Efficient Rostering of Mobile Nodes in Intermittently Connected Passive RFID Networks

Zhipeng Yang, *Student Member*, *IEEE*,
Ting Ning, *Student Member*, *IEEE*, and Hongyi Wu, *Member*, *IEEE*

Abstract—This paper focuses on the problem of rostering in intermittently connected passive RFID networks. It aims to report a list of tagged mobile nodes that appear in given interested area(s) and time interval(s). Such rostering faces several unique challenges. First, the network consists of two dramatically different types of nodes: powerful static readers and extremely resource-constrained mobile tags. Communication can be established from a reader to a tag only, but not tags to tags or readers to readers. Therefore, the connectivity is very low and intermittent. Besides connectivity, the tag's computation power is also intermittent. It is available only for a short interval when the tag is powered up by a nearby reader, rendering any continuous functions impossible. Moreover, the capacity of tags is so limited that it becomes the critical network resource and communication bottleneck. To address the above challenges, we propose a rostering algorithm that employs a dynamic space-efficient coding scheme to construct hypothetic packet candidates, appraises their values according to information redundancy and tag mobility, and establishes a 0-1 Knapsack model to choose the best set of packets, which together maximize their total (redundancy-excluded) value, but do not exceed the capacity of a tag. We carry out experiments that involve 38 volunteers for nine days and perform large-scale simulations to evaluate the proposed rostering scheme.

Index Terms—Passive RFID, delay-tolerant network, node rostering, intermittently connected network

1 Introduction

While wireless sensors have been broadly used for tracking large animals including zebras [1], whales [2], deers [3], and European badgers [4], earlier investigation has revealed that about 81 percent of bird species and 67 percent of mammal fauna cannot carry any active devices (such as GPS receivers or crossbow sensors), since the weight of the sensors must be under 5 percent of the weight of the animal otherwise such overweight additions often lead to high mortality rates of the animals being studied [5]. Although various efforts have been made to develop miniature sensors, the lowest weight of any active sensor is bounded inevitably by its battery and casing. The former must be sufficient for achieving the desired communication range and lifetime, while the latter must be heavy duty to protect power source and electronic circuits under harsh environments. The strict weight constraint becomes a key hurdle that limits the applicability of active sensors in various applications, not to mention extra hassle for ensuring adequate battery power.

1.1 An Overview of Featherlight Information Network with Delay-Endurable RFID Support (FINDERS)

Built upon Radio Frequency IDentification (RFID) technologies, the FINDERS has been introduced in [6], [7] to address the weight constraints discussed above. It exploits passive RFID tags that are ultralight, durable, and flexible, without power supply for long-lasting pervasive communication and computing applications. A FINDERS system is illustrated in Fig. 1, consisting of two types of nodes, readers and tags. The deployment of readers is carefully planned according to specific applications. For instance, in wildlife and biological studies, readers can be installed at "hot spots" where the animals visit frequently or "choke points" that they have to move through because of significant movement barriers otherwise. The readers are powerful nodes in FINDERS, with large storage space and high computing capacity. While all readers are fixed, the tags are attached to moving targets and thus become mobile (see Tags 1-8 in Fig. 1).

The communication in FINDERS is extremely challenging. Since FINDERS aims to support applications in remote fields, infrastructure-based communication networks (such as cellular, WiMAX, and telemetry systems) are unavailable. Moreover, due to harsh and complex wildlife environments with obstacles, it is not practically viable for most readers to access satellites; neither can they establish reliable connections (e.g., via Wifi, Zigbee, or VHF/UHF radio) to communicate with each other. As a result, they become isolated readers, or IRs (see IRs 1-4 in Fig. 1).

Only a few readers at convenient locations can access reliable network connections. They are dubbed gateway readers (or GRs), serving as gateways between FINDERS and conventional network infrastructure (see GRs 1 and 2 in Fig. 1).

• *The authors are with the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, LA 70504. E-mail: {zxy1767, txn6704, wu}@cacs.louisiana.edu, txn6704@louisiana.edu.*

Manuscript received 2 Dec. 2011; revised 25 June 2012; accepted 21 July 2012; published online 1 Aug. 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-12-0650. Digital Object Identifier no. 10.1109/TMC.2012.170.

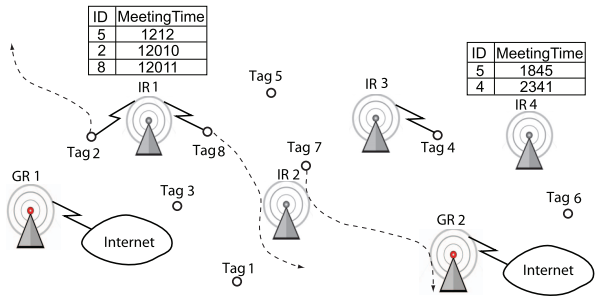


Fig. 1. An overview of FINDERS.

Since GRs are well connected, they can share data with a negligible delay compared with the average time to deliver data from one IR to another and thus can be viewed as a single virtual node. On the other hand, as IRs are isolated, their communication is enabled by the mobile tags that establish time-varying opportunistic links with nearby readers and thus form an intermittently connected delay-tolerant network (DTN [8]) for data delivery. An example of such distinctive communication paradigm is illustrated in Fig. 1. Since Tags 2 and 8 are located in IR 1's read/write range, IR 1 may read the data from Tag 2 and then write them into Tag 8. When Tag 8 passes by IR 2, the former unloads its data to the latter. IR 2 subsequently writes the data into Tag 7. With its trajectory through GR 2, Tag 7 can thus deliver the data to its destination via this gateway.

1.2 System Configuration

Many off-the-shelf passive tags that are light and durable and have sufficient reading/writing ranges can be employed in FINDERS. For example, we have adopted the Alien passive RFID system for this research (see Fig. 6). The readers are powerful devices, with sufficient storage and battery capacity. An Alien ALR-9900 reader possesses of 64-MB RAM and 64-MB flash memory. The trial data show that a typical car battery of $12\ \mathrm{V} \times 60\ \mathrm{Ah}$ can supply the reader for 20 to 35 hours with its scanning frequency ranging from 1 to 1/60 Hz. With a suitable solar charger, the battery is sufficient to sustain the continuous function of the reader in a wide range of wildlife applications. The reader is also equipped with an interface for extended computing power and storage capacity. On the other hand, being very thin and light, passive RFID tags are attached to mobile objects (such as the wildlife being studied). For example, an Alien ALN-9540 tag adopted in our experiments measures $8.15 \times 94.8 \times 0.05\ \mathrm{mm}$ and weights less than 1 gram. A tag has extremely limited storage space. The Alien tag can hold up to 20 bytes only. A block-based scheme will be introduced in Section 3 to expand tag capacity by leveraging the aggregated storage space of multiple passive tags, achieving a total capacity of tens to hundreds of bytes. Without loss of generality, we simply refer to a tag that can be a single tag or a block of tags, and let W denote its storage capacity in the following discussions.

Our field experiments have revealed that the Alien system can achieve a reading/writing distance of some 6 meters.

Each reader or tag is associated with a unique ID. A reader periodically scans nearby tags. When a reader detects a tag, it records a meeting event in its *local meeting table*. A collection of meeting events intrinsically provide a discrete sampling of the movement of mobile nodes. A local meeting table of a reader includes the IDs of the mobile nodes and their meeting time. Fig. 1 illustrates examples of the local meeting tables of IRs 1 and 4. As can be seen, a mobile node may be involved in multiple meeting events with different IRs at different time. Each entry of the table takes about 6 bytes. Since the meeting frequency is low in our target applications (generally lower than once per minute), the flash memory of the reader can hold such data for years without overflow.

1.3 Rostering of Mobile Nodes in FINDERS

This paper focuses on the rostering problem based on the FINDERS architecture. Rostering is fundamental to wildlife research. It aims to aggregate local information by individual RFID readers to collectively report a roster of tagged mobile nodes that appear in given interested area(s) and time interval(s). Rostering is different from the counting [9], [10], [11], [12] or count estimation [13], [14], [15], [16] problems, where only a headcount is desired by requiring a report of spatial and temporal view of the monitoring field. In addition, recent works on efficient RFID tag identification [17], [18], [19], [20], [21], missing tag detection [18], [22], [23], localization [24], [25], [26], [27], [28], and tag authentication and privacy [29], [30], [31], [32] are not applicable to the rostering problem either.

Rostering involves both communication and computing perspectives. When a mobile tag moves into the communication range of a reader, the latter detects the former and records a meeting event in its local meeting table. A collection of meeting events intrinsically provide a discrete sampling of the movement activity of the mobile nodes. A straightforward scheme is to transmit all meeting events to the GRs, which are accessible by end users anytime. However, this naive approach is impractical, due to continuous updates on meeting events and extremely tight constraints on communication capacity in FINDERS. In particular, while the GRs have reliable network connections and thus are ready for access at any time, it is nontrivial to retrieve all meeting events from the intermittently connected IRs. As a result, the local meeting events must be kept at individual readers in a distributed manner. Consequently, the rostering problem can be viewed from a distributed database perspective. A user's query request is sent to a GR, which subsequently instructs selected IRs to report a set of selected meeting events. Such meeting events must be efficiently aggregated at the source and intermediate nodes for economical communication.

To this end, the rostering problem can be viewed from a distributed database perspective, where collections of data are stored at networked servers distributed across multiple physical locations, which are similar to the local meeting tables maintained by scattered readers. A GR serves as a portal for external applications to query the database. For example, a rostering request can be mapped to a typical SELECT statement in database systems, aiming to create a roster of mobile nodes that visit a set of readers in a time interval:

```
SELECT tagID
FROM MeetingTable
WHERE timestamp > begin time
AND timestamp < end time
AND readerID > 'id1'
AND readerID < 'id2'
```

Clearly, one may extend the above statement by specifying multiple time intervals and multiple sets of readers. A set of readers represent a sampled area that the mobile nodes visit. A time interval can be past, current, or future. If the current clock is greater than the end time, the rostering process is based on previously recorded meeting events. When the clock is less than the start time, the request alerts readers to initiate rostering during a future interval. Otherwise, with the current clock between the start time and the end time, it is an ongoing rostering. Part of the meeting events are already available, while new events will be added until the end time.

In addition to query, a GR may send other commands too, for example, to request IRs to delete some obsolete events, or to reconfigure IR's scanning frequency, or to modify algorithmic parameters in rostering and communication. All of them can be viewed as typical statements in data manipulation language, data definition language, or data control language in database systems.

1.4 Challenges

The detection of meeting events largely follows standard RFID operations. The reader periodically scans nearby tags. Its scanning frequency (i.e., number of scans per second) may be tuned according to the requirement of applications. The higher the scanning frequency, the more meeting events are detected, providing a higher resolution of mobile node's movement.

On the other hand, the transmission of meeting events and other control messages is nontrivial, becoming the key challenge for achieving efficient rostering. The problem stems from the intermittent connectivity in FINDERS. As discussed earlier, the tags serve as transportation vehicles, carrying data packets from one reader to another. Since the mobility of tags is uncontrolled and the storage space of a tag is small (ranging from tens to hundreds of bytes), the communication capacity is extremely limited and the communication links are opportunistic. Therefore, a reader must fully exploit the capacity of tags whenever they are available. Since the capacity of a tag is fixed, the challenge is to fit as much useful information as possible onto the tag. The reader has many possible options to pack its information of meeting events, creating various hypotheses of data packets, with different amount of valuable information and packet lengths. A distributed algorithm is needed to determine the set of packets to be written into the tag, to maximize its effective information per bit (IPB) transmitted (i.e., the entropy). Moreover, in addition to delivering data from IRs to GRs, commands and feedbacks must be transmitted from GRs to IRs, which have not been studied before in the context of a passive RFID network like FINDERS [6]. To this end, several critical issues must be carefully addressed:

- Data packets must bear a compact format, which is tailored for adaptive data aggregation in rostering.
- Part of the data maintained by different readers can be redundant for a given rostering request. More redundancy is usually generated during data transmissions. Therefore, individual readers must prioritize their data for efficient utilization of the precious communication capacity.
- Mobile nodes with different mobility patterns are suitable for carrying different packets. For example, a mobile node likely moving toward a GR is more efficient for delivering meeting events, while the mobile nodes traveling away from GRs are more effective to serve rostering commands and feedbacks. An online learning mechanism is desired to estimate the mobility patterns of mobile nodes.
- A distributed algorithm should be devised to choose the best set of packets according to information redundancy and tag mobility, aiming to maximize their total value and at the same time do not exceed the capacity of a tag.

The above challenges are addressed in this work to develop an effective algorithm for mobile node rostering, based on several communication and computing techniques specifically tailored for FINDERS. When a communication opportunity becomes available between a reader and a tag, a dynamic space-efficient coding scheme is employed to construct hypothetic packet candidates, which are appraised according to information redundancy and tag mobility. A distributed algorithm based on 0-1 Knapsack model is devised to choose a set of packets, which together maximize their total (redundancy-excluded) value but do not exceed the capacity of the tag. We have carried out experiments that involve 38 volunteers for nine days and performed large-scale simulations to evaluate the proposed rostering scheme.

The rest of the paper is organized as follows: Section 2 introduces our proposed rostering algorithm. Section 3 describes implementation and testbed experiments. Section 4 presents simulation results. Finally, Section 5 concludes the paper.

2 Proposed Rostering Algorithm

In this section, we first present an overview of our proposed scheme and then elaborate algorithmic details for creating, appraising, and choosing data packets for effective rostering.

2.1 Overview of the Rostering Algorithm

Since the data transmission opportunities are available only when tags meet readers, the performance of the system largely depends on the decision made by the reader on its reading and writing operations. Fig. 5 illustrates the basic procedure for a reader to communicate with a tag. The reader periodically scans the channel. If it finds a tag (or a block of tags attached to an object), the information kept on the tag is read out and a meeting event is logged by the reader. After the tag is identified, the reader generates a candidate packet list and calculates the weight and value for each candidate packet based on the best known

CMD ID	Gen Time	CMD Counter	Priority	Rostering Command
1	1823	1	10	T=[1024,5523] A=[IR1,IR3]
3	2421	2	3	T=[5224,9800] A=[IR2,IR5]

(a) Command Table

ID	Reply Time t_j^R	Acked	FB Time t_j^F	Reply Counter m_j^R	FB Counter m_j^F
1	2102	Y	4233	5	3
3	3222	N	—	2	0
10	4258	N	—	1	0

(b) Roster1 (for Command1)

Fig. 2. Tables maintained at a reader.

information of data priority and redundancy. Finally, the optimal set of packets are chosen and written onto the tag.

We introduce three types of packets for rostering:

- *Command packet*: A rostering command is issued by a GR. Similar to the *SELECT* statement discussed in Section 1.3, a *Command* packet includes the interested time intervals (denoted by T) and areas (denoted by A). The latter is represented by readers's IDs. A command without A is meant for the entire area. A globally unique sequence number (called Command ID) is associated with each command.
- *Reply packet*: Zero to multiple *Reply* packets may be created by an IR, in response to a rostering command. A *Reply* packet contains the IDs of the mobile nodes with meeting events that satisfy T and A. The details of creating *Reply* packets will be elaborated in Section 2.2.
- *Feedback packet*: One or multiple *Feedback* packets are generated by a GR, corresponding to a command. A *Feedback* packet contains the mobile node IDs that have been received by GRs, facilitating IRs to eliminate unnecessary redundant data for efficient channel utilization.

In a nutshell, *Command* packets are dispersed from GRs to IRs. Consequently, *Reply* packets are created by IRs and delivered to GRs for rostering. Meanwhile, *Feedback* packets are distributed from GRs to IRs to filter out node IDs that have been received. More specifically, each reader (either a GR or IR) maintains a *local meeting table* as discussed in Section 1.2. In addition, it keeps a command table that includes all active rostering commands it has received (see Fig. 2a) and the rosters it has learned so far for each command (as illustrated in Fig. 2b). A roster entry includes the ID of a mobile node, the time when the corresponding meeting event was extracted for the command (i.e., "ReplyTime," denoted by t_j^R for ID j), the number of times that ID j has been transmitted by the reader in *Reply* packets (i.e., "ReplyCounter," m_j^R), a feedback flag (i.e., "Acked") to indicate if ID j has been received by GRs, the time that the feedback is generated (i.e., "FBTime," t_j^F), and the number of times that ID j has been transmitted by the reader in *Feedback* packets (i.e., "FBCounter," m_j^F). Note that while m_j^R and m_j^F are local knowledge and thus known by the reader accurately, t_j^R, t_j^F, and "Acked" are based on best known information, which is not always precise. Similarly, an entry in a command table includes a command ID, the time issuing the command, the number of times it is transmitted by the reader, its priority, and a description of the command. Algorithm 1 outlines the procedure for the IR to update its local meeting table, and related table at the time a tag is identified and the packet is received. The local meeting table keeps all meeting events of the IR, while the Id2Rostering table contains the rostering tables for all rostering commands. Each rostering table is indexed with the command ID. Once a packet is received, the local meeting table is updated, followed by possible updating of Id2Rostering table and command table.

Algorithm 1: Tag Identification

Input:
p: the packet received
$p.tagId$: tag id, $p.queryId$: rostering command id
$p.type$: packet type, $p.ids$: rostering ids
t: current timestamp
localMTable: the local meeting table of the IR
cmdTable: the table to store rostering command
id2RTable: the table of rostering table for each command

```
1  put < p.tagId, t > to localMTable
2  if p.type == Query then
3      if p.queryId not exist in Id2RTable then
4          table←search the localMTable according to
           condition defined in p;
5          id2RTable←(p.queryId, table);
6          put < p.queryId, time > to the cmdTable if
           no identical query exists;
7  else if p.type == Reply or p.type == Feedback then
8      table← search id2RTable with p.queryId;
9      if table is null then
10         table← create an empty rostering table with
           p.queryId;
11         Id2RTable←< p.queryId, table >;
12     foreach id in p.ids do
13         if p.type == Feedback then
14             table(id).Acked←Yes;
15             if table(i).FBTime is not set then
16                 table(id).FBTime←time;
17         else
18             if table(i).ReplyTime is not set then
19                 table(id).ReplyTime←time;
```

A user may send a *Command* packet to any GR, which serves as the head GR for the command. We assume the GRs are connected to a reliable network infrastructure (e.g., the Internet), and thus can always synchronize their rostering commands and rosters. A tag may carry a mixed set of *Command*, *Reply*, and *Feedback* packets. When a tag meets a GR, the GR first reads the information on the tag. If the tag contains *Reply* packets, the GR obtains the mobile node IDs therein and aggregates them into the corresponding rosters. Then, it writes *Command* and/or *Feedback* packets into the tag. When a tag meets an IR, the IR again

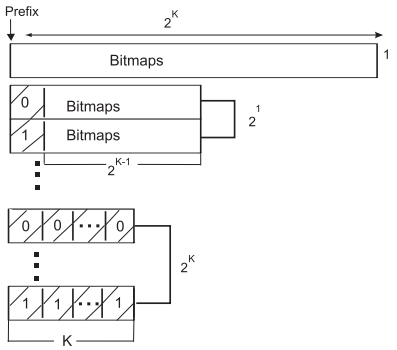


Fig. 3. Hypothetic packet candidates.

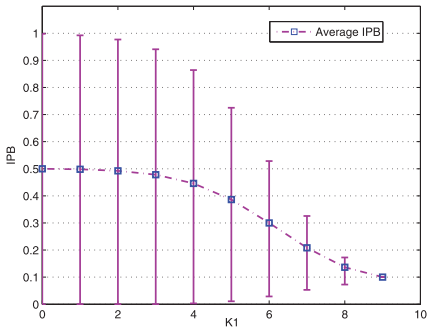


Fig. 4. IPB under different K_1 ($K = 9$).

first reads the information on the tag. If there is a *Command* packet and the command is new to the IR, it inserts the command into its command table and creates an empty roster for the command. It then immediately looks up its *local meeting table* to extract meeting events that satisfy the command and inserts corresponding mobile node IDs into the created roster. If the tag contains *Reply* and/or *Feedback* packets, the IR updates its rosters by inserting new entries (according to the new mobile node IDs in *Reply* packets) or updating feedback flags of existing entries (according to *Feedback* packets). Then, it creates a set of hypothetic packet candidates, appraises their values, and decides a set of most valuable packets to be written into the tag for delivering mobile node IDs to GRs, and/or disseminating commands or feedbacks. The details will be discussed in the next sections. Finally, the head GR replies to the user with the roster it learns after a given time period.

As can be seen, a reader not only maintains local meeting events but also receives information from remote readers. The commands, replies, and feedbacks are duplicated during their transmissions. It is common to have multiple copies of the same information at different readers. Therefore, different data may have different values. For example, if a mobile node ID already has many copies across the network, it will be less valuable to include it in a *Reply* packet, because such information might have been received by a GR or will be soon delivered to a GR by other tags in the network. As discussed in Section 1.4, the key challenge in rostering is to efficiently utilize the extremely limited communication capacity for transmitting *Command*, *Reply*, and *Feedback* packets. A reader must determine the best set of packets to be written into a tag whenever such communication opportunity becomes available, arriving at a resource optimization problem to be addressed next.

Note that multiple rostering commands can be executed simultaneously. Moreover, they may be prioritized such that more communication bandwidth (i.e., the capacity of tags) is allocated to important or urgent command.

2.2 Packet Format and Hypothetic Candidates

A tag may carry one or multiple packets. A packet must be appropriately formatted to effectively utilize the extremely limited tag capacity for transportation of valuable information. Each packet includes four fields, a *Type* field of 2 bits, a *Command ID* field with a fixed length, a *Data* field with a variable length, and a *Timestamp* field. The *Type* indicates whether it is a *Command* or *Reply* or *Feedback* packet. The *Command ID* contains the sequence number of the command for which a *Command* packet requests or to which a *Reply* or *Feedback* packet responds. The *Data* field of the *Command* packet simply includes the interested time intervals and areas as introduced in Section 2.1. On the other hand, the *Data* fields of *Reply* and *Feedback* packets are worth further elaboration, because they are specially designed to suit rostering.

First, we give a simple example to show the impact of format in *Reply* and *Feedback* packets. Assume that an IR has detected two mobile nodes with IDs of 11010 and 11011 that satisfy a rostering command. In a straightforward approach, the IR may create two *Reply* packets. Each of them contains an ID, consuming a total of 10 bits (if other fields are ignored here). Alternately, we note that the higher 4 bits of the two IDs are identical. Therefore, the IR may create a *Reply* packet with a prefix of 1101 plus two bitmap bits, i.e., $1101ab$. The bitmap bits (i.e., a and b) correspond to the last bits of the mobile node ID. If a is set to 1, it indicates that there is an ID with a prefix of 1101 and the last bit of 1 (i.e., 11011); or there is not such an ID if $a = 0$. Similarly, b is set to indicate the existence of 11010. Based on this approach, the IR can create a *Reply* packet with only 6 bits, i.e., 110111, for the above example. However, it does not always save space by employing bitmaps. For instance, if the two IDs are 11010 and 10100, then they only share a prefix of 1 bit, and it will take 16 bits to construct a bitmap for the remaining 4 bits in the IDs, summing up to 17 bits in total.

Generally, to support N mobile nodes, each ID needs $K = \lceil \log_2 N \rceil$ bits. We adopt a hybrid bitmap code to format *Reply* and *Feedback* packets, where the higher K_1 bits ($0 \leq K_1 \leq K$) of IDs are chosen as prefix, and 2^{K-K_1} bits are appended as bitmaps. When $K_1 = K$, it is a complete bitmap as shown at the top of Fig. 3. On the other hand, it yields simple mobile node IDs when $K_1 = 0$ (see the last group of codes in Fig. 3). There are up to $2^{K+1} - 1$ hybrid bitmap codes.

Here, we introduce the *IPB* for a *Reply* or *Feedback* packet, defined as the number of IDs it carries divided by the length of its *Data* field. The IPB under $K_1 = 0$ is a constant of $\frac{1}{K}$ with no regard to the number of mobile IDs, because each packet always contains one ID and consumes K bits. When $K_1 = K$, the bitmap consumes 2^K bits, and its IPB varies from 1 to $1/2^K$, depending on the presences of mobile node IDs. If there are 2^K IDs, every bit of the bitmap is set, achieving an IPB of 1; however, if there is only one ID, it becomes very inefficient with an IPB of as low as $1/2^K$. Fig. 4 illustrates the variation of IPB under different K_1. In

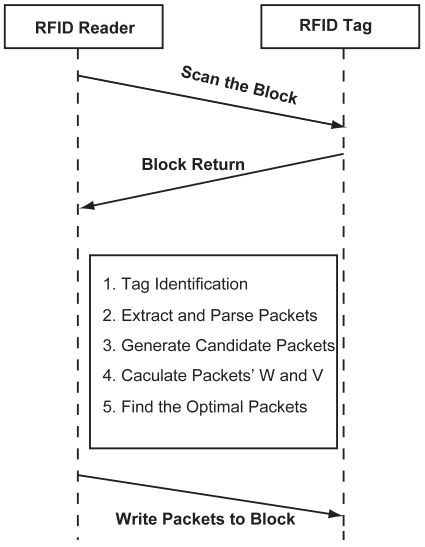


Fig. 5. Sequence diagram for communication between a reader and a tag.

addition, the possible packet format is also limited by the maximum length of a packet (which is usually constrained by the storage capacity of tags).

The *Reply* and *Feedback* packets differ in their *Type* fields only. They share the same hybrid bitmap format discussed above in their *Data* fields. The length of the prefix is included at the beginning of the *Data* field. Since the maximum prefix length is K, the first $\lceil log_2 K \rceil = \lceil \log_2 \log_2 N \rceil$ bits of the *Data* field are reserved to indicate the length of the prefix.

Based on the packet format introduced above, a reader creates hypothetic packet candidates, i.e., the possible packets to be transmitted. It creates a hypothetic *Command* packet candidate for each command in its command table, and hypothetic candidates for *Reply* and *Feedback* packets according to the rosters it has learned for each rostering command. For a given roster, the entries with "Acked=N" are used to build *Reply* packets, while others are for *Feedback* packets. There are $K+1$ different ways to create *Reply* (or *Feedback*) packets, with different lengths of prefix in the hybrid bitmap code (as shown in Fig. 3 with K_1 varying from 0 to K). As a variation of the standard hybrid bitmap code, the zeros at the end of a code can be discarded to further reduce packet length.

The compression scheme proposed above is based on the ID compressibility and will naturally favor the continuous IDs. To prevent starving of some less-compressible IDs, the redundancy level of each ID will be considered. The value of an ID depreciates when more redundancy is spread across the network. The more times an ID is transmitted, the less value the ID will have, aiming to achieve the desired fairness for data transmission. The next section discusses how to determine the values of the IDs and data packets.

2.3 Appraisal and Selection of Hypothetic Packet Candidates

Till now, we have obtained a set of hypothetic packet candidates. It is generally infeasible to write all of them into a tag due to limited tag capacity. As a result, the reader must choose a subset of them with the highest value for transmission. To this end, we formulate such optimization as a 0-1 Knapsack problem. More specifically, the available storage capacity of a tag is W. Each hypothetic packet candidate is associated with a weight and a value, denoted by w_i and v_i, respectively, for Packet i. Therefore, we have

$$\begin{aligned} &Maximize: \sum_{i=1}^{n} v_i x_i \\ &Subject\ to: \sum_{i=1}^{n} w_i x_i \le W,\ x_i \in \{0,1\}, \end{aligned} \tag{1}$$

where n is the total number of hypothetic packet candidates and $\{x_i \mid 1 \le i \le n\}$ are 0-1 variables to be determined.

In the above formulation, W is known. It is the capacity of a tag in bits, excluding the space reserved for control information (e.g., the mobile node ID). w_i can be readily obtained for each hypothetic packet candidate by counting its length in bits. The value, i.e., v_i, is appraised according to information redundancy and tag mobility. We first discuss the appraisal for *Reply* and *Feedback* packets and then *Command* packets, followed by the adjustment according to tag mobility.

2.3.1 Appraisal of Reply and Feedback Packets

The valuable information contained in the *Reply* and *Feedback* packets is the mobile node IDs. So, the value of a *Reply* or *Feedback* packet is defined as the sum of the values of individual IDs it contains, i.e., $v_i = \sum_{j=1}^{c_i} u_j$, where c_i is the number of IDs contained in Packet i and u_j is the value of the jth ID.

As discussed in Section 2.1, data are duplicated during their transmissions in FINDERS, creating redundancy. The information depreciates when more redundancy is spread across the network. Therefore, u_j should be determined according to the amount of redundant copies of its information. However, it is extremely costly to keep track of such redundancy. In this work, we estimate the redundancy by two factors. First, from the global perspective, the longer the information has been propagated, the more redundancy is usually generated in the entire network. Second, each reader records how many times it has transmitted the same information (i.e., *ReplyCounter* or *FBCounter* shown in Fig. 2b), which serves as a local estimation of redundancy. We define $u_j = P_1(1-\eta_1)^{\lfloor \frac{t-t_j}{\Delta} \rfloor} / m_j$, where P_1 is a priority parameter (to differentiate different rostering tasks), η_1 is a depreciation factor, t is the current time, t_j is the time that the information was generated, Δ is a constant, and m_j is the number of times the reader has transmitted the mobile node ID (i.e., j) for this rostering command. $t_j = t_j^R$ and $m_j = m_j^R$ for *Reply* packets (or $t_j = t_j^F$ and $m_j = m_j^F$ for *Feedback* packets) are available in the roster (see Fig. 2b). Note that since there is no separate time stamp for each ID, t_j^R and t_j^F are estimated by the packet-level time stamp and thus usually larger than the true value. The calculation of appraisal for Reply and Feedback packets is summarized in Algorithm 2.

Algorithm 2: Appraisal Calculation for Reply and Feedback Packets

Input:
l: list of candidate packets
t: current timestamp
querytable: the rostering table for the specific id at an IR
Output:
l: list of candidate packets

```
foreach packet in l do
    foreach record in querytable do
        if record.id can be represented by packet then
            Set packet's corresponding bitmap to 1;
            if packet.type == Reply then
                valueofbit ← calculate the value of the Reply information;
            else
                valueofbit ← calculate the value of the Feedback information;
            packet.appraisal←packet.appraisal+valueofbit;
```

2.3.2 Appraisal of Command Packets

A *Command* packet is different from *Reply* and *Feedback* packets, because it does not contain individual mobile node IDs. Therefore, the reader directly determines the value of the whole packet. Similar to above discussions, we have $v_i = P_2(1-\eta_2)^{\lfloor \frac{t-t_i}{\Delta} \rfloor}/m_i$, where P_2 is the priority parameter, η_2 is the depreciation factor, t_i is the time that the command was issued, and m_i is the number of times the reader has transmitted this rostering command.

2.3.3 Appraisal Adjustment

So far, we have discussed how to calculate the value for a packet according to redundancy. But note that the appraisal aims to facilitate the selection of a set of packets to be written onto a tag. As a result, the value depends not only on the information itself, but also on the tag's mobility.

In general, the precise mobility patterns of the tags are unknown to our algorithm. However, it is very beneficial to learn at least a roughly estimated moving pattern of a tag. In particularly, if a tag has a high probability to move toward GRs, it may serve as an excellent vehicle to carry *Reply* packets (whose destination is GRs). Otherwise, if it moves away from the GRs, it will be more efficient to transport *Command* or *Feedback* packets for disseminating such information to IRs. To this end, we employ the effective delivery capability (EDC) to reflect the node's cascaded probability to "reach" GRs. Let ξ_i denote the EDC of Node i. The EDC of a GR is always 1, and the initial EDC of an IR or a tag is zero. For any two Nodes i and j (in which one must be a reader and the other must be tag), if Node i is not a GR and meets Node j with $\xi_j > \xi_i$ at time t, ξ_i is updated to $(1-\eta_3)^{\lfloor \frac{t-t_o}{\Delta} \rfloor}\xi_i + \eta_3\xi_j$; if Node i does not meet any Node j with $\xi_j > \xi_i$ at time t, ξ_i is updated to $(1-\eta_3)^{\lfloor \frac{t-t_o}{\Delta} \rfloor}\xi_i$, where η_3 is a constant to keep partial memory, Δ is a shaping constant, and t_o is the time when ξ_i was updated last time.

For example, if Node i that is a tag with $\xi_i = 0$ meets Node j that is a GR with $\xi_j = 1$, such an update will increase the EDC of Node i to reflect its chance to meet the GR. When the tag subsequently meets an IR with an EDC of zero, the latter will make an update according to the former. As a result, the IR gains a cascaded probability to "contact" the GR. The process repeats. It has been proven in [7] that if the nodal mobility is statistically stable, the EWMA exhibits long-term stability, and thus, the mean of an EDC will converge.

When a tag meets an IR, their EDCs are compared. If the IR has a higher EDC than the tag has, it implies a low opportunity for the tag to deliver data to GRs either directly or indirectly. Thus, it will be inefficient to let the tag carry *Reply* packets. Similarly, if the tag has a higher EDC, it is unsuitable for *Feedback* or *Command* packets. In addition, since only relative value matters, we need to make adjustment on the value of either *Reply* or *Feedback/Command* packets, but not both. Based on the above observations, we define the following function to adjust the value of *Reply* packet i:

$$\tilde{v}_i = \begin{cases} \max\left(\beta_1 \frac{\xi_{tag}}{\xi_{IR}}, \beta_2\right) v_i, & \xi_{IR} < \xi_{tag} \\ \min\left(\beta_1 \frac{\xi_{tag}}{\xi_{IR}}, \frac{1}{\beta_2}\right) v_i, & \xi_{IR} \geq \xi_{tag}, \end{cases} \quad (2)$$

where β_1 and β_2 are constants to shape the adjustment.

Note that EDC reflects the long-term probability. It does not ensure that a tag with higher EDC always moves toward the GRs or vice versa. However, it offers a probabilistic prediction about the moving direction of the tag. Given such an intermittently and opportunistically connected network as FINDERS, a node cannot precisely predict the availability of a communication link. As a result, data transmission must be based on a probabilistic approach. In other words, the decision to transmit (or not to transmit) a data packet is made to maximize the likelihood of successful data delivery. It does not guarantee that every decision on data transmission is correct. Our results show that the probabilistic prediction based on EDC effectively increases the efficiency to utilize the scarce communication opportunities and thus improve the overall network performance.

2.3.4 Optimization by a 0-1 Knapsack Model

Till now, the reader has known W, w_i, and v_i, and thus is ready to solve the 0-1 Knapsack problem given in (1). The 0-1 knapsack problem is NP-complete. A dynamic programming solution that runs in pseudopolynomial time is adopted here [33]. Algorithm 3 outlines the method for solving the 0-1 Knapsack problem. Note that the total value of a set of *Reply* or *Feedback* packets is not the simple sum of individuals' values. It is calculated according to the union of the mobile node IDs of those packets. The algorithm determines 0-1 variables, $\{x_i \mid 1 \leq i \leq n\}$, i.e., the set of packets to be written into the tag, which together do not exceed the capacity of the tag and at the same time maximize the total value of the information being carried.

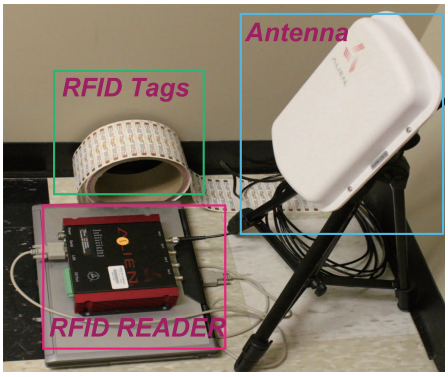


Fig. 6. Our Alien passive RFID gear.

Algorithm 3: Dynamic Programming to get optimal packaging solution

Input:
p: list of candidate packets
n: number of candidate packets
W: available space in the tag
Output: S[n,W]: Solution contains the optimal packets

```
for i ← 0 to W do
    S[0,i].value = 0;
for j ← 1 to n do
    S[j,0].value = 0;
for i ← 1 to n do
    for w ← 0 to W do
        if p[i].weight ≤ w then
        packets←S[i-1,w-p[i].wieght].packets;
        addsolution←packets⊕p[i];
        if addsolution.value≥S[i-1,w].packets.value
        then
        S[i,w].packets.add(p[i]);
        S[i,w].value←addsolution.value;
        else  S[i,w].value = S[i-1,w].value;
return S[n,W];
```

3 Experiments and Results

To demonstrate the feasibility and empirically evaluate the efficiency of the proposed rostering algorithm, we have carried out experiments based on the off-the-shelf RFID equipment supplied by Alien Technologies. We have acquired five sets of Alien passive Class1Gen2 RFID systems with five ALR-9900 readers and 1,000 ALN-9540-WR Squiggle tags (see Fig. 6 for a photograph of the reader and tags). The readers are programmed by using the vendor's applications development kit. In this section, we first discuss the implementation issues and then introduce our experiments and results.

3.1 Implementation Issues

The implementation of our proposed rostering algorithm is largely straightforward by following the description in Section 2. It does not require any modification on the off-the-shelf tags or standard reader commands. Only a small amount of codes for hypothetic packet candidate creation, appraisal, and selection need to be added to reader's program.

However, it is worth a discussion on an implementation issue mentioned in previous sections but not yet addressed. The capacity of a low-cost passive RFID tag is extremely limited. For example, an Alien ALN-9540-WR "Squiggle" tag adopted in our experiments has a memory space of 160 bits, in which only 128 bits are usable for data storage. Such limited capacity leads to the communication bottleneck in FINDERS. Therefore, it is highly desirable to support expansion of tag memory, according to the communication needs in specific applications. To this end, we devise an adaptive expansion scheme-based off-the-shelf Alien ALN-9540-WR tags. Since the Alien tag is Class1Gen2 (C1G2) standard compliant, our scheme can be applied to other standard passive tags as well.

We introduce "blocks" to expand tag capacity. A block is an integral unit attached to a mobile node, consisting of a head tag and zero or multiple storage tags. The tags in a block share the same Node ID. The length of Node ID is chosen according to the estimated number of nodes in an application. For example, we allocate 10 bits for Node ID in our implementation. The tags within a block are uniquely identified by their Tag IDs, whose length depends on the maximum storage space needed for a block. For example, one may choose 4 bits for Tag ID to support up to 16 tags in a block. The Tag ID of the head tag is predefined to 1111. This design effectively shortens communication delay. To scan nearby tags, the reader first sets its mask to 1111 for head tags only, thus reducing undesired collisions between head and storage tags. Once a head tag is identified, the reader uses the Node ID and individual Tag ID to generate a unique mask for the next tag in the block, so on and so forth, achieving collision-free communications.

Since each tag weights less than 1 gram, the total weight of multiple tags is up to a few grams, which does not exceed the weight constraint. Note that the idea of constructing blocks is not imperative to the proposed rostering algorithm. A more capable tag with large capacity can be certainly employed, instead of using a block of small tags, if it is available.

3.2 Testbed Setting

An experimental testbed has been set up to gain useful empiric insights of mobile node rostering in FINDERS. For fair comparison, trace data are collected to run comparable schemes. Our testbed consists of five readers deployed in the building of Computer Science Department. Reader 1 is located at the entrance of a major classroom (Room 117) on the first floor. Reader 2 is installed at a large research lab (Room 228) on the second floor. Three readers (i.e., Readers 3, 4, and 5) are on the third floor, close to the doors of a small lab and two faculty offices. Each reader is equipped with two side-by-side 6dBi circular polarized antennae. Readers 1-4 serve as IRs, while Reader 5 is the GR. All readers scan nearby tags at a frequency of once per second. The parameters used in the testbed is: $P_1 = 1$, $P_2 = 8$, $\eta_1 = \eta_2 = \eta_3 = 0.001$, $\beta_1 = 2$, $\beta_2 = 10$, and $\Delta = 600$.

Thirty-eight volunteers had participated in our experiments, including faculty members, senior PhD students

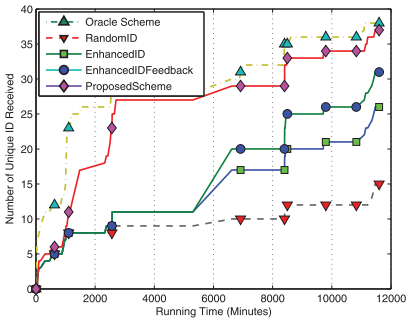


Fig. 7. Roster grows over time.

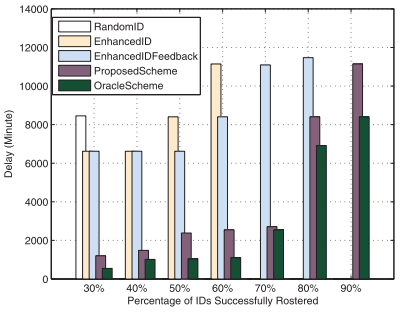


Fig. 8. Delay versus rostering rate.

(who do not have classes), graduate students at MS level (who go to classrooms regularly), and undergraduate students. Our experiment lasted nine days. Each participant carries a badge (a typical plastic badge used in conferences), with an Alien ALN-9540-WR tags enclosed. A tag contains four tag-level fields plus a number of packets. The tag-level fields are Node ID, Tag ID, EDC, and Timestamp, which consumes 10, 4, 14, and 20 bits, respectively. Thus, a tag has 80 bits left for packets. As discussed earlier, a packet includes four fields, a *Type* field of two bits, a *Command ID* field of 8 bits, a *Data* field with a variable length, and a packet-level *Timestamp* of 20 bits. The number of packets that can be carried by a tag depends on the *Data* field. For example, if the *Data* field contains a mobile node ID only, up to two packets can be written into a tag.

3.3 Experimental Results

To evaluate the performance of the proposed rostering algorithm, we have considered several other schemes for comparison. "RandomID" is a naive approach where an IR randomly chooses a set of mobile node IDs that satisfy a rostering command for transmission. It often results in unnecessary high redundancy and long delay. "IDEnhancedID" is similar to the RandomID scheme but with redundancy and EDC taken into account. The IR calculates the EDC for each ID and transmits the IDs with lowest EDCs. "IDEnhancedFeedback" further improves with a simple feedback mechanism. When a tag meets a GR, the GR writes the recently received IDs to the tag to inform IRs that they need no longer transmit such IDs. "ProposedScheme" is the proposed scheme. "Oracle Scheme" assumes that a mobile node ID is received as soon as it is detected by any reader, providing an unachievable performance upper bound. A command is issued to create a roster of mobile nodes in the entire area and experimental period.

Fig. 7 shows the number of unique mobile node IDs (i.e., the length of the roster) received by the GR over time. Generally, the roster grows with time. However, a flat interval is observed between 2,500 to 5,500 minutes, because that was Saturday and Sunday when few participants visited the Computer Science Building, resulting in nearly zero meeting events or communication opportunities. The proposed algorithm achieves a performance close to the oracle results, significantly outperforming other schemes. The difference between EnhancedIDFeedback and EnhancedID is subtle. The former employs feedback to remove some redundant information and thus is more efficient in channel utilization, in comparison with the latter. But it is effective for the IRs close to the GR only. RandomID leads to the worst performance. It yields a roster with less than half of the mobile node IDs, because it completely ignores the priority of IDs, and thus, the redundancy and overhead become overwhelming in the system.

Fig. 8 illustrates the delay for rostering x percent of unique mobile node IDs. Since no scheme (except the Oracle scheme) can achieve 100 percent ID rostering efficiency and most schemes perform similarly at the initial stage, we only consider x ranging from 30 to 90. As can be seen, the RandomID scheme can barely get 30 percent of IDs during the entire simulation. The EnhancedID and EnhancedIDFeedback schemes can roster 60 and 80 percent of the IDs, respectively, but with a longer delay compared with the proposed scheme.

Fig. 10 presents a closer look of the experimental results by illustrating the delays of individual nodes. The proposed scheme achieves the lowest delay for nearly all mobile nodes except Node 24. After analyzing the trace, we found that two IDs besides ID 24 were available at the IR for transmission. However, the packet candidates that contains ID 24 all have less value at that time, and thus, its transmission was delayed. There are also several IDs (e.g., 14, 20, 23, 29, and 38) that experience the same delay under all rostering schemes. The top of the figure shows the missed IDs. As can be seen, none of the schemes yield a complete roster. In particular, Node 8 is missed under all approaches. It was a very inactive node. It was detected only once by IR 2 at a late stage of the experiment. Therefore, there was not enough time to deliver it to the GR. We expect that it would be added to the roster if the experiment was extended for a few more hours.

Fig. 9 shows the impact of IR failures on the performance of our proposed rostering algorithm. As can be seen, when IR1 fails, the number of successfully rostered IDs drops

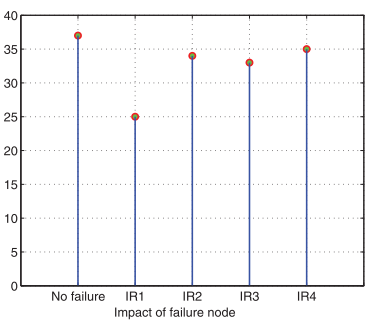


Fig. 9. Impact of IR failures.

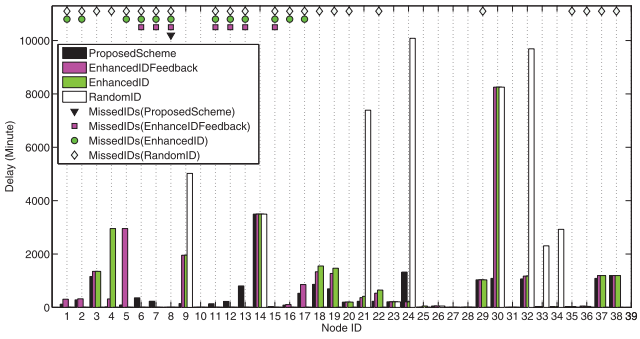


Fig. 10. Delay distribution.

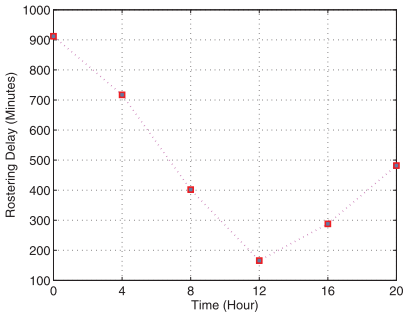


Fig. 12. Hourly delay on Wednesday.

dramatically (by about 1/3). This is because IR1 was installed at the front door of a large class room. The absence of IR1 results in much inefficient ID detection. On the other hand, the loss of IR2, IR3, or IR4 has only marginal impact because IR3 and IR4 acts as backup for each other in the network, and the unique visits to IR2 are rare.

To further study the impact of nodal mobility on rostering, we have designed an experiment by limiting the command to only roster IDs for the past two days. In other words, a roster command only intends to collect IDs detected during the previous 48 hours. Fig. 11 shows the average rostering delay varies on each day of a week. The results closely match the activity pattern of students and faculty. From Monday through Thursday, more nodes are present in our experimental field, and thus, the rostering delay is low. The rostering commands initiated at Friday afternoon experience a longer delay as the tag activity begins to decrease. The rostering commands generated on Saturday and Sundays exhibit very long delay since no much data can be delivered during the weekends, and the longest delay is found for such commands issued on Saturday because they have to wait for about two full days before being served. Fig. 12 further zooms in to show the delay of rostering during a 24 hour period. The first Wednesday of our experiment is chosen as an example, while similar results are observed on other weekdays as well. The rostering delay is low during daytime and high at night, which again shows the performance of rostering depends on the activity of mobile nodes that carry tags.

The data rostering delay is also location dependent. Fig. 13 shows the average delay of the rostering served by different readers. Since IR3 and IR4 are installed on the third floor with high connectivity to the GR (i.e., Reader 5), it replies to rostering commands quickly, resulting in the lowest delay. Deployed on the first floor and second floor, IR1 and IR2 are farther away from the GR and naturally experience longer rostering delay.

4 Simulation Results

Besides the experiments discussed above, extensive simulations are indispensable for a comprehensive evaluation of rostering in FINDERS with a large number of readers and tags, which are not practical to build in labs.

We simulate a network of 5×5 cells. The mobile nodes move according to power-law distribution, which is deemed as one of the most realistic mobility models for delay-tolerant mobile networks [34]. More specifically, each mobile node has a home cell, which is randomly assigned in our simulations. Node i makes its decision to stay in the current cell or move to one of the neighboring cells in every time slot. For example, if it is currently in Cell 0, it may move into one of four adjacent cells (i.e., Cells 1-4) or stay in Cell 0 in the next time slot. Its probability to be in Cell x is $P_i(x \mid 0) = P_i(x)/\sum_{z=0}^{4} P_i(z)$, where $x = 0, 1, 2, 3, 4$ and $P_i(x) = k_i(\frac{1}{d_i(x)})^{\beta}$. k_i is a constant, and β is the exponent of the power-law distribution, respectively. We set $\beta = 1.2$ in our simulations. $d_i(x)$ denotes the distance from Cell x to the home cell of Node i.

By default, 125 nodes are randomly distributed in the field, each with a capacity of 128 bits. There are one GR and five IRs in the system, with a scanning frequency of 1 Hz. Other parameters are similar to our experiment configuration. We focus on the delay for rostering 90 percent mobile nodes, i.e., the interval from the time when the GRs issue a rostering command to the time when they receive the IDs of 90 percent of mobile nodes. We vary several key parameters to observe their impact. All results show the average of 10 simulation runs and corresponding confidence intervals.

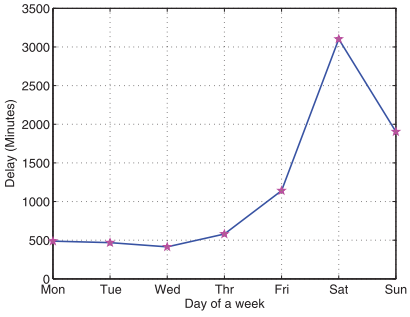


Fig. 11. Delay variation during a week.

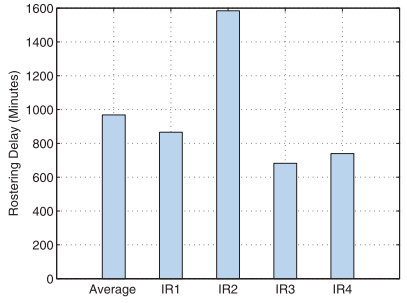
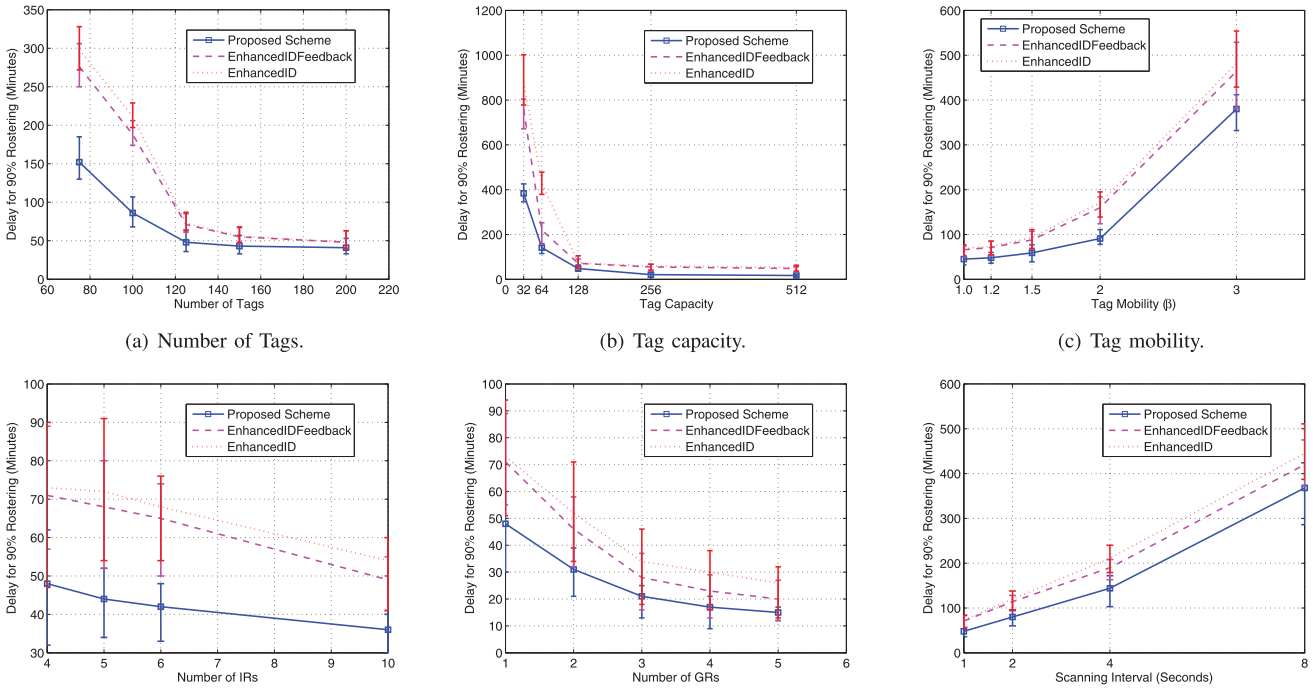


Fig. 13. Average rostering delay at IRs.



(a) Number of Tags. (b) Tag capacity. (c) Tag mobility.

(d) Number of IRs. (e) Number of GRs. (f) Scanning Interval.

Fig. 14. Simulation results.

Since the delay under "RandomID" is much higher than other schemes, its results are omitted here.

As illustrated in Fig. 14a, the delay decreases with the increase of the number of tags, because more tags result in more communication opportunities and accordingly faster delivery of data packets. However, the gain becomes smaller when the tag density is already high, since additional tags will carry unnecessarily duplicated data and thus do not further improve rostering performance. Our proposed scheme performs the best, followed by the IDEnhancedFeedback and IDEnhanced. The IDEnhancedFeedback benefits by the feedback packets that reduce last-hop redundancy and thus achieves better performance than IDEnhanced.

Fig. 14b shows that the performance of rostering generally improves by increasing the tag capacity, because more data can be written into and read from a tag. But note that when the tag capacity increases beyond 256 bits, the improvement becomes negligible, due to the saturation of the network. On the other hand, if the tag capacity is reduced to lower than 64 bits, the performance drops dramatically, especially for the IDEnhancedFeedback and IDEnhanced schemes. This is understandable because they are less efficient in term of IPB and thus consumes more capacity to deliver rostering data.

The power-law factor β determines tags' mobility. With a larger β, a tag stays closer to its home cell, i.e., with a lower probability to move to other (especially remote) cells. Since the communication in FINDERS largely depends on the mobility of tags, lower mobility leads to lower network capacity and accordingly longer rostering delay (see Fig. 14c). The tag mobility has similar impact on all schemes.

Figs. 14d and 14e illustrate the results by increasing IRs and GRs, respectively. Clearly, more IRs can capture more meeting events and accordingly improve the performance rostering. With more GRs, they together promote the chance to directly detect the tags and make it easier to collect data from IRs, thus achieving a lower rostering delay. We have also studied the impact of readers' scanning frequency (see Fig. 14f). The lower the scanning frequency (i.e., the larger the scanning interval), the less the meeting events, which leads to fewer tags to be detected and lower communication capacity. As a result, a longer average rostering delay is observed.

Besides above results for rostering 90 percent nodes, we have also done simulations to roster other desired percent of nodes and observed similar performance trend. With a lower target percentage, the rostering delay is shorter as one may expect. The proposed scheme shows consistent performance gains over other schemes.

5 Conclusion

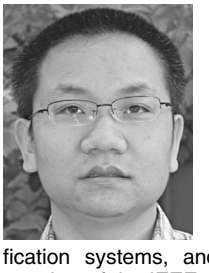
We have studied the problem of rostering in intermittently connected passive RFID networks, aiming to report a list of tagged mobile nodes that appear in given interested area(s) and time interval(s). We have proposed a rostering algorithm based on several communication and computing techniques tailored specifically to address the challenges due to sporadic wireless links, asymmetric communication, intermittent computation, and extremely small memory of tags in such unique networks. The proposed algorithm employs a dynamic space-efficient coding scheme to construct hypothetic packet candidates, appraises their values according to information redundancy and tag mobility, and establishes a 0-1 Knapsack model to choose the best set of packets, which together maximize their total (redundancy-excluded) value but do not exceed the capacity of a tag. We have carried out experiments that involve 38 volunteers for nine days and performed large-scale simulations to evaluate the proposed rostering scheme.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation under grant CNS-0831823. Part of this work was presented at the Ninth Annual IEEE International Conference on Pervasive Computing and Communication (PerCom), Seattle, 21-25 March 2011.

REFERENCES

[1] http://www.princeton.edu/mrm/zebranet.html, 2013.
[2] T. Small and Z.J. Haas, "The Shared Wireless Infostation Model—A New Ad Hoc Networking Paradigm (or Where There Is a Whale, There Is a Way)," *Proc. ACM MobiHoc,* pp. 233-244, 2003.
[3] http://www.wu.ece.ufl.edu/projects/DeerNet/DeerNet.html, 2013.
[4] V. Dyo, S.A. Ellwood, D.W. Macdonald, A. Markham, C. Mascolo, B. Pasztor, S. Scellato, N. Trigoni, R. Wohlers, and K. Yousef, "Evolution and Sustainability of a Wildlife Monitoring Sensor Network," *Proc. ACM Conf. Embedded Networked Sensor Systems,* 2010.
[5] M. Wikelski, R.W. Kays, N.J. Kasdin, K. Thorup, J.A. Smith, and G.W. Swenson, "Going Wild: What a Global Small-Animal Tracking System Could Do for Experimental Biologists," *J. Experimental Biology,* vol. 210, pp. 181-186, 2007.
[6] Z. Yang and H. Wu, "Featherlight Information Network with Delay-Endurable RFID Support (FINDERS)," *Proc. Sixth Ann. IEEE Comm. Soc. Conf. Sensor, Mesh and Ad Hoc Comm. and Networks (SECON '09),* pp. 55-63, 2009.
[7] Z. Yang and H. Wu, "FINDERS: A Featherlight Information Network with Delay-Endurable RFID Support," *IEEE/ACM Trans. Networking,* vol. 19, no. 4, pp. 961-974, Aug. 2011.
[8] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay Tolerant Network Architecture," IETF RFC 4838, 2004.
[9] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-Based Aggregation in Large Dynamic Networks," *ACM Trans. Computer Systems,* vol. 23, pp. 219-252, 2005.
[10] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast Computation by Population Protocols with a Leader," *Proc. 20th Int'l Symp. Distributed Computing,* pp. 61-75, 2006.
[11] J. Aspnes and E. Ruppert, "An Introduction to Population Protocols," *Bull. of the European Assoc. for Theoretical Computer Science,* vol. 93, pp. 98-117, 2007.
[12] B.D. Walker, J.K. Glenn, and T.C. Clancy, "Analysis of Simple Counting Protocols for Delay-Tolerant Networks," *Proc. ACM Workshop Challenged Networks (CHANTS),* pp. 19-26, 2007.
[13] M. Kodialam and T. Nandagopal, "Fast and Reliable Estimation Schemes in RFID Systems," *Proc. ACM MobiHoc,* pp. 322-333, 2006.
[14] C. Qian, H.-L. Ngan, and Y. Liu, "Cardinality Estimation for Large-Scale RFID Systems," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '08),* 2008.
[15] T. Li, S. Wu, S. Chen, and M. Yang, "Energy Efficient Algorithms for the RFID Estimation Problem," *Proc. IEEE INFOCOM,* pp. 1019-1027, 2010.
[16] M.S. Kodialam, T. Nandagopal, and W.C. Lau, "Anonymous Tracking Using RFID Tags," *Proc. IEEE INFOCOM,* pp. 1217-1225, 2007.
[17] J. Myung and W. Lee, "Adaptive Splitting Protocols for RFID Tag Collision Arbitration," *Proc. ACM MobiHoc,* pp. 202-213, 2006.
[18] T. Li, S. Chen, and Y. Ling, "Identifying the Missing Tags in a Large RFID System," *Proc. ACM MobiHoc,* pp. 1-10, 2010.
[19] H. Vogt, "Efficient Object Identification with Passive RFID Tags," *Proc. Int'l Conf. Pervasive Computing,* 2002.
[20] V. Namboodiri and L. Gao, "Energy-Aware Tag Anti-Collision Protocols for RFID Systems," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '07),* pp. 23-36, 2007.
[21] L. Xie, B. Sheng, C.C. Tan, H. Han, Q. Li, and D. Chen, "Efficient Tag Identification in Mobile RFID Systems," *Proc. IEEE INFOCOM,* pp. 1001-1009, 2010.
[22] W. Luo, S. Chen, T. Li, and S. Chen, "Efficient Missing Tag Detection in RFID Systems," *Proc. IEEE INFOCOM,* pp. 356-360, 2011.
[23] C.C. Tan, B. Sheng, and Q. Li, "How to Monitor for Missing RFID Tags," *Proc. Int'l Conf. Distributed Computing Systems (ICDCS),* 2008.
[24] L.M. Ni, Y. Liu, Y.C. Lau, and A.P. Patil, "LANDMARC: Indoor Location Sensing Using Active RFID," *Proc. IEEE Conf. Pervasive Computing and Comm. (PERCOM '03),* pp. 407-415, 2003.
[25] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and Localization with RFID Technology," *Proc. IEEE Int'l Conf. Robotics and Automation,* pp. 1015-1020, 2004.
[26] K. Yamano, K. Tanaka, M. Hirayama, E. Kondo, Y. Kimuro, and M. Matsumoto, "Self-Localization of Mobile Robots with RFID System by Using Support Vector Machine," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems,* pp. 3756-3761, 2004.
[27] Y. Zhao, Y. Liu, and L.M. Ni, "VIRE: Active RFID-Based Localization Using Virtual Reference Elimination," *Proc. Int'l Conf. Parallel Processing (ICPP '07),* p. 56, 2007.
[28] C. Wang, H. Wu, and N.-F. Tzeng, "RFID-Based 3-D Positioning Schemes," *Proc. IEEE INFOCOM,* pp. 1235-1243, 2007.
[29] D. Henrici and P. Müller, "Providing Security and Privacy in RFID Systems Using Triggered Hash Chains," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '08),* pp. 50-59, 2008.
[30] T. Dimitriou, "A Secure and Efficient RFID Protocol That Could Make Big Brother (Partially) Obsolete," *Proc. IEEE Int'l Conf. Pervasive Computing and Comm. (PERCOM '06),* pp. 269-275, 2006.
[31] L. Yang, J. Han, Y. Qi, C. Wang, T. Gu, and Y. Liu, "Season: Shelving Interference and Joint Identification in Large-Scale RFID Systems," *Proc. IEEE INFOCOM,* pp. 3092-3100, 2011.
[32] L. Yang, J. Han, Y. Qi, and Y. Liu, "Identification-Free Batch Authentication for RFID Tags," *Proc. IEEE Int'l Conf. Network Protocols (ICNP '10),* pp. 154-163, 2010.
[33] *Introduction to Algorithms,* pp. 382-283. MIT, 2001.
[34] J. Leguay, T. Friedman, and V. Conan, "DTN Routing in a Mobility Pattern Space," *Proc. ACM Special Interest Group on Data Comm. (SIGCOMM '05),* pp. 276-283, 2005.



Zhipeng Yang received the BS and MS degrees in computer science from Tianjin University, China, in 2001 and 2004, respectively. He has been working toward the PhD degree in computer science at the Center for Advanced Computer Studies, University of Louisiana, Lafayette, since 2007. From 2004 to 2006, he was a software engineer in Nortel and Lucent China. His current research interests include delay-tolerant networks, radio frequency identification systems, and wireless sensor networks. He is a student member of the IEEE.

Ting Ning received the BS and MS degrees in computer science from Lanzhou University, China, in 2005 and 2008, respectively. She has been working toward the PhD degree in computer science at the Center for Advanced Computer Studies, University of Louisiana, Lafayette, since 2008. Her current research interests include delay-tolerant networks, the application of game theory, and radio frequency identification systems. She is a student member of the IEEE.

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