Enabling Predictable Wireless Data Collection in Severe Energy Harvesting Environments

Zheng Dong[†], Yu Gu[*], Jiming Chen[§], Shaojie Tang[†], Tian He[‡], and Cong Liu[†]
[†]The University of Texas at Dallas, [*]IBM Watson Health
[§]Zhejiang University, [‡]University of Minnesota

***Abstract*—Micro-powered wireless embedded devices are widely used in many application domains. Their efficiency in practice, however, is significantly constrained by the dual limitations of low harvesting rates and tiny energy buffer. Recent research presents a network stack that efficiently fragments a large packet into many smaller packets that can fit within the available energy in the energy buffer of limited size. While this fragmentation technique represents a major step forward in solving the minuscule energy budget problem, it also introduces a tremendous practical challenge where potentially many fragmented packets belonging to different devices may contend for the communication channel. Designing purely heuristic-based packet transmission protocol is undesirable because the resulting per-packet and end-to-end transmission delay are unknown, thus causing unpredictable system performance which is unacceptable for many applications with real-time constraints. In this paper, we first formulate this packet transmission scheduling problem considering physical properties of the charging and transmission processes. We then develop a novel packet prioritization and transmission protocol NERF that yields tight and predictable delay bounds for transmitting packets from multiple micro-powered devices to a charger. We have implemented our protocol on top of the WISP 4.1 platform [3] and the SPEEDWAY RFID READER [2], and conducted validation experiments. Our experiments validate the correctness of our implementation and show that NERF can reduce the total collection delay by 40% when compared to an existing protocol ALOHA. We have also performed extensive data trace-driven simulations. Simulation results demonstrate the effectiveness of our proposed protocol. On average, our protocol yields an over 30% improvement in terms of runtime transmission delay compared to existing methods, while being able to guarantee tight and provable response time bounds.**

I. Introduction

Recent advancements in sensing, wireless communication and charging, and embedded systems are fueling the development of a rich set of micro-powered wireless rechargeable embedded devices and application scenarios, e.g., radio frequency identification (RFID) tags and RFID readers commonly used in tracking inventory in warehouses.

A fundamental challenge of using such micro-powered embedded devices is due to the limited amount of energy they can harvest, the limited size of the device's energy buffer, and the possibly low energy harvesting efficiency in practice [12], [14], [21], [23], [29]. It has been shown that the amount of harvested power using a micro-energy harvester is of the order of *nano*Watts to μWatts, which is three to six orders of magnitude lower than the average power draw of a Mote; while the Intel WISP [5] has energy buffers that are 4-6 orders of magnitude smaller than a coin cell. Due to these limitations, tasks need to be small enough to fit within the available energy in the energy buffer. In a recent work [33], a MAC layer protocol called QuarkNet is presented. The key idea behind QuarkNet is to efficiently fragment a backscatter network stack into its smallest atomic units. That is, QuarkNet can dynamically fragment a larger packet transfer into multiple frames that can be as small as a single bit that can fit within the available energy in the energy buffer of limited size.

Key motivation of this work. Although QuarkNet represents a major step forward in solving the minuscule energy budget problem via efficient fragmentation, it also introduces a tremendous practical challenge at the data link layer, where potentially many fragmented packets belonging to different devices may compete for the communication channel. This is due to the fact that only one packet can be transmitted through a communication channel at any time point. Transmitting these many packets from multiple devices simultaneously may cause significant interference among packet transmissions and unpredictable or even unbounded delay for those transmissions. To resolve this challenge, this paper investigates the following question: how to design a data link layer protocol on top of QuarkNet to transmit packets simultaneously so that predictable packet transmission delay can be achieved? Using a heuristic is undesirable because it cannot guarantee predictable system performance: system designers may not know neither the delay bound for transmitting packets of any device, nor the end-to-end delay for completing all packet transmissions. Guaranteeing predictable system performance with bounded per-packet delay and system-wise delay is important in many application settings, such as real time data collection using sensors in critical infrastructure monitoring systems [26], [28].

This problem is rather challenging due to the physical properties of the energy harvesting process. Specifically, a packet can be transmitted only if its corresponding sensor device has harvested enough energy for the transmission. In practice, unfortunately, the energy harvesting process is compromised by the facts that **(i) the energy harvesting rate gradually decreases as the charging time increases** [33], and **(ii) energy overflow may occur since a sensor device has a limited energy buffer size** [33]. It is thus possible for a packet to be delayed due to another packet's transmission; during such delayed periods, the corresponding sensor device is experiencing a lower energy harvesting rate or energy overflow, which will further delay the transmission of its next packet. Thus, a fundamental challenge is to quantify and alleviate the negative impact on delay performance due to the gradually degraded harvesting rate and the energy overflow issues.

[*]Work supported by NSF grant CNS 1527727.

978-1-5090-5303-2/16 $31.00 © 2016 IEEE
DOI 10.1109/RTSS.2016.13

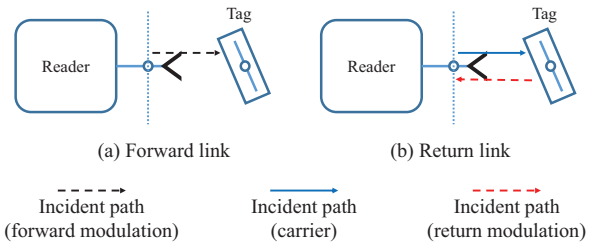

Fig. 1: The two links of ISO 18000-6C RFID communication, shown for the monostatic (shared transmit and receive antenna) case. In the forward link (a), a reader sends a modulated request to a tag, which rectifies the incident wave to power its circuitry. In the return link (b), the tag reflects a modulated reply to the reader.

In this paper, we answer the above-discussed research questions. We develop a fast packet prioritization and transmission protocol considering practical constraints that can be easily implemented in practice, which yields provably low and predictable delay bounds for transmitting packets from multiple micro-powered embedded devices to a charger (also the receiver). Our specific contributions are listed as follows:

- To the best of our knowledge, this is the first work studying the problem of designing a packet transmission protocol in energy harvesting environments, which yields predictable transmission delay bounds.
- We first formally formulate this packet transmission problem considering the complicated issues due to the gradually decreasing harvesting rate and the potential energy overflow issues. We then develop a fast packet prioritization and transmission protocol with low time complexity that can be easily implemented in practice. Despite its simplicity, our protocol is proved to possess desirable strong properties. Specifically, the protocol guarantees to yield a rather tight delay bound for transmitting packets.
- We have implemented our protocol on top of the WISP 4.1 platform [3] and the SPEEDWAY RFID READER [2], and conducted validation experiments. Our experiments validate the correctness of our implementation and show that NERF can reduce the total collection delay by 40% when compared to an existing protocol ALOHA. We have also performed extensive data trace-driven simulations. Simulation results demonstrate that our protocol is superior to existing methods for a wide range of scenarios, yielding an over 30% improvement on average in terms of packet transmission delay.

II. System Models and Preliminaries

In this section, we describe the charging and transmission models used in this work and necessary background.

A. Physical Layer Operation between a Reader and a Device

A backscatter radio is designed for the reader to both provide power (i.e., charging) to a passive device as well as to enable communication (i.e., collecting packets). As shown in Fig. 1, the reader provides a carrier wave, which can be reflected by a passive device back to the reader with its own information bits. This makes backscatter a considerably more energy-efficient communication mechanism compared to active radios, and ideally suited to the constraints of micro-powered devices. The Intel WISP [5] and UMass Moo [32] are examples of backscatter-enabled sensor platforms. Note that the transmission process is non-preemptive, i.e., an in-process packet transmission cannot be preempted and later resumed by other waiting packets.

B. System Model and Implementation Issues.

We consider a rechargeable wireless network with a set $S = \{s_1, s_2, s_3, \ldots, s_n\}$ of n stationary wireless micro-powered devices. Each device s_i has a specific amount of data to be transmitted. A charger R is placed in this sensor network at a given location L^* to collect data packets from all devices that also harvest energy from this charger. A common application scenario that can be accurately represented by this model is the monitoring system in an inventory warehouse, where a RFID reader is collecting data packets from multiple RFID tages in an inventory warehouse; while these RFID tasks harvest energy from the reader simultaneously whenever they are not transmitting packets to the reader.

Note that micro-powered devices do not need to communicate or synchronize with each other. This is because under the considered problem scenario, the charger serves as a centralized decision maker and decides when to collect which devices' packets based on the relevant information the charger collects from the devices. To collect packets, the charger needs to know the amount of energy stored on each device, because the energy status of a device determines whether the device has harvested enough energy to transmit a data packet to the charger. For implementation purposes, the charger can track this required energy information of each device by performing the following 3 steps: (i) localizing the location of devices using the techniques introduced in [25], (ii) calculating the distances between devices and the charger based on the location information and then calculating the amount of harvested energy of each device over time using Eq. (2), and (iii) when a device transmits a packet to the charger, the amount of energy remained in that device can be calculated using the amount of energy harvested by that device minus the energy used to transmit a packet. We assume that the reader can successfully receive each transmitted packet. We note that if the wireless link loss probability is known, our framework can integrate this information and derive the resulting response time bounds for packet transmissions. Intuitively, in order for our framework to schedule packet transmissions with bounded response times, it is only necessary to know the total number of packet transmissions (including failure ones) needed for each device.

C. Charging-transmission Period

In order to understand how a micro-powered device harvests energy from the reader to send packets, we first look at a simple scenario where one device s_i communicates with one charger. As shown in Fig. 2, s_i must sleep for a certain time

*Determine the optimal location for placing the reader is out of scope of this paper and left as future work.

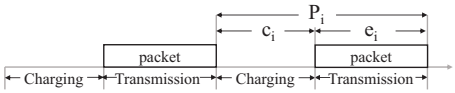

Fig. 2: A charging-transmission period of device s_i.

period during which it harvests energy from the reader, and then wakes up and transmits a packet. Note that the device can not harvest energy while transmitting data. Let F_i denote the packet size and b_i denote the bandwidth between the device s_i and the reader. Let e_i denote the transmission time of a packet, i.e., $e_i = \frac{F_i}{b_i}$.

Definition II.1. *Let P_i denote s_i's charging-transmission period, which is composed by the transmission window with length e_i and a charging period with length c_i, where c_i is determined by the charging rate and the amount of required energy to transmit a packet. P_i represents the minimum delay between any two consecutive packet transmissions made by s_i.*

Definition II.2. *Based on the definition of period P_i and transmission time e_i for device s_i, the channel utilization of s_i is denoted as $u_i = \frac{e_i}{P_i}$. We require $u_i \leq 1$, for otherwise the delay on transmitting packets of s_i is unbounded. The total channel utilization is given by $U = \sum_{i=1}^{n} u_i$.*

D. Non-linear Harvesting Rate

According to the above charging and transmission model, the energy harvesting rate has significant impact on the communication throughput, because a higher harvesting rate implies that more energy can be harvested for data transmission within a charging cycle. We conducted a measurements-based experiment using a customized WISP node depicted in Fig. 3. The size of a WISP node is similar to a USD quarter coin, which can be easily attached to containers, packages or other daily objects. The experiment results on energy harvesting rate using the WISP node are plotted in Fig. 4 as denoted by "experiment", while "Fitting" denotes the curve after performing curve fitting on top of the experiment curve. A key observation herein is that the harvesting rate achieves its maximum value after a certain amount of time, and then sharply drops. The interpretation is that when the amount of stored energy in the device's energy buffer becomes larger than the threshold, the energy harvesting rate sharply drops from a constantly high level to 0. This, this implies that if a packet is not transmitted immediately when the device harvests enough energy, the energy harvesting rate may drop sharply and the amount of energy the device can harvest and buffer in the capacitor may increase at a rather slower rate afterwards.

These measurement results have also been validated in recent work [33], and can be explained by investigating how a device's capacitor buffers the harvested energy. The charging voltage to a capacitor follows the equation $V = V_{max}(1 - e^{-t_s/\tau})$ [33], where t_s is the charging time, τ is the RC circuit time constant, and V_{max} is the maximum voltage to which the capacitor can be charged. The device's energy harvesting rate after being charged for t_s time units, denoted as $H(t_s)$, can be calculated by [33]:

$$H(t_s) = C \times V_{max}^2 \times \tau^{-1}(1 - e^{-t_s/\tau})e^{-t_s/\tau}. \quad (1)$$

When the harvesting conditions are fixed (i.e. V_{max} and τ are fixed), H is a concave function of t_s, as shown in Fig. 4. Note that, under the same hardware settings, V_{max} is determined by the distance between the reader and the micro-powered device [10]. In other words, the closer the device is from the charger, the greater V_{max} it will receive.

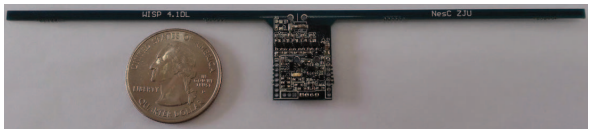

Fig. 3: WISP node used in our lab.

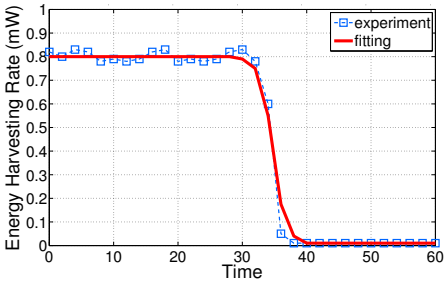

Fig. 4: The energy harvesting rate curve. At time 0, the energy storage is empty.

E. Optimal Packet Size

As discussed earlier, to resolve the dual limitations of low harvesting rates and tiny amount of harvested energy in reservoirs, Zhang *et al.* [33] introduced QuarkNet, which fragments a larger packet transfer into μ frames that can be as small as a single bit under severe energy constraints. We leverage this fragmentation technique to obtain an optimized packet size (via fragmentation) for each device according to the amount of harvested energy.

A packet consists of a header field of α bits long, an information field of l bits long and a tailer of β bits long. It has been shown that sending longer packets requires larger amount of energy, while transmitting short packets suffers from great overhead (additional data of header and tailer) [22]. Thus, by considering the trade-off between the packet overhead and the charging rate, we next derive a proper packet size that enables the device to achieve a balanced transmission rate.

According to [33], the amount of energy harvested by the device s_i over time is:

$$E^h(t) = \int_0^t H(t_s)d(t_s) \quad (2)$$

$$= \int_0^t C \times V_{max}^2 \times \tau^{-1}(1 - e^{-t_s/\tau})e^{-t_s/\tau}d(t_s) \quad (3)$$

$$= C \times V_{max}^2 \times \frac{(e^{-t/\tau} - 1)^2}{2}. \quad (4)$$

Let F_i denote the packet size. Based on the packet format and the energy model, prior works [24], [22] have proved that F_i is a linear function of the amount of energy harvested by s_i during $[0, t]$. Let κ denote the linear coefficient and $F_i = \kappa \times E^h(t)$. Then, if s_i sends a packet to the reader at t, the

amount of useful information sent to the reader is $F_i - \alpha - \beta$. Thus, the transmission rate is $\frac{F_i-\alpha-\beta}{P_i}$. To find an optimal packet size that maximizes the transmission rate, we derive the following optimization formulation.

$$\max \quad \frac{F_i - \alpha - \beta}{P_i} \tag{5}$$

$$F_i = \kappa \times E^h(t) \tag{6}$$

$$P_i = t + e_i \tag{7}$$

$$E^h(t) = C \times V_{max}^2 \times \frac{(e^{-t/\tau} - 1)^2}{2}. \tag{8}$$

where t is the charging time and the corresponding packet size is F_i. Solving this formulation yields an optimal packet size for a device to transmit packets to the charger (e.g., using the technique given in [4] to solve it, which can be performed offline). For the rest of the paper, let F_i denote the optimal packet size of s_i, t_s^i denote the charging period, i.e., $P_i = t_s^i + e_i$ (note that t_s^i equals to t when Eq. (5) is maximized), n_i denote the number of packets in s_i with a packet size of F_i.

III. Main Design

In this section, we present the proposed packet prioritization and transmission protocol that guarantees predictable transmission delay bounds.

A. Motivation and Design Overview

Non-linear harvesting-rate-induced delay. If all devices can send packets to the reader at their optimal frequency simultaneously (i.e., transmitting one packet for every P_i time units), then intuitively we know that the throughput of the whole network is maximized and the collection delay becomes predictable. However, when the reader simultaneously collects data packets from multiple devices, transmission delay may occur due to channel contention. This is due to the fact that only one packet can be transmitted at a time. The eligibility for s_i to transmit a packet to the reader depends on the amount of energy it harvests from the reader. Let $R_{i,k}$ denote the k^{th} packet release time when s_i harvests enough energy to send the k^{th} packet. For example, in Fig. 2, the release time of the first packet is at the end of its first charging period, denoted as $R_{i,1}$. Due to transmission delay, the k^{th} packet of s_i may not be able to transmitted at $R_{i,k}$ when enough energy has been harvested by s_i to transmit the packet, because the transmission channel may be occupied by another packet.

The above-mentioned issue also appear in similar problem context such as uniprocessor task scheduling, where a task may experience scheduling delay due to the competition of the resource. However, a unique challenge existed in our packet transmission problem is that ***the transmission delay experienced by a sensor for sending a packet may further cause longer delays for sending subsequent packets due to the non-linear energy harvesting rate as discussed in Sec. II-D***. For the above example, the harvesting rate for s_i to harvest energy for sending its $(k+1)^{th}$ packet decreases because at time $R_{i,k}$, the energy capacitor is not empty, instead it stores the amount of energy for transmitting the k^{th} packet of s_i. According to the charging model (Fig. 4), not only will the k^{th} packet be delayed, but s_i will need more time to harvest enough energy to transmit the next packet due to a decreased harvesting rate.

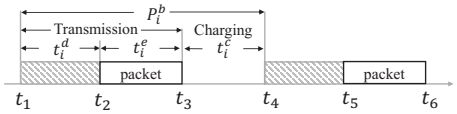

Fig. 5: A typical packet transmission cycle $[t_3, t_6]$ in a multi-device scenario.

A packet transmission cycle with length P_i denotes the ideal scenario for any device s_i, because transmitting a packet for every P_i time units is possible only if s_i's transmissions would never be delayed by other devices. With multiple devices in the system, the common case is that multiple devices contend for the same communication channel, thus causing transmission delays. The resulting time interval between two actual consecutive transmissions can thus become larger than P_i.

Motivated by the above discussions, our goal is to design a packet scheduling and transmission protocol that considers the non-linear harvesting rate and yields predictable delay bounds. We would like to first clearly explain what delay we need to bound, and then present an overview on the analysis steps.

Fig. 5 illustrates a typical packet transmission cycle in a multi-device scenario. At time t_1, the device s_i has harvested enough energy to send a packet but the reader is busy at that time. The transmission of this packet is thus delayed until t_2. The data packet starts transmitting at t_2 and completes its transmission at t_3. Then s_i starts to harvest energy to send its next packet and has harvested enough energy to send the successive packet at t_4. Such a packet transmission cycle denoted by P_i^b naturally defines the delay for collecting a packet. We formally define P_i^b as the *per-packet collection delay*.

We develop a packet scheduling protocol that yields predictable bounds on two kinds of time delay: the per-packet collection delay and the total collection delay for transmitting all devices' packets. Fig. 5 illustrates these concepts. As seen in the figure, the per-packet collection delay can be divided into three components: a transmission delay t_i^d, a transmission time t_i^e and a charging time t_i^c, i.e., $P_i^b = t_i^d + t_i^e + t_i^c$.

To bound the per-packet collection delay P_i^b, we first propose a packet prioritization protocol in Sec. III-B, namely NERF , to prioritize packets (which determines the packet transmission order). We prove that under NERF, $t_i^d + t_i^e$ can be bounded by a constant value when $U \leq 1$ (theorem III.1.) Then, we further bound P_i^b when $U \leq 1$ in Sec. III-C. In Sec. III-D, we extend the NERF protocol to handle the general case where the total channel utilization exceeds 1, where we prove that the extended NERF protocol can still yield a total collection delay bound.

B. The NERF Protocol

If multiple packets of different devices are contending for the transmission channel, it is important to assign priorities to these packets so that the inefficiency caused by nonlinear harvesting-rate-induced delay is minimized. Ideally, a packet

priority assignment and transmission protocol shall have at least three properties: (i) it yields a tight and bounded per-packet collection delay, (ii) it can reduce the total collection delay for transmitting all packets of each individual device, and (iii) it should be simple enough to implement in practice. The first property guarantees a predictable system performance. The second property enables a higher energy harvesting rate for each device. The third property is necessary, since in practice there may be a large number of devices and data packets. A complicated packet assignment may cause significant amount of runtime overheads.

Next, we present a protocol that is able to achieve these three properties. Our design is motivated by the classical uniprocessor real-time sproadic task scheduling problem, which can be considered as an equivalent problem as our packet prioritization problem.

Uniprocessor real-time sporadic task scheduling. This classical problem is to schedule a set of sporadic tasks on a single processor such that all the required deadlines are met. A real-time sporadic task releases computational jobs in a periodic manner. Each task T_i is characterized by a minimum job inter-arrival time p_i (traditionally known as the task period), and an execution time $e_i < p_i$, and a relative deadline $D_i = p_i$. The j^{th} job of task T_i, denoted by $T_{i,j}$, is released at time $r_{i,j}$. Each job $r_{i,j}$ has a absolute deadline $d_{i,j} = r_{i,j} + p_i$. Every task may continuously release job instances with any two consecutive invocations separated by at least p_i time units. Each task T_i has a utilization of e_i/p_i, which characterizes the long-term processor capacity requested by T_i (i.e., a task with a utilization of 0.5 is expected to utilize half of the processor capacity over the long term).

Similarity among the two problems. The packet/device model can be equivalently viewed as the real-time sporadic task model. Each device can be viewed as a sporadic task, where a device keeps releasing packets to be transmitted to the charger while a sporadic task releases jobs to be executed on the CPU. The transmission time and period of a device can be equivalently viewed as a task's per-job execution time and period. The transmission channel can be equivalently viewed as the CPU capacity, where each device has a channel utilization while each sporadic task also has a utilization.

The non-preemptive earliest-deadline-first (NPEDF) scheduler. Under the NPEDF scheduler, a job $T_{i,j}$ with the smallest $d_{i,j}$ has the highest priority. Ties are broken by task ID. At each time instance, NPEDF selects the job with the highest priority to be executed on the CPU. During a job $T_{i,j}$'s execution, another job $T_{l,k}$ with a higher priority cannot preempt the execution of $T_{i,j}$.

Motivated by the above problem transformation, we apply NPEDF in our problem context to prioritize packets. Since packets do not have deadlines, we denote our protocol by NERF instead, where the release time of the $(j+1)^{th}$ packet of s_i can be equivalently viewed as the deadline of the j^{th} packet of S_i. Specifically, let $r_{i,j}$ denote the releasing time of $s_{i,j}$ which is the j^{th} packet of s_i. That is, $r_{i,j}$ represents the time point when device s_i harvests enough energy to send $s_{i,j}$ to the charger. Ideally, s_i's $(j+1)^{th}$ packet will release at $r_{i,j} + P_i$. Then, let $P_{i,j} = r_{i,j} + P_i$ be the priority point of $s_{i,j}$. For any two packets $s_{i,j}$ and $s_{l,k}$, $s_{i,j}$ has a higher priority than $s_{l,k}$ if $P_{i,j} \leq P_{l,k}$, or $P_{i,j} = P_{l,k}$ and $i < l$. Moreover, whenever a device starts transmitting a packet, this transmission cannot be preempted by any other packet even with a higher priority (i.e., enforcing non-preemptivity of the packet transmission).Under NERF, a packet $s_{i,j}$ with a smaller priority value $P_{i,j}$ has a higher priority. When the reader is idle, it will collect the packet with the highest priority.

The rest of this section focuses on proving that NERF is able to yield a provable bound on the transmission delay plus the transmission time, i.e., $t_i^d + t_i^e$ for each s_i. Note that this part of the proof is new and has never been considered before even by researchers in the real-time systems community. For the real-time sporadic task scheduling problem, most works focus on the hard real-time case, where any deadline miss is unacceptable. Devi first showed in [7] that NPEDF is able to yield a response time bound (while allowing deadlines to be missed but any such miss can be analytically bounded) on $m \geq 2$ processors and $m \geq 2$ is a fundamental requirement for the analysis to work (see page 10 in [7]). However, in our problem, we have only one charger, thus $m = 1$. We next provide a new analysis framework that can provide such a response time bound under the $m = 1$ case.

Our new proof strategy is to compare the packet transmission schedule yielded by NERF with a corresponding CPU task execution schedule under a real-time CPU task scheduling algorithm, preemptive earliest-deadline-first (PEDF). PEDF is the same as NPEDF except that under PEDF, jobs with higher priorities are allowed to preempt the execution of a lower-priority job that starts executing. PEDF is proved to be an optimal scheduler for scheduling a set of real-time sporadic tasks on a single CPU, which guarantees all deadlines can be met [17]. The key intuition behind this comparison is that a PEDF schedule represents an ideal reference schedule where every packet $s_{i,j}$ will finish its transmission by $s_{i,j} + P_i$. Intuitively, a fundamental reason why PEDF yields an optimal schedule is due to its preemptive nature [17]. However, under NERF, transmission delay may occur due to non-preemptivity. Thus, a packet $s_{i,j}$ may not complete its transmission by $s_{i,j} + P_i$ because another packet with lower priority may start transmission earlier. The key is thus to quantify the maximum delay caused by the non-preemptive nature of packet transmission under NERF.

The following Theorem III.1 proves that when the total channel utilization is at most 1, the transmission delay plus the transmission time for each packet can be bounded in a tight manner. A key insight behind proving this theorem is that that at most one lower-priority job may preempt $s_{i,j}$. Thus, intuitively, the maximum transmission delay for any packet under NERF can be proved to be at most $P_i + e_{max}$, where e_{max} represents the maximum transmission time of any packet in the system. The P_i term is due to the fact that under PEDF any packet's transmission delay is bounded by the corresponding device's ideal charging-transmission period; while the e_{max} term is due to the fact that at most one job with lower priority can delay the transmission of a packet under NERF.

Definition III.1. *A time instant t is busy (resp. non-busy) if the reader is collecting (resp. is not collecting) a packet at t. A time interval is busy (resp. non-busy) for a packet set J if each instant within it is busy (resp. non-busy) for J.*

Definition III.2. *Packet $s_{i,j}$ is pending at time t, if $r_{i,j} < t < f_{i,j}$.*

Lemma III.1. *For a concrete system, its PEDF schedule and NERF schedule have the same busy/non-busy intervals.*

Proof: We prove by contradiction. Assuming that for a concrete system, its PEDF schedule and NPEDF schedule do not have the same busy/non-busy intervals. Suppose the first different time point between these two schedules happens at time t. Before t NERF and PEDF have the same busy intervals, implying that they complete the same amount of workload before t. Since the task system releases the same amount of workload (i.e., the same set of jobs with the same release times) by *t*regardless of the scheduling policy, both PEDF and NERF must be either busy or non-busy at t as both are work-conserving (i.e., a job will be immediately executed if the processor is idle). A contradiction is thus reached. ■

Theorem III.1. *Under NERF, for any device s_i, for any packet $s_{i,j}$, $t_i^d + t_i^e \leq P_i + e_{max}$.*

Proof: Let $B_1, B_2, \ldots$ denote a series of busy intervals in the NERF schedule w.r.t. time from left to right. Suppose $s_{i,j}$ is transmitted during an interval B_k. Since all the jobs released before B_k have been completed before B_k, only the jobs released in B_k may delay $s_{i,j}$. Thus, we only consider the jobs released in B_k. Let job set T_B denote all jobs released during B_k. Based on Lemma III.1, we know that the PEDF schedule of T has a corresponding busy interval, denoted as B'_k, which has the same start time and end time with B_k. Jobs in T_B may also execute during B'_k in PEDF schedule, but the jobs' execution order may be different from those in B_k. Let t_B denote the start time point of B_k. Let $d_{i,j} = r_{i,j} + P_i$ denote the priority value of $s_{i,j}$. $s_{i,j}$'s transmission cannot be delayed by the jobs in T_B with releasing time later than $r_{i,j}$ and priority lower than $d_{i,j}$. Therefore, we remove such jobs from T_B. When $s_{i,j}$ is released at $r_{i,j}$, $s_{i,j}$ may not be transmitted immediately. In order to bound $s_{i,j}$'s transmission delay, we need to consider two different cases.

Case 1: All packets transmitted by the charger during $[t_B, r_{i,j}]$ have higher priorities than $s_{i,j}$ in the NERF schedule. In this case, the jobs in T_B with priorities lower than $s_{i,j}$ cannot delay $s_{i,j}$. Those higher-priority jobs that can delay $s_{i,j}$ in the NERF schedule can also delay $s_{i,j}$ in the PEDF schedule. Thus, $s_{i,j}$ is transmitted in the NERF schedule at the same time as in the PEDF schedule. Therefore, $s_{i,j}$ completes in the NERF schedule at the same time as it completes in PEDF schedule. In this case, $s_{i,j}$'s transmission delay is at most $P_i < P_i + e_{max}$ [17].

Case 2: Some packets transmitted by the charger during $[t_B, r_{i,j}]$ have lower priorities than $s_{i,j}$ in the NERF schedule. In this case, the jobs in T_B with priorities lower than $s_{i,j}$ may disallow $s_{i,j}$ to be transmitted at $r_{i,j}$, since they may occupy the channel at $r_{i,j}$ due to an earlier release time. Let $s_{l,k}$ denote the latest job released before $r_{i,j}$ in the NERF schedule with a priority lower than $s_{i,j}$. Suppose that $s_{l,k}$ starts transmission at t_l in the NERF schedule. This implies that no jobs with priorities higher than $s_{l,k}$ are pending at t_l. We thus know that in the PEDF schedule, $s_{l,k}$ also starts transmission at t_l.

The difference is that under NERF, $s_{l,k}$ is executes non-preemptively until the entire packet is received by the charger; while under PEDF, $s_{l,k}$ may be preempted by higher priority packets and $s_{i,j}$ can only be preempted by the jobs with priority points earlier than $d_{i,j}$ and executing after $r_{i,j}$. The total amount of preemption time experienced by $s_{i,j}$ under PEDF is at most $P_i - e_i$, because $s_{i,j}$ meets its deadline under PEDF. In the following, we analyze the total amount of preemption time experienced by $s_{i,j}$ under NERF.

Under NERF, the total amount of preemption time experienced by $s_{i,j}$ can be divided into three parts:

- the total amount of preemption time experienced by $s_{i,j}$ due to higher-priority packets, which is at most $P_i - e_i$ as the same amount of preemption time experienced by $s_{i,j}$ under PEDF (for otherwise $s_{i,j}$ would miss its deadline under PEDF which is a contradiction).
- the total amount of delay experienced by $s_{i,j}$ due to packets with lower-priorities than $s_{i,j}$, which is at most $s_{l,k}$'s transmission time (i.e., e_l).

Since the transmission time of $s_{i,j}$ is e_i, $s_{i,j}$'s transmission delay is at most $P_i - e_i + e_l + e_i = P_i + e_l$. Given that $s_{l,k}$ is an arbitrary packet, $s_{i,j}$'s transmission delay is at most $P_i + e_{max}$. This theorem is thus proved. ■

C. Bounding the Per-packet Collection Delay

In Sec. III-B, we have derived the transmission delay bound for each packet. Based upon this bound, we derive a bound in this section on the per-packet collection delay P_i^b for any packet $s_{i,j}$. We will then derive a total collection delay bound when $U \leq 1$.

We now identify the scenario where P_i^b reaches its maximum value. As seen in Fig. 6 (as a combination of Fig. 4 and Fig. 5), according to the charging model, the required amount of energy E_i to send a packet from s_i to the reader equals the area of the region enclosed by the curve $H(t)$ (Eq. 2), the y-axis and the x-axis between 0 and t_1. Thus, the amount of harvested energy by s_i during intervals t_i^d and t_i^c also equals E_i. However, the charging rate of s_i during t_i^d is much lower than the charging rate during $[0, t_x]$, where t_x denotes a time point such that $t_i^d + t_i^c = t_2 - t_x$. This definition of t_x also implies that at time t_3, s_i has the same charging rate as its charging rate at time t_x. Since the charging rate during t_i^d is lower than that during $[0, t_x]$ (the charging power is maximal, when the energy buffer is empty), we know that $t_2 - t_1 > t_x$ and $t_2 - t_1 - t_x$ increases when $t_2 - t_1$ increases. In Fig 6, t_1 is a constant time point and t_2 is determined by the delayed period t_i^d. Thus, $t_2 - t_x$ achieves its maximum value when t_2 achieve its maximum value, i.e., when $t_i^d + t_i^e = P_i + e_{max}$ (by theorem III.1).

Based on the key observation above, when $t_i^d + t_i^e = P_i + e_{max}$, i.e. $t_i^d = P_i + e_{max} - e_i$, P_i^b achieves its maximum value. In other words, $t_2 - t_1 = P_i + e_{max} - e_i$. Since t_1 is known by Eq. 5, t_2 can be calculated. Based on the charging curve $H(t_s)$, the area of the shadowed region between t_1 and t_2 can be calculated, which equals the area of the shadowed region between 0 and t_x. Then, t_x can be calculated. Since $t_i^c = t_1 - t_x$ (by the definition of t_x), we can calculate t_i^c when P_i^b reaches its maximum value. Thus, we have

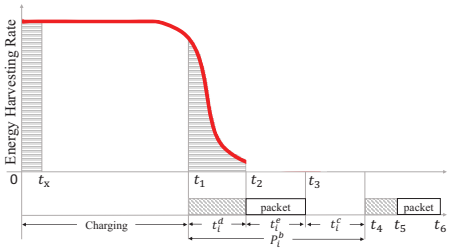

Fig. 6: The relationship between the charging rate and the harvested energy.

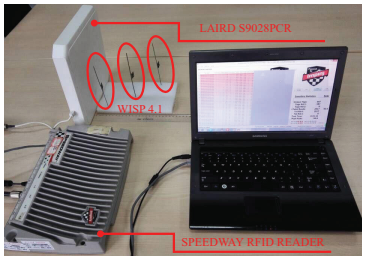

Fig. 7: Testbed.

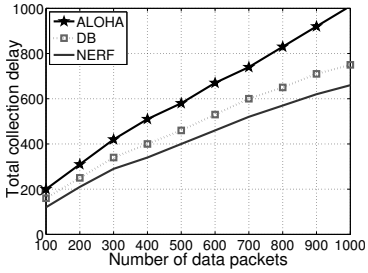

Fig. 8: Delay VS. Number of Data packets.

$$P_i^b = t_c^i + P_i + e_{max}, \tag{9}$$

Thus, the delay bound of collecting all the packets in s_i is at most $N_i \times P_i^b$, where N_i denotes the number of packets of s_i. Since the reader collects data packets from devices simultaneously, the total collection delay when $U \leq 1$ is upper bounded by:

$$D = \max_{i=1}^{n}\{N_i \times P_i^b\}. \tag{10}$$

D. Total Collection Bound for the General Case

We now derive a total collection delay using NERF for the general case, where $U > 1$ is possible. We develop a grouping scheme (GS) based on NERF that groups devices into smaller subsets, and transmit packets of the devices in each group in turn. GS consists of five steps, which are described as follows:

- **Step 1**: If $U \leq 1$, schedule transmissions using NERF.
- **Step 2**: If $U > 1$, sort all the devices according to their transmission time with non-increasing order. Suppose the sorted list is denoted by $s_1, s_2, \ldots, s_n$.
- **Step 3**: Select the first m devices in the sorted list, whose total channel utilization is at most 1 while adding any more device would cause the resulting $m+1$ devices to have a total channel utilization greater than 1. Schedule the packets in the selected m devices using NERF. We call these m devices as the ready device set. We call the remaining $n - m$ devices as the waiting device set.
- **Step 4**: When some device in the ready device set has sent all its packets to the charger, GS seeks to find and insert some devices in the waiting device set into the ready device set. The criteria is that after any insertion, the total channel utilization of devices in the ready device set shall be still at most 1.
- **Step 5**: Step 4 repeats until all packets are transmitted.

Deriving an upper bound on the total collection delay under GS. To derive this upper bound, we first construct another device grouping scheme similar to GS, denoted by GS', then derive an upper bound on the end-to-end delay of transmitting all packets under GS'. Finally we prove that this upper bound is also a safe upper bound for using GS.

The GS' policy description. Suppose the total utilization of all devices is X. Let $K = \lceil X \rceil$. Sort all the devices according to their transmission time in decreasing order. Group devices into k groups from the devices with the largest transmission time and ensure that the total channel utilization of devices in any group is no larger than 1. We know that $K \leq k \leq 2K$ since $u_i \leq 1$ for any device s_i. Then, the charger collects data packets from the devices in each group. For any group G_j where $j \in [1, k]$, we can calculate the total collection delay bound for this group according to Eq. 10. Let D_j denote this delay bound of group G_j. Then, the total collection bound for collecting packets in all groups is given by $D_{GS'} = \sum_{i=1}^{k} D_j$.

Let D_{GS} denote the total collection delay under GS. A lower bound on D_{GS} can be easily obtained by assuming the channel is busy all the time, which represents an ideal scenario. Thus, we have $D_{GS} \geq \sum_{i=1}^{n} e_i \times n_i$. An upper bound on D_{GS} is proved by the following theorem to be $D_{GS'}$. Intuitively, this is because the grouping strategy under GS and GS' is the same. But, GS' processes the next group of devices only after completing the transmissions of all devices' packets for the current group, which may leave the channel idle; while under GS, for the current group being processed, whenever a device's packets finish the transmission, GS will pick the packet from the next group that has the maximum transmission time. Thus, GS yields a total collection time that is at most the total collection time given by GS', as formally proved in the following theorem.

Theorem III.2. $\sum_{i=1}^{n} e_i \times n_i \leq D_{GS} \leq D_{GS'} = \sum_{i=1}^{k} D_j$.

Proof: In Sec. III-C, we have derived that the packet delay of s_i is $P_i^b = t_c^i + P_i + e_{max}$. According to this equation, we have two observations: (1) The packet delay bounds of devices are determined by their own parameters and the largest transmission time of any device in the system, and (2) The packet delay bound of a device with larger transmission time will not be impacted by devices with smaller transmission times.

We now prove that $D_{GS} \leq D_{GS'}$. Suppose we have n sensor nodes at the beginning and they are grouped under GS'. Under both GS and GS', the charger collects packets starting from group 1 using NERF. The difference between using GS and GS' occurs when any device in group 1 finishes sending all its packets to the charger. GS will try to add any device in any waiting device set into group 1 while guaranteeing that the total channel utilization of group 1 is still at most 1. On the other hand, GS' simply complete transmitting all packets of devices in group 1, after which group 2 will be considered. Clearly, GS' yields a more pessimistic transmission schedule in the sense that while considering group 1, even if some

devices in group 1 finishes the transmission and the charger has idleness that can transmit packets from some additional devices not belonging to group 1, GS' will simply leave the charger to be idle until all devices in group 1 complete all packet transmissions. Thus, the end-to-end delay bound under GS must be no greater than the one under GS' because GS seeks to utilize any idleness (in step 4 of GS) by inserting eligible devices into group 1. ■

IV. Implementation and Experimental Evaluation

To assess the performance of our proposed techniques, we have conducted experiments based on real implementations.

A. Experiment setup and baseline

Fig. 7 shows our indoor experimental testbed. We utilize an indoor RFID antenna (LAIRD S9028PCR [1]), a RFID Reader (SPEEDWAY RFID READER [2]) and 3 rechargeable sensor nodes (WISP 4.1 [3]) which are deployed in a 1 m $\times$ 1 m square area. Each sensor node harvests energy from the micro waves emitted by the RFID antenna to send its data packets to the reader. The distances between the 3 sensor nodes and the RFID antenna are 4 cm, 11 cm and 18 cm respectively. Given Eq. 2, the reader can calculate the amount of energy harvested by each sensor node over time. The reader collects data packets by assigning priorities to the sensor nodes according to NERF and the ALOHA protocol [16]. We use a Samsung laptop connecting to the reader to record the total collection delay of all packets from the sensor nodes.

Baseline. We implemented the ALOHA protocol [16] for data collection as a baseline in our experiments: when multiple sensor nodes have harvested enough energy to send data packets, the reader randomly chooses one of these packets to collect. ALOHA has been widely applied to avoid the collision of simultaneous transmissions in many application scenarios where one reader communicates with multiple sensor nodes [15], [30], [35]. In the experiment, we compare the analytically calculated delay bound under NERF, the runtime collection delay under NERF, and the runtime collection delay under ALOHA, denoted by DB, NERF, and ALOHA, respectively.

The evaluation goal is to investigate (*i*) correctness and efficiency: whether the reader can serve as a centralized decision maker and decide when to collect which sensor nodes' packets according to the NERF protocol in practice, as well as how much improvement NERF can achieve compared to ALOHA, and (*ii*) tightness: the tightness of the analytical collection bound DB.

B. Correctness and efficiency validation.

In the experiment, when a packet was received by the reader, the reader recorded its "next-release" information according to the energy status of the same sensor node. Based on such information, the actual priorities assigned to all data packets in the experiment can be obtained. We validate the correctness of our implementation of NERF by comparing the actual priorities assigned to all data packets to such priorities that are assigned according to NERF algorithmically. The experiment results show that all packets are correctly collected under NERF. Fig. 8 shows the experiment results that compare the total collection delay between ALOHA and NERF. As seen in this figure, NERF reduces the total collection delay by 40% compared to ALOHA under almost all scenarios. Another observation in Fig. 8 is that the total collection delay under both ALOHA and NERF increases as the total number of packets increases. This is because when the number of data packets increases, each device needs to harvest more energy to send data to the reader, thus requiring more transmission time.

C. Tightness.

As seen in Fig. 8, the gap between DB and NERF is reasonably small under any number of data packets. For example, when the number of data packets is 500, the analytical collection delay bound DB is less than 10% more than the actual runtime collection delay yielded under the NERF protocol. This implies that the analytical collection delay bound of NERF is reasonably tight. Moreover, DB is smaller than ALOHA under all scenarios, which implies that the analytical collection delay bound of the NERF protocol is actually more efficient than the actual runtime collection delay achieved under the ALOHA protocol.

V. Simulation Evaluation

We have conducted extensive experiments to evaluate the proposed NERF scheduler and GS policy under various system settings.

A. Experiment Setup

We consider a $\pi \times 20 \times 20\ m^2$ circular field (the ceiling of a charger's operational range is $20\ m$ [33]) where 20 to 200 RFID tags are deployed, and one charger is deployed for charging RFID devices and collecting packets. The charger's charging power is variable from 17dBm to 25.7dBm, which represents the range of the RF power that can be generated using the USRP RFX900 daughterboard [33]. Under the charging model where $H(t_s) = C \times V_{max}^2 \times \tau^{-1}(1-e^{-t_s/\tau})e^{-t_s/\tau}$, both τ and C are constants. According to the experiment data collected in Sec. II-D, we set $\tau = 4.32 \times 10^4$ and $C = 0.2316$. V_{max} is set to be $1.8V$ when the RFID's energy capacitor is set to be 100 μF [11]. The total channel utilization is varied within the range $[5, 50]$ and the amount of data packets on each RFID device is varied within the range $[0.5m, 20m]$. Using these settings, we simulate the charging and packet transmission process of the charger and devices. RFID tags transmit data packets to the charger when the charger's communication channel is available and the corresponding packet has the highest priority. The simulation is implemented in Matlab. For each experiment, the results are averaged with 100 runs.

Compared approaches. Since the problem of enabling simultaneous data collection in energy harvesting environments is relatively new, we compare our proposed approach with two baseline methods: random collection (RC) and grouped random collection (GRC). Under the RC scheme, the charger randomly collects data packets from devices that have sufficient harvested energy to transmit a packet; while under the GRC scheme, the charger first groups devices as our proposed GC and then randomly collects data packets from devices in each group.

In addition, we also calculate the upper bound and lower bound on the transmission delay presented in Sec. III-D,

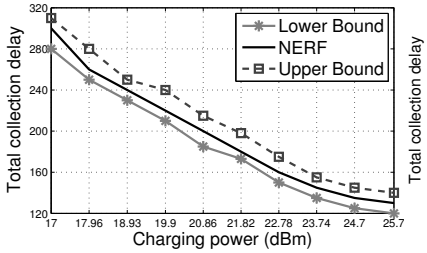

Fig. 9: Delay VS. Charging Power

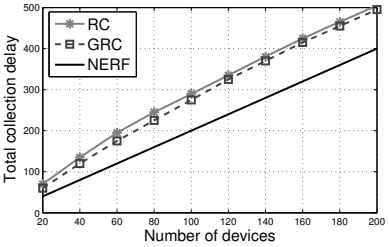

Fig. 10: Delay VS. Number of Device

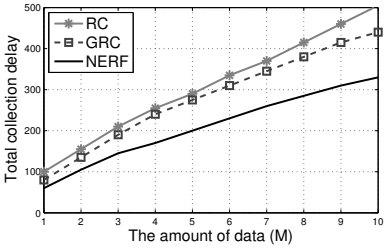

Fig. 11: Delay VS. Amount of Data

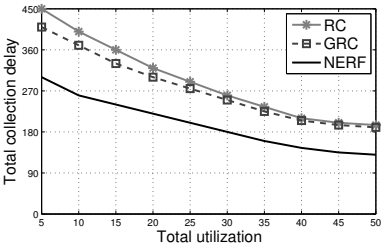

Fig. 12: Delay VS. Utilization

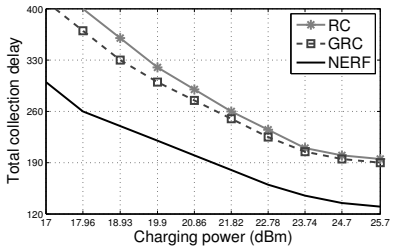

Fig. 13: Delay VS. Charging Power

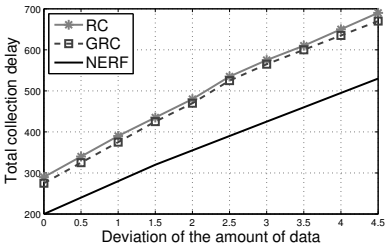

Fig. 14: Delay VS. Deviation

denoted by "Lower Bound" and "Upper Bound" in the result graphs, respectively. These bounds are used to demonstrate the gap between the runtime transmission delay and the theoretical delay bounds.

B. Performance Evaluation

1) Collection delay bound VS. Simulated Collection Delay: Fig. 9 shows the total delay for the charger to collect all data packets, when the charging power varies from 17 dBm to 25.7 dBm. As seen from the figure, the observed delay using NERF is only $5\% - 8\%$ greater than the theoretical lower bound, and our derived upper bound of the total collection delay is relatively tight. One observation is that total collection delay decreases as the charging power increases. This is because with a greater charging power, the charging period of each device becomes smaller and devices can send data packets to the charger in a more frequent fashion.

2) Impact of Device Number: The number of rechargeable devices is a critical factor affecting the performance of NERF. Thus, in the following, we evaluate whether NERF can operate effectively when a large set of devices are present. The resultant total collection delay by NERF, RC and GRC are shown in Fig. 10, with the device numbers varying from 20 to 200. NERF yields the best performance in all cases. We can see that for all three methods, the total collection delay increases as more devices are involved in the system. The reason is because as more devices are involved in the system, more data packets are needed to be collected. The reader thus needs more time to complete the collection. Another observation is that NERF operates with much smaller transmission delay than both RC and GRC, which indicates NERF achieves a much better performance. For example, when the device number is 400, the total collection delay of NERF is 320 minutes while RC and GRC costs 410 minutes and 420 minutes, respectively.

3) Impact of the amount of data: Fig. 11 shows the experimental results investigating the impact due to the variation of the amount of data on each device. In this experiment, the amount of data on each sensor node is varied from 1 M to 10 M. NERF achieves the best performance among all approaches. As seen in Fig. 11, the total data collection delay increases as more data is required to transmit by each device. This is because when the amount of data increases, each device needs to harvest more energy to send the data to the reader. The energy harvesting power and the bandwidth between the reader and each device are fixed, thus the total transmission delay increases. As observed in this figure, NERF achieves better performance than RC and GRC.

4) Impact of Total Utilization: The total channel utilization is related to the charging power and bandwidth between the reader and each device. Thus, different topologies of the same devices will have different total channel utilization. And the total utilization impacts the total transmission delay, which is shown in Fig. 12. In all cases, NERF achieves the best performance. Moreover, with a higher total channel utilization and a fixed number of devices and the same amount of data to be transmitted, a smaller total transmission delay can be achieved. This claim is validated by experiments as shown in Fig. 12, where a clear decreasing trend for NERF, RC and GRC can be observed along with the increasement of the total channel utilization. Moreover, NERF achieves a much better performance than the other two methods: the total transmission delay under NERF is about 20% and 25% less than the ones under RC and GRC, respectively.

5) Impact of Charging Power: We now investigate the impact of charging power of NERF, and the resultant total collection delay is shown in Fig. 13, with charging power varying from 17 dBm to 25.7 dBm. NERF performs consistently better than the other two approaches. As seen in the figure, the total collection delay decreases as the charging power is larger. This is because with a larger charging power, each device can harvest energy faster, and thus the devices can send packets to the reader more frequently. We also notice that when the charging power is larger than 23.74 dBm, the total collection delay decreases slower over time. The reason is

that the bandwidths between the reader and devices are fixed, when the charging power is large enough, the bandwidths will be the bottleneck of the transmission rates between devices and the reader.

6) Impact of data imbalance: In this set of experiments, we study the impact due to data imbalance on devices by varying the deviation of the amount of data from 0 to 4.5 M, and the results are shown in Fig. 14. NERF continues to perform consistently better than the other two approaches. As seen in the figure, the total collection delay increases as the deviation increases. This is because with a larger deviation, the amount of data on devices is more imbalance. The reader may need to collect more data packets from devices far from it with very low utilization. Since the total collection delay is decided by the collection time of the last packet, thus, with fixed charging power, the reader need more time to complete the collection.

VI. Related Work

Research on rechargeable wireless networks has received much recent attention due to its potential for battery-less perpetual sensing [9], [14], [18], [21], [23], [29], [8]. In [9], [18], [31], the authors proposed solutions for fair and high throughput data collection in the presence of renewable energy sources, which aims to achieve the maximum data collection rate for each device. Chen, *et al.* [6] considered the problem of maximizing throughput over a finite-horizon time period for a sensor network with energy replenishment. Liu, *et al.* [19] studied the problem of joint energy management and resource allocation in rechargeable sensor networks to maximize network utility while maintaining perpetual operation. In addition, energy harvesting techniques have been considered along with the traditional data collection with a sink node [13], [20], [27], [34]. However, these works do not consider how to schedule packet transmissions to enable the reader to collect data packets from different devices simultaneously with predictable delay bounds.

VII. Conclusion

We present in this paper a novel packet prioritization and transmission protocol that yields probably predictable delay bounds for transmitting packets from multiple micro-powered devices to a single charger. Extensive experimental results demonstrate the efficacy of our proposed protocol, which yields considerable improvements in terms of collection delay compared to existing methods. An interesting future work we plan to investigate is the charger placement problem, i.e., how to place the charger such that both the energy harvesting rate and the collection delay can be improved.

References

[1] Laird s9028pcr. http://www.lairdtech.com/products/s9028pcr.

[2] Speedway rfid reader. http://www.impinj.com/products/readers/.

[3] Wisp 4.1. https://wisp.wikispaces.com/WISP+4.1+DL.

[4] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[5] M. Buettner, R. Prasad, A. Sample, D. Yeager, B. Greenstein, J. R. Smith, and D. Wetherall. RFID sensor networks with the Intel WISP. In *ACM SenSys*, 2008.

[6] S. Chen, P. Sinha, N. B. Shroff, and C. Joo. Finite-horizon energy allocation and routing scheme in rechargeable sensor networks. In *INFOCOM 2011*.

[7] U. C. Devi. *Soft real-time scheduling on multiprocessors*. PhD thesis, University of North Carolina at Chapel Hill, 2006.

[8] Z. Dong, Y. Gu, L. Fu, P. Cheng, L. He, C. Yuen, W. Gao, T. He, and C. Liu. Energy synchronized task assignment in rechargeable sensor networks. In *SECON'16*.

[9] K.-W. Fan, Z. Zheng, and P. Sinha. Steady and fair rate allocation for rechargeable sensors in perpetual sensor networks. In *Sensys 2008*.

[10] L. Fu, P. Cheng, Y. Gu, J. Chen, and T. He. Minimizing charging delay in wireless rechargeable sensor networks. In *INFOCOM 2013*.

[11] J. Holleman, D. Yeager, R. Prasad, J. R. Smith, and B. Otis. Neuralwisp: An energy-harvesting wireless neural interface with 1-m range. In *BioCAS 2008*.

[12] Y. Hou, J. Ou, Y. Zheng, and M. Li. Place: Physical layer cardinality estimation for large-scale rfid systems. In *INFOCOM'15*.

[13] S. Ji and Z. Cai. Distributed data collection and its capacity in asynchronous wireless sensor networks. In *INFOCOM'12*.

[14] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith, and D. Wetherall. Wi-fi backscatter: Internet connectivity for rf-powered devices. In *SIGCOMM'14*, 2014.

[15] T. F. La Porta, G. Maselli, and C. Petrioli. Anticollision protocols for single-reader RFID systems: Temporal analysis and optimization. *IEEE Transactions on Mobile Computing (TMC)*, 10(2):267–279, 2011.

[16] S.-R. Lee, S.-D. Joo, and C.-W. Lee. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification. In *IEEE MobiQuitous*, 2005.

[17] J. W. Liu. Real-time systems. 2000.

[18] R.-S. Liu, K.-W. Fan, Z. Zheng, and P. Sinha. Perpetual and fair data collection for environmental energy harvesting sensor networks. *IEEE/ACM Transactions on Networking*, 19(4):947–960, 2011.

[19] R.-S. Liu, P. Sinha, and C. E. Koksal. Joint energy management and resource allocation in rechargeable sensor networks. In *INFOCOM'10*.

[20] J. Luo, J. Panchard, M. Piórkowski, M. Grossglauser, and J.-P. Hubaux. Mobiroute: Routing towards a mobile sink for improving lifetime in sensor networks. In *Distributed Computing in Sensor Systems*, pages 480–497. Springer, 2006.

[21] A. N. Parks, A. Liu, S. Gollakota, and J. R. Smith. Turbocharging ambient backscatter communication. In *SIGCOMM'14*, 2014.

[22] Y. Sankarasubramaniam, I. F. Akyildiz, and S. McLaughlin. Energy efficiency based packet size optimization in wireless sensor networks. In *Sensor Network Protocols and Applications, 2003*.

[23] L. Shangguan, Z. Yang, L. Alex, and Y. Liu. Relative localization of rfid tags using spatial-temporal phase profiling. In *USENIX NSDI*, 2015.

[24] E. Shih, S.-H. Cho, N. Ickes, R. Min, A. Sinha, A. Wang, and A. Chandrakasan. Physical layer driven protocol and algorithm design for energy-efficient wireless sensor networks. In *MobiCom '01*.

[25] Y. Shu, P. Cheng, Y. Gu, J. Chen, and T. He. Toc: Localizing wireless rechargeable sensors with time of charge. *TOSN*, 11(3):44, 2015.

[26] A. A. Somasundara, A. Ramamoorthy, and M. B. Srivastava. Mobile element scheduling for efficient data collection in wireless sensor networks with dynamic deadlines. In *RTSS'04*.

[27] Z. M. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli. Exploiting sink mobility for maximizing sensor networks lifetime. In *HICSS'05*.

[28] M. Xia, Y. Dong, W. Xu, X. Li, and D. Lu. Mc 2: Multimode user-centric design of wireless sensor networks for long-term monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 2014.

[29] L. Xie, H. Han, Q. Li, J. Wu, and S. Lu. Efficiently collecting histograms over rfid tags. In *INFOCOM'14*.

[30] L. Yang, Y. Chen, X.-Y. Li, C. Xiao, M. Li, and Y. Liu. Tagoram: Real-time tracking of mobile rfid tags to high precision using cots devices. In *Mobicom'14*.

[31] L. Yang, Q. Lin, X. Li, T. Liu, and Y. Liu. See through walls with cots rfid system! In *Mobicom'15*.

[32] H. Zhang, J. Gummeson, B. Ransford, and K. Fu. Moo: A batteryless computational RFID and sensing platform. *UMass, Tech. Rep*, 2011.

[33] P. Zhang and D. Ganesan. Enabling bit-by-bit backscatter communication in severe energy harvesting environments. In *NSDI'14*.

[34] Y. Zheng and M. Li. Read bulk data from computational rfids. In *INFOCOM'14*.

[35] Y. Zheng and M. Li. Fast tag searching protocol for large-scale rfid systems. *In IEEE/ACM Transactions on Networking (TON)*, 2013.