Mobile Node Rostering in Intermittently Connected Passive RFID Networks

Zhipeng Yang and Hongyi Wu
The Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, LA 70504
Email: {zxy1767,wu}@cacs.louisiana.edu

***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 9 days and perform large-scale simulations to evaluate the proposed rostering scheme.**

I. Introduction

It has been discussed extensively in literatures to apply wireless sensor network technologies for wildlife research [1]. However, earlier investigation has revealed that about 81% of bird species and 67% 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% of the weight of the animal otherwise such overweight additions often lead to high mortality rates of the animals being studied [2]. 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.

This work is supported in part by the National Science Foundation under grant CNS-0831823.

A. An Overview of FINDERS

Built upon Radio Frequency IDentification (RFID) technologies, the Featherlight Information Network with Delay-Endurable RFID Support (FINDERS) has been introduced in [3] to address the weight constraints discussed above. It exploits passive RFID tags that are ultra light, 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). 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. 5 and Sec. II-A).

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 or Zigbee) 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). 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

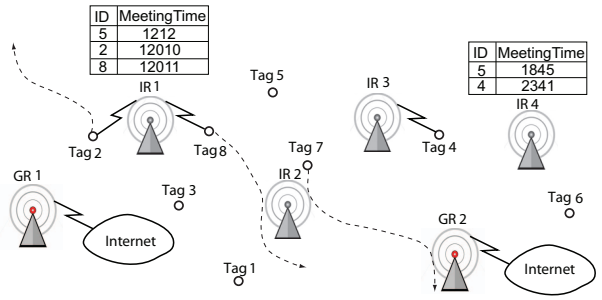


Fig. 1. An overview of FINDERS.

thus form an intermittently connected delay-tolerant network (DTN [4]) 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.

FINDERS is a unique pervasive system. It consists of two very different types of nodes: powerful static readers and extremely resource-constrained mobile tags. The communication in FINDERS can be established between a tag and a reader only, but not tags to tags or readers to readers. The communication must be initiated by a reader, in contrast to symmetric transmissions in most conventional networks. Due to the short communication range and dynamic tag mobility, the connectivity of FINDERS is very low and intermittent, forming a sparse network where a tag is connected to a reader only occasionally. Besides connectivity, the computation at the tag is also intermittent. It is available when the tag is powered up by a nearby reader, and once charged, the tag can hold its energy for up to about 5 seconds only [5]. Thus, such continuous functions necessary to many protocols as timers cannot be implemented here. In addition, the storage space of tags (that are the main vehicle for data transportation) is so limited that it becomes the critical network resource and communication bottleneck.

B. Rostering of Mobile Nodes in FINDERS

We have introduced the architecture of FINDERS in [3]. This paper focuses on a problem called rostering 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 simple counting where only a headcount is desired [6]–[9].

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. 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 at anytime. However, this naive approach is impractical, due to continuous updates on meeting events and extremely tight constraints on communication capacity in FINDERS. 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 support efficient rostering, a reader must fully exploit the capacity of tags whenever they are available. Since a tag's capacity is fixed, the challenge is how to fit as much useful information as possible onto the tag. The reader has many possible ways to pack its data, creating various hypotheses of packets, with different amount of valuable information and different packet lengths. A distributed algorithm is needed to determine the set of packets to be written into the tag, in order to maximize its effective information per bit (i.e., the entropy) carried. 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 [3].

To this end, we propose a rostering algorithm based on several communication and computing techniques specifically tailed 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 9 days and performed large-scale simulations to evaluate the proposed rostering scheme.

The rest of the paper is organized as follows. Sec. II introduces the problem and challenges. Sec. III describes our proposed rostering algorithm. Sec. IV elaborates implementation issues and testbed experiments. Sec. V presents simulation results. Finally, Sec. VI concludes the paper.

II. Problem Description and Challenges

We introduce the problem of mobile node rostering in this section, including an overview of system configuration, a description of the problem from a distributed database perspective, and the unique challenges for achieving efficient rostering in intermittently connected passive RFID networks.

A. System Configuration

As introduced in Sec. I, FINDERS is pervasive system that consists of fixed readers and mobile tags. The former are powerful devices, with sufficient storage and battery capacity. For example, the Alien's ALR-9900 reader employed in our experiments possesses of 64 MB RAM and 64 MB flash

memory. The trial data show that a typical car battery of $12V \times 60Ah$ 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 $8.15 \times 94.8 \times 0.05$mm and weights less than one 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 Sec. IV 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 20 ft.

A reader and a 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 six 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.

When a tag meets a reader, it not only generates a meeting event, but also provides a communication opportunity. The reader reads the data currently on the tag and writes appropriate information into it for transmission to another reader.

B. Problem Description

Rostering is a pervasive computing problem, aiming to enable a user to access the distributed *local meeting tables* and aggregate their meeting events to report a list of the mobile nodes that appear in given area(s) and time interval(s). Clearly rostering gathers more information than simple counting where only a headcount is desired [6]–[9].

A straightforward scheme is to transmit all local meeting tables to the user or a server that is fully accessible by users. However, this naive approach is impractical, due to continuous updates on local meeting tables 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 tables must be kept at individual readers in a distributed manner. A user's 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 on-going 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, e.g., 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 (DML), data definition language (DDL), or data control language (DCL) in database systems.

C. 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 introduced in Sec. I, the tags serve as transportation vehicles, carrying data packets from one reader to another. Since the mobility of tags are 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

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	Acked	FB Time	Reply Counter	FB Counter
1	2102	Y	4233	5	3
3	3222	N	—	2	0
10	4258	N	—	1	0

(b) Roster1 (for Command1)

Fig. 2. Tables maintaining by a reader.

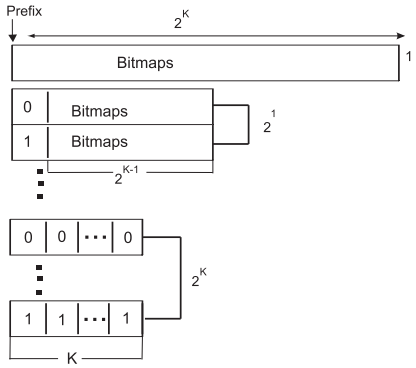


Fig. 3. Hypothetic packet candidates.

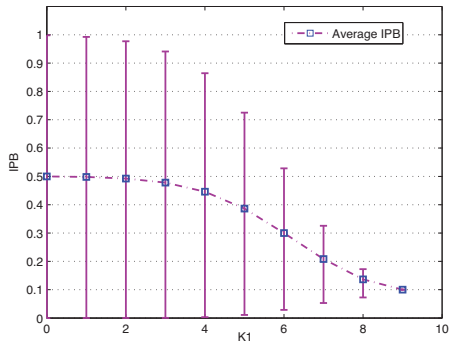


Fig. 4. IPB under different K_1 (with K=9).

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, in order to maximize its effective information per bit transmitted (i.e., the entropy). 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 in order to develop an effective algorithm for mobile node rostering.

III. 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.

A. Overview of the Rostering Algorithm

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 Sec. II-B, 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 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 Sec. III-B.
- *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 Sec. II-A. In addition, it keeps a command table that includes all active rostering commands it has received (see Fig. 2(a)) and the rosters it has learnt so far for each command (as illustrated in Fig. 2(b)). 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.

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

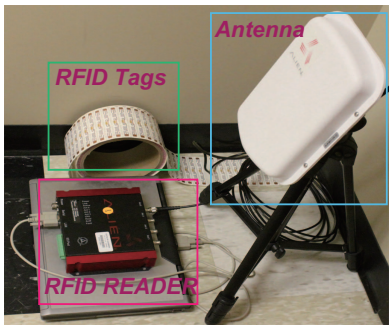


Fig. 5. Our Alien passive RFID gear.

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 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 next subsections. 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 Sec. II-C, 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.

B. Packet Format and Hypothetic Packet 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 two 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 Sec. III-A. On the other hand, the *Data* fields of *Reply* and *Feedback* packets are worth further elaboration, since 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 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 four 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 correspond to the last bit of the mobile node ID. If a is set to 1, it indicates 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 alway 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 *information per bit (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 $K_1 = 0$, 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 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 learnt 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.

C. 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:

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

where n is the total number of hypothetic packet candidates and $\{x_i | 1 \leq i \leq n\}$ 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.

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 j^{th} ID.

As discussed in Sec. III-A, 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 tracking 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. 2(b)), 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. 2(b)). Note that since there is no separate timestamp for each ID, t_j^R and t_j^F are estimated by the packet-level timestamp and thus usually larger than the true value.

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.

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 the tag's mobility. If the tag moves away from the GRs, it won't be much effective for transmitting *Reply* packets. Instead, it will be more efficient to write *Command* or *Feedback* packets to the tag 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" GR's. Let ξ_i denote the EDC of Node i. The EDC of a GR is always 1 and the initial EDC of IRs and tags is zero. If Node i is not a GR and meets Node j with $\xi_j > \xi_i$ at time t, ξ_i is updated to $(1-\eta_3)^{\lfloor \frac{t-t_o}{\Delta} \rfloor}\xi_i + \eta_3 \xi_j$, where t_o is the time when ξ_i was updated last time.

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:

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

where β_1 and β_2 are constants to shape the adjustment.

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 Eq. (1). The 0-1 knapsack problem is NP-complete. A dynamic programming solution that runs in pseudo-polynomial time is adopted here [10]. But 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, $\{x_i | 1 \leq i \leq n\}$, 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.

IV. 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 gears supplied by Alien Technologies. We have acquired five sets of Alien passive Class1Gen2 RFID systems with five ALR-9900 readers and 1000 ALN-9540-WR Squiggle^{TM} tags (see Fig. 5 for a photo of the reader and tags). The readers are programmed by using the vender's Applications Development Kit (ADK). In this section, we first discuss the implementation issues and then introduce our experiments and results.

A. Implementation Issues

The implementation of our proposed rostering algorithm is largely straightforward by following the description in Sec. III. 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 four 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.

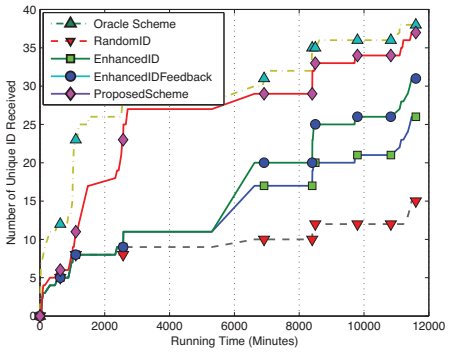


Fig. 6. Comparison of rostering efficiency.

B. 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.

Thirty eight volunteers had participated in our experiments, including faculty members, senior Ph.D. students (who do not have classes), graduate students at M.S. level (who go to classrooms regularly), and undergraduate students. Our experiment lasted 9 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.

C. 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 in order to inform IRs that they need no longer transmit such IDs. "ProposedScheme" is the proposed scheme. "Oracle Scheme" assumes 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. 6 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 2500 to 5500 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. 7 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.

V. 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 [11]. More specifically, each mobile node has

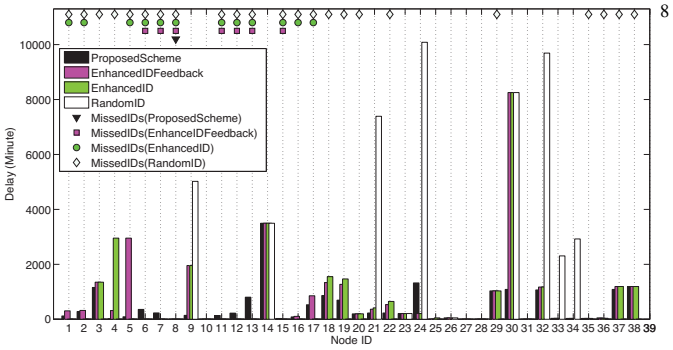


Fig. 7. Delay distribution.

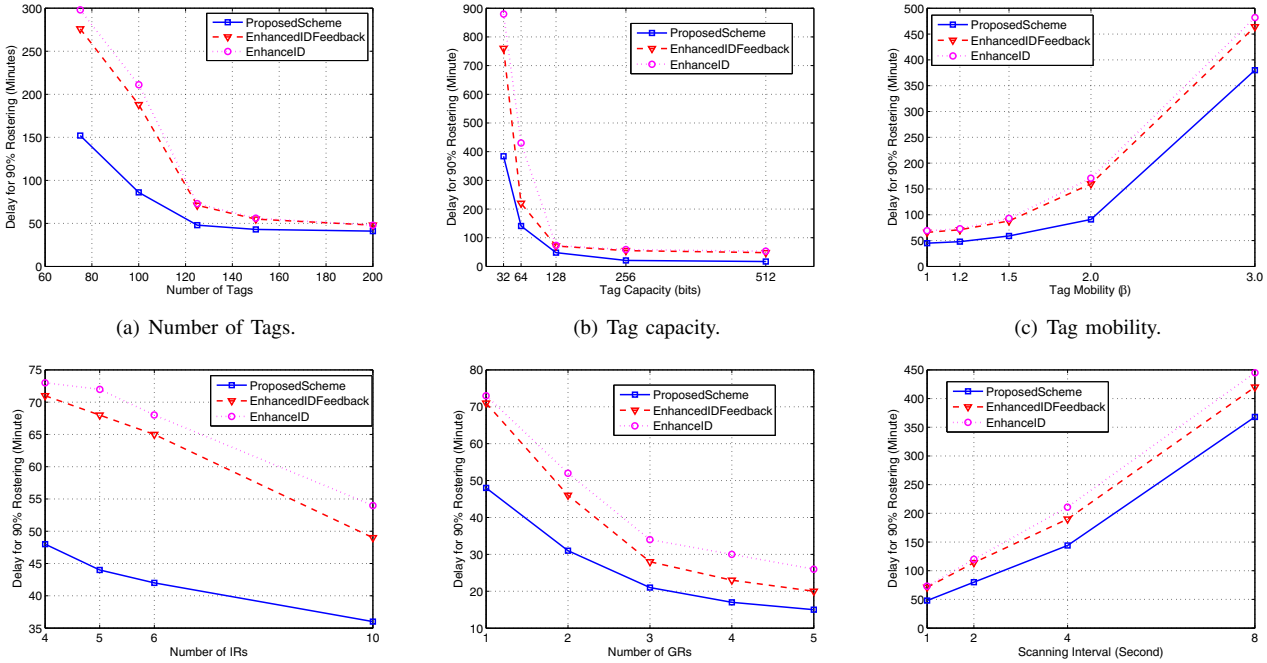
a home cell, which is randomly assigned in our simulations. Node i makes its decision to stay in the current cell or move to one of the neighboring cells in every time slot. For example, if it is currently in Cell 0, it may move into one of four adjacent cells (i.e., Cells 1-4) or stay in Cell 0 in the next time slot. Its probability to be in Cell x is $P_i(x|0) = P_i(x)/\sum_{z=0}^{4} P_i(z)$, 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 β=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 5 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% 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% of mobile nodes. We vary several key parameters to observe their impact. All results are the average of 10 simulation runs. Since the delay under "RandomID" is much higher than other schemes, its results are omitted here.

As illustrated in Fig. 8(a), 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. 8(b) 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) Number of Tags. (b) Tag capacity. (c) Tag mobility.

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

Fig. 8. Simulation results.

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. 8(c)). The tag mobility has similar impact on all schemes.

Figs. 8(d) and 8(e) 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. 8(f)). 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.

VI. 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 9 days and performed large-scale simulations to evaluate the proposed rostering scheme.

References

[1] 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," in *Proc. of ACM Sensys*, 2010.

[2] 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," *The Journal of Experimental Biology*, pp. 181–186, 2007.

[3] Z. Yang and H. Wu, "Featherlight Information Network with Delay-Endurable RFID Support (FINDERS)," in *Proc. of IEEE SECON*, pp. 55–63, 2009.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Delay Tolerant Network Architecture." draft-irtf-dtnrg-arch-02.txt, 2004.

[5] UHF Class 1 Gen 2 Standard v. 1.1.0.

[6] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based Aggregation in Large Dynamic Networks," in *Proc. of ACM Transactions on Computer Systems*, vol. 23, pp. 219–252, 2005.

[7] D. Angluin, J. Aspnes, and D. Eisenstat, "Fast Computation by Population Protocols with A Leader," in *Proc. of The 20th International Symposium on Distributed Computing*, pp. 61–75, 2006.

[8] J. Aspnes and E. Ruppert, "An Introduction to Population Protocols," *Bulletin of the European Association for Theoretical Computer Science*, vol. 93, pp. 98–117, 2007.

[9] B. D. Walker, J. K. Glenn, and T. C. Clancy, "Analysis of Simple Counting Protocols for Delay-Tolerant Networks," in *Proc. of ACM Workshop on Challenged Networks (CHANTS)*, pp. 19–26, 2007.

[10] *Introduction to Algorithms*, pp. 382–283. The MIT Press, 2001.

[11] J. Leguay, T. Friedman, and V. Conan, "DTN Routing in a Mobility Pattern Space," in *Proc. of ACM SIGCOMM'05 Workshop on Delay Tolerant Networking and Related Topics*, pp. 276 – 283, 2005.