Fog-based Data Offloading in Urban IoT Scenarios

Pranvera Kortoçi*, Liang Zheng†, Carlee Joe-Wong‡, Mario Di Francesco*, and Mung Chiang§

*Dept. of Computer Science, Aalto University
†Dept. of Electrical Engg., Princeton University
‡Electrical and Computer Engg., Carnegie Mellon University
§Electrical and Computer Engg., Purdue University

***Abstract*—Urban environments are a particularly important application scenario for the Internet of Things (IoT). These environments are usually dense and dynamic; in contrast, IoT devices are resource-constrained, thus making reliable data collection and scalable coordination a challenge. This work leverages the fog networking paradigm to devise a multi-tier data offloading protocol suitable for diverse data-centric applications in urban IoT scenarios. Specifically, it takes advantage of heterogeneity in the network so that sensors can collaboratively offload data to each other or to mobile gateways. Second, it evaluates the performance of this offloading process through the amount of data successfully reported to the cloud. In detail, it provides an analytical characterization of data drop-off rates as a random process and derives a light-weight yet efficient method for collaborative data offloading. Finally, it shows that the proposed fog-based solution significantly decreases the data drop-off rate through both analysis and extensive trace-driven simulations based on human mobility data from real urban settings.**

***Index Terms*—Fog networking, collaborative offloading, data drop-off rate, Internet of Things**

I. Introduction

Our life increasingly relies on smart objects that collect data from the environment and perform actions on the physical world [1]. These objects form the so-called Internet of Things (IoT) by interacting with each other through Internet-based standards and a global communication infrastructure [2]. In turn, the IoT is the foundation for several emerging applications and services in cyber-physical systems, from intelligent transportation to the industrial Internet and smart cities [3].

Many IoT devices are deployed in cities as well as large metropolitan areas for, e.g., security surveillance, traffic and pollution monitoring, infotainment and energy management [4, 5]. This proliferation of urban IoT applications creates an unprecedented volume of data containing physical quantities sampled from the environment [6]. As IoT devices are resource-constrained and battery-powered, they generally cannot process such data locally and usually rely on the cloud to perform data analysis and long-term storage [3]. However, a long-range wireless Internet connection is either very energy hungry (as in Long Term Evolution or WiFi) or provides only a very limited bitrate (as with Narrowband IoT and LoRa) [7]. To overcome these limitations, the concept of opportunistic IoT has been introduced by applying the paradigm of delay-tolerant networking to urban scenarios [8, 9]. Accordingly, quantities measured by IoT sensors are collected by mobile gateways, e.g., cars or people carrying smartphones [10].

IoT sensors can effectively support urban applications only when they successfully transfer their data to the gateways in the first place. However, these sensors' limited storage resources prevent them from indefinitely keeping the data they generate. It is then crucial that they transfer their data to the gateways before running out of storage; this issue is referred to as the "data drop-off problem." Yet designing solutions to prevent data drop-off is very challenging: effective solutions require careful co-operation between sensors and gateways, but the sensors' limited computing and bandwidth resources preclude coordination mechanisms with significant overhead. Moreover, gateway mobility in urban IoT scenarios is often uncontrolled and unpredictable, inducing significant dynamic heterogeneity between different sensors' environments. The sensors then need to adapt to changes in gateway mobility, again without significant overhead.

Many solutions have been proposed for data collection in wireless sensor networks and opportunistic communications in urban scenarios [11]. In particular, mobile nodes have been leveraged for energy-efficient data collection in wireless sensor networks [12]. Several data offloading schemes have been devised for mobile devices too, with a particular focus on content distribution and caching [13]. However, most of these solutions assume that end devices are homogeneous and directly communicate with each other or with an Internet-connected gateway: thus, they do not fully address today's urban IoT scenarios (Section VI). Instead, this work proposes the first data offloading scheme based on a *fog networking* architecture [14] to solve the data drop-off problem.

The fog is an extension of the traditional cloud paradigm that addresses the distinctive challenges of the IoT, where multiple edge and end-user devices collaboratively carry out a substantial amount of computation, storage, communication and management [4]. Fog networking explicitly considers the resources available on each device, as well as latency, efficiency, and network cost, while attempting to meet application-specific performance requirements [15]. Thus, unlike existing sensor data collection schemes, an approach built on fog can handle the heterogeneity of dynamic environments.

This work focuses on optimizing communications in fog networks where devices have limited buffering capability. First, it leverages the fog networking paradigm to devise a **multi-tier data offloading protocol** suitable for diverse data-centric applications in urban IoT scenarios (Section II). Specifically, it takes advantage of heterogeneity in the network

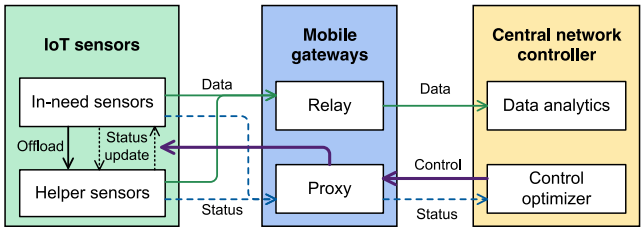


Fig. 1: Interactions in the reference three-tier fog network.

so that sensors can collaboratively offload data to each other or to mobile gateways. Second, it **quantifies the performance of this offloading process** through the amount of data successfully reported to the cloud (Section III). In particular, it derives a tractable stochastic model for the expected data drop-off rates, using this analytical characterization to adapt the offloading protocol to the prevailing environment. Finally, it shows that the proposed fog-based solution significantly decreases the data drop-off rate through **both analysis and extensive trace-driven simulations** based on human mobility data from real urban settings (Section V).

II. Background

This section introduces the reference fog architecture employed by the proposed solution, then details its system model.

A. Reference Architecture

A three-tier fog network architecture is considered in this work [14], as illustrated in Figure 1. The bottom tier comprises *IoT sensors*, namely, wireless embedded devices with limited battery, computing, and storage resources. Sensors are static, periodically collecting data from the environment through short- or medium-range communication technologies (such as Bluetooth Low Energy [16, 17]) and storing it in a local buffer. They forward the data to other sensors or to the intermediate tier based on their buffer occupancy and communication opportunities. The intermediate tier of the network consists of *mobile gateways*, namely, mobile devices in the urban area (for instance, smartphones carried by people or vehicles). These gateways have two radio transceivers, one for exchanging data with the IoT sensors and another for Internet connection (e.g., Long Term Evolution or WiFi). They receive data from the IoT sensors and forward them to the top tier located in the cloud. A *central network controller* therein stores the collected messages and carries out data analytics. The controller also performs network management by dynamically reconfiguring the devices in the lower tiers.

The interactions between the system components are illustrated in Figure 1. To reduce coordination overhead between the sensors and gateways, sensors are logically divided into *in-need* and *helper* categories, based on their buffer availability and expected probability to be in contact with a gateway over time. In-need sensors are rarely in reach of a mobile gateway, resulting in a high data drop-off rate due to unavailable buffer space. In contrast, helper sensors encounter mobile gateways more frequently and can temporarily store the data collected by other sensors in their own buffers. Accordingly, in-need sensors offload their data to helper sensors; individual sensors may switch between categories as they learn about contact opportunities with the mobile gateways. These gateways collect messages from all sensors as well as status information (including contact opportunities) and forward them to the central network controller. The latter processes the received data and sends control messages back to the sensors through the mobile gateways. In particular, the central network controller dynamically partitions IoT sensors into in-need and helpers. Section IV proposes methods for this partitioning based on an analysis of the resulting data drop-off rates.

TABLE I: Summary of used notation.

Symbol	Definition
$\mathcal{S}$	Set of M sensors $\mathcal{S} = \{s_1, \ldots, s_M\}$
$\mathcal{U}$	Set of N mobile gateways $\mathcal{U} = \{u_1, \ldots, u_N\}$
B	Buffer size of each sensor
d	Size of individual data items sampled in a time slot
L	Maximum number of data items that fit in the buffer
$p_m(t)$	Probability that sensor m meets a gateway in time slot t
φ_m	Probability of data drop-off for sensor m
$b_m(t)$	Buffer occupancy of sensor m at time slot t
$\mathcal{H}$	Set of K helper sensors $\mathcal{H} = \{s_1, \ldots, s_K\} \subset \mathcal{S}$
$\mathcal{I}$	Set of $M-K$ in-need sensors $\mathcal{I} = \{s_{K+1}, \ldots, s_M\} \subset \mathcal{S}$
ϵ	Threshold of data drop-off rate to distinguish between helper and in-need sensors
δ	Amount of data items offloaded from an in-need sensor to a neighboring helper sensor at once

B. System Model

The network includes the set $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ of $M = |\mathcal{S}|$ static IoT sensors deployed at fixed locations, as well as the set $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ of $N = |\mathcal{U}|$ mobile gateways. The mobility of the gateways is exogenous to the system [12]. Sensors may not all have physical neighbors, however, each group of connected sensors is visited at least once by a mobile gateway during the lifetime of the network [18], which is realistic for densely populated urban scenarios [17].

Each sensor has a buffer of size B and periodically samples items of $d \ll B$ bytes each. Sensors operate on a cyclical schedule. The time between two consecutive cycles is called a *time slot*. The size of a time slot is fixed and such that a sensor can transmit all the data stored in its buffer to a gateway within a single contact; this is a reasonable assumption given currently available technologies (see [19]). The buffer occupancy $b_m(t) \leq B$ is defined as the amount of data stored by sensor m at time slot t. Sensors do not have any prior knowledge about the gateways' mobility patterns, so $p_m(t)$ is used to express the probability[1] that sensor m meets any of the N gateways at time slot t, which can be updated over time. These probabilities are considered independent across different groups of sensors due to the unpredictable mobility of the gateways. Moreover, the gateways have sufficient resources to

[1]For clarity, the dependency on the time slot is dropped from the notation when there is no ambiguity.

store and forward the data received from the sensors to the cloud [9, 20] without any data loss.

Table I summarizes the notation used in this work. Unless otherwise specified, time and data size are measured in seconds and bytes, respectively.

III. Analysis of Data Drop-off Rates

This section characterizes the communications between IoT sensors and mobile gateways as a random process. For the sake of clarity in the exposition, the simple case of no offloading is addressed first; the analysis is then extended to cover collaborative data offloading.

A. Baseline Scheme: No Offloading

The following discussion derives first the probability of meeting a mobile gateway at a certain time slot as a function of the data generation process. It then characterizes the data drop-off rate when sensors do not collaborate with each other.

The probability p_m of meeting a gateway at time slot t is:

$$p_m(t) = \left(1 - \frac{\lambda}{t}\right) p_m(t-1) + \frac{\lambda}{t} \mathbb{1}^t_{\{s_m \to u_n\}}, \qquad (1)$$

where $\lambda \in (0, 1]$ encodes the relative weight on the probability of meeting a gateway recently compared to further in the past, while $\mathbb{1}^t_{\{s_m \to u_n\}}$ is equal to one if sensor m is in contact with a gateway at time slot t and zero otherwise. Eq. (1) can be rewritten in the equivalent form of $p_m(t) = (1-\lambda)p_m(t-1) + \lambda \hat{p}_m(t)$, where $\hat{p}_m(t) = \big((t-1)p_m(t-1) + \mathbb{1}^t_{\{s_m \to u_n\}}\big)/t$ is the empirical probability of meeting a gateway before time slot t. The analysis below assumes that the $p_m(t)$ have converged to a probability p_m.

Figure 2 illustrates the state transitions of sensor m without any cooperation mechanisms. The sensor sends all sampled data in its buffer to a mobile gateway once in reach; sampled data are otherwise stored in the local buffer until it is filled. If the buffer is full, older data are dropped and replaced with more recent data until a gateway is met. In particular, $L = B/d$ is the maximum number of time slots[2] a sensor can wait for a gateway without incurring a buffer overflow. Once the buffer occupancy hits Ld, it remains in this state until a gateway is met, after which the occupancy resets to zero. The state transitions in Figure 2 form an ergodic Markov chain [21].

The data drop-off rate at a given time t can be derived as the probability of buffer overflow according to:

$$\varphi_m = p_m(1-p_m)^L \sum_{t=1}^{\infty} (1-p_m)^t = (1-p_m)^{L+1}. \qquad (2)$$

Correspondingly, the expected amount of dropped data is:

$$d^{\varphi}_m = p_m(1-p_m)^L \sum_{t=1}^{\infty} t(1-p_m)^t d = \frac{(1-p_m)^{L+1}}{p_m} d. \qquad (3)$$

An infinite buffer size $B = +\infty$ leads to $\varphi_m = 0$ in Eq. (2) and $d^{\varphi}_m = 0$ in Eq. (3) irrespective of the actual p_m. Since the buffer size B is finite in practice, however, the amount of dropped data is determined by the value of p_m.

[2]It is reasonable to assume $\lfloor B/d \rfloor \simeq B/d$ due to $d \ll B$.

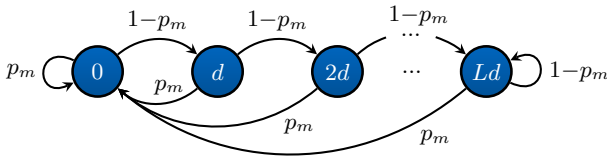


Fig. 2: State transitions for sensor m without data offloading. Each circle signifies an independent state of buffer occupancy, while arrows represent the probability of changes in the buffer occupancy between two consecutive time slots. The probabilities at the bottom of the figure realize when the sensor meets a gateway, while those on the top realize when the sensor adds a new item of size d to its buffer. The rightmost arrow denotes buffer overflows, implying that the buffer occupancy does not change (as it is equal to the buffer size).

The buffer occupancy in the steady-state can be derived following the ergodicity of the Markov chain in Figure 2. In particular, the expected buffer change between two adjacent time slots can be reformulated as $\mathbb{E}\left[b_m(t+1) - b_m(t)\right] = (1-\varphi_m)\big((1-p_m)d - p_m\mathbb{E}\left[b_m(t)\right]\big) + \varphi_m\big(-p_m\mathbb{E}\left[b_m(t)\right]\big)$. Solving this equation leads to

$$\lim_{t\to\infty} \mathbb{E}\left[b_m(t)\right] = \left(1 - (1-p_m)^{L+1}\right) \frac{1-p_m}{p_m} d, \qquad (4)$$

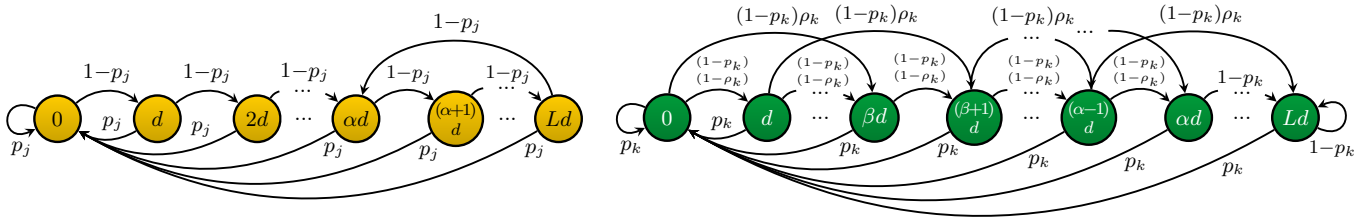
which clearly decreases with p_m. Moreover, $\sum_{t=1}^{\infty} t(1-p_m)^{t-1}p_m = 1/p_m$ is the average number of time slots that a sensor must wait between consecutive contacts with a gateway. Thus, a sensor with smaller p_m is expected to have a higher expected data drop-off rate.

B. Data Offloading for IoT Scenarios

The analysis in the previous section assumed no collaboration between sensors (i.e., no offloading). However, IoT scenarios are highly heterogeneous; sensors have different buffer availability and likelihood to meet a mobile gateway over time, since some of them may be deployed in locations visited more frequently by the gateways. We use these individual characteristics to divide sensors into two categories: in-need and helpers, as discussed in Section II-A.

Formally, the set of helper sensors is denoted as $\mathcal{H} = \{s_1, s_2, \ldots, s_K\}$. The set of all other sensors (i.e., in-need sensors) is indicated as $\mathcal{I} = \{s_{K+1}, s_{K+2}, \ldots, s_M\}$ ($K \leq M$, $\mathcal{I} \cap \mathcal{H} = \emptyset$ and $\mathcal{I} \cup \mathcal{H} = \mathcal{S}$). Helper and in-need sensors are referred to with indices k and j, respectively. In addition, the subset of in-need sensors that are neighbors of the helper sensor k is indicated by $I_k \subset \mathcal{I}$; similarly, the subset of helper sensors neighboring in-need sensor j is denoted by $H_j \subset \mathcal{H}$. Due to the assumption on network connectivity (see also Section II-B), all sensors are either in-need or helpers for at least one or possibly multiple other sensors.

The following derives the probability that an in-need sensor offloads data and a helper sensor receives the data. The discussion here considers static parameters; Section IV-A presents dynamic adaptation of network-related parameters over time.



(a) In-need

(b) Helper

Fig. 3: State transitions for a sensor performing collaborative offloading by role: (a) in-need and (b) helper, with the same notation used in Figure 2. For the sake of readability, buffer occupancy is indicated through auxiliary variables $\alpha = L - \delta$ and $\beta = \delta + 1$. The probabilities corresponding to the top-most transitions in the diagrams realize when data is offloaded from an in-need to a helper sensor.

1) In-need sensors: An in-need sensor only offloads data when it cannot store any further sensed items since its buffer is full. If this happens, the sensor broadcasts the least recent δd amount of data to its neighbors and removes the data from its buffer. It then takes another consecutive δ time slots to fill the buffer again. This is described by the last δ circles in Figure 3a. Thus, the probability that the buffer at the in-need sensor j is full is given by:

$$\Pr(b_j = Ld) = p_j(1-p_j)^L \sum_{t=0}^{\infty}(1-p_j)^{(\delta+1)t} = \frac{p_j(1-p_j)^L}{1-(1-p_j)^{\delta+1}}.$$

At each time slot, a helper sensor stores the data sent by an in-need sensor, as long as it is not currently transmitting to a gateway and has enough buffer to fit the offloaded data. The rest of the data transmitted by in-need sensors is either received by other helper sensors or dropped. Since all in-need sensors are independent, the probability of in-need sensor j successfully offloading its data to helper sensor k is given by:

$$\begin{aligned}&\Pr(s_j \to s_k) = \\ &(1-p_k)\Pr(b_k \le (L-\delta)d) \sum_{n=0}^{|I_k \setminus s_j|-1} \sum_{[I_k \setminus s_j]^n \subset I_k \setminus s_j} \frac{1}{n+1} \\ &\prod_{s_m \in [I_k \setminus s_j]^n} \Pr(b_m = Ld) \prod_{s_i \in \{I_k \setminus s_j - [I_k \setminus s_j]^n\}} \big(1 - \Pr(b_i = Ld)\big),\end{aligned} \tag{5}$$

where $|I_k \setminus s_j|$ and $[I_k \setminus s_j]^n$ are the cardinality and n-subset of $I_k \setminus s_j$, respectively. The derivation of $\Pr(b_k \le (L-\delta)d)$ in Eq. (5) is discussed in the next subsection. In this communication process, data is dropped only when no neighboring helper is available to store it. The data drop-off rate for in-need sensor j is then derived according to the following proposition.

Proposition 1: The steady-state data drop-off rate for in-need sensor j is given by:

$$\varphi_j = \Pr(b_j = Ld) \prod_{s_m \in H_j} \big(1 - \Pr(s_j \to s_m)\big).$$

2) Helper sensors: A helper sensor k has neighboring in-need sensors with data to offload with probability

$$\rho_k = 1 - \prod_{s_j \in I_k} \big(1 - \Pr(b_j = Ld)\big).$$

Figure 3b shows that helper sensor k is able to store the δd amount of offloaded data (and keep it together with its own newly-sensed data item of size d) when it has enough available buffer to offer help, i.e., $b_k \le (L-\delta-1)d$. In fact, there is a probability of $(1-p_k)\rho_k$ that helper sensor k adds $(\delta+1)d$ amount of data to its own buffer. However, the helper sensor does not collect any data from in-need sensors when $b_k \ge (L-\delta)d$; thus, the increase in buffer occupancy is only due to its own sensing task with probability $1-p_k$. This creates the following dynamic system for the probabilities associated with the buffer state of helper sensor k:

$$\begin{aligned}&\Pr(b_k = \tau d) = \\ &\begin{cases} p_k\big((1-p_k)(1-\rho_k)\big)^{\tau}, & \text{if } \tau \le \delta \\ (1-p_k)(1-\rho_k)\Pr(b_k = (\tau-1)d) & \\ \quad +(1-p_k)\rho_k \Pr(b_k = (\tau-\delta-1)d), & \text{if } \delta < \tau \le L-\delta \\ (1-p_k)\Pr(b_k = (\tau-1)d) & \\ \quad +(1-p_k)\rho_k \Pr(b_k = (\tau-\delta-1)d), & \text{if } L-\delta < \tau \le L-1, \end{cases}\end{aligned}$$

The steady-state probabilities of the buffer states at $\delta < \tau \le L-\delta$ are solved according to the following lemma.

Lemma 1: The probability that the buffer of helper sensor k contains τd data at the steady-state is given by

$$\begin{aligned}\Pr(b_k = \tau d) = p_k \sum_{i=0}^{\lfloor \frac{\tau}{\delta+1} \rfloor} &\binom{\tau - \big(\lfloor \frac{\tau}{\delta+1} \rfloor - i\big)\delta}{\lfloor \frac{\tau}{\delta+1} \rfloor - i} \big((1-p_k)\rho_k\big)^{\lfloor \frac{\tau}{\delta+1} \rfloor - i} \\ &\big((1-p_k)(1-\rho_k)\big)^{\tau - (\lfloor \frac{\tau}{\delta+1} \rfloor - i)(\delta+1)}\end{aligned} \tag{6}$$

for $\delta + 1 \le \tau \le L - \delta$.

Eq. (6) can be better understood by recognizing that the first term within the summation is the binomial coefficient for counting the permutations for which the buffer grows by an amount d for $\lfloor \frac{\tau}{\delta+1} \rfloor - i$ times and by an amount $(\delta+1)d$ for $\tau - \big(\lfloor \frac{\tau}{\delta+1} \rfloor - i\big)(\delta+1)$ times. As such, the data drop-off rate is equivalent to the probability of not meeting a gateway after the buffer reaches Ld:

Proposition 2: The steady-state data drop-off rate for helper sensor k is given by

$$\varphi_k = \frac{1-p_k}{p_k} \sum_{\tau=L-\delta}^{L} (1-p_m)^{L-\tau+1} \rho_k \Pr(b_k = (\tau-\delta-1)d)$$

where $\frac{1-p_k}{p_k}$ is calculated from $\sum_{t=1}^{\infty}(1-p_k)^t$ and $\Pr(b_k = (\tau-\delta-1)d)$ is obtained from Eq. (6).

IV. Fog-based Control of Data Offloading

This section presents an adaptive data offloading strategy and a supporting protocol to dynamically change network-related parameters over time.

A. Adaptive Network Control

The central network controller calculates two network-wide offloading parameters, ϵ and δ, which express the likelihood of a sensor to drop data and the number of data items a sensor offloads, respectively. Sensors locally update the probability p_m to meet a gateway and whenever a sensor meets a gateway, such a value is then reported to the central network controller. In turn, the central network controller updates ϵ and δ based on the considerations detailed next.

As helper sensor k is expected to meet a gateway in $1/p_k$ time slots, the expected fraction of the buffer needed to store its own data should be less than the total size (i.e., $1/p_k < L$) to ensure that these helper sensors will have some available buffer to receive offloaded data from in-need sensors. To guarantee that this condition is met, the threshold probability p_m in Eq. (10) should be larger than $1/L$.

Lemma 2: The threshold ϵ of the data drop-off rate for a helper sensor must satisfy

$$\epsilon < \left(1-\frac{1}{L}\right)^{L+1}. \tag{7}$$

Observing Eq. (7) reveals that the condition on ϵ is relaxed with larger values of L, meaning that a larger buffer size improves the data drop-off rate, as expected.

With similar considerations to those leading to Lemma 2, the amount of offloaded data should be less than the expected available buffer – i.e., $B - \lim_{t\to\infty}\mathbb{E}[b_k(t)] \geq \delta d$ – at any helper sensor k, to ensure that the buffer of a helper sensor can accommodate the data received from at least one in-need sensor. Combining the expression above with $p_k > 1/L$ for the helper sensors and substituting Eq. (4) leads to the condition in the following lemma.

Lemma 3: The amount of data δd offloaded to helper sensors must satisfy:

$$\delta < 1 + (L-1)\left(1-\frac{1}{L}\right)^{L+1}.$$

The central network controller chooses the values of ϵ and δ, subject to the constraints of Lemmas 2 and 3, based on the p_m values reported by the gateways. These values should be chosen carefully: if ϵ is too large, only a few sensors would be allowed to offload their data, and some helper sensors might drop data due to not meeting gateways frequently enough. As ϵ decreases, however, some sensors which could have stored offloaded data may instead be designated as in-need sensors, again resulting in dropped data. Similarly, a large value of δ may lead to frequent buffer overflows, whereas a small value may trigger in-need sensors to offload too much data.

Algorithm 1 Data offloading protocol run by sensor m

```
1  foreach time slot t do
2      if a gateway is in range then
3          p_m ← (1 − λ/t) p_m + λ/t
4          transmit all data and p_m to the gateway
5          obtain new ε and δ from the gateway
6      else
7          p_m ← (1 − λ/t) p_m
8          if p_m ≤ 1 − ε^{1/(L+1)} and b_m(t) = B then
9              extract and broadcast δd data // IN-NEED
10         else if p_m > 1 − ε^{1/(L+1)} and b_m(t) < (L − δ)d then
11             store received δd data // HELPER
```

A simple and intuitive way to divide the sensors into in-need and helpers is to set ϵ to be the average drop-off rate of all sensors without data offloading, according to Eq. (2). However, this choice can lead to a disparity between the numbers of helper and in-need sensors. For instance, if most sensors experience a very small p_m, employing the average would lead to too few helper sensors. Consequently, ϵ is set to be the *harmonic* mean of the probabilities of buffer overflow:

$$\epsilon = \min\left\{\frac{M}{\sum_{s_m\in\mathcal{S}}(1-p_m)^{-(L+1)}}, \left(1-\frac{1}{L}\right)^{L+1}\right\}, \tag{8}$$

where the minimum operator ensures that the condition derived in Lemma 2 is satisfied. Note that the harmonic mean enables a conservative decision (i.e., ϵ is likely to be small) as the expression in Eq. (2) is dominated by its small arguments. Furthermore, δ is set to the ratio between the expected buffer available at helper sensors and the data dropped by in-need sensors [refer to Eq. (3) and Eq. (4) for more details]:

$$\delta = \min\left\{\left\lfloor\frac{\sum_{s_k\in\mathcal{H}}\left(L-\left(1-(1-p_k)^{L+1}\right)\frac{1-p_k}{p_k}\right)}{\sum_{s_j\in\mathcal{I}}\frac{(1-p_j)^{L+1}}{p_j}}\right\rfloor, 1+(L-1)\left(1-\frac{1}{L}\right)^{L+1}\right\}. \tag{9}$$

Here p_k and p_j are the probabilities that a helper and an in-need sensor meet a gateway when ϵ is set by Eq. (8). Thus, the expected available buffer capacity for offloaded data is just equal to the amount of offloaded data.

B. Data Offloading Process

The data offloading process is carried out locally at individual sensors as detailed in Algorithm 1. When a gateway is in range (lines 2-5), the sensor first updates its observed probability p_m of encountering a gateway according to Eq. (1). It then transfers p_m and all buffered data to the gateway, which

replies with the updated values of the operating parameters, ϵ and δ. When no gateway is in range (lines 6-11), the sensor similarly updates its observed probability p_m. It then derives its class (i.e., *in-need* or *helper*) based on the its last known value of the parameter ϵ and acts accordingly. Namely, a sensor is in-need when $\varphi_m \geq \epsilon$, i.e., the probability of a buffer overflow from Eq. (2) is greater than ϵ. Equivalently, we have:

$$p_m \leq 1 - \epsilon^{\frac{1}{L+1}}. \tag{10}$$

In this case, the sensor extracts an amount of δd data from its buffer and broadcasts it to all neighboring helper sensors when its buffer is full (lines 8-9). Otherwise, the sensor is a helper, thus, available to store the data broadcasted by other sensors as long as there is enough room in its buffer (lines 10-11).

For the sake of simplicity, the data offloading process in Algorithm 1 is explained as a set of operations repeated at every time slot. In practice, gateway discovery and sensor cooperation could be performed asynchronously. Moreover, IoT sensors could employ power management mechanisms to save energy for communication. In practice, sensors could use Bluetooth Low Energy, which is particularly suitable for opportunistic scenarios [19, 20]. In fact, it is robust to interference; it has a very low power consumption and high-enough bitrate. According to [19], a Bluetooth Low Energy sensor can transfer a few megabits of data for several years despite the overhead of neighbor discovery.

V. Performance Evaluation

A performance evaluation of the proposed data offloading scheme is conducted next. First, the data drop-off rates are evaluated numerically based on the analytical model of the fog network in Section III. Next, experimental results under dynamic network conditions are performed through trace-driven simulations in a realistic urban scenario. In all cases, $\lambda = 0.01$ and ten replications are performed. The figures report the average values over all sensors and the ten runs along with the related standard deviations as error bars when meaningful.

A. Numerical Results

The following evaluates the data drop-off rate in the network by applying the analytical model in Section III-B with $L = 10$. A network with a varying number of sensors is considered based on the following representative scenarios, which express different levels of sensor heterogeneity in terms of their connectivity with mobile gateways.

- *Uniform*: all sensors have a probability p_m to meet a gateway uniformly distributed between zero and one.
- *Helper bias*: 80% of the sensors have a probability p_m to meet a gateway uniformly distributed between 0.3 and 0.5, while $p_m < 0.2$ for all other sensors. This setting induces a strong bias towards sensors being helpers.
- *In-need bias*: 80% of the sensors have a probability p_m to meet a gateway below 0.2, while p_m is uniformly distributed between 0.3 and 0.5 for all other sensors. This setting induces a strong bias towards sensors being in-need rather than helpers.

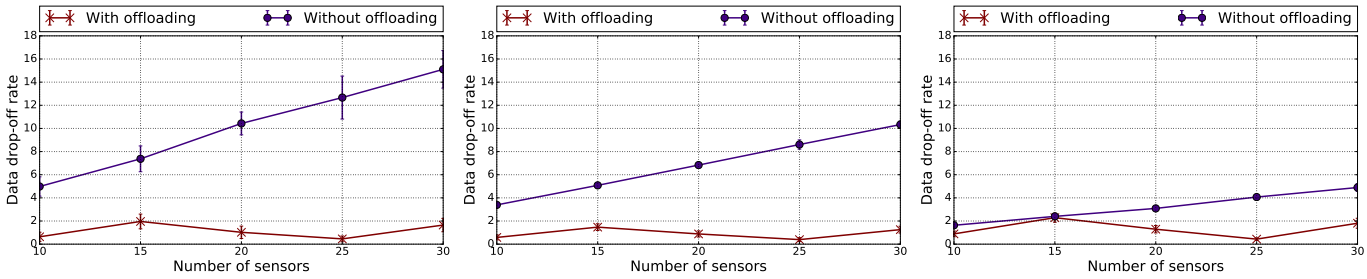
Figure 4 illustrates the data drop-off rate as a function of the number of sensors for the different scenarios. All figures show the baseline approach when no offloading is employed (i.e., when individual sensors transfer their own data directly to the gateway) as well as the collaborative data offloading approach. Clearly, the proposed scheme outperforms no offloading, achieving data drop-off rates consistently below 2.5% as compared to drop-off rates up to 15% without offloading. Moreover, collaborative offloading exhibits almost constant data drop-off rates with few fluctuations as the number of sensors increases, while the drop-off rate without offloading increases with the number of sensors. However, these fluctuations are smoother for the helper bias scenario, implying that the proposed data offloading protocol scales better to larger deployments by leveraging network heterogeneity.

The network-wide offloading parameters ϵ and δ are shown in Figures 5a and 5b, respectively, as a function of the number of sensors and for different scenarios. Recall that sensors with drop-off rates below the threshold ϵ are designated as helper sensors. This is reflected in the results reported in Figure 5a: the availability of more helpers results in lower values of ϵ, since setting a lower threshold will still leave many helper sensors. The figure also shows that ϵ is less sensitive to the network size when there are more helper sensors. In fact, setting a low value of ϵ always yields enough helper sensors to offload data in this case, even if there are fewer total sensors in the network. Similarly, the presence of more helpers allows sensors to maintain lower buffer occupancies overall, which in turn allows in-need sensors to offload more data (i.e., a higher parameter δ). This trend is apparent from Figure 5b, where the scenarios with more helpers incur in higher values of δ.

B. Trace-based Simulations

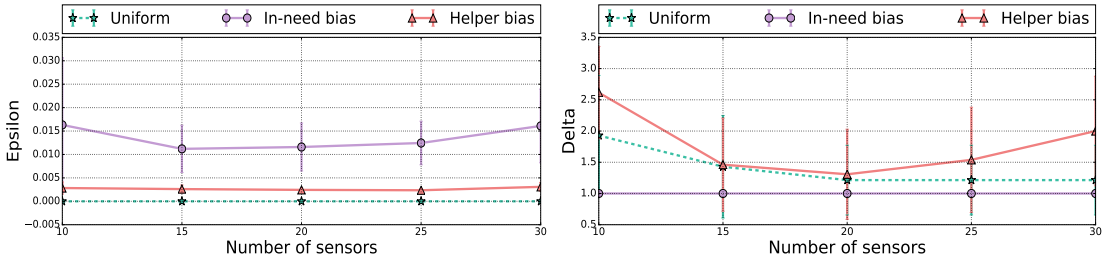
A custom Python simulator was employed to assess the performance of the proposed data offloading scheme according to the protocol presented in Section IV-B (i.e., Algorithm 1) in dynamic conditions. The considered urban IoT scenario is represented by a varying number of sensors randomly deployed in a metropolitan area of 4.5 by 3.5 kilometers and two gateway densities: 50 and 100 mobile gateways. The ONE simulator v1.6.0 was employed to generate mobility traces based on the roads in the city of Helsinki and pedestrians walking with a speed between 0.5 and 1.5 m/s along streets as well as pedestrian paths. All sensors used a transmission range of 100 m, a buffer size $B = 5,000$, and items of size $d = 10$. All sensors were initialized with the same value of $p_m = 0.002$ to be in-need at the beginning of the simulation; each sensor then dynamically updated the observed probability of meeting a gateway over time and set their role accordingly. The length of the time slot was set to one second and the experiments lasted for 5 hours of simulated time.

The proposed collaborative offloading approach is compared against three other schemes: an approach with no offloading (as a baseline); a *naive* approach wherein sensors try to offload one data item every time their buffer is full; and a *static* approach wherein sensors try to offload 20% of their buffer



(a) Uniform (b) 80% helpers (c) 80% in-need

Fig. 4: Data drop-off rate as a function of the number of nodes in the network for three representative scenarios: (a) uniform distribution of in-need and helper sensors, (b) 80% helper sensors, and (c) 80% in-need sensors.



(a) (b)

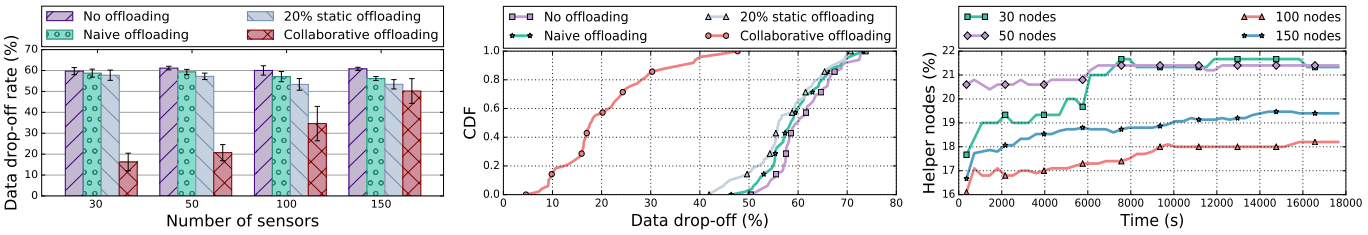
Fig. 5: Network-wide offloading parameters: (a) likelihood ϵ to drop data and (b) number δ of offloaded data items as a function of the number of sensors in the network for the considered scenarios.

size, irrespective of the local network conditions. Data in the naive and static approaches is assumed to be dropped if there is no nearby gateway or sensor with available buffer.

1) Data Drop-off Rate: Figures 6a and 7a illustrate the data drop-off rate as a function of the network size (i.e., the number of sensors) and two gateway densities for the four considered offloading schemes. Note that the data drop-off rate without offloading is roughly constant irrespective of the network size, whereas it slightly decreases with the number of deployed sensors when naive and static offloading (20% of the full buffer) is performed. Given a gateway density, the gap between the data drop-off rate without and with naive (or static) offloading increases with the number of deployed sensors. A higher number of sensors leads to a higher chance for a single sensor to have neighbors that, in turn, would have a higher chance to meet a gateway – in other words, neighbors able to help. The static offloading scheme performs better than the naive one for the case with 50 gateways, while there is no significant difference between the two schemes for the case with 100 gateways as a result of sensors' buffers being less full. The data drop-off rate with collaborative offloading is clearly well below the data drop-off rate of the other three schemes. However, the drop-off rate increases with the number of sensors for the collaborative schemes. This trend can be explained based on the clustering of sensors with respect to gateway movements. An analysis of the simulation data revealed that gateways tend to frequent specific locations (e.g., cross intersections) much more than others. As the network size increases, a smaller fraction of sensors is located near these spots, leading to a faster increase of in-need sensors compared to helpers and a larger data drop-off rate. However, as observed in Figure 7a, the collaborative offloading scheme performs better in networks with higher gateway density, since a larger number of gateways still allows sensors to have more contact opportunities. More data can then be offloaded to helper sensors, leading to lower data drop-off rates.

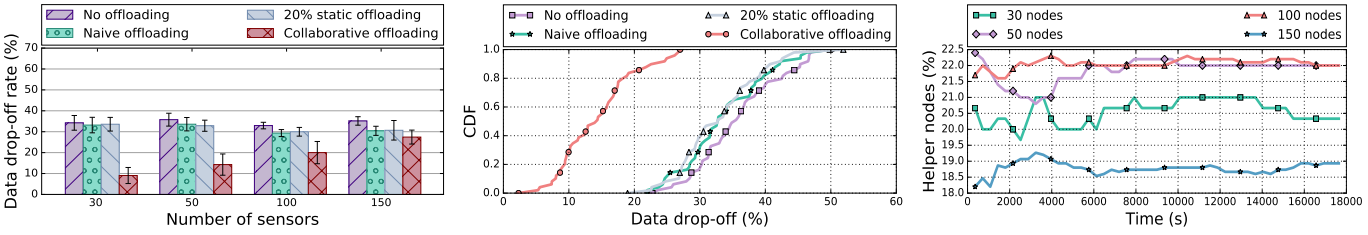
Figures 6b and 7b illustrate the CDF (cumulative distribution function) of the data drop-off rate across the sensors for a fixed network size of 50 sensors and varying gateway densities. Such a CDF is shown for four different schemes: without, naive, static, and collaborative offloading. Figure 6b shows that collaborative offloading with 50 gateways results in most of the sensors having a data drop-off rate below 20%, whereas no, static and naive offloading lead to data drop-off rates as high as 55% for most of the sensors. Moreover, as seen in Figure 7b, as the gateway density reaches 100, the data drop-off rate decreases for all the four cases. However, the collaborative offloading scheme still outperforms the other three, offering low data drop-off rates of less than 15% for most of the sensors and drop-off rates of less than 30% for all of them. The large gap in the data drop-off rate holds for all the sensors in the network. The performance of the naive and static offloading scheme closely follows the no offloading one, with only slightly lower data drop-off rates.

2) Fraction of Helper vs. In-need Sensors: Figures 6c and 7c show the fraction of the helper sensors as a function of simulated time for different values of the network size and gateway densities in the network. According to the



(a) (b) (c)

Fig. 6: (a) Drop-off as a function of the number of sensors in the network from the trace-driven simulations, (b) the CDF of the data drop-off rate, and (c) the fraction of helper sensors over time for 50 gateways.



(a) (b) (c)

Fig. 7: (a) Drop-off as a function of the number of sensors in the network from the trace-driven simulations, (b) the CDF of the data drop-off rate, and (c) the fraction of helper sensors over time for 100 gateways.

initialization process described above, all sensors are in-need at the beginning of the simulation. Later, they start reporting the observed values of p_m to the gateways; in turn, the central network controller recalculates and disseminates the offloading parameters ϵ and δ to the network. Accordingly, sensors change their role over time: while they initially tend to offload data often, they eventually become helpers [Eq. (10)].

In fact, the collaborative offloading process relies on sensors that *learn* and *update* their status accordingly – the fraction of helper sensors for both gateway densities slowly increases over time. Note that the fraction of helpers increases faster for lower network densities. Such a trend occurs as more sensor nodes in the network lead to a higher number of sensors per cluster. Consequently, one helper sensor would have to serve more in-need sensors, resulting in a rapid increase of buffer occupancy and a lower capability to remain a helper sensor for a long time. Moreover, comparing Figures 6c and 7c reveals that a higher gateway density leads to a higher fraction of helper sensors for all network densities. This derives directly from the fact that when more gateways are present, in-need sensors are more likely to meet a gateway, possibly triggering them to act as helpers. In fact, for a gateway density of 50, the fraction of helper sensors is roughly between 16 and 21.5% (Figure 6c); whereas for 100 gateways, the fraction of helper nodes is approximately between 18 and 22.5% (Figure 7c).

VI. Related Work

Data offloading has been widely studied in opportunistic communication scenarios [22]. For instance, Li et al. [23] proposed an optimal offloading strategy under buffer constraints for heterogeneous end-users. After showing that the considered problem is NP-hard, they devise and evaluate several offloading heuristics. However, they consider data dissemination from the infrastructure to the mobile devices, while the focus of this work is on sending data from devices to the cloud infrastructure, resulting in different data drop-off patterns. Lu et al. [13] proposed collaborative data offloading to address the reliability of communications from mobile devices to the infrastructure. This article similarly targets reliable data communication through device cooperation, but our solution explicitly targets collaboration between static sensors visited by mobile gateways, rather than between mobile devices.

This work is very related to data collection in wireless sensor networks by means of mobile sinks [12, 24, 25]. In this context, Gao et al. [18] attempted to maximize the data collected by mobile sinks by designating nearby nodes as intermediate data collectors. Their solution assumes that nodes have enough buffer to store all data, whereas sink mobility is fixed and known in advance. Similarly, Maia et al. [26] rely on more powerful nodes to store replicated data, which are then collected by mobiles sinks. Wen et al. [27] constructed an energy-aware path for mobile sinks to collect data from sensors, under the assumption that the location of the sensors and of the mobile sinks are known. In contrast, this article specifically addresses potential buffer overflows and attempts to learn the gateway arrivals over time. A few works have focused on learning the sink mobility pattern to improve sensor data collection. For instance, Shah et al. [28] employed distributed reinforcement learning for sensor nodes to predict arrivals of a mobile sink for sparse scenarios where at most one node is simultaneously in contact with the mobile sink. Our work instead addresses scenarios where groups of nodes

can collaboratively forward data to mobile gateways. Pozza et al. [29] considered opportunistic IoT scenarios by means of a prediction framework based on temporal-difference learning, but do not address communication reliability.

Finally, this work shares some similarities with the literature on mobile crowdsensing [30]. Jin et al. [31] addressed the deployment of mobile access points to support mission-oriented sensing, but did not leverage mobile users for sensor data collection, as done here. Xiao et al. [32] considered mobile users that perform pre-assigned sensing tasks with a certain probability depending on their mobility relative to specific points of interest. Our work does not consider pre-determined sensing tasks carried out by mobile users, but rather periodic data reporting by IoT devices.

VII. Conclusion

Fog networking brings forth new opportunities to improve the efficiency of IoT scenarios with resource-constrained devices by locating functionalities close to the network edge. In such a context, this work studies a multi-tier IoT network composed of sensors, mobile gateways, and a network controller. It leverages the fog networking paradigm to introduce a collaborative data offloading scheme that is adaptive and network-driven while minimizing the data drop-off rate. The proposed data offloading protocol accounts for the unpredictable mobility of gateways, allowing the sensors to reconfigure their roles according to their real-time data drop-off rates. This collaborative data offloading is shown to significantly reduce the data drop-off rates in the considered IoT scenario, by both numerical results and trace-driven simulations.

Future research might introduce a new control parameter to trade off between energy expenditure and drop-off rates for extremely low-power sensors. More broadly, one may utilize a similar collaborative architecture for data pre-processing or other computing needs within a fog context.

Acknowledgments

This work was partially supported by: the Academy of Finland under grants number 299222 and 319710; the US National Science Foundation under grants CNS-1751075, CNS-1759652, and CNS-1759655; and the Defense Advanced Research Projects Agency (DARPA) DCOMP program under contracts number HR001117C0052 and HR001117C0048.

References

[1] D. Miorandi, S. Sicari, F. D. Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, *Fog Computing: A Platform for Internet of Things and Analytics*, pp. 169–186. Springer International Publishing, March 2014.

[4] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities," *IEEE IoT Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[5] D. Yuan, S. S. Kanhere, and M. Hollick, "Instrumenting wireless sensor networks a survey on the metrics that matter," *Pervasive and Mobile Computing*, vol. 37, pp. 45–62, 2017.

[6] T. Yu, X. Wang, and A. Shami, "A novel fog computing enabled temporal data reduction scheme in IoT systems," in *The 2017 IEEE Global Communications Conference*, pp. 1–5, Dec 2017.

[7] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[8] B. Guo *et al.*, "Opportunistic IoT: Exploring the harmonious interaction between human and the Internet of Things," *Journal of Network and Computer Applications*, vol. 36, no. 6, pp. 1531–1539, 2013.

[9] G. Aloi *et al.*, "Enabling IoT interoperability through opportunistic smartphone-based mobile gateways," *Journal of Network and Computer Applications*, vol. 81, pp. 74–84, 2017.

[10] R. Pozza, M. Nati, S. Georgoulas, K. Moessner, and A. Gluhak, "Neighbor discovery for opportunistic networking in Internet of Things scenarios: A survey," *IEEE Access*, vol. 3, pp. 1101–1131, 2015.

[11] B. Rashid and M. H. Rehmani, "Applications of wireless sensor networks for urban areas: A survey," *Journal of Network and Computer Applications*, vol. 60, pp. 192–219, 2016.

[12] M. Di Francesco, S. K. Das, and G. Anastasi, "Data collection in wireless sensor networks with mobile elements: A survey," *ACM Transactions on Sensor Networks*, vol. 8, August 2011.

[13] Z. Lu, X. Sun, and T. L. Porta, "Cooperative data offload in opportunistic networks: From mobile devices to infrastructure," *IEEE/ACM Transactions on Networking*, vol. 25, pp. 3382–3395, Dec 2017.

[14] OpenFog Consortium, "The OpenFog Consortium Reference Architecture: Executive Summary." https://goo.gl/cIrzuT, 2017.

[15] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for Internet of Things services," *IEEE Internet Computing*, vol. 21, pp. 16–24, Mar 2017.

[16] M. Uddin, S. Mukherjee, H. Chang, and T. V. Lakshman, "SDN-based service automation for IoT," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, pp. 1–10, Oct 2017.

[17] M. Tomasini *et al.*, "On the effect of human mobility to the design of metropolitan mobile opportunistic networks of sensors," *Pervasive and Mobile Computing*, vol. 38, pp. 215–232, 2017.

[18] S. Gao, H. Zhang, and S. K. Das, "Efficient data collection in wireless sensor networks with path-constrained mobile sinks," *IEEE Trans. on Mobile Computing*, vol. 10, no. 4, pp. 592–608, 2011.

[19] S. Aguilar, R. Vidal, and C. Gomez, "Opportunistic sensor data collection with Bluetooth Low Energy," *Sensors*, vol. 17, no. 1, 2017.

[20] H. Wirtz, J. Rüth, M. Serror, J. A. Bitsch Link, and K. Wehrle, "Opportunistic interaction in the Challenged Internet of Things," in *The 9th ACM MobiCom Workshop on Challenged Networks*, pp. 7–12, 2014.

[21] J. G. Kemeny *et al.*, *Finite Markov chains*, vol. 356. van Nostrand Princeton, NJ, 1960.

[22] D. Xu *et al.*, "A survey of opportunistic offloading," *IEEE Comm. Surveys & Tutorials*, vol. 20, no. 3, 2018.

[23] Y. Li, M. Qian, D. Jin, P. Hui, Z. Wang, and S. Chen, "Multiple mobile data offloading through disruption tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 13, pp. 1579–1596, July 2014.

[24] S. Yang, U. Adeel, Y. Tahir, and J. A. McCann, "Practical opportunistic data collection in wireless sensor networks with mobile sinks," *IEEE Trans. on Mobile Computing*, vol. 16, no. 5, pp. 1420–1433, 2017.

[25] A. Mehrabi and K. Kim, "Maximizing data collection throughput on a path in energy harvesting sensor networks using a mobile sink," *IEEE Transactions on Mobile Computing*, no. 3, pp. 690–704, 2016.

[26] G. Maia *et al.*, "A distributed data storage protocol for heterogeneous wireless sensor networks with mobile sinks," *Ad Hoc Networks*, vol. 11, no. 5, pp. 1588–1602, 2013.

[27] W. Wen, S. Zhao, C. Shang, and C.-Y. Chang, "EAPC: Energy-aware path construction for data collection using mobile sink in wireless sensor networks," *IEEE Sensors Journal*, vol. 18, no. 2, pp. 890–901, 2018.

[28] K. Shah, M. Di Francesco, and M. Kumar, "Distributed resource management in wireless sensor networks using reinforcement learning," *Wireless Networks*, vol. 19, pp. 705–724, July 2013.

[29] R. Pozza, M. Nati, S. Georgoulas, A. Gluhak, K. Moessner, and S. Krco, "CARD: Context-aware resource discovery for mobile Internet of Things scenarios," in *IEEE WoWMoM 2014*, pp. 1–10, June 2014.

[30] B. Guo, Z. Wang, Z. Yu, Y. Wang, N. Y. Yen, R. Huang, and X. Zhou, "Mobile crowd sensing and computing: The review of an emerging human-powered sensing paradigm," *ACM Computing Surveys*, vol. 48, pp. 7:1–7:31, September 2015.

[31] H. Jin, H. Huang, L. Su, and K. Nahrstedt, "Cost-minimizing mobile access point deployment in workflow-based mobile sensor networks," in *The 22nd Int. Conf. on Network Protocols*, pp. 83–94, Oct 2014.

[32] M. Xiao, J. Wu, H. Huang, L. Huang, and C. Hu, "Deadline-sensitive user recruitment for mobile crowdsensing with probabilistic collaboration," in *The 24nd Int. Conf. on Network Protocols*, pp. 1–10, Nov 2016.