y. Han and H. Wu are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504. E-mail: {yxh0499, wu}@cacs.louisiana.edu.*
- *T. Luo is with I2R, A*STAR, Singapore 138632. E-mail: luot@i2r.a-star.edu.sg.*
- *D. Li is with the School of Electronic Information, Wuhan University, Wuhan, China 430079. E-mail: dsli@whu.edu.cn.*

Manuscript received 25 Mar. 2015; revised 28 Dec. 2015; accepted 20 Jan. 2016. Date of publication 3 Feb. 2016; date of current version 31 Oct. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2016.2524590

To this end, Device-to-Device (D2D) communication has been identified as an effective complementary solution to address these challenges, by utilizing the short-range (e.g., license-free Wi-Fi or licensed mmWave) wireless links, to establish opportunistic connections between mobile users for data delivery. In such mobile opportunistic D2D networks, the endpoints (i.e., the source and destination) are not always continuously connected. As a matter of fact, the network is generally disconnected, while most nodes communicate with each other only occasionally. In order to facilitate data transfer, the nodes adopt a store-and-forward mechanism to gradually forward data across the network.

The D2D networks will not replace the infrastructure-based B2D (i.e., base-station-to-device) communication, since it is obviously incapable to support general communication needs of mobile users (especially for real-time voice and data delivery). However, it does have a niche (complementing to the conventional B2D infrastructure) in some application settings, such as crowdsourcing as to be discussed next.

1.1 Crowdsourcing in D2D

Crowdsourcing is emerging as a new data-collection, solution-finding, and opinion-seeking model that obtains needed services, ideas, or content by soliciting contributions from a large crowd of public participants. Recent examples of crowdsourcing range from web-based platforms such as Amazon Mechanic Turk, Microworkers, Kickstarter, and TaskCN, to popular mobile apps including TaskRabbit, Placemeter, Weather Signal, Wave, and Blablacar to name a few.

Crowdsourcing does not depend on any specific underlying network. But D2D based crowdsourcing is particularly

desired when the initiator cannot directly reach out to the participants or the conventional approaches for data transportation are costly. First, there are scenarios where infrastructure-based wireless networks (including cellular and Wi-Fi) are unavailable, and thus we have to completely rely on mobile opportunistic D2D to announce the crowdsourcing task and to transport the requested data. Even in a developed country (such as the United States), vast rural areas (including many popular national parks) are not covered by wireless infrastructure. Consider a scenario in Yellowstone National Park, where the rangers are trying to locate and rescue an elk that has been reported with serious wound. The ranger office can initiate a crowdsourcing task, requesting the participants to provide photos of suspected animals. The mobile opportunistic D2D network is perhaps the best way to carry out such crowdsourcing.

Second, although most non-rural areas are covered by cellular systems, the cost and bandwidth often become the bottleneck for achieving effective crowdsourcing. Wi-Fi is a good alternative. However, the Wi-Fi coverage in most small cities, towns, and villages in US (and around the world) is still very limited. Consider a crowdsourcing task to collect 1-minute video clips of traffic congestion around the world, where each video clip is about $20MB$. Given its nature of long-term, large-scale data gathering with low QoS requirement (i.e., tolerance for relatively long delay and low reliability), it obviously prefers the communication network with the lowest cost if multiple options are available. To this end, when crowdsourcing is introduced in cellular networks, it is more desirable to operate over D2D rather than regular B2D channels, for the sake of saving communication cost of mobile participants, service providers and the end-user who initiates the crowdsourcing task (i.e., the initiator), especially when the task involves a vast number of participants and a massive volume of data for a long period of time.

Moreover, we would also like to point out that our system model and proposed solution (as to be introduced next) are generally applicable to networks with hybrid mobile and static nodes. In particular, the nodes in D2D do not have to be mobile, and our problem formulation can generally cover the scenario with fixed wireless APs or other infrastructure nodes. Such static node can be treated as a D2D node with zero mobility. Since there exist many other mobile nodes, they can still establish opportunistic links. For example, just like a mobile node, a fixed AP can forward data to/from other mobile nodes when they pass by. If multiple static nodes are connected to each other in a cluster or to the infrastructure via stable links, they can be merged and treated as a single virtual node if we ignore the delay which is much shorter than the delay over opportunistic links. Such a virtual node has intermittent connectivity with other mobile nodes, and can serve as either a seed or normal participator for different applications. Obviously, this virtual node often has more contacts with other nodes since it consists of a set of physical nodes (i.e., APs). If a mobile node has constant connectivity with the APs, it is considered as part of the virtual node as well. In a general case, there may be multiple sets of well-connected nodes (including APs and/or mobile nodes). Each of them is treated as a virtual node. With such virtual nodes in the network model, the proposed solution applies in the same way. So there is essentially no difference between the static and mobile nodes and thus we do not need to differentiate them.

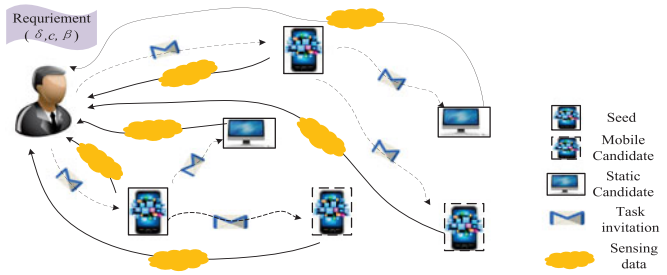


Fig. 1. An example scenario for quota based data delivery.

Although crowdsourcing has been extensively discussed in recent years, the marriage of crowdsourcing and D2D creates new, interesting research problems, mainly due to the unique non-deterministic setting in D2D. In particular, incentive is playing a pivotal role to enable large-scale crowdsourcing applications by attracting sufficient participation. Since the crowdsourcing participants need to consume their resources (such as battery and computing power, storage space, and communication bandwidth) and are subject to undesired risks (e.g., potential exposure to privacy threats when sharing their data), the mobile users are often reluctant to participate in crowdsourcing, unless they are compensated with satisfactory rewards [2]. To this end, a variety of incentive models have been developed recently [3], [4], [5], [6], [7] to stimulate the collaboration between nodes. These models, however, are not applicable in D2D. Given the non-deterministic nature of D2D networks, it is often impossible for the crowdsourcing initiator to collect sufficient information from potential participants or for individual participants to negotiate with the initiator directly, precluding the use of a class of well-studied solutions based on auction or game models [8], [9], [10], [11], [12], [13]. In this work, we propose to formulate a practical and interesting crowdsourcing problem in D2D networks.

1.2 Problem Overview

There are a class of mobile applications that involve large-scale data gathering from individual mobile devices. For example, assume a crowdsourcing initiator aims to collect data to study noise pollution experienced by the residents of a city. So she announces a crowdsourcing task, asking for noise pollution data gathered by the microphones on mobile handsets (as depicted in Fig. 1). Note that crowdsourcing applications often require certain coverage or granularity of the data and must receive the data within a given time window (or delay budget). For instance, the initiator may desire to collect noise pollution measurement from 0.1 percent of the total one million residents in the city (or 1,000 samples) for a period of one week starting from the time of announcement.

An incentive is offered by the initiator to stimulate broad participation of the crowdsourcing task. More specifically, the initiator announces the task along with a minimum payment per participant. In fact, many traditional data acquisition mechanisms are based on a similar approach. For instance, in traditional surveys (which can be considered as a sort of crowdsourcing), the initiator often provides some fixed incentives (such as monetary reward or coupons) to each participant. Likewise, fixed

incentives are usually announced to recruit participant for medical trials. In D2D, the initiator often has limited ability to reach out to a large number of mobile nodes directly. As an incentive to complete high-quality crowdsourcing that achieves the desired data coverage, the initiator can offer higher per-person payment if more nodes (toward 0.1 percent of the total population) join the crowdsourcing task, such that a node will not only participate by itself but also help recruit other participants. However, if more samples (than the desired 0.1 percent of the total population) are received, the surplus samples are not useful for the initiator and thus will not be paid.

At a given instance of the time window after the announcement is made, assume a small set of nodes have already joined the crowdsourcing task. They are called seeds. The seeds may learn the crowdsourcing task via various means, e.g., by scanning a Q-R code posted by the initiator on printouts or TVs or by D2D-based electronic announcement. Note that when a seed joins the crowdsourcing effort, it does not always have a direct connection with the initiator. So even it begins immediately to sense the requested data and deliver them to the initiator, it often takes a long and nondeterministic delay before the data are received by the initiator. The delay depends on the distance between them and the availability of opportunistic links. Whenever the initiator receives data from a participant, it immediately issues a voucher to the latter. The actual payment will be made to the corresponding participant by the end of the time window (e.g., by mailing a check to the participants or depositing the fund to their bank account).

Under the aforementioned incentive scheme, the seeds are obviously motivated to recruit other nodes to participate in the crowdsourcing. But at the same time, the initiator has predetermined a desired number of samples and/or is under a fixed budget. Therefore, she can pay up to a given number of participants only (i.e., recruitment quota). In other words, the initiator essentially chooses the payees in a first-come-first-serve manner, but the actual payment per person depends on the total number of participants. If many newly recruited nodes deliver data to the initiator before the seeds do, then some seeds are at the risk of losing their payment. Note that, although the seeds generally begin their crowdsourcing effort before the newly recruited nodes, there is no guarantee that the former can deliver data to the initiator earlier than the latter do, especially when the latter are closer to and have better connections with the initiator. The problem is further complicated since the recruiting is performed through opportunistic links and thus the accurate number of recruited nodes is often unknown by the seeds.

In this work, we formulate the problem from the perspective of the seeds, which face the dilemma of how to carefully invite additional participants in order to maximize their gain while keeping the risk of losing their payment low.[1] The detailed problem formulation will be introduced in Section 2.

1. Multiple crowdsourcing tasks may be initiated by different nodes in the network. They can be treated separately. At the same time, a node may serve as an initiator, or a seed, or a participant in different crowdsourcing tasks.

1.3 Contributions

This is the first work that investigates a realistic incentive mechanism for participant recruitment for delay-sensitive data crowdsourcing in D2D networks. The problem is unique due to the opportunistic network setting and the competition among mobile nodes. Since seeds intend to maximize their own benefit, the general principle is to recruit the qualified participants that are able to deliver data to the initiator within a delay budget (e.g., one week given in the above example), but at the same time do not create threats to the seeds. Besides this offline problem formulation, there is an online version of the problem, as once a node is recruited, it becomes one of the updated set of seeds and again needs to intelligently decide how to further recruit additional participants.

In this paper, we first formulate the offline problem from a centralized perspective, and then propose a dynamic programming algorithm to solve it. The centralized algorithm offers useful insights but are impractical to implement in real network settings due to high communication and computation cost. To this end, we propose two distributed alternatives. The first one is based on a divide-and-conquer approach by partitioning the network into opportunistic Voronoi cells and running an optimization algorithm in each cell. The second is a task-splitting scheme, tailored for the online version of the problem, which recursively delegates the recruiting responsibility to newly joined nodes. To evaluate the feasibility and efficiency of the proposed algorithms, we implemented a prototype based on Android and carried out experiments using 25 Dell Streak tablets for 24 days. Moreover, we also conducted extensive simulations in larger scales and more diverse settings than the experiment. Our results demonstrate that the proposed approaches approximate the overall optimization objective while satisfying the delay and penalty constraints.

The rest of the paper is organized as follows. Section 2 presents the problem formulation. Section 3 discusses a centralized dynamic programming algorithm. Section 4 describes two distributed solutions. Sections 5 and 6 discuss experimental and simulation results, respectively. Finally, Section 7 concludes the paper.

2 System Model and Problem Formulation

In a mobile opportunistic D2D network, the delays of the opportunistic links depend on nodal mobility and are random variables, denoted by T_{ij}, where i and j indicate Nodes i and j, respectively. The distribution of T_{ij} is general but known, which can be learnt by the mobile nodes in a distributed manner (as discussed in Section 4).

Assume an initiator announces a crowdsourcing request, aiming to recruit up to c participants within a time window. There are n nodes in the network that are interested in the crowdsourcing. The n nodes include $s \leq c$ seeds and $m = n - s$ other nodes. The seeds have received invitation and begin immediately to sense the requested data and deliver them to the initiator. As discussed earlier, it often takes a long and nondeterministic delay before the data can be received by the initiator. At the same time, the seeds initiate the recruiting process by inviting other nodes to join the crowdsourcing. In general, the initiator selects the nodes

that have provided the best service in the past or have close connectivity with other candidate nodes as seeds.[2] If historic information is unavailable, the seeds are often randomly chosen from a diverse set of nodes.

Since we intend to determine an optimal set of nodes to be invited to join the task, we define an $n \times 1$ vector X as the nodes recruiting strategy, where each element is a 0-1 variable to be optimized. If $X_i = 1$, an invitation will be sent to Node i; otherwise, Node i will not be invited. To keep the notation consistent, we set $X_i = 1$ for all seeds. Note that, $X_i = 1$ only means an invitation is sent to Node i but does not guarantee it succeeds in the competition. To secure the payment, it must be one of the first c nodes that deliver data to the initiator within a predefined delay budget δ.

Let R_X be the number of nodes that participate the crowdsourcing under X. In the opportunistic communication setting, R_X is a random variable. It is nontrivial to compute its distribution. We will discuss how to obtain it in the next section. The utility (i.e., the overall benefit of the seeds) is defined as a function of R_X. Its exact form is obviously subject to applications. Here we consider a general increasing function denoted by $f(\cdot)$, which can be, for example, $f(\cdot) = x$. The average utility is thus $\bar{f}_X = \sum_{x=1}^{c} f(x) Pr\{R_X = x\}$. Accordingly, the utility function is formulated as $U_X = \bar{f}_X$.

Let P_X denote the probability that at least one seed fails to join the task. A seed may lose the task due to two reasons. First, it takes longer than δ to deliver its data to the initiator. Second, its data reaches the initiator within δ after c or more nodes. Again, it is nontrivial to compute P_X because of the competition among nodes for becoming one of the final participants, as will be elaborated in the next section. The penalty can be a general increasing function of P_X. Without loss of generality, it is simply defined to be P_X.

Therefore the optimization problem is formulated as follows where β is the maximal tolerable penalty:

$$\begin{aligned} Maximize: &\quad U_X, \\ S.t.: &\quad P_X \leq \beta. \end{aligned} \tag{1}$$

Apparently, the goal is to maximize the utility while keeping the risk no greater than the maximal tolerable level. Notice that the optimization is from the perspective of seed nodes thus they could make the most beneficial decision to themselves.

3 Centralized Problem Formulation and Solution

This section investigates the problem from a centralized perspective. Although centralized solutions require global information, which are usually hard to obtain in real application settings, they could offer valuable insights into exploring the solution space and provide useful guidelines for developing distributed counterparts. This work focuses on discovering an optimal solution for the competition-based crowdsourcing problem in D2D, while assuming the existence of an underlying routing scheme, e.g., shortest path with the delivery probability within delay budget δ as the link weight, which finds a path between any two nodes and all nodes are cooperative in data forwarding. Routing in similar context have been studied extensively [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] and is out of the scope of this paper.

We first elaborate the details of a 0-1 nonlinear programming optimization for the centralized problem formulation introduced in Section 2, which yields optimal results but suffers high computation complexity. Then, we present a dynamic programming algorithm with reduced computation time.

3.1 0-1 Nonlinear Programming Optimization

While the problem formulated above appears simple, it is nontrivial to be solved, since the nondeterministic network setting dramatically increases the complexity to derive U_X and P_X. In the following discussion, we assume a deterministic underlying communication protocol, which generates a given path to deliver the invitation to a node (e.g., Node i) and to send data from Node i back to the initiator.[3] We also assume the average duration for message transmission over an available wireless communication link is negligible, in comparison with the time waiting for the communication opportunity. This assumption is valid since the messages involved in the node-recruiting process are all very short. The end-to-end delay of the path is a random variable, denoted by τ_i, where a path could include one or multiple hops. Its distribution is the convolution of the delay distributions of the links along the path. Since link delay distributions (e.g., T_{ij}) are given, we can readily compute the probability for the initiator to receive data from Node i within δ, i.e., $Pr\{\tau_i \leq \delta\}$.

Next, we first derive the distribution of R_X. Obviously, R_X ranges from 0 to c. We have

$$Pr\{R_X = x\} = \begin{cases} Y(x), & 0 \leq x < c \\ \sum_{y=c}^{n} Y(y), & x = c, \end{cases} \tag{2}$$

where

$$Y(z) = \sum_{i=1}^{\binom{n}{z}} \prod_{u \in \phi_i} Pr(\tau_u \leq \delta X_u) \times \prod_{v \in \overline{\phi}_i} Pr(\tau_v > \delta X_v). \tag{3}$$

$Y(z)$ shows the probability to have exactly z nodes successfully send data to the initiator within delay budget δ. In Eq. (3), we consider all possible combinations of selecting z out of total n nodes, i.e., $\binom{n}{z}$. Each combination is represented by a set ϕ_i, while $\overline{\phi}_i$ indicates the unselected $n - z$ nodes. Obviously, if a node in ϕ_i is not invited (i.e., $X_u = 0$), then $\prod_{u \in \phi_i} Pr(\tau_u \leq \delta X_u)$ is simply 0. Similarly, if a node in $\overline{\phi}_i$ is not invited (i.e., $X_v = 0$), then $Pr(\tau_v > \delta X_v) = 1$. Therefore, the term inside the summation of Eq. (3) shows the probability that all invited nodes in ϕ_i have their delays no greater than δ while the delays of other invited nodes are

2. The optimal seed selection is out of scope of this paper and may be studied in our future work.

3. If Node i is a seed, it already has the invitation, so the path is simply from itself to the initiator.

greater than δ. The entire equation gives the total probability that the initiator receives data from z out of n nodes within δ. Once $Pr\{R_X = x\}$ is known, U_X can be readily calculated as discussed in the previous section.

Second, we compute P_X. The complexity to derive P_X is due to the competition among nodes. More specifically, only the first c nodes that receive the invitation and deliver data to the initiator within δ can successfully secure the payment. In other words, the initiator ranks the nodes according to the times (i.e., delays) of receiving their data, and decides the payees and their payment after the process terminates (i.e., when the number of received data samples reaches c or δ expires).

Let ψ_j be a set that includes all seeds and up to $c-s$ additional nodes. Each ψ_j has a probability of P_{ψ_j} to be ranked before other nodes. The size of ψ_j may vary from s to c. Under each size, we consider all possible combinations of ψ_j. Thus the summation of all P_{ψ_j} yields the probability that the seeds can successfully secure their payments, i.e.,

$$P_{suc} = \sum_{i=s}^{c} \sum_{j=1}^{\binom{n-s}{i-s}} P_{\psi_j}. \tag{4}$$

Accordingly, we have $P_X = 1 - P_{suc}$.

The probability P_{ψ_j} in the above equation can be straightforwardly formulated as follows:

$$P_{\psi_j} = \sum_{t=1}^{\delta} Pr(\tau_u \overset{*}{\leq} tX_u | \forall u \in \psi_j) \times Pr(\tau_v > tX_v | \forall v \in \overline{\psi}_j), \tag{5}$$

where, $Pr(\tau_v > tX_v | \forall v \in \overline{\psi}_j)$ is calculated as:

$$\prod_{v \in \overline{\psi}_j} Pr(\tau_v > tX_v), \tag{6}$$

and $Pr(\tau_u \overset{*}{\leq} tX_u | \forall u \in \psi_j)$ is:

$$\sum_{k=1}^{i} \sum_{m=1}^{\binom{i}{k}} \prod_{u_a \in \psi_j^a} Pr(\tau_{u_a} = tX_{u_a}) \prod_{u_b \in \psi_j^b} Pr(\tau_{u_b} < tX_{u_b}). \tag{7}$$

Eq. (7) intrinsically considers all possible cases where a subset of ψ_j, denoted by ψ_j^a, consists of nodes with their delay equal to t, while the rest of ψ_j (denoted as ψ_j^b) has delay less than t. Obviously, only the nodes being invited (i.e., with $X_x = 1$) affect the above probability calculations.

Plugging U_X and P_X into Eq. (1), we arrive at a 0-1 nonlinear optimization problem. It can be numerically solved by using existing algorithms such as Branch and Bound and Backtracking [29], [30]. For example, we have employed the available Matlab solver to obtain results to verify the effectiveness of the optimization approach.

3.2 Dynamic Programming

In the 0-1 nonlinear optimization model introduced above, both the utility function and penalty function are nonlinear. Although existing techniques can be applied to search the constrained space, they are often time consuming, thus unscalable to large networks.

At the first glance, node-recruiting process is similar to 0-1 Knapsack problem in the sense that they both intend to choose a subset of candidates to maximize a utility function with subject to a constraint. However, there exists a critical difference between them. In Knapsack, each item has independent value and weight, which remain invariant regardless of the set of selected items. Thus once an item is chosen, the algorithm can simply add its value to the total value and deduct its weight from the remaining capacity. In the competition-based participant recruitment problem, however, a node does not have an explicitly addable utility or deductible penalty, because the utility and penalty functions depend on which set of nodes receive the invitations. A node has different contribution to the utility and penalty, when the set of invitees are different.

In this research, we devise an efficient dynamic programming algorithm to solve the centralized node recruiting problem. More specifically, let Φ be the set of nodes that have been chosen as invitees, while Ψ be the set of undetermined candidate nodes. The initial Φ, denoted as Φ_o, includes the set of s seeds; and Ψ initially includes other $n-s$ candidate nodes, denoted as Ψ_o.

Let $U(\Phi, \Psi, \beta)$ denote the utility of the optimal solution, under the following conditions:

- First, all nodes in Φ have been selected as invitees;
- Second, any subset of candidate nodes in Ψ can be selected as invitees; and
- Third, the total penalty is no greater than β.

In other words, $U(\Phi, \Psi, \beta)$ denotes the best possible solution when the nodes in Φ have been determined as invitees, while other nodes can be freely chosen.

Consider a random node $x \in \Psi$. Obviously, x is either included or excluded in the optimal solution, depending on which case results in larger utility value. Thus we have

$$U(\Phi, \Psi, \beta) = max\{U(\Phi, \Psi - \{x\}, \beta), U(\Phi + \{x\}, \Psi - \{x\}, \beta)\}. \tag{8}$$

However, since neither $U(\Phi, \Psi - \{x\}, \beta)$ nor $U(\Phi + \{x\}, \Psi - \{x\}, \beta)$ is known, $U(\Phi, \Psi, \beta)$ cannot be immediately determined. The algorithm recursively derives $U(\Phi, \Psi, \beta)$ according to Eq. (8), until $\Psi = \emptyset$. When $\Psi = \emptyset$, $U(\Phi, \Psi, \beta)$ can be computed based on the set of chosen invitees in Φ, according to the equations in Section 3.1. Note that, in Section 3.1, we have derived the formula for U_X with X as unknown variables, and then determine X to maximize U_X. Here, when $\Psi = \emptyset$, X is already determined, so we can straightforwardly compute the corresponding utility, i.e., $U(\Phi, \Psi, \beta)$.

The algorithm then traces back to calculate $U(\Phi_o, \Psi_o, \beta)$, i.e., the maximum utility under the overall optimal solution. Note that the sequence to examine the nodes in Ψ (i.e., x in above discussions) does not affect the final results.

An example of the algorithm is illustrated in Fig. 2. Assume a network with 5 nodes: Node 1 to Node 5. The seeds include $\{4, 5\}$. Obviously $\Phi_o = \{4, 5\}$ and $\Psi_o = \{1, 2, 3\}$.

The computation complexity of dynamic programming is polynomial to the input node number n and the penalty threshold β. Each calculation of $U(\Phi, \emptyset, \beta)$ requires a computation of n^{c+1}. Therefore, the overall complexity is βn^{c+2}.

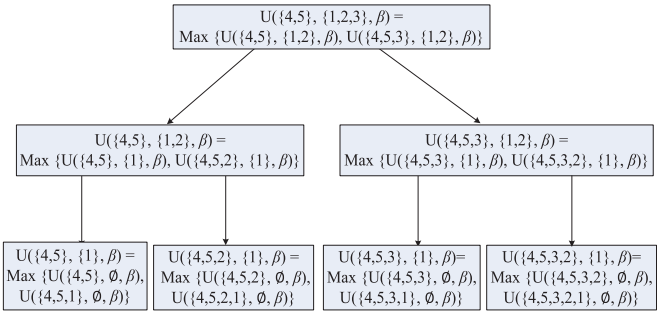
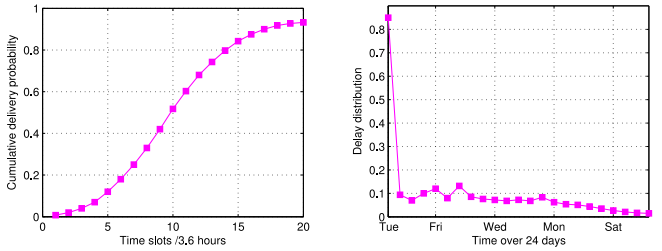


Fig. 2. An example of the dynamic programming algorithm.



(a) Delay distribution. (b) Delay distribution variation.

Fig. 3. Delay distribution between node 1 and 24 during two experimental days.

4 Distributed Solutions

In general, the centralized algorithm offers useful insights but is impractical to implement in real network settings. It demands global network information resulting in significant communication overhead. Even if the global network information is available, the centralized algorithm is computationally expensive, because it considers all potential crowdsourcing participants in the D2D network. To this end, two distributed algorithms are developed for making efficient node recruiting decisions. The first one is based on a divide-and-conquer approach by partitioning the network into opportunistic Voronoi cells and then let each cell run the optimization with much fewer candidate nodes. The second one is a task-splitting scheme which recursively delegates the recruiting responsibility to newly joined nodes.

Note that the locations of seeds generally vary over time, because the nodes are mobile and the whole work is based on opportunistic D2D connections. Moreover, the Voronoi partition of the network is not based on geographic location but the stochastic delay distribution between nodes. This is why it is named opportunistic Voronoi diagram. The nodes are grouped in the same cell because they have the probabilistically closest connections to the cell generator (i.e., the seed of the cell). While nodes are mobile, their delay distributions are relatively stable. Thus the opportunistic Voronoi diagram does not change during a crowdsourcing task.

4.1 Cell-Based Distributed Solution

Under the distributed setting, each node maintains a set of "neighbors" with which it has direct (i.e., one-hop) contact and the corresponding delay distributions. The delay distribution of a direct link between two nodes can be built through their historical inter-meeting times and updated upon a new meeting event. In our implementation, we adopt discrete time slots to construct approximate delay distributions, where each slot is Δ minutes. The distribution of direct contact between Nodes i and j is represented by $[P_{ij}^1, P_{ij}^2, \ldots, P_{ij}^D]$, where P_{ij}^d indicates the probability that their inter-meeting time is between $(d-1)\Delta$ and $d\Delta$. It is required that $D\Delta$ be no less than delay budget δ. Such approximate delay distribution can be obtained via a trivial online learning algorithm. For example, Fig. 3a shows the delay distribution between two nodes (Nodes 1 and 24) during the experiment (with the experimental setting deferred to Section 5). Since they are close to each other, the total delivery probability within $\delta = 3$ days is high, around 0.935. Most of consecutive contacts happen within two days. Furthermore, Fig. 3b shows the convergence of delay distribution. As can be seen, the delay distribution converges fast. There is small fluctuation around Sunday due to the reduced nodal contacts during weekend. But after the first week, the variation becomes insignificant and keeps stable.

To realize the cell-based approach, we first partition the network according to an opportunistic Voronoi diagram with the seed nodes serving as generating points. The network partitioning can be initiated by the seeds and performed in a distributed manner via local communications. More specifically, each seed broadcasts an announcement that includes its identity. The announcement is essentially flooded with a predefined time-to-live window. When the announcement is forwarded from Node i to Node j, the delay distribution of the link is appended to the announcement.

A candidate node may receive announcements from multiple seeds; and from each seed, it may receive multiple copies of the same announcement. Each announcement (including different copies from the same seed) contains a path to reach the seed and the delay distributions of all links along the path. The end-to-end delay distribution of a path is the convolution of the corresponding links' delay distributions. To partition the network, an appropriate metric must be defined to indicate the "distance" between nodes. The metric should take end-to-end delay distribution into consideration. In this work, we let the "distance" between two nodes be the inverse of the probability that their end-to-end delay is no greater than δ. A candidate node chooses the closest seed and joins its cell. Algorithm 1 outlines the cell establishment procedure and how the the candidates interact with the seeds and initiator. The algorithm essentially forms opportunistic Voronoi cells with the seeds as generating points, as shown in Fig. 4.

Once the opportunistic Voronoi cells are formed, the generating point (i.e., seed) of each cell can run the algorithms discussed in Section 3.2 to make a local decision for new nodes invitation. Since the algorithm is limited in individual cells, both the communication and computational overheads are much reduced compared with the centralized solution.

While the idea is straightforward, it is tricky to choose proper inputs to run the algorithm. First, the recruitment quota for each cell must be decided. Obviously, if every seed assumes a quota of c available, each of them will intend to aggressively recruit new nodes to join the crowdsourcing task. Although the penalty constraint (i.e., β) appears satisfied in each cell, the actual penalty is high, due to the excessive

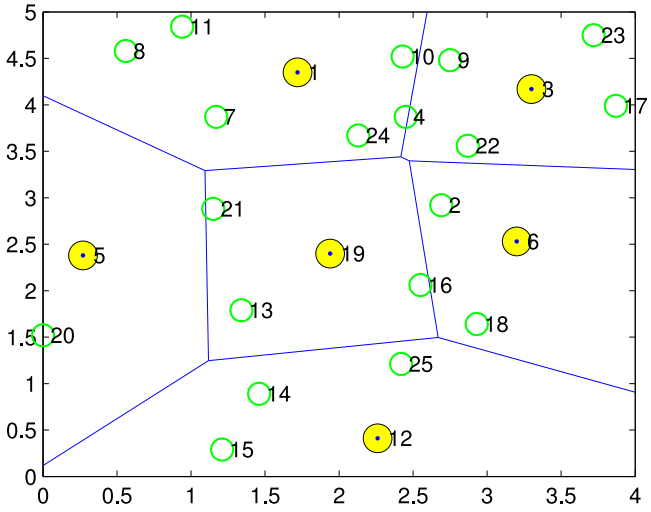


Fig. 4. An example of cells formed by 25 nodes in the experiment, where the solid dots indicate seeds while hollow circles represent candidate nodes. The distance between two nodes is inversely proportional to their meeting probability within δ. The blue lines show the boundaries between cells.

nodes being invited in the entire network. Therefore, we propose to allocate a quota to each cell, in proportion to the cell size, i.e., $c_i = \frac{x_i}{n-s}c$ where x_i is the number of candidate nodes in the cell i, $n-s$ is the total number of candidate nodes.

Second, the set of seeds as the input of the algorithm must be decided. To this end, we have explored two options. One approach is to isolate the cells. To optimally invite nodes in each cell, the algorithm considers only one seed, the proportionally allocated quota, and the candidate nodes in the cell. We call it *single-seed cell-based distributed* approach. Since this approach is run by an individual seed, without consideration of other seeds in the network, it essentially loosens the penalty constraint and thus tends to be aggressive. The decision made by a seed may result in high penalty cost of other seeds. More specifically, a seed may invite a number of candidates that are ranked higher than other seeds, thus increasing the probability that such seeds lose the task.

Another approach is to achieve joint optimization across cells. The algorithm still considers the candidate nodes in a given cell only, but takes the whole set of seeds into account for computing the overall penalty. When the algorithm is applied in Cell i, it assumes a quota of $c_i + s - 1$ available, including c_i for the cell and $s-1$ for other seeds. It tends to be more cautious, since it considers the entire set of seeds in the optimization and makes sure the invited candidate nodes do not result in a penalty greater than β based on all seeds. We call it *whole-seeds cell-based distributed* solution.

The comparison between different strategies will be further discussed in Section 5. No matter which approach is adopted, once a seed decides a set of candidates in its cell, it sends invitation to them, which upon receiving the invitation, send data to the initiator.

In the cell-based distributed approach discussed above, each seed makes its own decision based on local knowledge about the network. Apparently, such decision is not necessarily optimal, due to the incomplete inputs. However, it serves as a good approximation as to be illustrated in Section 5. The computation complexity at a seed is $\beta x_i^{c_i+2}$, where $c_i << c$ and $n_i << n$. The communication overhead in each cell is proportional to the length of delay distribution maintained by each node (in the order of $O(x_i)$) and the number of nodes in each cell (i.e., x_i). Since there is only one seed in a cell, candidate nodes only need to deliver their delay distribution to the seed so that the seed can calculate U_X and P_X. Therefore, the overall communication cost is $O(x_i^2)$.

Algorithm 1. Cell-based Recruitment

Input: p: the packet received, $p.type$: packet type, $p.srcID$: packet originator, Δt: time interval until now, γ: predefined reply collection period, $recATable$: received announcements from seeds, $NewMem$: newly recruited nodes.

if $Node_i.type == initiator$ **then**
  Select seeds based on contact history and send an invitation to them;
else if $Node_i.type == seed$ **then**
  while $\Delta t < \gamma$ **do**
    if $p.type == cellReply$ **then**
      if $p.candID$ not received before **then**
        $candTable \leftarrow p.canID$;
  $c_i = x_i \times c/(n-s)$;
  $NewMem \leftarrow Optimization(\delta, c_i, \Phi, canTable)$;
  Send $invitation$ to candidates in $NewGrpMem$;
else if $Node_i.type == candidate$ **then**
  while $\Delta t < \gamma$ **do**
    if $p.type == cellAnnounce$ **then**
      if $p.seedId \notin recATable$ **then**
        $recATable \leftarrow p$;
  Sort $recATable$ based on the delivery probability with $p.seedID$ in descending order;
  $Node_i.seedID \leftarrow recATable[0].p.seedID$;
  Send a reply to $Node_i.seedID$;
  if $p.type == invitation, p.srcID = Node_i.seedID$ **then**
    Send $confirmation$ back to the initiator;
    $Node_i.type = NewMem$.

4.2 Task Splitting: An Online Distributed Approach

The cell-based scheme discussed above adopts a divide-and-conquer approach by partitioning the network into cells and then running the optimization algorithm in each individual cell. The complexity apparently depends on cell size. In general, the more seed nodes, the smaller the average cell size. But in an extreme case, the complexity of the cell-based scheme can be similar to the centralized solution (e.g., when there is only one seed). Intrinsically, the cell-based scheme has such problem because it relies on the seeds to recruit new nodes to join the task. Based on this observation, we further propose an online distributed approach, which delegates the recruiting responsibility to newly joined nodes. We call it *task-splitting* scheme, as outlined below.

Each node maintains a "popularity" metric, which can be as simple as the number of valid contacts per time unit.[4] Let α_i denote the popularity of node i. Similar to previous

4. Note that when two nodes are connected for a long period, they may continuously find each other as "contacts" if they perform periodical neighbor discovering. Such continuous contacts between the same nodes are treated as one contact only.

discussions, let Φ be the set of nodes that have been chosen as invitees, which is initialized to be the set of seeds (i.e., Φ_o). Initially, the total recruitment quota (i.e., c) is allocated to the seeds, proportionally to their popularity. For a seed i, its allocation is $c_i = c \times \frac{\alpha_i}{\sum_{k\in\Phi_o}\alpha_k}$. As to be discussed next, c_i will be updated during the task-splitting process.

When a node $i \in \Phi$ meets another node $j \notin \Phi$, it needs to make two decisions.

- First, it must decide if Node j should be recruited.
- Second, if Node j is recruited, how to delegate future recruiting responsibilities to it.

To make the first decision, Node i essentially runs the optimization (introduced in Section 3.2) based on the partial information it has. More specifically, it keeps an incomplete Φ based on its observation, denoted by Φ^i, which includes the initial seeds (that are known by every seed) and the new members recruited by itself. Node j is considered as the only candidate node. The quota is assumed to be $c_i + |\Phi^i|$. The outcome of the optimization determines whether Node j should be recruited in.

Algorithm 2. Task Splitting Based Recruitment

Input: p: the packet received, $p.type$: packet type, $p.srcID$: packet originator, $NewMem$: newly recruited nodes
if $Node_i.type == seed, Node_j.type == candidate$
then
  $boolean\ flag = Optimization(Node_i, Node_j, \Phi, c_i)$;
  if $flag == true$ **then**
    $c'_i = c_i \times \alpha_i/(\alpha_i + \alpha_j)$; $c'_j = c_i \times \alpha_j/(\alpha_i + \alpha_j)$;
    $\Phi \leftarrow \Phi + Node_j$; $p.type = invitation$;
    send p to $Node_j$;
else if
$Node_i.type == candidate, p.type == invitation$ **then**
  Send $confirmation$ back to initiator;
  $Node_i.type = NewMem$.

If Node j is recruited, Node i will update Φ^i by including Node j. At the same time, it will make copy of the updated Φ^i for Node j to initialize Φ^j. Moreover, it must decide how to delegate future recruiting responsibilities to Node j. This is done by simply splitting the current c_i between the two nodes proportionally to their popularity metrics, i.e., $c'_i = c_i \frac{\alpha_i}{\alpha_i+\alpha_j}$ and $c'_j = c_i \frac{\alpha_j}{\alpha_i+\alpha_j}$, where c'_i and c'_j are the quotas to be used by Nodes i and j, respectively. On the other hand, when Node j joins, it immediately sends data to the initiator, as demonstrated in Algorithm 2.

The splitting procedure continues until the delay budget δ expires. The computing complexity of the algorithm is obviously low, involving local computation only. Since there is only one candidate node, the computation complexity is a constant. Similarly, the overall communication cost is also a small constant.

Comparing the two schemes, the *cell-based* is more applicable in scenarios with more seed nodes while the *task-splitting* is preferred when fewer seed nodes present. We will further discuss this observation in Section 5 (see Fig. 10 and related discussions).

4.3 Performance Analysis

To theoretically compare the proposed algorithms, we investigate the performance bound, i.e., to show the bound of the $Utility$ of each distributed algorithm relative to the centralized approach. To facilitate the analysis, we assume the delay distribution among nodes are uniform and let α be the probability that the overall delay is not greater than δ between two nodes. In practical application, α is always greater than 0.5. In the following discussion, "Central" stands for the centralized dynamic programming algorithm; "Cell-Single" means the single-seed cell-based distributed approach; "Cell-Whole" indicates the whole-seeds cell-based distributed solution; and "Split" is the task-splitting algorithm.

Theorem 4.1. *The utility of the cell-based and split-based approach is no less than* $O(\frac{1}{(1-\alpha)^{n-n/s}})$ *and* $O(\frac{1}{c(1-\alpha)^{n-s}})$ *of the centralized algorithm, respectively.*

Proof. First, for $Central$, the average utility can be derived from Eq. (3) as: $Utility(Central) = c\alpha^{s+k}(1-\alpha)^{n-(s+k)}$, where k is the number of recruited nodes. k should be determined subject to the penalty constraint given in Eq. (4). The detailed derivation of k is omitted here because as to to be shown next, k will be canceled in the calculation of the bound.

For $Cell-Single$, there are totally s cells, each with the allocated quota of $\frac{c}{s}$ and the overall $Utility(Cell-Single) = c\alpha^{1+k'}(1-\alpha)^{n/s-(1+k')}$, where k' is the number of newly recruited nodes in each cell.

Therefore, we have $\frac{Utility(Cell-Single)}{Utility(Central)} = \alpha^{k'+1-s-k}(1-\alpha)^{s+k+\frac{n}{s}-n-k'-1}$. Given $\alpha > 0.5$, we have $\frac{Utility(Cell-Single)}{Utility(Central)} > (1-\alpha)^{k'+1-s-k}(1-\alpha)^{s+k+\frac{n}{s}-n-k'-1} > \frac{1}{(1-\alpha)^{n-\frac{n}{s}}}$, thus $Utility(Cell-Single) = O(\frac{1}{(1-\alpha)^{n-n/s}})Utility(Central)$.

The only difference between $Cell-Whole$ and $Cell-Single$ is that in each cell, all seeds (i.e., s) are taken into account and the quota is $\frac{c}{s}+s-1$. Similarly, we can derive $Utility(Cell-Whole) = O(\frac{1}{(1-\alpha)^{n-n/s}})Utility(Central)$, showing the same bound as $Cell-Single$.

$Split$ is similar to $Cell-Whole$, but with dynamically increasing cells because the newly recruited nodes are also eligible to recruit new nodes, which in turn, renders the size of a cell smaller. But the calculation of utility is still similar. We have derived $Utility(Split) = c\alpha^{s+\hat{k}}(1-\alpha)^{1-\hat{k}}$ where $\hat{k}$ is 0 or 1. Accordingly, we have $\frac{Utility(Split)}{Utility(Central)} = \frac{1}{c}\alpha^{\hat{k}-k}(1-\alpha)^{s+k+1-n-\hat{k}} > \frac{1}{c}(1-\alpha)^{s+1-n}$. Therefore, $Utility(Split) = O(\frac{1}{c(1-\alpha)^{n-s}})Utility(Central)$.

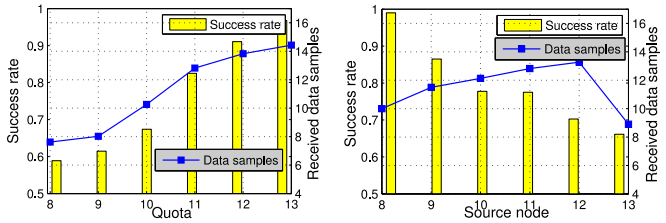
Until now, we have shown the utility of the cell-based and split-based approach is respectively bounded by $O(\frac{1}{(1-\alpha)^{n-n/s}})$ and $O(\frac{1}{c(1-\alpha)^{n-s}})$ of the centralized algorithm. This completes the proof of Theorem 4.1. □

5 Prototype and Experiments

To empirically demonstrate the feasibility and evaluate the efficiency of the proposed competition based participant

TABLE 1
Performance Comparison

	Success Rate	No. of Data Samples	Ave. Utility	Ave. Delay
Central	0.822	9.35	9.35	15.76 h
Cell-Whole	0.792	9.87	9.9	15.18 h
Cell-Single	0.778	11.25	10	13.92 h
Split	0.761	12.54	10	13.31 h



(a) Variation of c. (b) Different initiators.

Fig. 5. Success rate under different quota values and different initiators.

recruiting nodes for crowdsourcing in D2D, we have implemented a prototype on Android and carried out testbed experiments with 25 Dell Streak tablets for 24 days.

5.1 Testbed Setup

The experiment was carried out on campus with the participation of student volunteers frequenting laboratories, classrooms, dormitories, and a variety of other locations. Twenty five mobile devices tagged as Node 1 to 25 were distributed to students for a period of 24 days. Each tablet has 16 GB internal storage to keep experimental data. The devices communicate with each other via 1 Mbps Bluetooth channels with sufficient buffer. The duration of a contact is always sufficient to let nodes complete their data transmission.

A set of nodes, including $\{1, 3, 5, 6, 12, 19\}$, are randomly chosen as initiators. Each initiator launches a new crowdsourcing task periodically. In each task, the initiator randomly chooses a subset of up to 10 nodes as seeds. To add dynamics, Nodes 1 and 3 generate one task per hour, while other initiators generate one task per four hours. Note that these packets (varying between 10 bytes and 200 bytes) are the announcement packets from the initiators. Such packets describe the crowdsourcing tasks, e.g., the type of data solicited by the initiator, the deadline, the payment, etc. They are not the actual data collected by the participants, which apparently vary depending on the task. For example, in a crowdsourcing task to collect 1-minute video clips during the New Year Day celebration, each video clip is about 20 MB. Given the enormous amount of total data, it is cost-effective to transport them via D2D networks.

Based on preliminary tests, we configure the default protocol parameters as follows: the penalty threshold β is 0.2; δ is 3 days; and c varies from 8 to 13.

In order to facilitate a fair comparison between different schemes, they should be tested under the same nodal mobility. However, it is apparently impossible to run multiple algorithms simultaneously. Thus, we opt to implement one of them, i.e., the task-splitting algorithm, on the mobile devices. At the same time, we extract detailed mobility trace from the experiment, and run other algorithms based on the trace.

5.2 Experimental Results

The interested performance metrics include the success rate, the average utility, the number of data samples, and the average delay of receiving the data. The success rate is defined as the fraction of crowdsourcing tasks where all seeds successfully obtain the payments. The number of data samples indicates the average number of data received by the initiator under each crowdsourcing task.

The performance comparison is summarized in Table 1. Since this is the first work that investigates the problem of competition-based crowdsourcing in D2D, there are no competing schemes to compare with. Because different recruitment quota (i.e., c) and seeds (i.e., s) lead to different data samples and utility the table shows the results with $c = 10$ and $s = 8$. The variation of the parameters will be illustrated in later figures.

As can be seen in Table 1, only "Central" keeps the group success rate safe above 0.8 (i.e., $1 - \beta = 0.8$) because the complete global information enables the centralized algorithm to make an optimal decision which maximizes the expected utility while meeting the penalty constraint. "Cell-Whole" achieves a success rate close to 0.8. The other two distributed schemes are more aggressive in recruiting new nodes to join the crowdsourcing task, and thus resulting in higher probability that at least one seed fails to obtain the payment. Meanwhile, under a more aggressive approach, the initiator receives more data samples with shorter delay, and the average utility is higher. When the number of data samples reaches c, the average utility arrives at the peak. Furthermore, the centralized approach is the most conservative because it tries the best to protect the benefit of seeds. In other words, it may avoid choosing a candidate strongly connected to the initiator in order to make sure the seeds will not be rated behind the candidate and eventually fail to obtain the payment. On the other hand, the other approaches are more aggressive, which may choose the strongly connected candidates at the sacrifice of some success rate. As a result, their average latency is lower than the central one.

Next, we will show the impact of various parameters in the performance.[5] In general, when we study the impact of one parameter, we set other parameters to be constants. All results presented below are gathered from the testbed experiment (i.e., based on the distributed task-splitting algorithm).

Fig. 5 shows the impact of recruitment quota. Intuitively, with the same seeds, the larger quota, the easier for every seed to secure a payment. Therefore, the success rate increases. At the same time, more nodes can be recruited into the group. As a result, the number of data samples also increases. The increase of tasks also naturally leads to a higher average utility (although this is not shown in Fig. 5a).

We also observe diverse performance of the crowdsourcing tasks launched by different initiators (see Fig. 5b). To understand the result, please also refer to Fig. 6, which shows the popularity of the nodes. The nodal popularity is

5. Due to space limit, the following figures have double-ticked Y-axis. To ease readability, please find the corresponding legend for each Y-axis.

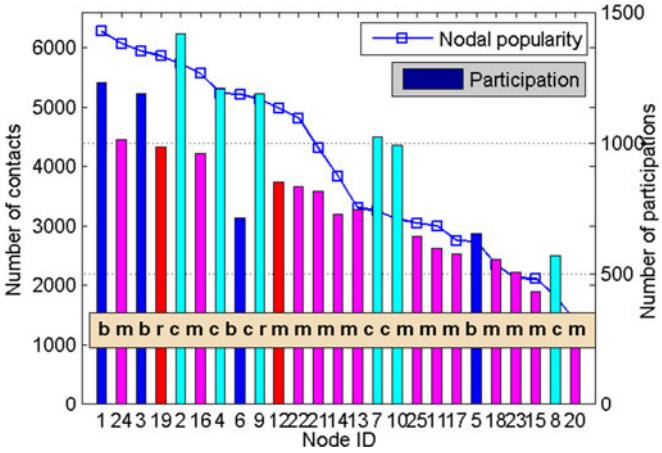
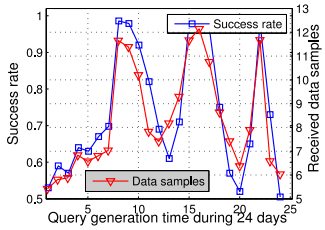


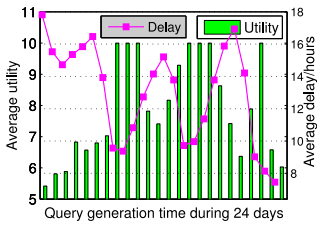
Fig. 6. Relation between nodal popularity and participation. Nodes 1, 3, 5 and 6 are both initiator and seed nodes (in different crowdsourcing tasks), marked by *blue(b)*; other initiators 19 and 12 are *red(r)*; seed nodes 2, 4, 9, 7, 10, and 8 are indicated by *cyan(c)*, and all other nodes are depicted by *magenta(m)*.

based on the valid contacts between nodes as introduced in Section 4.2. Besides the popularity, Fig. 6 also shows nodal participation, i.e., the number of times each node participates in the tasks. In general, the initiator with higher popularity have more frequent contact with others and thus are able to acquire more precise network knowledge. Consequently, they can make accurate decisions, with high success rate and relatively low overhead, as shown in Fig. 5b. Although with exceptions, this general trend holds for most initiators. The small number of data samples for initiator 5 is because the student carrying the node was absent from the third Thursday to Saturday, and thus some data samples destined to it were dropped.

Fig. 7 illustrates the performance during 24 days. First of all, we can see a clear periodicity in weeks for the success rate and number of data samples in Fig. 7a, where the X-axis shows the time when such crowdsourcing tasks are generated. In the first week, the nodes are in the process of discovering neighbors and learning the link delay distributions. Therefore, many decisions are based on incomplete and unstable network information. The performance is naturally poor. From the second Monday, nodes have gathered stable network information, so the optimization results become more accurate, yielding higher success rate. It is worth mentioning that two students carrying Nodes 5 and 15 are absent from the third Thursday to Saturday, which significantly degrade the performance.



(a) Success rate and number of data samples.



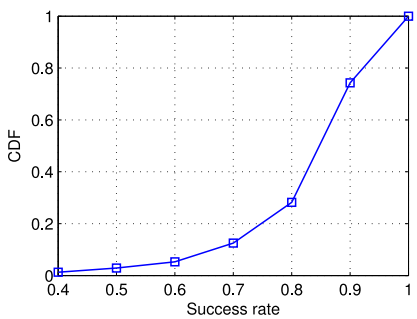
(b) Delay and utility.

Fig. 7. Performance variation under different task generation time.

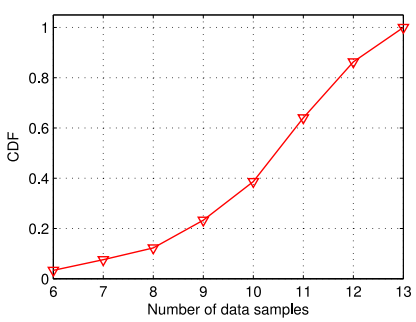
In general, nodes have more frequent contacts during weekdays than weekends. When a crowdsourcing task is generated on Monday, the seeds have sufficient time to deliver data to the initiator. As a result, the success rate is often the highest. If a task is initiated in later days of a week, the probability of delivering data within δ becomes smaller. In particular, if it is during weekends, some seeds have to wait until next Monday before they are able to send out the data or recruit new participants due to the lack of contacts, thus some tasks are simply terminated without success. The delay and utility also follow a similar pattern as depicted in Fig. 7b. The delay drops during the last three days, because the activities launched have not enough time to get finished, but only those that are finished before the end of the experiment are counted.

Fig. 8 illustrates the performance distributions. As discussed earlier, although the desired penalty threshold is $\beta = 0.2$, the distributed approach does not guarantee to achieve it due to incomplete local inputs and the approximated algorithm. However, as shown in Fig. 8a, more than 70 percent of crowdsourcing tasks reach the target success rate of 0.8. The number of data samples ranges from 6 to 13, with an average of 10. Therefore, the utility concentrates around 10.

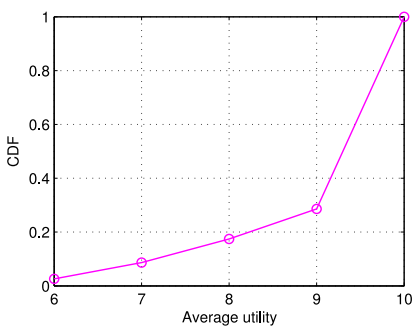
Next, we change the delay budget δ to observe its impact on the performance as shown in Fig. 9. The delay budget directly affects the routing selection and the optimization process. With a larger δ, more data can be delivered to the initiator and the average delay is naturally longer. Meanwhile, since more candidate nodes become available under larger δ, better optimization result (i.e., higher success rate and/or higher utility) is achievable. When δ is sufficiently large (e.g., when it reaches three days in this experiment), the gains in utility and success rate tend to become saturated.



(a) Success rate distribution.

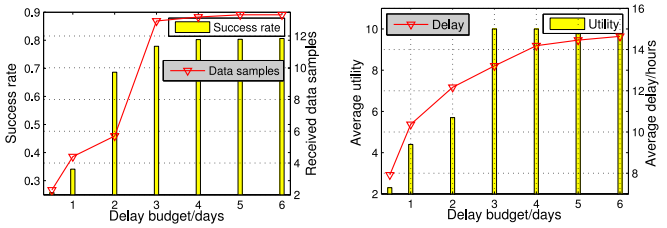


(b) Distribution of number of data samples.



(c) Utility distribution.

Fig. 8. The cumulative distributions of performance metrics.



(a) Success rate and data samples.

(b) Delay and utility.

Fig. 9. Performance under different delay budget.

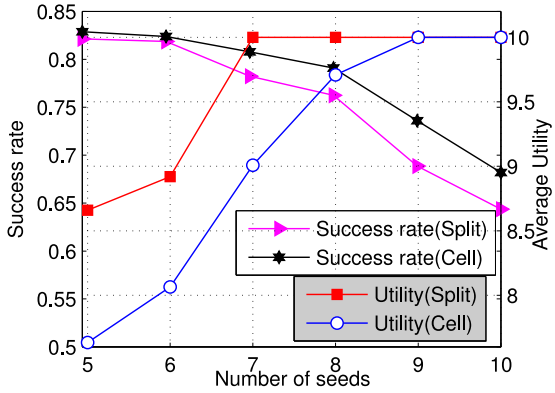


Fig. 10. Performance under different number of seeds.

Another important parameter is the number of seeds as illustrated in Fig. 10. In general, when there are more seeds, it is more difficult to ensure each of them secure a payment. Thus the success rate decreases. Besides, more seeds often help more efficiently recruit new nodes to join the crowdsourcing, thus enhancing the average utility. We also observe that when the number of seeds is low (e.g., not greater than 6), $task-splitting$ can satisfy the required success rate 0.8, while keeping a higher utility than $cell-based$. Thus it is better than $cell-based$ scheme (which has a unnecessarily higher success rate at the sacrifice of certain utility). When there are more seeds, $cell-based$ scheme performs better since it achieves the success rate closer to the expected threshold than the $task-splitting$ approach.

Besides the overall performance, we are also interested in the behaviors of seeds and other nodes. With a larger c, the seeds more aggressively recruit new nodes to participate the crowdsourcing. As a result, more invitations are sent, and the total number of data samples and final participants naturally increase. As can be seen in Fig. 11a, as c increases, the algorithm allows bigger room to let new nodes join the competition, thus more candidate nodes successfully obtain the payment. Similarly, seeds dominate when δ is small (see Fig. 11b). After $\delta = 3$ days, almost every seed can secure a payment, thus more room is given to other participants. The effect of the number of seeds is straightforward as illustrated in Fig. 11c. Suppose the total recruitment quota is fixed, with more seeds, there is less necessity to recruit other nodes. The number of invitations shows a steady increase because seeds always receive invitations from the initiator.

6 Simulations

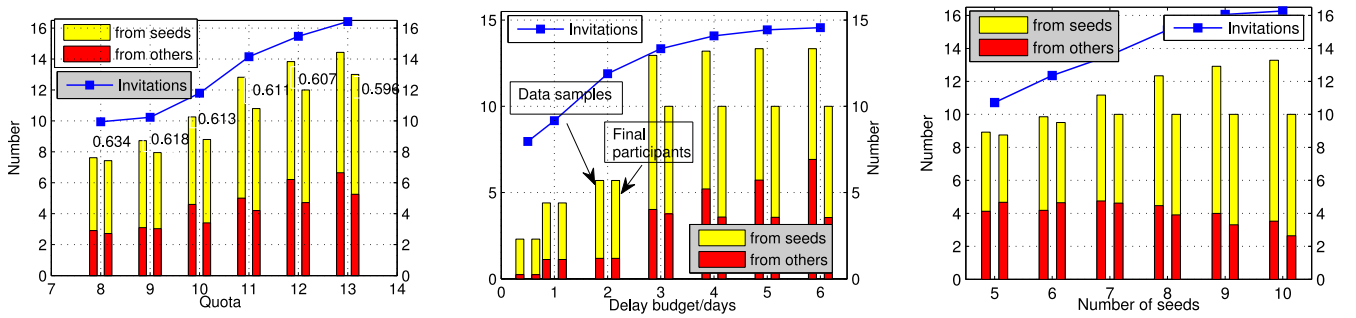
Besides the experiments, we have extracted the algorithm codes from our prototype and carried out extensive simulations based on Haggle trace [31] and DieselNet [32] trace by varying the penalty threshold and network size, which are difficult to test in the experiments.

First, the result based on Haggle trace is presented in Figs. 12 and 13. As shown in Fig. 12, a larger penalty threshold (β) allows the seeds to recruit more nodes. So there is a higher expected utility. At the same time, more nodes will compete with the seeds, increasing the probability that some seeds fail to secure their payments. Thus the success rate decreases.

In addition, we also investigate the effect of network size (in terms of the total number of nodes in a given area). A higher network density often boosts the communication opportunities. Consequently, all approaches achieve higher success rate, because the data have a better chance to be successfully delivered. With more data samples, more nodes secure their task, thus the average utility increases.

As observed in Figs. 14 and 15, the result under DieselNet trace has a similar trend as that under Haggle trace. The only difference is that due to low node density (given a small number of buses in a large area), the valid contacts (refer to Section 4.2) between nodes are fewer than people's meeting events under Haggle trace. Therefore, the overall success rate is lower while the utility is also lower. When network size is less than 20, some seed nodes even can not find a path for itself to reach the initiator within δ.

Finally, Fig. 16 illustrates the communication cost under different schemes. The centralized approach results in the highest communication overhead because each node must

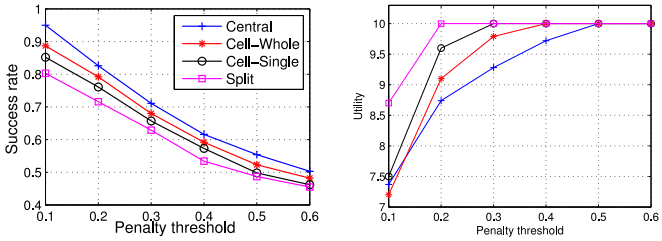


(a) Under different quota.

(b) Under different delay budgets.

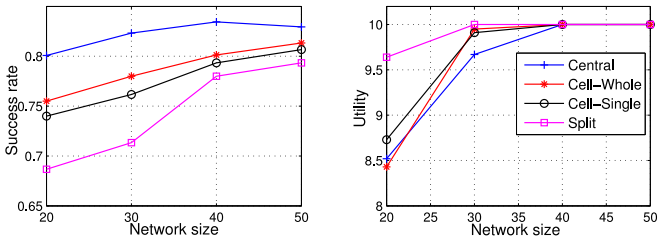
(c) Under different number of seeds.

Fig. 11. Performance comparison between seeds and other nodes. The numbers above the bars in (a) illustrate the ratio of the number of successful participants from seeds to all successful participants. From the ratio, we can observe that when c is small, a higher percentage of successful participants come from the seeds, and the percentage decreases when c increases.



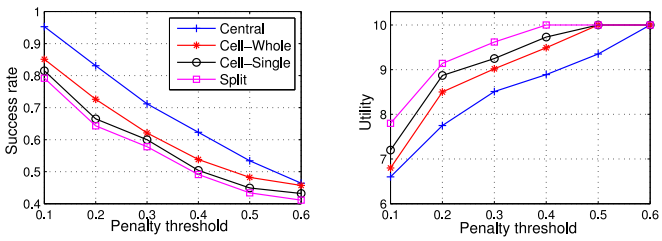
(a) Success rate. (b) Average utility.

Fig. 12. Simulated performance for Haggle under different β, with c=10, δ=3 hours, 30 total nodes, and 8 seed nodes.



(a) Success rate. (b) Average utility.

Fig. 13. Simulated performance for Haggle under different network size, with β=0.2, c=10, δ=3 hours, and 8 seed nodes.



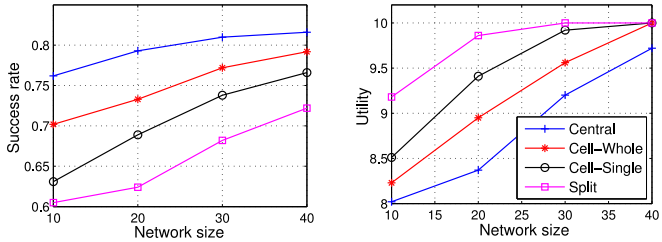
(a) Success rate. (b) Average utility.

Fig. 14. Simulated performance for DieselNet under different β, with c=10, δ=2 days, 30 total nodes, and 8 seed nodes.

exchange delivery distribution table regarding all other nodes. In distributed solutions, the communication cost is proportionally reduced due to the smaller size of optimization problem. Notice that *Cell-Single* displays slightly lower cost because it doesn't need to exchange the information with other group members compared to *Cell-Whole*, but both of them have to maintain the delay distribution of their cell members. Therefore, their communication cost is higher than *Split*.

7 Conclusion

In this paper, we have investigated the competition-based participant recruitment for large-scale, delay-sensitive data crowdsourcing in Device-to-Device networks, which are characterized by their highly nondeterministic and intermittent connectivity. We have formulated the problem from the perspective of the seeds, i.e., a set of nodes already in the crowdsourcing and face the dilemma of inviting additional participants in order to maximize their gain while keeping the risk of losing their payment low. We have proposed a dynamic programming algorithm as a first attack to this problem, followed by two distributed alternatives. The first



(a) Success rate. (b) Average utility.

Fig. 15. Simulated performance for DieselNet under different network size, with β=0.2, c=10, δ=2 days, 30 total nodes, and 8 seed nodes.

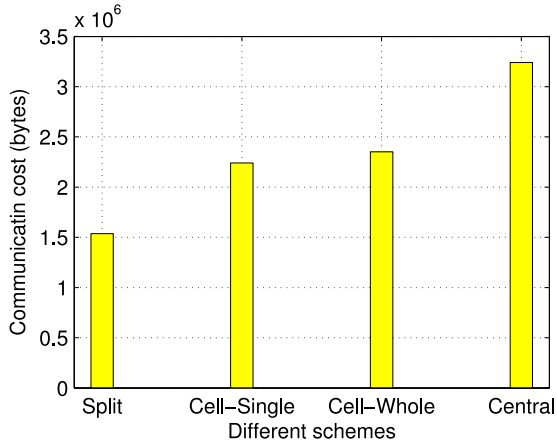


Fig. 16. Communication cost under different schemes.

one is a divide-and-conquer scheme by partitioning the network into opportunistic Voronoi cells and let each cell run the optimization, while the second is a task-splitting scheme, which recursively delegates the recruiting task to newly joined nodes. We have implemented a prototype and carried out experiments using 25 tablets for 24 days and run simulations for a more extensive evaluation under larger scales and more diverse settings. The results have shown that the distributed schemes approximate the optimization with affordable complexity, which can be adaptively chosen based on different network settings and nodal resource constraints (i.e., computation capability and storage limitation).

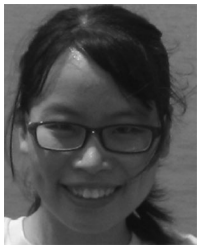
Acknowledgments

This work was supported in part by the National Science Foundation under Grant CNS-1528004.

References

[1] Ericsson, "5G radio access," in *Ericsson White Paper*, 2013.
[2] D. Yang, G. Xue, X. Fang, and J. Tang, "Crowdsourcing to smartphones: Incentive mechanism design for mobile phone sensing," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 173–184.
[3] T. Ning, Z. Yang, H. Wu, and Z. Han, "Self-interest-driven incentives for ad dissemination in autonomous mobile social networks," in *Proc. INFOCOM*, 2013, pp. 2358–2366.
[4] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proc. Annu. Int. Conf. Mobile Comput. Netw.*, 2000, pp. 255–265.
[5] B. Chen and M. Chan, "Mobicent: A Credit-based incentive system for disruption tolerant network," in *Proc. INFOCOM*, 2011, pp. 3119–3127.

[6] S. Zhong, J. Chen, and Y. R. Yang, "Sprite, a simple, cheat-proof, credit-based system for mobile Ad-hoc networks," in *Proc. INFOCOM*, 2003, pp. 1987–1997.
[7] L. Buttyan, L. Dora, M. Felegyhazi, and I. Vajda, "Bater based cooperation in delay-tolerant personal wireless networks," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, 2007, pp. 1–6.
[8] L. Song, D. Niyato, Z. Han, and E. Hossain, "Game-theoretic resource allocation methods for device-to-device (D2D) communication," *IEEE Wireless Commun. Mag.*, vol. 21, no. 3, pp. 136–144, Jun. 2014.
[9] J. Sun, X. Chen, J. Zhang, and Y. Zhang, "SYNERGY: A game-theoretical approach for cooperative key generation in wireless networks ," in *Proc. INFOCOM*, 2014, pp. 997–1005.
[10] X. Zhuo, W. G. nd Guohong Cao, and S. Hua, "An incentive framework for cellular traffic offloading," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 541–555, Mar. 2014.
[11] K. Zhang, R. Wang, and D. Qian, "AIM: An auction incentive mechanism in wireless networks with opportunistic routing," in *Proc. Int. Conf. Comput. Sci. Eng.*, 2010, pp. 28–33.
[12] I. Senturk, S. Yilmaz, and K. Akkaya, "Connectivity restoration in delay-tolerant sensor networks using game theory," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 11, no. 23, pp. 109–124, 2012.
[13] Q. Li, S. Zhu, and G. Cao, "Routing in socially selfish delay tolerant networks," in *Proc. INFOCOM*, 2010, pp. 1–9.
[14] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN routing as a resource allocation problem," in *Proc. ACM SIGCOMM*, 2007, pp. 373–384.
[15] D. Gunawardena, T. Karagiannis, A. Proutiere, E. Santos-Neto, and M. Vojnovic, "Scoop: Decentralized and opportunistic multicasting of information streams," in *Proc. 17th Annu. Int. Conf. Mobile Comput. Netw.*, 2011, pp. 169–180.
[16] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: The single-copy case," *IEEE Trans. Netw.*, vol. 16, no. 1, pp. 77–90, Feb. 2008.
[17] N. Banerjee, M. D. Corner, D. Towsley, and B. N. Levine, "Relays, base stations, and meshes: Enhancing mobile networks with infrastructure," in *Proc. 14th ACM Int. Conf. Mobile Comput. Netw.*, 2008, pp. 81–91.
[18] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks," in *Proc. 5th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2004, pp. 187–198.
[19] E. Miluzzo, N. D. Lane, K. Fodor, R. A. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, "Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application," in *Proc. 6th ACM Conf. Embedded Netw. Sens. Syst.*, 2008, pp. 337–350.
[20] P. Mohan, V. Padmanabhan, and R. Ramjee, "Nericell: Rich monitoring of road and traffic conditions using mobile smartphones," in *Proc. ACM Conf. Embedded Netw. Sens. Syst.*, 2008, pp. 323–336.
[21] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 7, no. 3, pp. 19–20, 2003.
[22] W. Gao and G. Cao, "User-centric data dissemination in disruption tolerant networks," in *Proc. INFOCOM*, 2011, pp. 3119–3127.
[23] U. Lee, S. Y. Oh, K.-W. Lee, and M. Gerla, "Relaycast: Scalable multicast routing in delay tolerant networks," in *Proc. IEEE Int. Conf. Netw. Protocols*, 2008, pp. 218–227.
[24] W. Gao, Q. Li, B. Zhao, and G. Cao, "Multicasting in delay tolerant networks: A social network perspective," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2009, pp. 299–308.
[25] S. Ioannidis, A. Chaintreau, and L. Massoulie, "Optimal and scalable distribution of content updates over a mobile social network," in *Proc. INFOCOM*, 2009, pp. 1422–1430.
[26] K. C. Lin, C. Chen, and C. Chou, "Preference-aware content dissemination in opportunistic mobile social networks," in *Proc. INFOCOM*, 2012, pp. 1960–1968.
[27] J. Fan, J. Chen, Y. Du, W. Gao, J. Wu, and Y. Sun, "Geocommunity-based broadcasting for data dissemination in mobile social networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 4, pp. 734–743, Apr. 2013.
[28] C. Boldrini, M. Conti, and A. Passarella, "Contentplace: Social-aware data dissemination in opportunistic networks," in *Proc. 11th Int. Symp. Modeling, Anal. Simul. Wireless Mobile Syst.*, 2008, pp. 203–210.
[29] A. H. Land and A. G. Doig, "an automatic method of solving discrete programming problems," *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960.
[30] D. E. Knuth, *The Art of Computer Programming*. Reading, MA, USA: Addison-Wesley, 1960.
[31] J. Scott, R. Gass, J. Crowcroft, P. Hui, C. Diot, and A. Chaintreau. (2006, Jan.). CRAWDAD trace cambridge/haggle/imote/infocom (v. 2006-01-31) [Online]. Available: http://crawdad.cs.dartmouth.edu/cambridge/haggle/imote/infocom
[32] J. Burgess, B. N. Levine, R. Mahajan, J. Zahorjan, A. Balasubramanian, A. Venkataramani, Y. Zhou, B. Croft, N. Banerjee, M. Corner, and D. Towsley. (2008, Sep.). CRAWDAD data set umass/diesel (v. 2008-09-14) [Online]. Available: http://crawdad.cs.dartmouth.edu/umass/diesel



Yanyan Han received the BS and MS degree in electronic information engineering from Shandong University, Jinan, China, in 2008 and Wuhan University, Wuhan, China, in 2011, respectively. She has been working toward the PhD degree in computer science at the Center for Advanced Computer Studies (CACS), University of Louisiana at Lafayette (UL Lafayette), since 2011. Her current research interests include delay-tolerant networks, wireless sensor networks, and mobile opportunistic networks. He is a student member of the IEEE.

Tie Luo received the PhD degree in electrical and computer engineering from the National University of Singapore. He is a scientist at the Institute for Infocomm Research (I2R), A*STAR, Singapore. His research interests include Internet of things, social computing, crowdsourcing, participatory sensing, machine learning, and security and trust. He served as the TPC Co-Chair of IEEE PerCom CASPer 2016, ICDCN ComNet-IoT 2016, and IEEE ISSNIP PSC 2014. He also served on the Organizing Committee of IEEE ISSNIP 2014 and 2015. He was a guest editor for *Mobile Information Systems* journal and the *Journal of Sensor and Actuator Networks* in 2016. He received the Best Paper Award of ICTC 2012, and was nominated for the Best Paper Award of IEEE INFOCOM 2015. He is a member of the IEEE.

Deshi Li received the PhD degree of computer applied technique in 2001 from Wuhan University, Hubei, China. He is currently a professor in the School of Electronic Information, Wuhan University as the dean of the department. His research interests include system on chip, wireless sensor network, internet of things, and underwater acoustic networks. He has (co-)authored around 40 research papers published. He has been a member of program committees of many international conferences.

Hongyi Wu received the BS degree in scientific instruments from Zhejiang University, Hangzhou, China, in 1996, the MS degree in electrical engineering, and the PhD degree in computer science from the State University of New York at Buffalo in 2000 and 2002, respectively. Since then, he has been with the Center for Advanced Computer Studies, University of Louisiana at Lafayette (UL Lafayette), where he is currently a professor and holds the Alfred and Helen Lamson Endowed Professorship in computer science. His research spans delay-tolerant networks, radio frequency identification systems, wireless sensor networks, and integrated heterogeneous wireless systems. He received the US National Science Foundation CAREER Award in 2004 and the UL Lafayette Distinguished Professor Award in 2011. He is a member of the IEEE.