A framework for fast RFID tag reading in static and mobile environments

V. Sarangan\textsuperscript{a,}\textsuperscript{☆}, M.R. Devarapalli\textsuperscript{a}, S. Radhakrishnan\textsuperscript{b}

\textsuperscript{a} Computer Science Department, Oklahoma State University, 212 Mathematical Sciences, Stillwater, Oklahoma 74078, United States
\textsuperscript{b} School of Computer Science, University of Oklahoma, Norman, Oklahoma 73019, United States

Received 14 September 2007; received in revised form 7 December 2007; accepted 18 December 2007
Available online 2 January 2008

Responsible Editor: J. Misic

Abstract

A framework for reducing the average reading time of passive RFID tags in dense environments is introduced. The proposed framework termed Accelerated Frame Slotted ALOHA (AFSA) can be used in conjunction with almost all RFID tag reading protocols that are based on frame slotted ALOHA. It is shown that AFSA reduces the tag reading time by avoiding the wastage in bit times due to collisions and idle slots. The implementation of AFSA in conjunction with two different ALOHA protocols – one with unlimited frame sizes and the other with limited frame sizes is discussed. For both these protocols, extensions of AFSA to read passive tags in a mobile setting are described. Simulation results show that AFSA reduces the average tag reading time by up to 40\% with respect to the stand alone ALOHA protocols under both static and mobile settings.

© 2007 Elsevier B.V. All rights reserved.

Keywords: RFID; Passive tag reading; Mobile environments; Slotted ALOHA

1. Introduction

RFID (Radio Frequency Identification) is an automatic identification scheme considered to be a better replacement of existing barcode technology. RFID possesses the advantages of contact-less identification and the ability to hold more data with respect to barcode technology. RFID uses silicon chips or tags to store electronic product codes (EPC). The tags also have radio transceivers using which they transmit the EPCs to the devices that probe these tags. RFID has applications in plethora of environments such as supply chain management, inventory control, supermarket checkout process, pet identification, and toll ways.

RFID tags can be categorized in to different classes based on their functional characteristics [1]:

[1] A preliminary version of this paper appeared in ACM QShine 2007, Vancouver. This work was supported by the US Federal Highway Administration under Grants SAFETEA-LU 1934 and SAFETEA-LU 1702.

☆ Corresponding author. Tel.: +1 405 744 5672; fax: +1 405 744 9097.

E-mail addresses: saranga@cs.okstate.edu (V. Sarangan), mallad@cs.okstate.edu (M.R. Devarapalli), sridhar@ou.edu (S. Radhakrishnan).
Class 0 and Class 1 (Identity Tags), Class 2 (Higher Functionality Tags), Class 3 (Semi-Passive Tags), Class 4 (Active Ad Hoc Tags), and Class 5 (Reader Tags). Out of these, tags belonging to classes 0, 1, and 5 find a wide spread usage. Class 0 and Class 1 tags are those that possess just the bare minimum functionality expected from a RFID tag, namely store and transmit functionality. Class 5 tags are those that have the ability to read the data stored in other classes of tags and are commonly referred to as readers.

Class 0 and Class 1 tags do not have any power source on their own and are usually powered by the readers through radio waves during the reading process. It is envisioned that the cost of these tags will come down to $0.05/unit [2] over the next several years and the cost effectiveness of these tags will enable them to be used almost ubiquitously, thereby creating dense RFID environments. In order to completely tap into the potential benefits of the RFID technology, it is easy to infer that tag reading mechanisms which can read large number of tags efficiently within a short time are indispensable.

To achieve this goal of efficient tag reading, various tag reading protocols or RFID MAC protocols have been proposed in the literature. They are classified into two categories: deterministic and probabilistic [3]. Deterministic protocols are based on binary trees where each root-to-leaf path represents a unique tag id. Several of these protocols have been proposed for Class 0 Generation 1 tags [4–7]. Probabilistic protocols are based on the slotted ALOHA framework, where the channel time is split into frames. A single frame in turn is divided into several time slots. During a frame, each tag randomly chooses a time slot and transmits its identifier to the reader in that slot. These protocols have been proposed for Class 1 Generation 1 tags. The focus of this paper is on probabilistic protocols since deterministic protocols have a relatively long identification delay [8–11]. Probabilistic tag reading protocols can achieve smaller identification delays provided the amount of time wasted due to collisions and idle time slots is reduced [12]. Collisions occur when more than one tag transmits in the same slot, in which case all their identifiers are lost. Idle time slots occur when none of the tags in the reader’s vicinity chose a particular slot for transmitting their identifier.

In this paper, we propose a framework called Accelerated Frame Slotted ALOHA (AFSA) which reduces the duration of bit times wasted due to collisions and idle slots in probabilistic tag reading protocols. AFSA can be used in conjunction with any of the ALOHA based protocols to reduce the tag reading time. In this paper, we will explain the design and working of AFSA by taking the case of two different ALOHA protocols – Dynamic Framed Slotted ALOHA (DFSA) [13], a protocol with unlimited frame sizes and Extended Dynamic Framed Slotted ALOHA (EDFSA) [14], a protocol with limited frame sizes. We also discuss extensions of AFSA, with respect to both DFSA and EDFSA, to read tags in a mobile setting wherein the tags lie on a conveyor belt and pass through the readers fixed above the conveyor. Simulation results show that AFSA can reduce the average tag reading time by up to 40% with respect to the stand alone ALOHA protocols under both static and mobile settings.

The remainder of this paper is organized as follows. Section 2 discusses the research related to the proposed work. Section 3 presents an overview of the two ALOHA protocols in conjunction with which the proposed framework is described. Section 4 introduces the proposed AFSA framework and Section 5 discusses how AFSA can be optimized in order to minimize the overall tag reading time. Section 6 describes the extensions for AFSA to read tags in mobile environments. Section 7 discusses the results obtained using AFSA under static and mobile settings. Section 8 concludes the paper and also outlines possible future directions for research.

2. Related work

Almost all of the probabilistic tag reading protocols are based on slotted ALOHA. In these protocols, each tag reading round comprises of three phases. The first phase is the advertisement phase, where the reader broadcasts the frame size to the tags. Frame size refers to the number of time slots available in a frame. The second phase is the transmission phase, during which each tag randomly chooses a time slot within the frame and transmits its EPC. If more than one tag chooses the same time slot, their transmissions collide and the slot will ultimately be wasted. A slot is also wasted when none of the tags in the reader’s vicinity choose a particular slot for transmitting their identifier.

In this paper, we propose a framework called Accelerated Frame Slotted ALOHA (AFSA) which reduces the duration of bit times wasted due to coll...
informs a tag if its transmission was successful or not. While the aforesaid working is common for all the probabilistic protocols, they differ in terms of the way in which they choose the frame size and the way in which they choose the tags that respond in a given round.

The basic frame slotted ALOHA (BFSA) [15] is the simplest of all the probabilistic protocols. It uses a fixed frame size for all the rounds, irrespective of the number of tags in the reader’s vicinity. Consequently, the system efficiency\(^1\) drops significantly in case of both large and small tag counts. Under large tag counts, lot of collisions occur while under small tag counts, many slots go unused. Dynamic Frame Slotted ALOHA (DFSA) overcomes this problem by dynamically changing the frame size as per the tag count. It gathers and uses information such as number of successful slots, idle slots, and collision slots from the previous round to determine the appropriate frame size for the next round.

DFSA has many variants, with each differing in terms of how they adjust the frame size. The first variant changes the frame size as per the number of collisions observed. If this number is greater than a given threshold, the reader increases the frame size. If the number of collisions is less than another threshold, the reader decreases the frame size [15]. The second variant uses an initial frame size to start the reading process. If none of the tags are read in the first round (which can happen if all the slots are wasted in collisions), it then doubles the frame size. It continues to double the frame size until at least one tag is successfully read. As soon as a tag is read, it starts the following round with the initial minimum frame size [15].

Cha and Kim [13] propose another variant of DFSA which reduces the overall tag reading time by choosing a frame size that maximizes the probability of a slot being successful. In this variant, before each round, the number of unread tags in the system is first estimated by the reader. Using this estimated tag count, the frame size that maximizes the probability of a slot being successful is determined. This frame size is then used in the following round. ASAP [16] is another protocol which similar to [13] in principle. However, ASAP uses slots of varying lengths and also a new tag estimation procedure. In ASAP, if a transmission slot is perceived to be idle, it is prematurely ended thus saving some time. The number of unread tags in the system is estimated using a expectation estimator and the frame size that maximizes the temporal efficiency for this tag count is then chosen. Temporal efficiency in ASAP is defined as the ratio of time actually spent in reading the tags over the wasted time.

Tree slotted ALOHA (TSA) [17] is a novel tag reading protocol which is a hybrid between slotted ALOHA and binary tree based tag reading techniques. The basic idea of TSA is to resolve a collision as soon as it happens. In Framed Slotted ALOHA protocols, two tags not colliding with each other in a given frame, can collide in the next frame. TSA avoids this situation by making sure that when a collision occurs in a slot, only those tags that generated the collision in that slot are queried in the next read cycle. This results in a better performance.

While the above DFSA schemes perform well, they assume that the frame size can be increased indefinitely. However, in practice, it may not be possible to do so [18]. In addition, large frame sizes can increase the interference between readers in a multi-reader environment. Consequently, schemes that can minimize the reading time with limited frame sizes are required. Enhanced Dynamic Frame Slotted ALOHA (EDFSA) [14] is one such scheme that guarantees a high tag reading rate even with a limited frame size. In EDFSA, under the assumption of equal slot durations, the frame size and the number of tags participating in a given round are chosen so as to maximize the probability of a slot being successful.

The framework proposed in this paper – Accelerated Frame Slotted ALOHA (AFSA), decreases the amount of time wasted in ALOHA based tag reading protocols due to collisions and idle slots by employing bitmaps. AFSA thus brings down the overall time taken to read a tag. While AFSA can be used in conjunction with any of the ALOHA based protocols to reduce the tag reading time, we explain the design and working of AFSA by taking the case of two different ALOHA protocols – the DFSA variant proposed in [13] (which assumes unlimited frame sizes) and Extended Dynamic Framed Slotted ALOHA (EDFSA) [14] (which assumes limited frame sizes). For ease of reference, hence forth, we will refer to the protocol proposed in [13] as DFSA. The following section provides a brief overview of DFSA and EDFSA.

\(^{1}\) System efficiency is defined as the percentage of successful slots in a given frame.
3. Overview of DFSA and EDFSA

3.1. DFSA

The working of DFSA [13] is same as that of most other dynamic frame slotted ALOHA protocols. Each round in DFSA consists of three phases – advertisement, transmission, and acknowledgment. Let $K$ be the actual number of unread tags present in the system. Before initiating the advertisement phase of a round, the reader estimates the value of $K$. An estimate, $\hat{K}$, is obtained by using the number of collisions observed by the reader in the previous round ($Z_c$) and is given by $\hat{K} = 2.3922 \times Z_c$. Once the tag count has been estimated, the frame size ($N$) for the next ensuing round is chosen as $N = \hat{K}$. It can be shown that if $K = \hat{K}$, this choice of frame size indeed maximizes the probability of a slot being successful in the following round. The reader then advertises this seemingly optimal frame size to the tags. The transmission and acknowledgment phases then follow as described in the beginning of Section 2.

3.2. EDFSA

As with DFSA, in EDFSA [14] too, the reader estimates the unread tag population in the beginning of each round. For determining $\hat{K}$, the estimation technique suggested by Vogt in [12] is used. When the estimated tag population is small, the reader sets the frame size ($N$) according to the estimated population count as $N = \hat{K}$. For large estimated tag populations, the reader does not increase the frame size indefinitely as in DFSA. Instead, it restricts the number of tags that can participate in the following round so as to achieve better reading rates even with small frame sizes. After estimating the tag count, EDFSA divides the tag population into various groups. The number of groups $M$ depends on the maximum frame size $N_{\text{max}}$ and $\hat{K}$ as $M = \lceil \hat{K}/N_{\text{max}} \rceil$.

Once the reader determines the $M$ value, it broadcasts this value along with the frame size to all the tags. The tags then generate a random number and carry out a modulo operation of this random number with the advertised group size. Tags with a zero remainder alone participate in that round and the rest of the tags stay muted. In other words, the grouping mechanism transforms a tag population which is too dense for the available frame size into a population which is approximately the optimum for the available frame size. All participating tags then generate a second random number in the range $1$–$N$ to decide the slot number during which they should transmit their data. The transmission and acknowledgment phases then continue as in DFSA. In EDFSA, the frame sizes and the group numbers for different tag counts are estimated such that the probability of a slot being successful is maintained close to the maximum possible value of 36.8%. Assuming a maximum frame size of 256, Table 1 gives a list of the appropriate frame sizes and the number of groups for various unread tag populations [14].

<table>
<thead>
<tr>
<th>Number of unread tags ($K$)</th>
<th>Frame size ($N$)</th>
<th># of Groups ($M$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>708–1416</td>
<td>256</td>
<td>4</td>
</tr>
<tr>
<td>355–707</td>
<td>256</td>
<td>2</td>
</tr>
<tr>
<td>177–354</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>82–176</td>
<td>128</td>
<td>1</td>
</tr>
<tr>
<td>41–81</td>
<td>64</td>
<td>1</td>
</tr>
<tr>
<td>20–40</td>
<td>32</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 1

4. Proposed framework: Accelerated Framed Slotted ALOHA (AFSA)

4.1. System model

Generation 1 RFID systems work either in the 13.56 MHz ISM band or 900 MHz UHF band, while Generation 2 systems can work in the 860–960 MHz UHF band. In this paper, we consider a collision limited 900 MHz UHF RFID system, where the tags communicate with the reader over a shared channel. However, we note that the proposed framework can be adapted for RFID systems that work in the 13.56 MHz or 860–960 MHz bands too by appropriately changing the slot timing values in the design expressions.

In the system considered in this work, the reader uses a 900 MHz continuous wave to power the passive tags. Each tag transmits its EPC (which is...
assumed to be 64 bits) along with a 16 bit CRC (cyclic redundancy check code) with a symbol duration of 4 μs [20]. Tags employ FSK backscatter modulation to transmit their data. Reader uses the symbols ‘0’, ‘1’, and ‘Null’ to communicate with the tags. The symbols ‘0’ and ‘1’ are used for constructing commands while the ‘Null’ symbol is used for marking the beginning of a command, the end of a command, and the end of a slot within a frame. The duration of these symbols is 12.5 μs [20]. Readers communicate with tags in ‘rounds’ whose structure is shown in Fig. 1.

4.2. Description

To begin with, all the tags are in unpowered or passive state. The reader sends the reset, oscillator calibration, and data symbol calibration signals listening to which the tags move to the active state. The reader then advertises the group size and the frame size to the tags. If the advertised group size is 1, all tags move to the select state. If the advertised group size is greater than 1, the tags carry out a modulo operation as described in Section 3.2. Tags that have a remainder zero move to the select state while other tags remain in the active state. Tags in the select state exchange a few bits of information with the reader. After this exchange, some of the tags in the select state transit to the transmit state while the remaining tags go back to the active state. In the transmit state, the tags send their EPC codes to the reader. If a tag’s transmission is successful, it goes to the identified state and if not, it goes back to the active state. The tag state machine under AFSA is shown in Fig. 2.

The proposed AFSA framework has five phases to complete a single round of tag reading. The overall structure of a round of operation under AFSA is shown in Fig. 1 and the individual phases are described below.

1. Advertisement phase: During this phase, the reader broadcasts the frame size ($N$) and the number of groups ($M$) to all the tags that are within its range. The parameters $N$ and $M$ are decided by the underlying ALOHA protocol (it can be either DFSA or EDFSA). In case of DFSA, the protocol assumes the frame size to be unlimited and does not follow any grouping. Therefore, the value of $M$ is set to 1. The value of $N$ value is set to $bK$, the estimated number of unread tags in the reader’s vicinity. In the case of EDFSA, the values for $N$ and $M$ are chosen as per Table 1 by approximating $K$ with $\hat{K}$. In addition to $M$ and $N$, the reader also advertises $n$, which is a parameter introduced by the AFSA framework (the significance of $n$ will become clear shortly$^5$).

Once the values of $N$ and $M$ are known, the tags first determine their eligibility to participate in the ensuing round. If $M = 1$, all tags are allowed to participate and if $M > 1$, each tag determines its eligibility as per the methodology described under Section 3.2. Each eligible tag transits to the select state and generates another random number which is uniformly distributed in the range of 1–$N$. This number identifies the slot that the tag has chosen to transmit its data. The structure of the advertisement phase is shown in Fig. 3.

2. Reservation phase: In this phase, the tags in the select state let the reader know about the slots

---

$^4$ The semantics of this exchange will become clear shortly.

$^5$ We have found that 3 bits are sufficient for the reader to advertise $n$ to the tags.
that they have chosen. Unlike other slotted ALOHA protocols, where the tags send their data right away in the chosen slot, in AFSA the tags transmit an \( n \) bit sequence in their chosen slot, where \( n \) is the number advertised by the reader in the advertisement phase. For a given value of \( n \), there are \( 2^n \) possible \( n \) bit sequences. A tag randomly picks one of these \( 2^n \) sequences and transmits it to the reader in the slot chosen by it. If the reader successfully receives an \( n \) bit sequence in a slot, it perceives that slot to be successfully reserved by some tag for transmitting its data. On the other hand, if the reader receives a garbled signal in a slot it understands that a collision has occurred between two or more tags in that slot. Fig. 4 shows the structure of this phase.

3. Reservation summary phase: During this phase, the reader informs the tags about the status of their slot reservations. The reservation status is advertised through a bitmap of length \( N \). For example, assuming \( N = 4 \), the bitmap 1001 indicates that an \( n \) bit sequence was successfully received by the reader in the slots numbered 1 and 4. This implies that, those tags that had chosen slots 1 and 4 were successful in their reservation. In other words, occurrence of bit 1 in location \( i \) indicates that only one tag had chosen slot \( i \). During slots 2 and 3, the reader did not successfully receive any \( n \) bit sequence. This can indicate one of the two possibilities: (i) collisions – these slots were chosen by more than one tag and when the tags transmitted their randomly chosen \( n \) bit sequences, collision occurred and consequently, the reader was not able to decode the received signal; or (ii) idle – the slots were not chosen by any of the tags. In other words, occurrence of bit 0 in location \( i \) indicate that slot \( i \) has been wasted either due to collisions or idleness. Fig. 5 shows the structure of this phase.

4. Data transmission phase: After the reservation summary phase, all the participating tags will be aware of the status of their reservations. Only those tags that find their reservations to be successful will move to the transmit state and will transmit their data in the data transmission phase. The remaining tags will go back to the active state. We note that if \( S \) is the total number of ‘1’s in the summary bitmap, it is clear that only \( S \) data transmissions are possible, where \( S \leq N \). Hence, it is enough if there are just \( S \) data transmission slots. A tag that finds a ‘1’ in the summary bitmap in the location corresponding to its chosen slot (say \( i \)) can easily determine its turn in the sequence of \( S \) transmissions by counting the number of ‘1’s in the summary bitmap starting from location one until location \( i \). For example, if the summary bitmap is 1001, the tag that had reserved the fourth slot in the reservation phase should transmit second in the data transmission phase. Fig. 6 shows the structure of this phase.

5. Acknowledgment phase: This is the final phase in which the reader acknowledges the data transmitted by the tags. The acknowledgment is sent in the form of bits – a ‘1’ indicates that the transmission in that corresponding slot was received successfully and a ‘0’ indicates that the transmission was not successful. All tags that received a positive acknowledgment from the reader will transit to the identified state and become muted. Tags that received a negative acknowledgment will go back to the active state and will participate in the reading process again. The structure of this phase is shown in Fig. 7.

Duration of a round: Let \( T \) denote the total duration of a round. Let \( T_{Ad} \), \( T_R \), \( T_{Su} \), \( T_D \), and \( T_{Ack} \)
respective denote the durations of the advertisement, reservation, summary, data transmission, and the acknowledgment phases. Now, $T$ can be written as
\[ T = T_{Ad} + T_R + T_{Su} + T_D + T_{Ack}. \] (1)

From the system model, it is clear that when the EPC is transmitted along with the CRC, the symbol duration is 4 µs. For all other transmissions, the symbol duration is 12.5 µs. Therefore, we can express $T_{Ad}$, $T_R$, $T_{Su}$, $T_D$, and $T_{Ack}$ as follows (all in µs):
\[ T_{Ad} = 12.5 \times (20 + \log_2 M + 3), \]
\[ T_R = 12.5 \times N(n + 1), \]
\[ T_{Su} = 12.5 \times (10 + N), \] (2)
\[ T_D = S \times 80 \times 4 + 12.5 \times S, \]
\[ T_{Ack} = 12.5 \times (10 + S). \]

**Remarks:** In the proposed framework, the total number of information bits that are exchanged between the reader and the tags is $Nn + N + 64S$. In the original protocols (DFSA or EDFSA), $64N$ number of bits are exchanged between the reader and the tags. Since $S \leq N$, by choosing $n \ll 64$, the total number of bits exchanged during a single round in AFSA can be made smaller than that of the base protocols. This in turn can lead to quicker rounds and smaller tag reading times.

### 4.3. Tag estimation

In order to advertise the appropriate frame size ($N$) and the group number ($M$), the reader has to arrive at $K$ – an estimate on the actual number of tags $K$ that need to be read. Several tag estimation strategies have been proposed in researchers in [12,13,19,21] and any of them can be used. In this paper, we follow the strategy suggested in [19] on account of its simplicity and adequate accuracy. For the sake of completeness, we reproduce the tag estimation strategy here.

Let $E[I]$ be the expected number of idle slots when the frame size is $N$ and the participating tag count is $K$. It is possible to express $E[I]$ as
\[ E[I] = N(1 - \frac{1}{N})^K. \] Let $Z_l$ be the total number of idle slots observed by the reader in a given round. Then by approximating $E[I]$ with $Z_l$, we can estimate the number of tags that would have participated in that round as
\[ \hat{K}_{Exp} = \frac{\log(\frac{Z_l}{N})}{\log(1 - \frac{1}{N})}. \] (3)

The number of tags that will participate in the next round can be arrived as $\hat{K}_{Exp} - Z_S$, where $Z_S$ refers to the number of successful slots observed in that round. In cases where no idle slots have been observed, i.e., $Z_l = 0$, the tag count is estimated to be the lower bound of $Z_S + 2Z_U$ where $Z_U$ refers to the number of collision slots observed in a given round.

It is to be noted that if the underlying protocol dictates that the group count $M$ be greater than 1, then only a fraction of the unread tags would have participated in a round. Therefore, in such a scenario, an estimate on the actual number of unread tags can be arrived at by multiplying the estimated tag count $\hat{K}_{Exp}$ with the number of groups $M$.

### 5. AFSA optimization

In the discussions in Section 4, it was mentioned that during the reservation phase, each tag transmits a randomly chosen $n$ bit sequence in the slot selected by it. If more than one tag chooses the same slot, a collision occurs. The reader can detect this collision only if the bit sequences transmitted by the tags that chose the same slot are different. However, if all the tags that chose the same slot transmit the same $n$ bit sequence, the reader cannot detect the collision occurring in that slot. The consequence of a collision going undetected is that the reader may wrongly interpret a slot reservation to be successful when in reality, it is not. This incorrect interpretation will lead to collisions in the data transmission phase. In other words, undetected collisions will increase the number of bits wasted in a round which in turn increases the round time and thereby the overall tag reading time.

The probability of a collision going undetected depends on two factors – (a) the probability of more than one tag choosing the same slot and (b) the probability of all the tags that chose the same slot selecting the same $n$ bit sequence. The former is decided by the value of frame length $N$ and the number of tags $K$ competing for the slots, while
the latter is decided by the value of the parameter $n$. Since the values of $N$ and $K$ are decided by the underlying protocol, AFSA has no control over them. The only parameter that can be varied to reduce the number of undetected collisions is $n$, the length of the bit sequence used by the tags to reserve their slots.

Intuitively, it is clear that by opting for a large value for $n$, the number of undetected collisions can be decreased. However, increasing the value of $n$ also implies that the tags have to transmit more number of bits in the reservation phase, thereby increasing the round time and overall tag reading time. Thus both smaller and larger values of $n$ increase the overall round time. We now set out to determine the optimal value $n^*$ that minimizes the total round time for given values of $N$ and $K$. A closer look at the different phases of the proposed framework will reveal that, the value of $n$ significantly affects the durations of the reservation, data transmission, and acknowledgment phases alone.

Once the durations of these phases are known in terms of $n$, $n^*$ can be determined by differentiating their sum with respect to $n$ and equating the outcome to zero. We now reproduce the expressions for $T_R$, $T_D$ and $T_{\text{Ack}}$ below:

\begin{align*}
T_R &= N(12.5n + 12.5), \\
T_D &= S(80 \times 4 + 12.5), \\
T_{\text{Ack}} &= (10 + S) \times 12.5.
\end{align*}

The parameter $S$ represents the total number of successful slot reservations as perceived by the reader in the reservation phase. It is possible to write $S$ as follows:

\begin{equation}
S = E[R] + E[UC],
\end{equation}

where $E[R]$ represents the average number of slots that were truly reserved (i.e., these are the total number of slots that were each selected by exactly one tag) and $E[UC]$ represents the average number of collision slots that were undetected by the reader. Since the tags choose one among the $N$ slots as per a uniform distribution, the value for $E[R]$ can be written as [14]:

\begin{equation}
E[R] = K\left(1 - \frac{1}{N}\right)^{K-1}.
\end{equation}

A collision goes undetected when more than one tag chooses the same slot and all the tags corresponding to a given slot select the same $n$ bit sequence. Let $X(i)$ be the expected number of slots that have been selected by exactly $i$ tags. $X(i)$ can be written as $X(i) = N\binom{K}{i} \left(\frac{n}{K}\right)^i \left(1 - \frac{n}{K}\right)^{K-i}$. Since the slot selection and the $n$-bit sequence selection are independent of each other, the expected number of undetected collisions that may happen in slots with tag tag occupancy $i$ can be written as: $E[UC|\text{occupancy} = i] = X(i)p_i$, where $p_i$ is the probability that all the $i$ tags in a slot choose the same $n$ bit sequence and is given by $p_i = \frac{1}{2^{n(i-1)}}$. Now the expected number of undetected collisions that may happen in all the $N$ slots can be written as

\begin{equation}
E[UC] = \sum_{i=2}^{K-E[R]} X(i)p_i,
\end{equation}

While the above expression gives the accurate value of $E[UC]$, it becomes little cumbersome to handle when we attempt to find $n^*$. We derive an alternate, simpler expression that gives an approximate value of $E[UC]$ and use this expression to determine $n^*$. Let $E[U]$ represent the total number of slots that were chosen by more than one tag. The value for $E[U]$ is given by

\begin{equation}
E[U] = N - E[I] - E[R],
\end{equation}

where $E[I]$ refers to the total number of slots that were not chosen by any of the tags (idle slots). Given that each tag independently selects any particular slot with equal probability, $E[I]$ can be written as $E[I] = N(1 - \frac{n}{K})^K$ [13]. It is easy to observe that if the reader fails to detect a collision in a particular slot, then that slot should be one of the $E[U]$ slots. Also by definition, each of the $E[U]$ slots is chosen by at least two tags (possibly more). Therefore, if $P_2$ represents the probability of two tags selecting the same $n$ bit sequence in the reservation phase, then $E[UC]$ can be written as

\begin{equation}
E[UC] \leq E[U]P_2.
\end{equation}

It can also be noted that $P_2$ is equal to $1/2^n$. We use the above upper bound to approximate $E[UC]$ in our $n^*$ estimation. Combining the above, and minimizing the sum of $T_R$, $T_D$ and $T_{\text{Ack}}$ with respect to $n$ gives us $n^*$ which can be written as

\begin{equation}
n^* = 3.32\log_{10}\left(\frac{19.14E[U]}{N}\right).
\end{equation}

Using the above value of $n^*$ in the protocol minimizes the total round time for given values of $N$ and $K$.

Remarks on the approximation: As mentioned earlier, for analytical tractability, we approximate
the actual value of $E[UC]$ with an upper bound. In order to get some perspective on the tightness of this bound, the actual values of $E[UC]/C_{138}$ and the upper bound used for approximation are shown in Fig. 8. In obtaining these values, the relation $N = K$ was followed, since this is the optimal frame size for ALOHA based tag reading protocols with equal slot sizes. From the figure, we can notice that the approximation used introduces an error of roughly 35%. Since we approximate $E[UC]$ by an upper bound, the resulting value of $n_{\text{approx}}$ will be conservative, i.e., it will be greater than or equal to the true value of $n^*$. In spite of using this conservative $n_{\text{approx}}$ value, we find that the performance of the AFSA framework is still better than that of stand alone ALOHA protocols.

6. m-AFSA: AFSA for mobile environments

One of the biggest challenges facing RFID tag reading protocols is to read the tags effectively and efficiently when the tags are mobile and are passing through the readers’ range as in the case of assembly lines. The challenge arises out of the fact that in a mobile setting, the reader has a limited time to complete the reading process. The tags get muted or killed not only after a successful read but also when they leave the reader’s field, whichever occurs first. The focus of this section is to extend the proposed AFSA framework to a mobile setting and design the initial tag population, tag arrival and departure rates such that a desired tag reading probability is guaranteed.

We adopt a system model similar to the one used in [19]. We consider a 900 MHz RFID system in which the tags arrive into the reader region on a conveyor belt moving with a constant velocity $v$. In stationary RFID systems, the reader calibrates and synchronizes the tags only at the beginning of the identification process. In mobile RFID systems, since new tags may continuously enter the reader’s field, the reader should be able to accommodate intermediate tag calibration and synchronization. In order to achieve this, at the beginning of each round, the reader transmits the ‘oscillator calibration’ signal to synchronize the newly arrived tags. The round structure adopted in $m$-AFSA is shown in Fig. 9 and the system setup is shown in Fig. 10.

The reader is located at a height of $h$ meters above the conveyor belt on which the tags arrive. The reader is assumed to be capable of communicating with a tag as long as the distance between the tag and the reader is less than $l_{\text{max}}$, the maximum operating range of the reader. The tags spend a total time of $t = t_c + t_o$ in the reader’s vicinity where, $t_c$ is the time duration during which tags get energized and synchronized and $t_o$ is the operating time available for the tags to transmit their data before they leave the reader’s field. It is easy to see that $t = 2\sqrt{\frac{l_{\text{max}}^2 - h^2}{v}}$. As in [19], we choose $t_c = T + T_{\text{cal}} + T_{\text{Null}}$ to guarantee that each new batch of tags get at least one calibration cycle before they can transmit their data. As before, $T$ refers to the duration of a round. Let $q$ denote the number of rounds a tag can participate in the reading process when it is passing through the reader’s range. It is easy to observe that:

$$q = \frac{t_o}{T + T_{\text{cal}} + T_{\text{Null}}}$$

$$= \frac{1}{T + T_{\text{cal}} + T_{\text{Null}}} - 1. \quad (11)$$

Let $P$ be the tag reading rate that is desired. If $p_t$ is the average probability of a tag being read in a given round, it is clear that $p_t$ should satisfy the following condition:
\[(1 - p_i)^q \leq 1 - P.\]  \hspace{1cm} (12)

We denote the group of tags that enter the reader’s field at the beginning of round \(i\) as group \(G_i\) tags. Let \(\psi\) be the arrival rate of the tags. Since the tags are moving on a conveyor with constant velocity, the tag arrival rate is equal to the tag departure rate.

Under this setup, to complete the design of \(m\text{-AFSA}\), we need to find \(G_i\), \(p_i\), \(T\), and \(\psi\) such that Eqs. (11) and (12) are satisfied. The design of \(m\text{-AFSA}\) is influenced by the base protocol under consideration. \(m\text{-AFSA}\) design is notably different for DFSA and EDFSA owing to the philosophical difference (in terms of using limited and unlimited frame sizes) in the base protocols themselves. In the following paragraphs, we delineate the design of \(m\text{-AFSA}\) for DFSA and EDFSA.

6.1. \(m\text{-AFSA for DFSA}\)

In DFSA, the number of tags participating in a round \(G_i\) influences the round duration \(T\). On the surface, it may seem that \(G_i\) can affect the value of \(p_i\) too. However, as the following arguments show, the success probability \(p_i\) remains unaffected by \(G_i\). This is because, in DFSA, under ideal conditions, the frame size \(N_i\) in a given round \(i\) is always chosen such that \(N_i = G_i\). With this relation, the expected number of tags to be successfully read in a round is given by

\[SG_i = G_i \left(1 - \frac{1}{G_i}\right)^{G_i - 1}\]

\[\approx G_i \cdot e^{-1}(\text{for large values of } G_i)\]

\[= 0.368G_i.\]  \hspace{1cm} (13)

Thus we can arrive at \(p_i = 0.368\), which is independent of \(G_i\). Having determined \(p_i\), the number of rounds required to meet the desired read percentage can be found from Eq. (12) as

\[q = \left\lfloor \frac{\log(1 - P)}{\log(1 - p_i)} \right\rfloor.\]  \hspace{1cm} (14)

From \(q\) and Eq. (11), the duration available for a single reading cycle \(T\) can be derived as

\[T = \frac{t}{q + 1} - T_{\text{cal}} - T_{\text{Null}}.\]  \hspace{1cm} (15)

Let \(G_1\) denote the number of tags participating in round 1. It is clear that at the end of round 1, 0.368 \(G_1\) tags will be successfully read by the reader. The key in \(m\text{-AFSA}\) is to keep the number of unread tags in each round to be approximately the same at \(G_1\), i.e., \(G_i = G_1\) for all \(i\). Since 0.368 \(G_1\) tags are read on an average in round 1, the number of new tags that enter the reader’s field in round 2 is set to 0.368 \(G_1\), i.e., \(\psi(T + T_{\text{cal}} + T_{\text{Null}}) = 0.368G_1\) from which \(\psi\) can be determined. This also means that 0.368 \(G_1\) tags leave the reader’s field at the end of every round. All that remains now is to determine the \(G_1\) value which will satisfy the \(T\) obtained above in Eq. (15).

Estimating \(G_1\): For convenience, we re-state Eq. (1) that defines \(T\) here:

\[T = T_{\text{Ad}} + T_R + T_{\text{Su}} + T_D + T_{\text{Ack}}\]

\[= (20 + \log_2 M + 3) \times 12.5 + N(n + 1)\]

\[\times 12.5 + (10 + N) \times 12.5 + S \times 80 \times 4 + S\]

\[\times 12.5 + (10 + S) \times 12.5.\]  \hspace{1cm} (16)

We now proceed to express\(^{6}\) \(N\), \(n\), and \(S\) in terms of \(G_1\). Since the frame size is always equal to the number of tags, \(N = G_1\). In AFSA, we always use the optimal value of \(n\) given in Eq. (10). By substituting \(N = G_1\) and \(K = G_1\) in this expression and through simple algebra, we can arrive at the result \(n^* = 2\), i.e., when \(N = K = G_1\), the optimal value for \(n\) is a constant and equals 2. Using this result and from Eqs. (5), (6), (8), and (9), we can arrive the following:

\(^{6}\) Recall that in DFSA, the number of groups \(M = 1\).
\[ S = E[R] + E[U] \cdot P_2 = 0.434G_1. \]  

(17)

Having expressed \( N, n, \) and \( S \) in terms of \( G_1, \) we can now write \( T \) (in \( \mu s \)) given in Eq. (16) as

\[ T = 537.5 + 199.73G_1 \]  

(18)

from which \( G_1 \) can be determined.

Recall that to complete the design of \( m-AFSA \) with DFSA as the underlying protocol, parameters \( G_i, p_i, T, \) and \( \psi \) have to be determined such that (11) and (12) are satisfied. It is clear that the above discussions precisely enable us to achieve this. Therefore, the design of \( m-AFSA \) with DFSA as the underlying protocol is complete.

6.2. \( m-AFSA \) for EDFSA

In EDFSA, on account of assuming an upper bound on the frame size, the number of tags participating in a round \( G_i \) not only affects the round time \( T \) but also the success probability, \( p_i. \) In designing \( m-AFSA \) for EDFSA, we severe the dependency of \( T \) on \( G_i \) by using a conservative value for \( T. \) The \( T \) value chosen is conservative in the sense it minimizes \( q, \) the number of rounds in which a tag can participate. Once a conservative value of \( q \) is estimated, we determine the \( G_i \) values such that the resulting probability \( p_i \) satisfies Eq. (12). This will complete the design of \( m-AFSA \) for EDFSA.

A conservative estimate for \( q: \) As Eq. (11) shows, \( q \) depends on \( t \) and \( T. \) The value of \( t \) is determined by the system architecture, while \( T \) depends on \( N, K, \) and \( n. \) The actual value of \( N \) (the number of reservation slots) is determined by the number of unread tags \( K \) that are present within the reader’s range. Since we use EDFSA as the base protocol, the \( N \) value varies as shown in Table 1. In a mobile setting, as there is a continuous stream of unread tags arriving into the system, it is advantageous to set \( N \) at the maximum possible value so as to increase the system throughput. This implies that \( N \) can be fixed at 256. With \( N \) set at 256, the value of \( T \) can be determined if the parameter \( S \) is known. The value of \( S \) depends on \( K \) and \( n^* \). The value of \( n^* \) used in AFSA is actually dynamic and it varies as per the value of \( K. \) To simplify the design, we approximate \( n^* \) by the constant 2 and as the simulation results will shortly reveal, it turns out that this is a reasonable approximation in terms of the overall tag reading time. With \( N \) and \( n^* \) fixed, \( S \) now depends only on \( K, \) the number of unread tags in the population. The value of \( K \) is chosen to be the one that maximizes \( S. \) Maximizing \( S \) will maximize \( T \) and therefore will minimize \( q. \) For \( N = 256, \) and \( n^* = 2, \) we find that \( K = 336 \) maximizes \( S \) and hence this \( K \) is used to estimate \( T. \) With afore stated parameters, the maximum value of \( T \) is estimated to be 52.92 ms using which a conservative estimate for \( q \) can be arrived at.

Estimating \( G_i: \) Having estimated a conservative value for \( q, \) all that remains to be estimated are the \( G_i \) values. Let \( G_i \) denote the number of tags participating in round 1 and \( SG_1 \) represent the number of tags that are successfully read in round 1. As per the EDFSA grouping statistics presented in Table 1, when \( N = 256, \) the optimal number of tags that can participate in the reading process is between 177 and 354. Following the same principle in our design, we set \( 176 \leq G_1 \leq 354. \) Under these conditions, it can be shown easily that \( 88 \leq SG_1 \leq 94 \) and that the average value of \( SG_1 \) is 92. The key in \( m-AFSA \) is to keep the number of unread tags in each round to be approximately the same at \( G_1, \) i.e., \( G_i = G_1 \) for all \( i. \) Since 92 tags are read on an average in a round, the number of new tags that enter the reader’s field in the next round is set to be 92, i.e., \( \psi(T + T_{cal} + T_{Null}) = 92 \) from which \( \psi \) can be determined. This also means that 92 tags leave the reader’s field at the end of every round. Therefore \( p_i, \) the average probability of a tag being read in a round, can be expressed as \( p_i = \frac{92}{G_1}. \) Having found the lower bound for \( q \) and the average value of \( p_i \) in terms of \( G_1, \) it now possible to re-write Eq. (12) as

\[ p_i = \frac{92}{G_1} = 1 - (1 - P)^{1/q}. \]  

(19)

From Eq. (19), \( G_1 \) can be determined and this completes the design of \( m-AFSA \) with EDFSA as the underlying protocol.

Remarks: For high values of \( P, \) it is possible that the value of \( G_1 \) obtained from Eq. (19) can be less than 177. If so, the frame size of \( N = 256 \) may no longer be optimal (as per the EDFSA tag grouping rule) for the mobile tag reading system. In this case, the design process is re-started with a smaller frame size of \( N = 128 \) and continued using the principles outlined above.

7. Simulation results

In this section, we present our simulation results that delineate the performance of the proposed AFSA framework. The simulations were done in C++ and the results presented here are the outcome of 50 different runs. The variations indicated in the graphs corre-
respond to the 100% confidence interval. Initially, we discuss the performance of the AFSA framework in a static setting and then move on to the mobile setting.

7.1. Performance in a static setting

To begin with, we study the impact of the parameter $n$ on the performance of the proposed AFSA framework. We discuss the results that are obtained with EDFSA as the base protocol. Similar results were obtained with DFSA as the base protocol and for the sake of brevity we do not discuss them. As mentioned earlier, the parameter $n$ plays an important role in determining the average tag reading time. Larger values of $n$ decrease the number of undetected collisions; however, they increase the duration of the reservation phase and thus may increase the overall round time. Smaller values of $n$ decrease the duration of the reservation phase but increase the number of undetected collisions; thus a $n$ value that is too small can increase the overall round time. This behavior is brought out in the simulation results presented in Fig. 11.

The figures show that the average tag reading time increases for both small and large $n$ values. We also note that minimizing the time spent in undetected collisions does not necessarily lead to minimum tag reading time. It can be seen that true to its objective, $n = n^*$ minimizes the overall tag reading time but not necessarily the time spent in collisions. While $n = n^*$ gives the best read time, we observe that the performance of $n = 2$ is very close to that of $n = n^*$. The reason for this can be found out from Fig. 12. This figure shows that for most of the $(N,K)$ combinations encountered during the read process, $n^*$ is actually 2 and hence the observed behavior. This result also validates to some extent the approximation $n^* \approx 2$ made in Section 6.2.

Having discussed the performance of AFSA for different $n$ values, we now compare the performance of AFSA with other protocols. The following protocols are compared in our simulation study: 1. DFSA [13], 2. EDFSA [14], 3. ASAP [19], 4. Query Tree (QT) [22], 5. Binary Tree (BT) [23], 6. AFSA + DFSA, and 7. AFSA + EDFSA. Note that QT and BT are tree based deterministic tag reading protocols. The protocols are compared in terms of the average tag reading times they achieve and their efficiency. Protocol efficiency is defined as the ratio of time spent in actual tag reading over the total time. We note here that the total time includes several components including actual tag reading time, time spent in collisions, idle time, and protocol overhead (in terms of transmitting reader commands and acknowledgments).

The graphs in Fig. 13 show the performance of different protocols in terms of the afore said parameters. In these set of experiments, the AFSA proto-

---

For the sake of clarity, the tag reading times of ALOHA based protocols and tree based protocols are shown in separate graphs.
cols use the optimal \( n \) value, i.e., \( n = n^* \). From the graphs, we can instantly recognize that all the ALOHA based protocols have smaller average tag reading times than the tree based protocols. Among the two tree based protocols, BT has a better reading time than QT. This is because, in QT, the reader has to transmit the query string in the beginning of each round which consumes bit times. In the case of BT, the reader need not transmit any query string since the tags are assumed to have memory to remember the tree path traversed by them during the reading process. This saves some bit times leading to smaller tag reading times.

Among the ALOHA based protocols, we notice that protocols that utilize the proposed AFSA framework deliver a superior performance in terms of the average tag reading time. AFSA + DFSA and AFSA + EDFSA achieve tag reading times close to 0.57 ms (equivalent to a rate of 1754 tags/s) while the base protocols’ reading times are in the range of 0.97 ms. The reading times achieved under AFSA + EDFSA and AFSA + DFSA are also better than those achieved under the ASAP tag reading protocol. Further, the AFSA protocols achieve the highest efficiency among the seven protocols that are compared. We also observe that the performance of AFSA + EDFSA and AFSA + DFSA are almost equal to each other. This shows that the proposed AFSA framework is equally good in improving the performance of protocols having

\[
\frac{n}{C^3}
\]

\[
P = 99\% \quad P = 99.9\%
\]

\[
\psi = 1032.01 \quad 1016.14
\]

\[
\psi = 1669.45 \quad 1663.32
\]

\[
\psi = 1734.26 \quad 1734.26
\]

\[
\psi = 1762.36 \quad 1755.14
\]

\[
\psi = 103.14 \quad 98.49
\]

\[
\psi = 1669.45 \quad 1663.32
\]

\[
\psi = 1734.26 \quad 1734.26
\]

\[
\psi = 1762.36 \quad 1755.14
\]

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Initial tag count ( (G_t) )</th>
<th>Rate ( (\psi) )</th>
<th>Achieved read %</th>
<th>Equiv. reading time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m-EDFSA )</td>
<td>184</td>
<td>1037.14</td>
<td>99.25</td>
<td>0.971</td>
</tr>
<tr>
<td>( m-DFSA )</td>
<td>181</td>
<td>1032.01</td>
<td>99.69</td>
<td>0.972</td>
</tr>
<tr>
<td>( m-ASAP )</td>
<td>285</td>
<td>1669.45</td>
<td>99.62</td>
<td>0.601</td>
</tr>
<tr>
<td>( m-AFSA + EDFSA )</td>
<td>288</td>
<td>1734.26</td>
<td>99.87</td>
<td>0.577</td>
</tr>
<tr>
<td>( m-AFSA + DFSA )</td>
<td>311</td>
<td>1762.36</td>
<td>99.87</td>
<td>0.569</td>
</tr>
</tbody>
</table>

Fig. 12. Sample variation of \( n^* \) values observed in one complete read.

Fig. 13. Performance of different tag reading protocols in a static setting.

Table 2

Performance of AFSA and other protocols in a mobile setting
limited frame sizes as well as unlimited frame sizes thereby highlighting its applicability to both the classes of ALOHA based tag reading protocols.

7.2. Performance in a mobile setting

In studying the performance of the AFSA framework in a mobile setting, we use the same system setup found in [19]. We set $v = 5$ m/s, $l_{\text{max}} = 2$ m, and $h = 1$ m. We study the system until 50000 tags have been read. Two different target read percentage values of $P = 99\%$ and $P = 99.9\%$ are used. The appropriate initial tag count $G_i$ and the tag departure (and arrival) rate $\psi$ are determined as per the discussions given under Section 6. The simulation is run for 20 times and the average values are reported. Table 2 presents the results of our simulation study.

From the results we see that the designed initial tag count allows all the protocols to achieve the desired tag reading percentage. We also notice that the tag departure rate under $P = 99.9\%$ is lower than the departure rate under $P = 99\%$. This is due to the fact that in order to meet a higher successful reading rate, contention in the system has to be reduced. Contention can be reduced only by allowing fewer tags to pass through the system. Hence, the lower departure rate for $P = 99.9\%$.

Further, we observe that the proposed AFSA framework significantly improves the system throughput in comparison to the base protocols. This is due to the fact that the protocols under the AFSA framework have a smaller round time which allows them to have more number of tag reading cycles. This in turn allows the protocols to have an higher initial tag count and a higher departure rate without compromising on the target read percentage.

It is also to be noted that when the tag departure rates achieved under the proposed design are translated into tag reading times (by taking the reciprocal of the product of tag departure rate and the achieved read percentage), they are almost equal to the average tag reading times achieved under a static setting. This shows that the proposed mobile system design is close to being the best one can hope for, since the tag reading time achieved for a given protocol under any mobile setting has to be at least the reading time achieved under a static setting.

7.3. Tag requirements for AFSA

In this section, we argue that the AFSA framework can be implemented on Class 1 generation 1 tags and Class 2 generation 2 tags with minimum hardware modification. We note that several ALOHA based anti-collision tag reading protocols have already been proposed for Class 1 Generation 1 and Class 1 Generation 2 tags. This implies that such tags currently possess the following abilities: (a) generate a random number within a given range and remember it; (b) identify any particular slot in a frame; (c) transmit the EPC in the chosen slot and (d) parse through the acknowledgment bit string advertised by the reader and understand the status of their EPC transmissions.

A quick analysis of the different phases in an AFSA round will reveal that in addition to the above capabilities, the tags should possess the following two features in order to implement the proposed framework: (i) ability to transmit a bit sequence other than their EPC – the tags should send $n^*$ number of bits in their reservation phase, where $n^* \ll 64$ (assuming the EPC length to be 64 bits); and (ii) ability to transmit more bits in a round than under existing ALOHA protocols – tags whose reservation was successful in the reservation phase end up transmitting a total of $n^* + 64$ bits under AFSA whereas in other ALOHA protocols, they need to transmit only 64 bits.

Feature (i) can be implemented by allowing the tags to transmit the first $n^*$ or the last $n^*$ bits of their EPCs, and thus eliminate the need for additional memory. While one may argue that $n^*$ is a parameter that varies between rounds, as the simulation results have shown, it is possible to approximate $n^*$ with 2 and still achieve a good performance. Feature (ii) can be achieved by allowing the reader to induce more energy in the tags by transmitting a stronger electromagnetic pulse. Apparently, one can think that this may result in additional power consumption at the reader. However, the improvement achieved in terms of smaller tag reading times may necessitate the reader to be active only for a shorter duration and the resulting energy savings may partially or fully offset the additional energy consumption. Also, in instances where the reader can be recharged periodically from the mains, energy consumption at the reader may not be an issue.

8. Conclusions

In this paper, we introduced a framework named Accelerated Frame Slotted ALOHA (AFSA) for
reducing the reading time of passive RFID tags in dense environments. AFSA can be used in conjunction with almost all RFID tag reading protocols that are based on frame slotted ALOHA. We discussed the working of AFSA in conjunction with two different ALOHA protocols – one with unlimited frame sizes and the other with limited frame sizes. AFSA reduces the tag reading time by using bitmaps and avoids the wastage in bit times due to collisions and idle slots. Further, we also described extensions of AFSA to read passive tags in a mobile setting. Our simulation results showed that AFSA can significantly reduce the average tag reading time with respect to the base protocols and achieve high tag reading rates under both static and mobile settings. The results also show that AFSA works equally well with protocols that use unlimited frame sizes as well as limited frame sizes. Currently, we are working towards extending AFSA by incorporating generic distributions for slot selection.

Acknowledgements

We thank all the reviewers for their insightful comments. Without doubt, their suggestions have greatly helped us to improve the quality of the paper.

References

Venkatesh Sarangan obtained his Ph.D., in computer science and engineering from The Pennsylvania State University, University Park, in August 2003. Since then he has been with the Computer Science Department at Oklahoma State University as an assistant professor. His research interests include Internet protocols, RFID Systems, wireless and sensor networks, and queuing theory. His research work is being supported by various agencies including National Science Foundation, Federal Department of Highway Administration, Federal Transit Administration, NASA-EPSCoR, Oklahoma Department of Transportation and local industry.

Mallareddy Devarapalli obtained his MS in Computer Science from Oklahoma State University, Stillwater, in July 2007. His interests include RFID systems and systems programming.

Sridhar Radhakrishnan is a Professor in the School of Computer Science at the University of Oklahoma, which he joined in 1990. He received his Ph.D. in Computer Science from Louisiana State University in 1990. He received his undergraduate degrees from Vivekananda College, Chennai, India and from University of South Alabama, Mobile, Alabama. His research interest is in the design of protocols for broadband, wireless and mobile networks. He has published many research articles in journals, conference proceedings, and book chapters. His research is supported by industry and government.