Resource-Aware Photo Crowdsourcing Through Disruption Tolerant Networks

Yibo Wu, Yi Wang, Wenjie Hu, Xiaomei Zhang, and Guohong Cao

Department of Computer Science and Engineering
The Pennsylvania State University, University Park
Email: {yxw185, yuw124, wwh5068, xqz5057, gcao}@cse.psu.edu

***Abstract*—Photo crowdsourcing with smartphone has attracted considerable attention recently due to the prevalence of smartphones and the rich information provided by photos. In scenarios such as disaster recovery or battlefield, where the cellular network is partly damaged or severely overloaded, Disruption Tolerant Networks (DTNs) become the best way to deliver the crowdsourced photos. Since the bandwidth and storage resources in DTN are very limited and not enough to deliver all the crowdsourced photos, it is important to prioritize more valuable photos to use the limited resources. In this paper, we design a resource-aware photo crowdsourcing framework in DTN, which uses photo metadata including the smartphone's location, orientation, and other built-in camera's parameters, to estimate the value of photos. We propose a photo selection algorithm to maximize the value of photos delivered to the command center considering bandwidth and storage constraints. Both prototype implementation and trace-driven simulations demonstrate the effectiveness of our design.**

I. Introduction

The prevalence of smartphones creates big opportunities for crowdsourcing, where someone issues the request for certain information, and others help collect, process and return such information to the requester [5], [11], [18], [19]. An important application of crowdsourcing is gathering pictorial information in emergency situations [17]. In a natural disaster or a battlefield, a command center may need information about some specific targets (e.g., buildings). Rescuers, survivors and soldiers in the field can use smartphones (or other mobile devices) to take photos and upload them to the command center. The information contained in the photos, such as the damage of a building or the extent of flooding, helps the command center make critical decisions on the assignment of manpower, equipment, and supplies.

Unfortunately, the communication between the command center and crowdsourcing participants can be extremely constrained in these scenarios. The cellular network may be partly damaged or overloaded with extensive requests, and hence not accessible to all participants. Human-carried satellite radios may be an option, but only a small portion of participants have satellite radios due to the high cost. As a result, it is better to use Disruption Tolerant Networks (DTNs) to transfer photos among participants, and once available, use the cellular network or satellite connections to upload photos to the command center. Although DTN cannot guarantee prompt data delivery, it may be the last resort and its cost-effectiveness also makes it a feasible solution in such resource constrained environments.

This work was supported in part by the National Science Foundation (NSF) under grant CNS-1526425, CNS-1320278, and CNS-1421578.

Even with DTN, how to save resources such as storage and bandwidth poses many challenges. Participants can take many megapixel photos which consume a lot of storage space. The photos may contain redundant information, or may be irrelevant to the targets of interest. These redundant or irrelevant data exacerbate the contention for storage resources. Moreover, data can only be transferred when two participants are within the wireless transmission range of each other. Transmitting redundant or irrelevant data reduces the chance to transmit other useful data since the wireless bandwidth is limited and the participants may move out of the transmission range quickly. Thus, it is important to eliminate redundant or irrelevant photos without sacrificing useful information related to the targets of interest.

To address these challenges, we need to tackle two separate problems. First, we need to identify redundant or irrelevant photos. A straightforward way is to process and recognize photos using computer vision algorithms on participants' smartphones. If the target is recognized in a photo (i.e., the photo is relevant) and the photo is not similar to other photos (i.e., the photo is not redundant), then the photo is useful. However, running computer vision algorithms on mobile devices may take much longer time and consume a large amount of energy, and the results may not be very accurate since the best algorithm may not be able to run on mobile devices. Moreover, running computer vision algorithms requires participants to download some images of the interested targets and photos taken by others, which consumes more bandwidth.

In this paper, we develop a *photo coverage model* to quantify the value of photos in a more resource-friendly way. Specifically, a photo is characterized by its geometric metadata such as the location, orientation, and field-of-view of the camera. These parameters can be obtained via the embedded sensors on smartphones, including GPS, accelerometer, magnetic field sensor, etc. [10], [12], [27]. Since the metadata are just a couple of floating point numbers, they are easy to transmit, store, and analyze. Given metadata, we can quickly infer whether and how a photo covers the targets and then determine its value accordingly. If there are redundant photos, their coverage will overlap and their value will be reduced.

The second problem is how to incorporate the value of photos into routing, so that the most useful photos are prioritized to use the limited storage and bandwidth resources. There are some existing utility-driven routing algorithms for DTN [2], [3], [13], and simply using photo coverage as the utility metric and running their algorithms could enable us to prioritize photos based on their individual coverage. However, this simple solution does not consider the redundancy between photos. If two similar photos both have high coverage, those algorithms will prioritize both photos in routing, without considering that it is only meaningful to deliver one of them. This unique feature, caused by the redundancy (or coverage overlap) between photos, motivates us to develop a *photo selection algorithm* different from previous utility-based approaches. Our algorithm considers both the coverage overlap between photos and the contact opportunities related to user mobility. It prioritizes the storage and transmission of the most useful photos, and thus significantly increases the value of the photos delivered to the command center.

We summarize the contributions of this paper as follows.

- We propose a resource-aware photo crowdsourcing framework based on DTNs, including a photo coverage model and a photo selection algorithm.
- The photo coverage model quantifies the value of photos using lightweight metadata. It is much more resource-friendly compared to computer vision techniques.
- The photo selection algorithm considers the coverage overlap between photos, which is unique for photo crowdsourcing and has not been addressed in previous utility-based routing algorithms.
- We evaluate our design through experiments based on Android smartphones and through extensive trace-driven simulations. The results demonstrate that our resource-aware framework works well in practice and significantly increases the value of the crowdsourced photos.

The rest of the paper is organized as follows. Section II presents the photo coverage model used to estimate the value of photos, and Section III details the photo selection algorithm. The prototype implementation is shown in Section IV, and the trace-driven simulations are presented in Section V. Section VI reviews related work, and Section VII concludes the paper.

II. Photo Coverage Model

In this section, we introduce the model used to estimate the value of photos (called *photo coverage*).

A. PoIs and Photos

The command center has some *Points of Interest (PoIs)* it wants to observe. It issues a PoI list containing the coordinates of the PoIs, and spreads it to as many participants as possible through DTN or other communication networks. The PoI list is denoted as $X = \{x_1, x_2, x_3, \cdots\}$, where each x_i is a PoI. Without ambiguity, x_i also means the location of the i-th PoI.

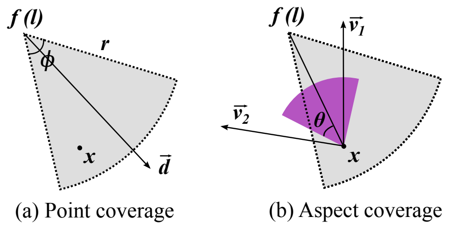


Fig. 1. (a) For point coverage, PoI x is covered by photo f, so $C_{pt}(x, f) = 1$. (b) For aspect coverage, aspect $\overrightarrow{v_1}$ is covered by photo f since $\angle(\overrightarrow{v_1}, \overrightarrow{xl}) < \theta$. $\overrightarrow{v_2}$ is not covered since $\angle(\overrightarrow{v_2}, \overrightarrow{xl}) > \theta$. In fact, all the aspects in the darker area are covered by photo f, so $C_{as}(x, f) = 2\theta$.

Each participant has some photos in the smartphone, called his/her *photo collection*. A photo collection is denoted as $F = \{f_1, f_2, f_3, \cdots\}$, where each f_j is a photo. A photo f is characterized by a tuple $(l, r, \phi, \overrightarrow{d})$, called *metadata*. Here l is the location where the photo is taken. r is the coverage range of the camera, beyond which people can hardly identify anything in the photo. ϕ is the field-of-view of the camera, which determines how wide the camera can see. $\overrightarrow{d}$ is the orientation of the camera when the photo is taken. It can be expressed as a vector coming out from the camera and vertical to the image plane. These four parameters are obtainable from smartphone APIs and built-in sensors, and they jointly determine the coverage area of a photo (gray area in Fig. 1(a)).

B. Point Coverage and Aspect Coverage

The crowdsourced photos should cover the PoIs. In traditional sensor networks, a target is covered if it is inside the coverage area of a sensor. Similarly, a PoI is covered if it is inside the coverage area of a photo, referred to as *point coverage*. For a PoI x and a photo collection $F = \{f_1, f_2, \cdots\}$, point coverage $C_{pt}(x, F) = 1$ if x is in the coverage area of any f_j from F; otherwise $C_{pt}(x, F) = 0$.

If $C_{pt}(x, F) = 1$, PoI x can be found in photo collection F. However, it is not clear how x appears in the photos. If the PoI is a building, we may see its front view, side view, or even back view. This is quite different from traditional sensor coverage, where the relative angle of sensors and targets does not matter. From the command center's perspective, only seeing the PoI is not good enough, and it is better to view the PoI from multiple directions so as to obtain omnibus information.

Therefore, we introduce *aspect coverage* to capture this unique property of photos. An aspect of a PoI, denoted by $\overrightarrow{v}$, is a vector that can be represented by an angle in $[0, 2\pi)$. Angle 0 represents the vector pointing to the right (east on the map), and it increases in the clockwise direction. For example, in Fig. 1(b), $\overrightarrow{v_1}$ and $\overrightarrow{v_2}$ are two aspects with angle 270° and 190°, respectively.

Aspect $\overrightarrow{v}$ of PoI x is covered by photo f if the following conditions hold: x is inside the coverage area of f, and the angle between $\overrightarrow{v}$ and $\overrightarrow{xl}$ is less than a predefined threshold θ (called *effective angle*). Here $\overrightarrow{xl}$ is the vector from the PoI to the camera, representing the viewing direction of the camera.

Then for a PoI x and a photo collection F, we have aspect coverage $C_{as}(x, F) = \int_0^{2\pi} 1_F(v)dv$, where $1_F(v) = 1$ if $\overrightarrow{v}$ is covered by any f_j from F, or 0 otherwise.

Aspect coverage shows how many aspects of a PoI are covered in the photos. With larger aspect coverage, more different views of the PoI can be seen, and thus more information can be obtained about the PoI. However, only considering aspect coverage may cause problems. A photo collection may have large aspect coverage on some PoIs but leave many other PoIs completely uncovered. Thus, we should first consider point coverage to ensure more PoIs are covered, and then maximize the aspect coverage of those PoIs.

C. Photo Coverage

Photo coverage combines point coverage and aspect coverage, and gives point coverage higher priority.

Definition 1 (photo coverage)**.** *Given a PoI x and a photo collection F, photo coverage C_{ph} is defined as*

$$C_{ph}(x, F) = (C_{pt}(x, F), C_{as}(x, F)),$$

where $C_{pt}(x, F)$ and $C_{as}(x, F)$ obey lexicographical order. In other words, $C_{ph1} > C_{ph2}$ if and only if 1) $C_{pt1} > C_{pt2}$ or 2) $C_{pt1} = C_{pt2}$ and $C_{as1} > C_{as2}$.

The above photo coverage is defined for one PoI. For a PoI list $X = \{x_1, x_2, \cdots\}$ and a photo collection F, photo coverage C_{ph} is defined as

$$C_{ph}(X, F) = \left(\sum_{x_i \in X} C_{pt}(x_i, F), \sum_{x_i \in X} C_{as}(x_i, F) \right).$$

In the rest of this paper, photo coverage is always calculated for the same PoI list, so we omit the notation X and use $C_{ph}(F)$ to denote the photo coverage of F.

Discussions: When a target is more important than other targets, or when a particular angle of a target (e.g., main entrance of a building) is more important than others, we can easily extend the above definition to assign different weights to different PoIs, or different weights to different aspects of a PoI. For example, if a PoI has weight w, a photo covering it has point coverage w instead of 1. Then, photos covering more important PoIs will have higher coverage, and thus will be prioritized in routing.

The value of photos can be affected by factors other than coverage. For example, a photo that is blurred or has improper brightness may not be useful even though it has good coverage. Also, the value of photos may diminish with time. These factors are highly application-dependent and hence is out of the scope of this paper. The applications could either use a continuous function to quantify those factors together with our photo coverage model, or use a binary threshold to filter out unqualified photos before using our model.

III. Photo Selection Algorithm

In this section, we first describe the optimization problem considered in this paper. We identify and address two important design challenges, and then present our photo selection algorithm.

A. Problem Description

At the beginning of a crowdsourcing event, the command center issues a PoI list describing its interests and a deadline indicating how long the PoI list will be valid. Before the deadline, photos are selected and delivered to the command center through DTN and other communication links. Specifically, when two nodes (participants) move within the wireless communication distance (Bluetooth or WiFi), they are in contact and photos can be transmitted between them. The command center can be seen as a special node, denoted by n_0, who can receive photos from other nodes when cellular or satellite connections are available.

Before the deadline, the command center has received a collection of photos F_0. Our objective is to maximize its photo coverage, i.e., $\max\ C_{ph}(F_0)$, under the following constraints: 1) limited contact opportunities and limited bandwidth in DTN, 2) limited availability of cellular and satellite connections, and 3) limited storage space of smartphones.

A centralized solution to the above problem will not work because the contacts among DTN nodes are not known a priori. Therefore, to maximize $C_{ph}(F_0)$ in a distributed way, nodes should optimize their local photo collections by exchanging photos with each other whenever a contact happens. This local optimization is the core of our photo selection algorithm, which will be presented in Section III-D. However, such optimization requires accurate calculation of photo coverage, which poses two new challenges. First, due to coverage overlap, the coverage of a photo depends not only on itself, but also on the presence of other photos. We address this challenge by proposing a metadata management scheme in Section III-B. Second, when a photo is in DTN, its *potential value to the command center* depends not only on its photo coverage, but also on how likely it can be delivered to the command center. We address this challenge by introducing *expected coverage* in Section III-C.

B. Metadata Management

Due to coverage overlap, the coverage of a photo depends on the presence of other photos. For example, a photo's coverage will decrease if another photo already covers the same PoI. To calculate photo coverage accurately, it is essential to take the presence of other existing photos into account. To this end, nodes should share the metadata of their photos with each other whenever a contact happens. Thus, every node maintains its knowledge about the metadata of every other node. Such metadata is cached and used later for photo coverage calculation. Note that when the metadata of the command center is shared, it works as an acknowledgment, telling nodes which photo has already been delivered to the command center.

Caching metadata costs very little storage space, but cache validation is a big challenge. After the node whose metadata has been cached leaves after a contact, its photos may change when it meets other nodes. However, these changes may not be known to those who have cached the metadata. Such inconsistency cannot be solved by traditional cache validation schemes due to the low connectivity of DTN. To address this issue, we propose a metadata management scheme as follows.

The inter-contact time T_{ab} between two nodes n_a and n_b has been shown to have exponential decay for many mobility models (e.g., random waypoint and Brownian motion) and real traces [4], [7], [30]. Hence, to capture the contact patterns of nodes (e.g., rescuers in the same team contact more often), we assume T_{ab} to be exponentially distributed with parameter λ_{ab}. Then, the inter-contact time between node n_a and any other node is a random variable $T_a = \min_{n_b \neq n_a} T_{ab}$, so it follows exponential distribution with parameter $\lambda_a = \sum_{n_b \neq n_a} \lambda_{ab}$.

In a contact between n_a and n_b, n_a sends n_b its photo metadata and parameter λ_a learned from historical contacts. Later, when n_b wishes to use n_a's metadata to calculate the coverage of its photos, it computes the time elapsed since their last contact, denoted as t. Then it calculates the probability that n_a has met another node within time t as

$$P\{T_a < t\} = 1 - e^{-\lambda_a t}. \tag{1}$$

The outcome $P\{T_a < t\}$ is compared with a predefined threshold P_{thld}. If it exceeds the threshold, the metadata is considered invalid and removed from the cache, because there is a high probability that node n_a has already met another node and therefore has updated its photos. Otherwise, the metadata of n_a is valid and can be used to calculate photo coverage. The value of P_{thld} is currently determined by simulations. It is difficult, if not impossible, to theoretically evaluate its effect, because it is hard to estimate how many photos have been updated during a contact.

C. Expected Coverage

By our model, the value of a photo can be estimated by its photo coverage. However, for photos in DTN, it is not clear if the photo coverage can be achieved since these photos may never reach the command center. Thus, the potential value of a photo depends not only on its photo coverage, but also on how likely it can be delivered to the command center.

We address this issue by jointly considering the photo coverage and the probability of data delivery from a node to the command center. To this end, we compute *delivery probability* as defined in the PROPHET routing protocol [16]. The delivery probability from node n_a to node n_b is a probabilistic metric of how likely n_a can deliver a packet to n_b. It is computed based on three heuristics: 1) the delivery probability should increase if two nodes encounter each other; 2) the delivery probability should decrease if two nodes do not encounter for a while; 3) if the delivery probability from n_a to n_b is high and that from n_b to n_c is high, then the delivery probability from n_a to n_c is also high (transitive property). The exact calculation of delivery probability can be found in Section III-A of [16]. Here, we use the delivery probability from n_i to the command center n_0, denoted as p_i, as an indicator of how likely n_i's photos can be delivered.

Intuitively, if we multiply a photo's coverage by the probability that it is delivered, the result will be the expectation of the achieved coverage, referred to as the *expected coverage*. Compared to the original photo coverage, the expected coverage is a better estimate of a photo's value, because it takes into account whether or not the photo can actually be delivered. However, the above definition of expected coverage only works for one single photo. It does not consider the possible coverage overlap between this photo and other existing photos. To consider the coverage overlap between photos and thus assess the value of photos more accurately, we extend the above definition of expected coverage to a set of photos, as detailed below.

In a contact between n_a and n_b, the two nodes can assess the value of their photos by calculating expected coverage. To consider coverage overlap between photos, expected coverage is calculated for a node set M that contains all nodes of which n_a and n_b have valid metadata. First, M includes n_a and n_b. Second, M includes the nodes whose metadata is valid according to equation (1). Third, M also includes n_0 because it is important to consider what has been received by the command center. We assume the command center does not drop photos, and thus the metadata of n_0 is always valid. The expected coverage of node set M is defined as follows.

Definition 2 (Expected Coverage)**.** *Given a node set $M = \{n_0, n_1, \cdots, n_{m-1}\}$, let F_i and p_i be the photo collection and the delivery probability of node n_i $(i = 0, 1, \cdots, m-1)$, respectively. Also let $b_i \in \{0, 1\}$ denote whether node n_i can deliver its photos to the command center. Then a binary string $B = b_0 b_1 \cdots b_{m-1}$ denotes one possible outcome with regard to whether each node can deliver its photos. The probability of this outcome is*

$$P_B = \prod_{i=0}^{m-1} p_i^{b_i} (1 - p_i)^{(1-b_i)}.$$

In this outcome, the photo coverage obtained by the command center is

$$C_B = C_{ph}\left(\bigcup_{b_i=1} F_i\right).$$

Hence, the expected coverage of M is defined as $C_{ex}(M) = \sum_{B \in \{0,1\}^m} P_B \cdot C_B$.

As an example, with $m = 3$, node set $M = \{n_0, n_a, n_b\}$. Since the command center always reaches itself ($b_0 = 1$), we have four cases, $B = 100, 101, 110, 111$. Therefore,

$$\begin{aligned} C_{ex}(M) &= C_{ph}(F_0) \cdot (1 - p_a)(1 - p_b) \\ &\quad + C_{ph}(F_0 \cup F_b) \cdot p_b(1 - p_a) \\ &\quad + C_{ph}(F_0 \cup F_a) \cdot p_a(1 - p_b) \\ &\quad + C_{ph}(F_0 \cup F_a \cup F_b) \cdot p_a p_b. \end{aligned} \tag{2}$$

From formula (2) we can see two advantages of using expected coverage. First, it always considers the photos already received by the command center (F_0). Second, it considers all possible cases about whether the nodes can reach the command center. In this example, there are two nodes (n_a and n_b), and each node may or may not reach the command center. Each of the four possible cases is considered with a weight indicating its probability of happening.

Via Definition 2 we calculate the expected coverage of the nodes in M. Since our initial purpose is to assess the value of photos in n_a and n_b. we also denote $C_{ex}(M)$ as $C_{ex}(F_a, F_b)$ to emphasize that purpose.

D. Photo Selection Algorithm

Recall that our objective is to maximize the photo coverage obtained by the command center. To achieve this goal in a distributed way, nodes should maximize the potential value (i.e., expected coverage) of their local photo collections by exchanging photos with each other. Specifically, when two nodes n_a and n_b are in contact, they reallocate their photos to maximize the expected coverage $C_{ex}(F_a, F_b)$. The reallocation is based on a selection pool that contains all the candidate photos, $F_a \cup F_b = \{f_1, f_2, \cdots, f_k\}$. Let s_j be the size of photo f_j. We have the following photo reallocation problem.

$$\begin{aligned} \max \quad & C_{ex}(F_a, F_b), \\ \text{subject to} \quad & \sum_{j=1}^{k} y_j s_j \le S_a, \sum_{j=1}^{k} z_j s_j \le S_b, \end{aligned}$$

where $y_j, z_j \in \{0, 1\}$ means whether photo f_j is selected into node n_a or n_b. S_a and S_b is the storage size of n_a and n_b.

The above problem is NP-hard since the standard 0-1 knapsack problem can reduce to it. Also, its objective function $C_{ex}(\cdot)$ is non-convex due to coverage overlap: the coverage of a photo depends not only on itself, but also on other existing photos. This makes the photo reallocation problem even more challenging. Here, we propose a greedy algorithm to solve it.

Without loss of generality we assume $p_a > p_b$, i.e., node n_a has more chances than n_b to deliver photos to the command center. Thus, n_a should have higher priority to select photos. We first fill up the storage of n_a by solving the following problem.

$$\begin{aligned} \max \quad & C_{ex}(F_a, \emptyset), \\ \text{subject to} \quad & \sum_{j=1}^{k} y_j s_j \le S_a. \end{aligned} \tag{3}$$

It is a non-convex optimization problem, so the solution is based on greedy heuristics. In each step of selection, one photo from the selection pool is selected and stored in n_a, such that the current expected coverage $C_{ex}(F_a, \emptyset)$ is greedily maximized. The selection proceeds for the remaining part of the selection pool until n_a's storage is full or no more benefit can be achieved.

Afterward, node n_b selects photos to its storage by solving a similar problem to (3). n_b considers what n_a has selected by maximizing $C_{ex}(F_a, F_b)$. Thus, n_b may not choose photos that are similar to (or the same as) those stored in n_a. This happens when n_a can reach the command center with a high probability, i.e., p_a is large. However, n_b uses the same selection pool as n_a, that is, the original $F_a \cup F_b$. It is possible that n_b selects a photo f_j that is already stored in n_a ($y_j = z_j = 1$). This happens when f_j is very useful but n_a cannot deliver it with a high probability.

The above algorithm assumes that the contact duration is long enough to complete the required photo transmission. In some cases, the assumption may not be valid; i.e., the contact duration is too short to transmit all required photos. With such network constraints, the algorithm should be adjusted as follows.

First, the two contacting nodes run the above algorithm to obtain a solution that maximizes the expected coverage. The solution can be different from their current photo collections. Then they transmit photos between each other so that their photo collections gradually become the same as the solution. To do this, the photos in the solution are considered one by one, in the order that they are selected. Specifically, we start by considering the first photo selected to n_a in the solution (given $p_a > p_b$). If the photo is already in n_a's photo collection, no transmission is needed. Otherwise, the photo is currently in n_b and is transmitted to n_a. Then we consider the second photo selected to n_a and check whether a transmission is needed. Similarly, all photos selected to n_a are considered and n_a's photo collection finally becomes the same as the solution. After that, the same operation proceeds for n_b. The contact may end at any time during this process, and any unfinished transmission will be discarded.

IV. Prototype Implementation

We have implemented a proof-of-concept prototype to test whether our photo selection algorithm works for real photos. It is based on a Google Nexus 4 running Android 4.2, and it enables automatic metadata acquisition when a photo is taken. In the following, we explain how photo metadata is automatically generated, and then present a demonstration with real photos taken from the phone.

A. Generating Metadata

Photo metadata includes GPS location, field-of-view, coverage range, and orientation. We discuss how these four parameters are generated automatically and accurately.

GPS location is easily accessible via built-in GPS receivers. Common GPS errors of 5-8.5m should be tolerable for big objects like buildings and roads [29]. Field-of-view may vary from photo to photo, but it can be accurately obtained via Android camera API [1].

Coverage range is the distance beyond which objects are no longer sufficiently recognizable in the photo. It is very difficult to estimate because recognizability is not a quantitatively defined concept, and it depends on many factors such as focal length, lens and camera vibration. Among these factors the most important one is focal length f. An object becomes larger in an image in the same rate as focal length increases. Therefore, we set coverage range $r \propto f$. We also know by

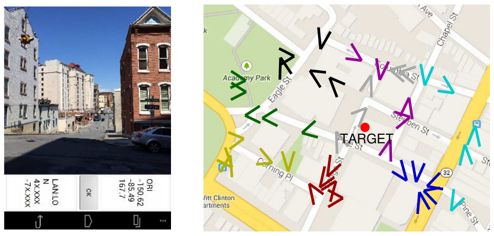


Fig. 2. (a) Screenshot of the programmed phone. (b) We take 40 photos and assign them to 8 nodes in the trace. The locations and orientations of the photos are shown in V shapes, where the photos assigned to different nodes have different colors.

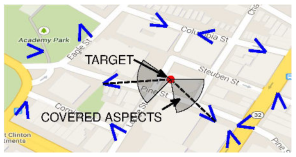
photography that $f \propto \cot(\phi/2)$, where ϕ is the field-of-view, so $r \propto \cot(\phi/2) = c\cot(\phi/2)$. The remaining problem is to determine coefficient c, which largely depends on the requirement of applications. As the objects in our experiments are buildings, we set c to be 50m. Thus, for field-of-view $\phi \in [30^\circ, 60^\circ]$, coverage range $r \in [87\text{m}, 187\text{m}]$. We find that for our purpose, objects taken within this coverage range are generally recognizable.

Orientation is a critical factor in our model so it has to be computed accurately. We adopt the method in SmartPhoto [27], which involves the use of accelerometer, magnetic field sensor and gyroscope. When a photo is taken and the phone is held static, the accelerometer measures the gravity direction and the magnetic field sensor measures the ambient geomagnetic field. These two measurements can be used to calculate an estimate of orientation. Since this method is susceptible to noise and errors, gyroscope is used as a compliment to improve accuracy. Gyroscope measures the rate of phone rotation. If the rate is multiplied by the period of time between two gyroscope readings, the result is the change of orientation. Thus, gyroscope provides another estimate of orientation, and the two estimates can be linearly combined to produce a more reliable result. This result is further enhanced by orthonormalization, and the final outcome achieves a maximum error of five degrees.

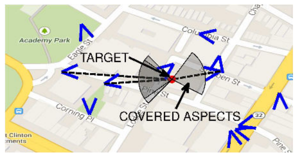
B. Demonstration With Real Photos

We apply the proposed photo selection algorithm to real photos to see the selection results. To run the selection algorithm, we extract the contact information among 9 nodes from an existing DTN trace, *MIT Reality*. The MIT trace records the contacts among users carrying handheld Bluetooth devices. The devices periodically detect their peers nearby, and a contact is recorded when two devices move into the transmission range of each other. Among the 9 nodes, 8 of them represent crowdsourcing participants, and the other one represents the command center. It can be considered as a rescuer carrying a satellite radio or a data mule that periodically moves back and forth between the area and the command center. We use the last 48 contacts among the 9 nodes to run the algorithm and collect photos, and use all previous contacts to learn the delivery probability of nodes.

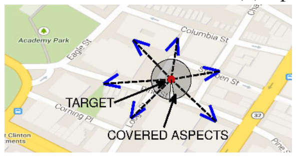
Furthermore, we take 40 photos using the programmed phone and assign five of them to each crowdsourcing participant in the trace, as if the node took these photos for the crowdsourcing task. The locations and orientations of the photos are shown in Figure 2(b) in V shapes, where the photos assigned to different nodes are in different colors. The target (PoI) is a historic church near the center of the area.



(a) PhotoNet



(b) Spray&Wait



(c) Our scheme

Fig. 3. Photos delivered and aspects covered.

Based on the photo metadata and the contact trace, we simulate the photo crowdsourcing process using three different algorithms: our scheme, PhotoNet, Spray&Wait. PhotoNet [22] is a picture delivery service that prioritizes the transmission of photos by considering location, time stamp, and color difference, with the goal of maximizing the "diversity" of the photos. Spray&Wait [20] is a classic DTN routing protocol that balances resource utilization and delivery ratio. It is used to represent a set of general-purpose DTN routing protocols that do not differentiate packets by their content.

In real crowdsourcing applications, a participant may possess hundreds of photos while the peer-to-peer communication bandwidth is at 1MB/s level and the smartphone storage is at 1GB level. In our experiments a participant has much fewer (five) photos, so the bandwidth and storage constraints must be scaled down accordingly. We limit the number of photos that can be transferred in a contact to be three and the number of photos that can be stored in a device to be five.

Figure 3 shows the photos delivered to the command center at the end of the trace. For photos covering the target, dashed lines show their viewing directions and gray areas show the covered aspects (effective angle $\theta = 40^\circ$). Both PhotoNet and Spray&Wait deliver 12 photos because there are four contacts between the participants and the command center, and three photos are transferred in each contact. PhotoNet delivers photos that are diverse in terms of location, time and color histogram, and thus the delivered photos spread across the area. However, it does not have much information about the target, with only two photos covering 160°. Spray&Wait does not consider photos by their content, and the results are not good either. Three of the 12 photos cover 171° of the target. Among the three schemes, ours has the best performance. Although 12 photos can be delivered, six of them are not helpful for increasing the photo coverage. Thus, our scheme only delivers the six most useful photos, which cover 346° of the target.

Figure 4 shows the real images delivered by our scheme. As can be seen, the photos cover the church from different

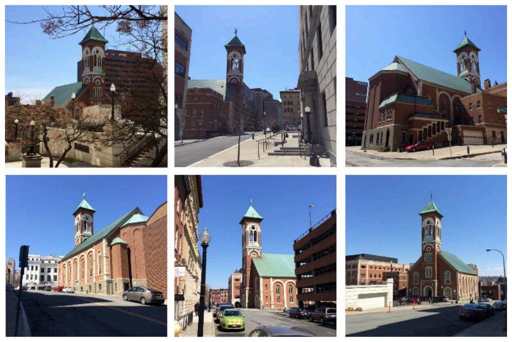


Fig. 4. Images delivered by our scheme.

angles, showing a quite complete view of the church. Having complete information about the target is important in a disaster recovery scenario, where the command center needs to identify any damage related to the building and determine whether it is safe for survivors. The images delivered by the other two algorithms are not shown due to space limitation. From the photo metadata and the aspect coverage shown in Figure 3, it is clear that they are not as good as those delivered by our scheme.

V. Trace-Driven Simulations

Real photo crowdsourcing applications have a much larger scale than our demonstration. In this section, we use trace-driven simulations to further evaluate the performance of our photo selection algorithm.

A. Simulation Setup

We assume participants move in a 6300m$\times$6300m square region, like a town or the central area of a city. The command center issues a list of 250 PoIs randomly located in the region.

The contact information between participants is retrieved from two traces: *MIT Reality* and *Cambridge06* [15]. Similar to the MIT trace, Cambridge06 records the contacts among users carrying handheld Bluetooth devices. The devices periodically scan their peers nearby and record a contact if a peer is discovered. The scan interval for the MIT trace is five minutes, and the scan interval for the Cambridge06 trace is two minutes.

Participants may be able to communicate with the command center through various ways such as DTN links, satellite radios, etc. To simulate such links, we randomly pick about 2% of the total participants and assume they can communicate with the command center. They can be rescuers in a disaster area or commanders in a battlefield who have satellite radios, or they can be data mules that move back and forth between the interested area and the command center.

Photos are randomly generated by the participants. Photo metadata and other simulation settings are shown in Table I.

TABLE I
Simulation Settings

Parameter	Notation	Value*
photo size		4MB
effective angle	θ	$30°$
orientation	$\vec{d}$	$[0°, 360°)$
field-of-view	ϕ	$[30°, 60°]$
coverage range	r	$[50, 100]\cot(\phi/2)$m
valid threshold	P_{thld}	0.8
PROPHET	P_{init}, β, γ	0.75,0.25,0.98
# of nodes		97/54
simulation time		300/200 hr

*A range means that the value is randomly generated in that range; two numbers with "/" means the value in the MIT/Cambridge06 trace.

B. Comparing With Other Schemes

In this subsection, we evaluate the performance of our scheme by comparing it with other solutions. The evaluation is based on the MIT trace, and the performance is measured in terms of the point coverage and aspect coverage obtained by the command center (normalized by the number of PoIs). Each data point is the average of 50 simulation runs.

The proposed photo selection algorithm (called our scheme) is compared with the following schemes.

- **BestPossible:** There is no storage or bandwidth constraint for this scheme. The only constraint is contact opportunity. Nodes will try to replicate every useful photo to everyone, and achieve the best possible coverage.
- **NoMetadata:** Disable the metadata caching and metadata management component from our scheme.
- **Spray&Wait:** The binary spray and wait protocol with four allowed copies, as described in [20].
- **ModifiedSpray:** It is based on Spray&Wait. When a node transmits photos to another node, it transmits the photo with the most photo coverage first. When a node receives a photo and its storage is full, it first removes the photo with the least photo coverage.

Fig. 5 shows the results, where the storage size is 0.6GB and the number of created photos is 250 per hour. As time goes, more photos are delivered and thus the coverage increases. Our scheme covers 70% of PoIs after 150 hours of crowdsourcing. This time would be significantly shorter in real world applications because there could be much more participants than those in the DTN traces.

Comparing the five schemes, BestPossible provides the performance upper bound, and our scheme performs close to it with a maximum of 10% less point coverage and 17% less aspect coverage. NoMetadata performs worse than our scheme, which shows that metadata caching and metadata management can improve performance even though the inter-contact time used in metadata management does not strictly follow exponential distribution in real traces. Spray&Wait has the worst performance because it is designed for routing general data and does not consider which photo is more useful. At 150 hours, it has 49% less point coverage and 69% less aspect coverage than our scheme. ModifiedSpray outperforms Spray&Wait because it considers the coverage of photos. However, like previous utility-based routing algorithms, ModifiedSpray prioritizes

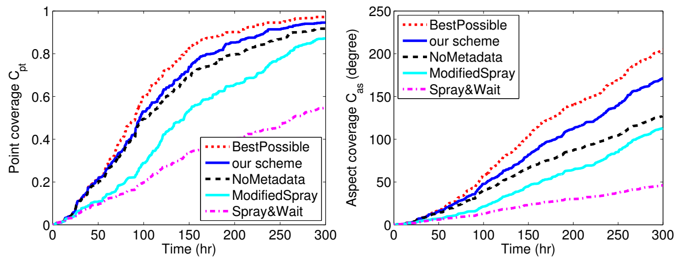


Fig. 5. Comparing our scheme with other four schemes.

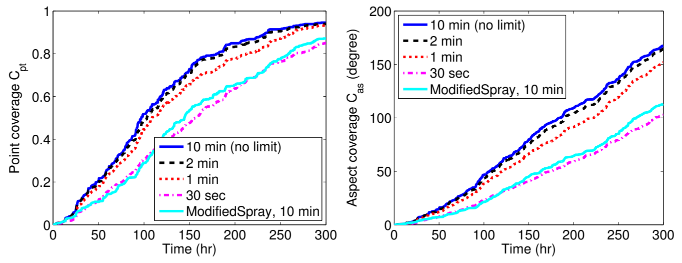


Fig. 6. The effects of short contact duration.

photos by their individual coverage without considering the overlap (redundancy) between photos. This is why it still underperforms our scheme. At 150 hours, ModifiedSpray has 26% less point coverage and 38% less aspect coverage than our scheme.

C. The Effects of Short Contact Duration

We usually assume the contact duration is long enough to do the required photo transmission. In this subsection, we evaluate how our scheme performs under short contact duration. The transmission bandwidth is set as 2MB/s, which can be achieved by Bluetooth 3.0+hs, WiFi ad hoc mode, and WiFi Direct. Thus, a 10 minute contact results in 1.2GB transmission capacity, enough for the transmission between two nodes with 0.6GB storage.

The results are shown in Fig. 6. Since 10 minutes means no limit on the contact duration, its performance is the same as before. When the contact duration is reduced to 2 minutes, the performance only decreases by about 1%. This is because our scheme prioritizes the transmission of important photos, and thus those photos are still delivered to the command center even though the contact duration reduces by 80%. Nonetheless, the performance drops faster when the contact duration is further reduced and becomes insufficient for transmitting important photos. In the 30 seconds case, only 5% of the photos can be transmitted, and the performance degrades to a similar level of ModifiedSpray with 10 minutes duration. In sum, our scheme maintains good performance unless the contact duration is drastically reduced (e.g., by 95%), in which case the performance is still comparable to ModifiedSpray with 10 minutes duration.

D. The Effects of Storage Capacity

In this subsection, we evaluate the effects of storage capacity. At the end of a simulation run, the point coverage and aspect coverage obtained by the command center are recorded (normalized by the number of PoIs). We also record the number of photos delivered to the command center.

Fig. 7 summarizes the results, where the number of generated photos is 250 per hour. Fig. 7(a)(b)(c) is based on the MIT trace and Fig. 7(d)(e)(f) is based on the Cambridge06 trace. From Fig. 7(a)(b)(d)(e), we can see that increasing storage capacity generally improves photo coverage since more photos can be delivered to the command center. For our scheme and NoMetadata, the performance is improved mainly because with larger storage space, a useful photo has more copies stored in different nodes, and thus it is more likely to be delivered. However, ModifiedSpray is not affected too much by storage space since the number of copies for a photo is limited to four.

Fig. 7(c)(f) shows the number of delivered photos in logarithmic scale. In our scheme and NoMetadata, photos are transferred to the command center only if they can contribute to the photo coverage. Thus, the number of delivered photos in our scheme and NoMetadata is dramatically less than that in ModifiedSpray and Spray&Wait.

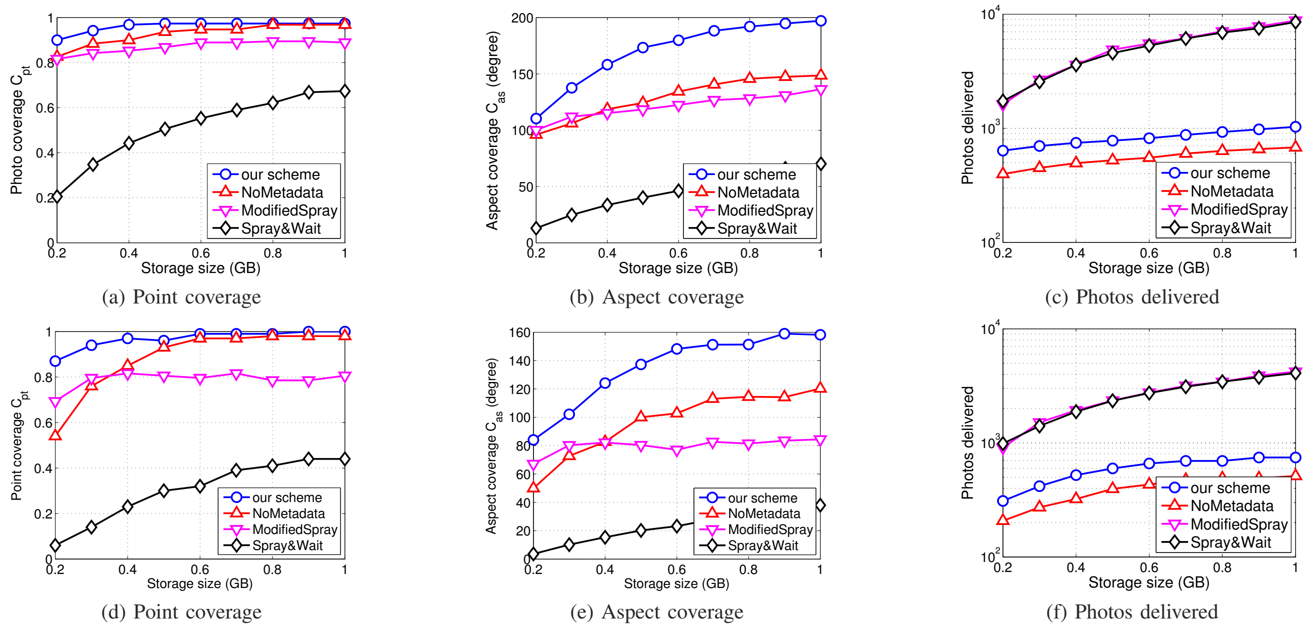
E. The Effects of the Number of Generated Photos

The number of generated photos has two opposite effects: 1) network and storage contention is exacerbated with more generated photos; and 2) photo coverage can be increased if more useful photos are generated and received by the command center.

Fig. 8 summarizes the results, where the storage size is fixed at 0.6GB. As shown in Fig. 8(a)(b)(d)(e), our scheme can significantly improve the point coverage and aspect coverage when more photos are generated. This is because our scheme is able to select an increasing amount of useful photos from an increasing amount of candidate photos, and thus the good effect (more candidate photos) overcomes the bad effect (more contention). Similarly, NoMetadata and ModifiedSpray follow the same trend because they consider photo coverage and thus can select useful photos from candidate photos. Spray&Wait fluctuates because it cannot select useful photos from candidate photos, hence suffering from the exacerbated resource contention.

Fig. 8(c)(f) shows the number of delivered photos in logarithmic scale. Again, our scheme and NoMetadata deliver much fewer photos than the other two schemes. However, these photos achieve good coverage and contain little redundancy. Consider the case where 250 photos are generated per hour in Fig. 8(c). Our scheme delivers around 800 photos to the command center. Since there are 250 PoIs in total, on average each PoI is covered by 3.2 photos. With 30° effective angle (Table I), the 3.2 photos would cover $3.2 \times 2 \times 30^\circ = 192^\circ$ if they do not overlap with each other at all. In fact, from Fig. 8(b), we can see that the actual aspect coverage on each PoI is about 180°, which shows that the redundancy between the 3.2 photos is only 12°.

To summarize the simulation results, our scheme performs close to the best possible results where no bandwidth or storage constraint is enforced. Our scheme significantly outper-



(a) Point coverage (b) Aspect coverage (c) Photos delivered

(d) Point coverage (e) Aspect coverage (f) Photos delivered

Fig. 7. The effects of storage capacity. (a)(b)(c) are based on the MIT trace and (d)(e)(f) are based on the Cambridge06 trace.

forms Spray&Wait, which represents a set of general-purpose routing protocols that do not consider packet utility in routing. Our scheme also outperforms ModifiedSpray, which is similar to previous utility-based approaches in that they prioritize packets by their individual utility, without considering the overlap (redundancy) between photos.

VI. Related Work

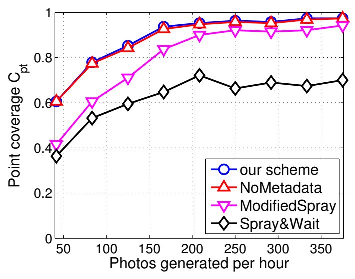
Sensor coverage has attracted lots of research [8], [9], [21], [28], from which we borrow the concept of point coverage and apply it to define our photo coverage. We also use the concept of aspect coverage proposed by Wang *et al.* in [24], [23], [25], where a point is *full-view covered* if it has 2π aspect coverage. This idea is important because it goes one step further from traditional sensor coverage to photo coverage: a photo not only shows an object, but also its front, side and back views.

To deliver photos to the command center, this research is also related to DTN routing. Early works in DTN routing assume that packets are equally important, and thus do not differentiate packets based on their content [20], [16], [14], [6]. Those works are not suitable for the scenario considered in this paper, where the value of each photo is different and must be considered in routing. More recent works consider the utility of packets and prioritize packets with high utility in routing decisions [2], [3], [13]. For example, [2] uses several variations of delay as the utility; [13] uses the average delay and the average delivery rate as the utility; and [3] routes prefetched webpages to DTN nodes, so it uses the probability that a link is actually clicked as the utility. A common property of the above utility metrics is that the utility of one packet is not explicitly affected by the presence of other packets. However, the utility metric in our work, photo coverage, is different. The coverage of a photo can change dramatically depending on the presence of other similar photos. Hence, existing utility-based approaches are not suitable for maximizing photo coverage.

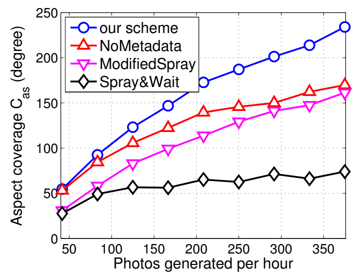
SmartPhoto [27], PhotoNet [22], and CARE [26] are three closely related papers as they also study photo crowdsourcing or photo delivery. SmartPhoto assumes that reliable communication such as cellular network is available to all users, and then develops centralized photo selection algorithms running on the server. Instead, our solution considers DTN scenarios, and thus we have to consider both photo coverage and delivery probability, and select photos in a distributed way. PhotoNet is a picture delivery service that prioritizes the transmission of photos by considering the location, time stamp, and color difference, with the goal of maximizing the "diversity" of the photos. Compared to their model, we consider direction and angle information, which are very important and unique features for photos and enable us to develop much finer optimization models. CARE leverages image similarity detection algorithms to eliminate similar-looking photos, and thus increase the delivery capability of DTNs. While those algorithms can detect similarity in terms of pixels, they do not necessarily detect similarity in terms of content or semantics. In comparison, our photo coverage model can detect a broader scope of similarity at a much lower computational cost, which is critical in resource constrained DTN scenarios.

VII. Conclusions

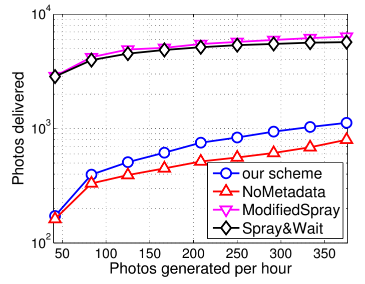
In this paper, we studied the photo crowdsourcing problem in scenarios like disaster recovery and battlefield, where the cellular network may be partly damaged or overloaded with extensive requests. As a result, photos related to points of interest have to be transmitted to the command center through DTNs. Due to storage and bandwidth limitations, nodes need to select and transmit the most useful photos, and we proposed to use photo metadata including the smartphone's location,



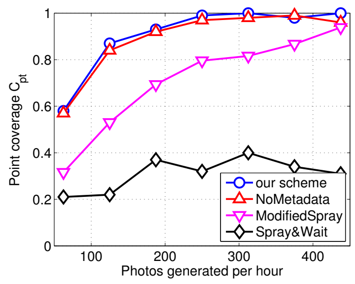
(a) Point coverage



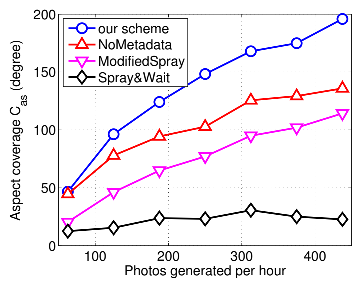
(b) Aspect coverage



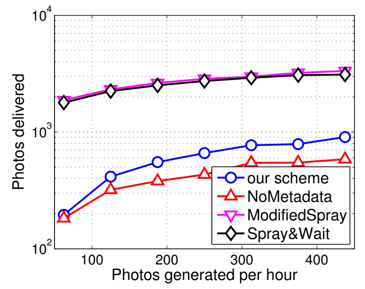
(c) Photos delivered



(d) Point coverage



(e) Aspect coverage



(f) Photos delivered

Fig. 8. The effects of the number of generated photos. (a)(b)(c) are based on the MIT trace and (d)(e)(f) are based on the Cambridge06 trace.

orientation, and other built-in camera's parameters, to estimate the value (coverage) of the photos. Then, we proposed a photo selection algorithm to maximize the photo coverage obtained by the command center. Results based on a proof-of-concept prototype and extensive simulations demonstrated the effectiveness of our design.

REFERENCES

[1] SensorManager - Android Developer. http://developer.android.com/reference/android/hardware/SensorManager.html.

[2] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Dtn routing as a resource allocation problem. In *ACM SIGCOMM*, 2007.

[3] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Enhancing interactive web applications in hybrid networks. In *ACM MobiCom*, 2008.

[4] H. Cai and D. Y. Eun. Toward stochastic anatomy of inter-meeting time distribution under general mobility models. In *ACM MobiHoc*, 2008.

[5] R. Gao, M. Zhao, T. Ye, F. Ye, Y. Wang, K. Bian, T. Wang, and X. Li. Jigsaw: Indoor floor plan reconstruction via mobile crowdsensing. In *ACM MobiCom*, 2014.

[6] W. Gao and G. Cao. On exploiting transient contact patterns for data forwarding in delay tolerant networks. In *IEEE ICNP*, 2010.

[7] W. Gao, Q. Li, B. Zhao, and G. Cao. Social-aware multicast in disruption tolerant networks. *IEEE/ACM Transactions on Networking*, 2012.

[8] X. Gong, J. Zhang, D. Cochran, and K. Xing. Barrier coverage in bistatic radar sensor networks: cassini oval sensing and optimal placement. In *ACM MobiHoc*, 2013.

[9] J. Hong, J. Cao, Y. Zeng, S. Lu, D. Chen, and Z. Li. A location-free prediction-based sleep scheduling protocol for object tracking in sensor networks. In *IEEE ICNP*, 2009.

[10] P. Jain, J. Manweiler, A. Acharya, and K. Beaty. Focus: Clustering crowdsourced videos by line-of-sight. In *ACM Sensys*, 2013.

[11] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos. User recruitment for mobile crowdsensing over opportunistic networks. In *IEEE INFOCOM*, 2015.

[12] E. Koukoumidis, L.-S. Peh, and M. R. Martonosi. Signalguru: Leveraging mobile phones for collaborative traffic signal schedule advisory. In *ACM MobiSys*, 2011.

[13] A. Krifa, C. Barakat, and T. Spyropoulos. An optimal joint scheduling and drop policy for delay tolerant networks. In *WoWMoM Workshop on Autonomic and Opportunistic Communications*, 2008.

[14] K. Lee, Y. Yi, J. Jeong, H. Won, I. Rhee, and S. Chong. Max-contribution: On optimal resource allocation in delay tolerant networks. In *IEEE INFOCOM*, 2010.

[15] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft. Opportunistic content distribution in an urban setting. In *ACM SIGCOMM Workshop on Challenged Networks*, 2006.

[16] A. Lindgren, A. Doria, and O. Schelen. Probabilistic routing in intermittently connected networks. In *ACM SIGMOBILE Mobile Computing and Communications Review*, 2003.

[17] S. B. Liu, L. Palen, J. Sutton, A. L. Hughes, and S. Vieweg. In search of the bigger picture: The emergent role of on-line photo sharing in times of disaster. In *ISCRAM*, 2008.

[18] T. Luo, S. S. Kanhere, H. P. Tan, F. Wu, and H. Wu. Crowdsourcing with tullock contests: A new perspective. In *IEEE INFOCOM*, 2015.

[19] P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, and M. Satyanarayanan. Scalable crowd-sourcing of video from mobile devices. In *ACM Mobisys*, 2013.

[20] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. In *ACM SIGCOMM workshop on Delay-tolerant networking*, 2005.

[21] P.-J. Wan, X. Xu, and Z. Wang. Wireless coverage with disparate ranges. In *ACM MobiHoc*, 2011.

[22] H. Wang, M. Uddin, G. Qi, T. Huang, T. Abdelzaher, and G. Cao. Photonet: A similarity-aware image delivery service for situation awareness. In *ACM/IEEE IPSN*, pages 135–136, 2011.

[23] Y. Wang and G. Cao. Barrier coverage in camera sensor networks. In *ACM MobiHoc*, 2011.

[24] Y. Wang and G. Cao. On full-view coverage in camera sensor networks. In *IEEE INFOCOM*, 2011.

[25] Y. Wang and G. Cao. Achieving full-view coverage in camera sensor networks. *ACM Transactions on Sensor Networks*, 10(1), 2013.

[26] U. Weinsberg, Q. Li, N. Taft, A. Balachandran, V. Sekar, G. Iannaccone, and S. Seshan. Care: Content aware redundancy elimination for challenged networks. In *ACM HotNets*, 2012.

[27] Y. Wu, Y. Wang, W. Hu, and G. Cao. Smartphoto: A resource-aware crowdsourcing approach for image sensing with smartphones. *IEEE Transactions on Mobile Computing*, 15(5), May 2016.

[28] O. Younis, M. Krunz, and S. Ramasubramanian. Coverage without location information. In *IEEE ICNP*, 2007.

[29] P. A. Zandbergen and S. J. Barbeau. Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones. *Journal of Navigation*, 64(3), 2011.

[30] H. Zhu, L. Fu, G. Xue, Y. Zhu, M. Li, and L. Ni. Recognizing exponential inter-contact time in vanets. In *IEEE INFOCOM*, 2010.