Performance Improvement of Generation-2 RFID Protocol

Chonggang Wang
University of Arkansas
Fayetteville, AR 72701
USA
cgwang@uark.edu

Mahmoud Daneshmand
AT&T Labs Research
Florham Park, NJ 07932
USA
daneshmand@att.com

Bo Li
Hong Kong University of Science and Technology
Hong Kong, China
bli@cse.ust.hk

Kazem Sohraby
University of Arkansas
Fayetteville, AR 72701
USA
sohraby@uark.edu

ABSTRACT

Radio frequency identification (RFID) provides a non-line-of-sight and contactless approach for object identification. But if there are multiple tags in the range of an RFID reader, *tag collision* can take place due to radio signal interference and therefore an anti-collision algorithm is required to resolve collisions. Recently, EPCglobal RFID generation-2 (Gen-2) protocol [1] is proposed for ultra-high frequency (UHF) passive tags and is being deployed. Gen-2 designs a slotted random anti-collision algorithm, especially, an adaptive slot-counter (Q) selection algorithm. The integer-valued parameter Q in Gen-2 plays a critical role in tag collision resolution. This adaptive algorithm dynamically adjusts the value of Q based on the type of replies from tags. In this paper, we propose an optimal Q algorithm that determines the optimal values of Q according to the number of remaining tags and in turn to optimize tag identification speed (TIS). It's been demonstrated through extensive simulations that the proposed algorithm achieves higher TIS than Gen-2 adaptive Q algorithm.

Categories and Subject Descriptors

C.3.3 [**Network Protocols**]: Applications, Protocol architecture, Protocol verification, and Routing protocols.

General Terms

Algorithms, Performance, Design.

Keywords

Radio Frequency Identification, Air Interface, Generation-2 RFID, Performance Analysis, Tag Identification Speed.

QShine 2008, July 28-31, 2008, Hong Kong, Hong Kong.
Copyright 2008 ICST ISBN 978-963-9799-26-4
DOI 10.4108/ICST.QSHINE2008.3926

1. INTRODUCTION

RFID has been existed for many years since its first application of "identification, friend, or foe" (IFF) in World War II [2]. Recently, it is being used in many fields including supply chain management, homeland security, military deployment management, healthcare industry and airline industry. A RFID system consists of the following important components: 1) *RFID tags (or transponders)*. Each tag consisting of a microchip and an embedded antenna contains a unique identity or called Electronic Product Code (EPC). An object or item affixed with one tag at least is identified when the tag is interrogated by a RFID reader. Tags could be classified into active tags, semi-active tags, and passive tags depending on whether they have embedded power or not and what the embedded power is used for; 2) *RFID readers (or Interrogators)*. A RFID reader usually has more than one separate antenna and is responsible to read potential tags around it. The communication between the reader and tags are established by an air-interface protocol that provides operation modes and procedures for both reader-to-tag and tag-to-reader directions. Air-interface interface protocol defines the command format, synchronous timing between the reader and tags, and determines how the frequency and time resource could be shared by the reader and tags; 3) *RFID Database*. Each record of RFID raw data may contain information such as reading time, location, and tag EPC. In business environments such as a distribution center, many pallets, cases and products attached with RFID tags could arrive during a very short time and therefore massive RFID data flow is produced, which needs to manage in an efficient and timely manner. RFID readers usually store some raw data at the front-end of the reading process, producing a filtered RFID database at the end of interrogation cycle. In fact, tags and readers form the frond-end communication subsystem, while database is a part of back-end software subsystem, as shown in Fig. 1.

As shown in Fig. 1, each reader can only interrogate tags within its vicinity, or called *interrogation region* [3]. When multiple tags scattering within the interrogation region, there are chances that reply signals from multiple tags occur simultaneously and therefore *tag collisions* occur. On the other hand, multiple readers can be deployed in reality and networked together into a reader network. In case of multiple readers placed close to each other, it is possible that the signal of one reader interferes with others. Such interference caused by the presence of multiple readers was called *reader collision* [4].

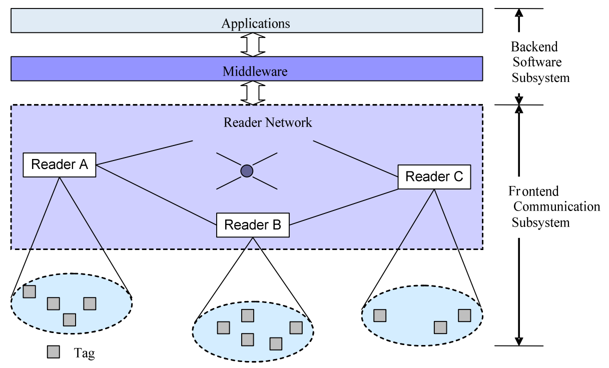

Figure 1. General RFID architecture.

This paper investigates tag collisions and proposes an optimal Q algorithms based on Gen-2 [1] protocol. The rest of this paper is organized as follows. Section II describes the anti-collision-related features of Gen-2 protocol. Section III presents the optimal Q algorithm. Section IV gives out simulation results for performance comparisons of our algorithm to Gen-2 adaptive Q algorithm. Finally, Section V concludes the whole paper.

2. GEN-2 PROTOCOL

Gen-2 protocol [1] offers a new air-interface protocol including physical and media access control (MAC) specifications for UHF RFID passive tags, which operates in the range of 860 MHz to 960 MHz. This protocol provides advanced new features designed for fast tag identification, flexibility, and security. First, Gen-2 protocol can partition the tag population into distinct sub-populations so that tags can associate separately and independently with each of the several readers.

2.1 Basic Operations of Tag Identification Process

Tag identification process consists of a number of inventory rounds. In [1], inventory round is defined as "the period between successive Query commands" issued by the reader and therefore the issuing of a new Query command implies the ending of the current inventory round and the beginning of new inventory round. According to [1], the reader can issue a Query command when system is powered up or when the channel is idle; therefore, if there are no tags identified in a particular round, the reader can issue a Query command to start a new inventory round. During each inventory round, the reader issues a set of QueryAdjust, or QueryRep commands to identify tags. When a tag is identified in a particular inventory round, it will cease to respond to commands from the reader in the same round.

Gen-2 protocol defines three types of query commands: Query, QueryAdjust, and QueryRep. Query carries the value of parameter Q and initiates an inventory round; Query triggers each tag to select a random number and store it in its Slot Counter (SC). QueryAdjust is used to ask all tags to adjust the value of Q and reselect their SC. With this command, Q is incremented by 1, decremented by 1 or remains unchanged according to the adaptive Q algorithm that will be explained later. QueryRep is used by the reader to notify all tags to decrement their SC by 1. Those tags which contain SC = 0, will decrement to 7FFF. In summary, Query and QueryAdjust inform all tags of the latest Q value and trigger them to reselect SC, while QueryRep instructs tags to decrement their SC by 1. The sending of Query by the reader implies that a new inventory round has begun. Within an inventory round, several QueryAdjust and/or QueryRep can be transmitted by the reader, in order to identify the remaining tags. According to [1], Query command can be issued when the system is powered on or if there are no replies from the tags both QueryAdjust and QueryRep can be issued either after a tag is successfully identified, or channel is in collision. Such operations are described in [1] as options for implementation consideration. The basic operations of successfully identifying a tag can be summarized as follows (See Fig. 2):

1) **Reader $\rightarrow$ Tags:** The reader initiates the process of an inventory round by sending a "Query" command to the population of tags it was selected to participate in the round. The Query command sends an integer-valued parameter Q and instructs tags to independently select a random integer from the uniform distribution [0, 2^Q-1] and respond a 16-bit random number (RN16) to the reader if 0 is chosen. The reader waits for replies from the tags.

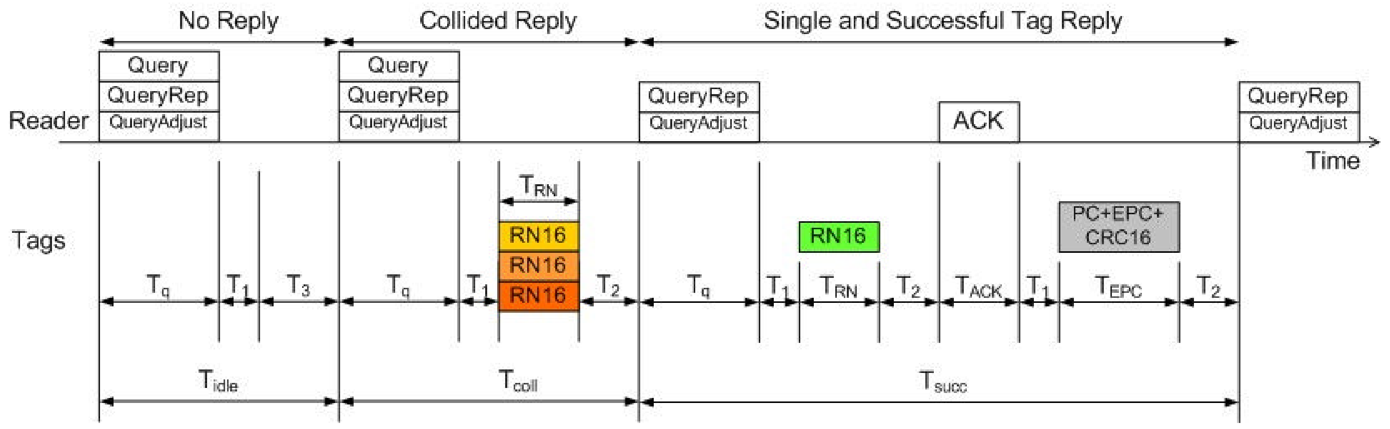

Figure 2. Tag identification process and timing relationship in Gen-2 protocol.

2) **Tags $\rightarrow$ Reader:** Tags' responses to the reader lead to either a Success *(S)* or a *Failure (F).* It is a *Success* if exactly one tag has selected number 0, and thus, that tag has being identified by the reader. Otherwise, it is a Failure and no tag is being identified.

3) **Reader $\leftrightarrow$ Tag:** If Success, then reader sends an acknowledgement (ACK) command back to the identified tag. The identified tag processes the received ACK and reports its EPC back to the reader.

4) **Reader $\rightarrow$ Tags:** Either Success or Failure the reader continues with new query commands (Query, QueryAdjust, or QueryRep) to identify remaining and newly arriving tags.

How tags respond to the reader is dependent on and controlled by the reader's commands. Since all tags respond independently, collision could occur among tags' responses and a "slotted random anti-collision" algorithm is described in [1] for its resolution. The following is a summary of the algorithm: Upon receiving a Query or QueryAdjust command, each tag deposits an integer-valued number in its slot counter, which is an integer selected at random from a uniform distribution [0, 2^Q-1], where Q is an integer-valued parameter. Q varies in the [0, 15] range, and is designated and adjusted by the reader. The value of Q is embedded in the Query command, and updated using QueryAdjust command. After selecting the random number, tags which have SC = 0 respond to the reader command. As shown in Fig. 2, if there is a collided or successful reply, the reader continues to issue QueryRep or QueryAdjust; if there is no reply, the reader could send a Query, QueryAdjust, or a QueryRep. These commands will instructs unidentified tags to either reselect or reduce their SC, or to restart and choose a new SC with a new Q value.

2.2 Adaptive Q Algorithm

When reader receives a reply from the tags after issuing a query command or the time T_1+T_3 has expired before receiving any reply (See Fig. 2), the algorithm in Fig. 3 is triggered by the reader to update Q based on the following rules. In this flow chart, suppose Q_{fp} is the float-point representation of Q. The value of Q is determined based on the integer nearest to Q_{fp}. The detailed operation is as follows:

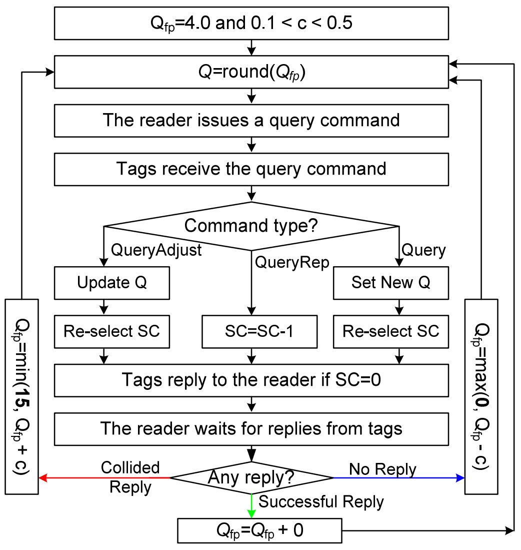
Q_{fp}=4.0 and 0.1 < c < 0.5
Q=round(Q_{fp})
The reader issues a query command
Tags receive the query command
Command type?
QueryAdjust
QueryRep
Query
Update Q
Set New Q
Re-select SC
SC=SC-1
Re-select SC
Tags reply to the reader if SC=0
The reader waits for replies from tags
Any reply?
Collided Reply
No Reply
Successful Reply
Q_{fp}=Q_{fp} + 0
Q_{fp}=min(15, Q_{fp} + c)
Q_{fp}=max(0, Q_{fp} - c)

Figure 3. Gen-2 adaptive Q algorithm.

- **Collided Reply:** This is due to the fact that more than one tag has selected SC=0, which in turn could imply that Q is too small and that the number of remaining tags is too large. In this case Q_{fp} is incremented by the value of parameter *c*, a real number. After this operation, if Q_{fp} exceeds 15, it is set to 15. The value of Q is the integer that is nearest to Q_{fp}, that is: Q=round(Q_{fp}).

- **No Reply:** This can be due to the fact that none of the tags has selected SC=0, which in turn could imply that Q is too large and that the number of remaining tags is too small. In this case Q_{fp} is decremented by *c*. After this operation, if Q_{fp} is negative, we let Q_{fp} =0. Then the value of Q is Q=round(Q_{fp}).
- **Successful Reply:** This means that only one tag has selected SC=0 and that the current value of Q is proper. In this case, Q_{fp} and Q remain unchanged.

According to the specifications in [1] (also shown in Fig. 3), the initial value of Q is Q_0=4. The typical values for *c* suggested by [1] are in the range of (0.1, 0.5). It is also suggested in [1] to use a small *c* when Q is large, and a large *c* when Q is small. Since *c*< 1, there are three possibilities after each update: Q increments by 1, Q decrements by 1, or Q remains unchanged. The reader uses QueryAdjust to notify tags of these possibilities. In our former paper [5], we designed a new slot-counter selection algorithm that uses different *c* values for cases of "collided reply" and "No Reply", respectively, and improves the tag identification speed therefore.

3. OPTIMAL Q ALGORITHM

The optical Q algorithm works as follows: 1) each time when a tag is identified and the number of remaining tags is n, the reader determines an optimal Q based on n; 2) then the reader notifies tags of this optimal value and instructs all tags to re-select their SC; 3) tags make response according the Gen-2 operations listed in Section II.1.

In order to determine the optimal Q value, let us first rewrite Q=Q(n) and deduce tag identification speed. We calculated TIS in [6] as follows:

$$TIS = N / \sum_{n=N}^{1} T(Q(n),n) \tag{1}$$

$$T(Q(n),n) = 1 * Ave\_T_{succ} + n_{q2}(Q(n),n) * Ave\_T_{idle} + (n_q(Q(n),n) - n_{q2}(Q(n),n) - 1) * Ave\_T_{coll}, \tag{2}$$

$$n_{q1}(Q(n),n) = 1 \text{ if n=N or 0 else,} \tag{3}$$

$$n_{q3}(Q(n),n) = n_q(Q(n),n) * p_i(Q(n),n)\ , \tag{4}$$

$$n_{q2}(Q(n),n) = n_q(Q(n),n) - n_{q1}(Q(n),n) - n_{q3}(Q(n),n)\ , \tag{5}$$

$$n_q(Q(n),n) = 1 / p_s(Q(n),n)\ , \tag{6}$$

$$p_s(Q(n),n) = n * (1/2^{Q(n)}) * (1 - 1/2^{Q(n)})^{n-1}\ , \tag{7}$$

$$p_i(Q(n),n) = (1 - 1/2^{Q(n)})^n\ , \tag{8}$$

where N is the number of total tags to be identified in an inventory round. When the number of remaining tags is n and Q=Q(n), p_i(Q(n), n) is the probability that a query command gets no reply, p_s(Q(n), n) is the probability to get a single/successful reply n_{q1}(Q(n), n) is the number of Query command used to identify a tag, n_{q2}(Q(n), n) is the number of QueryAdjust used to identify a tag, n_{q3}(Q(n), n) is the number of QueryRep used to identify a tag. Ave_T_{succ}, Ave_T_{idle}, and Ave_T_{coll}, are average duration of a successful reply, no reply and a collided reply, which can be easily deduced from Fig. 2.

Eq. (1) shows that TIS depends on Q and n. Now we consider optimizing TIS, given n. In order to maximize TIS, we need to find an optimal Q to minimize T(Q(n), n) in Eq. (2). Although it is difficult to get a closed form expression for the solution from Eq. (2), we can obtain the numerical result of the optimal Q by differentiating T(Q(n), n) with respect to Q(n) and setting the derivative to zero. We use Q^*(n) to denote the optimal Q value for TIS maximization. This optimal setting of Q to maximize TIS is our optimal Q algorithm.

Inputting Q^*(n) into Eq. (1), we can get TIS of our optimal Q algorithm. Fig. 4 shows Q^*(n) when the number of remaining tags varies between 0 and 1000. It can be seen that Q^*(n) increases with the increase of n. Another important observation is that Q^*(n) is not too sensitive to n. For example, when n is in the range of [350 700], Q^*(n)=10 when TRrate=15.625 Kbps. This observation makes our optimal Q algorithm be a highly practical approach to optimize tag reading performance because it does not need to know the exact value of n, although some accurate algorithms have been available to measure it [7].

Although our optimal Q algorithm needs to know the number of remaining tags n, we in this paper will not propose any estimation algorithm to predict the number of tags; however, some algorithms in literatures such as [7] are available to predict the number of tags. Instead, we will furthermore show in the next section though simulation that our optimal Q algorithm, although it is based on the number of remaining tags n, does not require its exact value.

4. SIMULATION RESULTS

Simulations are setup as follows: 1) RTrate is fixed at 64 Kbps and TRrate has four values – 125, 62.5, 31.25, and 15.625; 2) The number of tags N_t varies between 10 and 1000; below N_t=400, increments are 10 while between

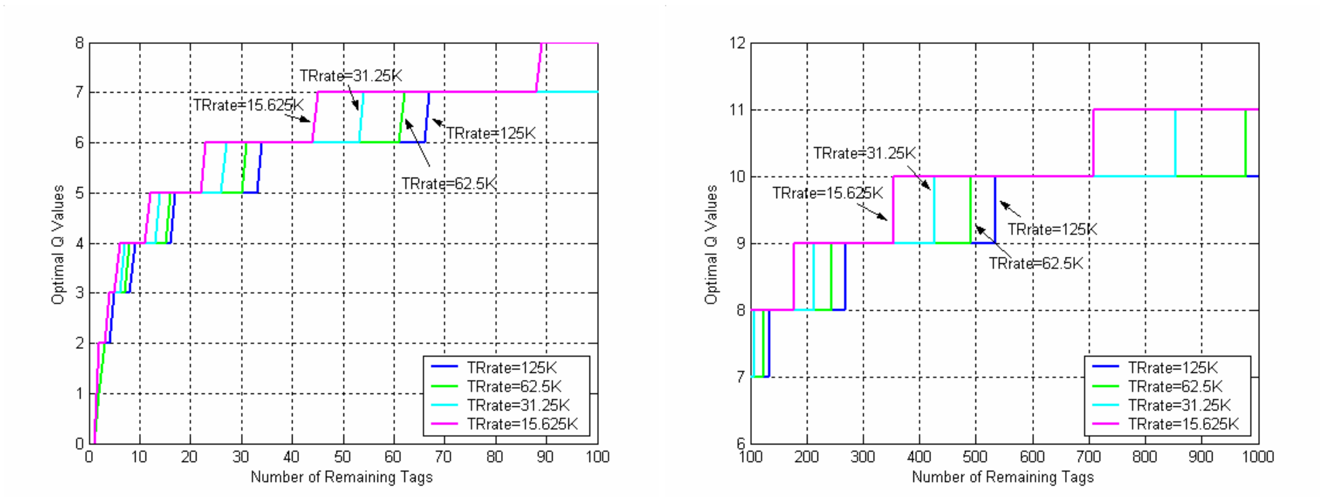

Figure 4. Optimal Q values $Q^*(n)$ (left: n is below 100; right: n is between 100 and 1000)

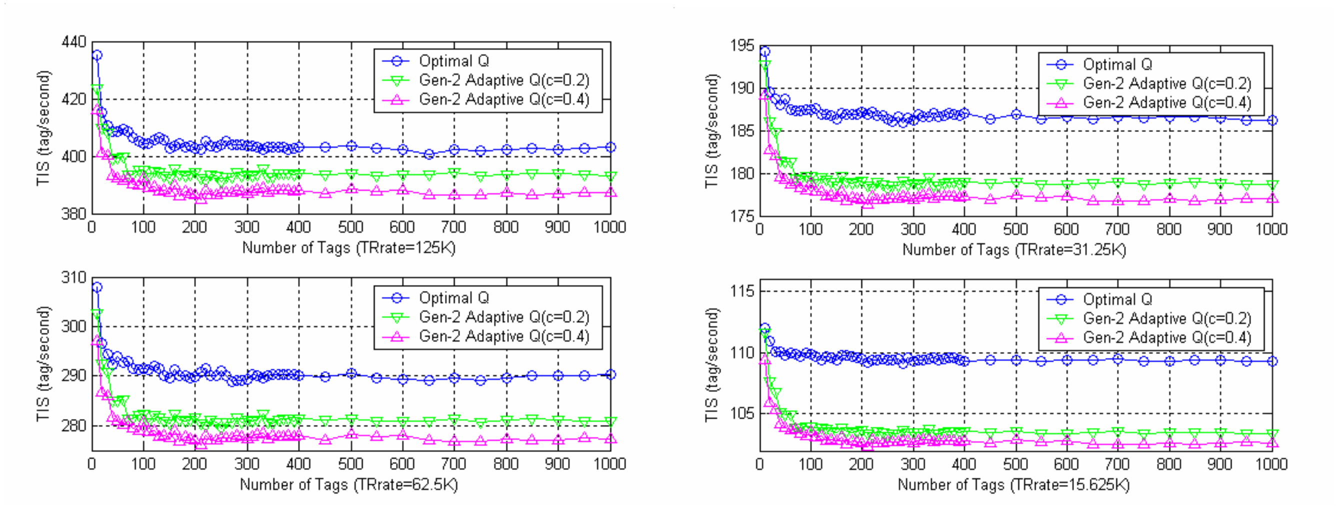

Figure 5. Performance comparisons between our optimal Q and Gen-2 adaptive Q.

N_t=400 and 1000 the increments are 50; 3) channel with zero bit error rate is assumed; 4) when there is a collided reply, the reader sends QeuryAdjust; when there is a successful reply, the reader issues QueryRep; when there is no reply, the reader uses QueryRep if Q has no change or QueryAdjust if Q gets changed. The typical values of other system parameters are given in Table 1. In simulation we measure tag identification speed as the ratio of the total number of identified tags over the total time consumed. As shown in Fig. 4, the optimal algorithm achieves higher TIS than Gen-2 adaptive Q algorithm, especially when the number of tags is large.

Since our optimal Q relies on the number of remaining tags *n*, we are interested in investigating the performance of both policies assuming that *n* is unknown. We note that approaches such as [7] can be used to estimate *n*. Simulation results in Fig. 6 are obtained by setting $n=n'*(1+f)$, where n' is the actual number of remaining tags and *f* is a real random number uniformly distributed in the range of [-0.4, +0.4]. In other words, we intentionally give false values of *n* to the optimal Q algorithm, which in turn calculate the optimal Q values based on these false values. As shown in Fig. 6, the optimal Q still achieves higher TIS

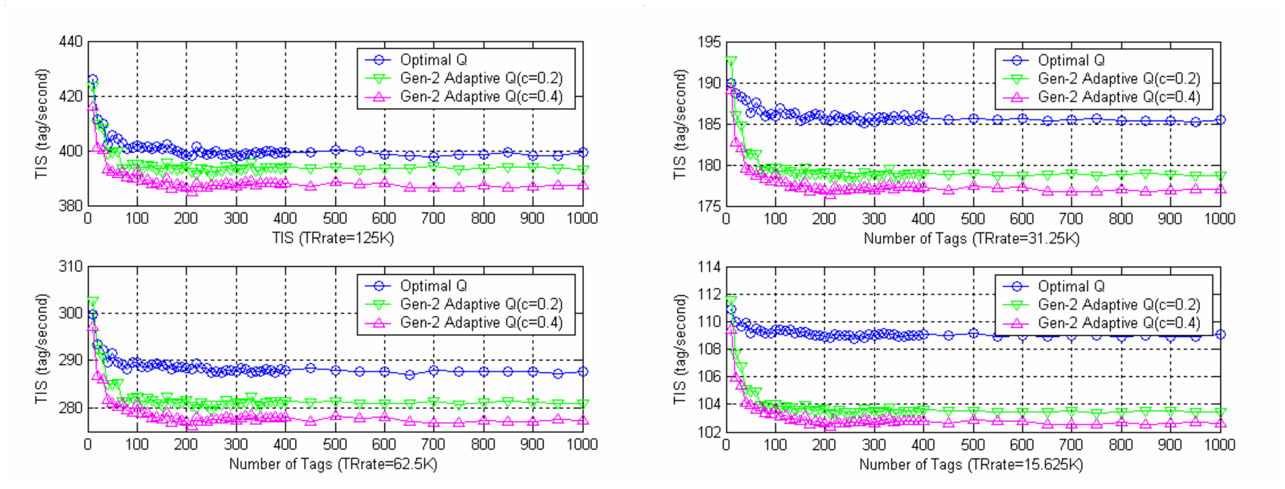

Figure 6. Performance comparisons between our optimal Q and Gen-2 adaptive Q considering that the number of remaining tags has a certain error.

than adaptive Q, even though there is an average 20% error in *n*, which implies that our optimal Q algorithm does not need accurate measurement of *n*.

5. CONCLUSIONS

This paper designs an optimal Q algorithm for RFID Gen-2 protocol, which adjusts the parameter Q based on the number of remaining tags. The new algorithm achieves better performance than Gen-2 adaptive Q algorithm in terms of tag identification speed.

6. ACKNOWLEDGMENTS

This research was supported by AT&T Labs Research.

7. REFERENCES

[1] EPCglobal Specification, "EPCTM radio-frequency identity protocols class-1 generation-2 UHF RFID protocol for communications at 860 MHz – 960 MHz," version 1.0.9, Jan. 2005. [Online] http://www.epcglobalinc.org

[2] J. Landt, "The history of RFID," IEEE Potentials, vol. 24, no. 4, Oct.-Nov. 2005, pp. 8 – 11.

[3] Z. Zhou, H. Gupta, S. Das, and X. Zhu, "Fast reading of RFID tags in multi-reader systems," Under Submission, 2006 [http://www.cs.sunysb.edu/~hgupta/ps/rfid.pdf]

[4] D. W. Engels and S. E. Sarma, "The reader collision problem," Proc. of IEEE International conference on Systems, Man and Cybernetics (SMC'02), Oct. 6-9, 2002.

[5] C. Wang, M. Daneshmand, and K. Sohraby, "A new slot-count selection algorithm for RFID protocol," Proc. of Chinacom 2007, August 22-24, 2007, Shanghai, China

[6] C. Wang, M. Daneshmand, and K. Sohraby, "Performance analysis of gen-2 RFID protocol," Technical Report, University of Arkansas, Jan 2008. [Online] http://comp.uark.edu/~cgwang/TR/RFID.pdf.

[7] M. S. Kodialam and T. Nandagopal, "Fast and reliable estimation schemes in RFID systems," Proc. of ACM Mobicom 2006, Los Angeles, CA, USA, September 23-29 2006, pp. 322-333.

Table 1. Typical values of Gen-2 system parameters

Parameters	**Value**
TARI	12.5 us
DATA0	1.0*TARI = 12.5 us
DATA1	1.5*TARI = 18.75 us
RTrate	64 Kbps
RTcal	31.25 us
TRcal	64.0 us
DR	8
LF	DR/TRcal = 125 KHz
M	1, 2, 4, 8
TRrate	LF/M = 125, 62.5, 31.25, 15.625 Kbps
T_{pri}	1/LF
T=>R Preamble	$6*T_{pri}$
T=>R End-of-Signaling	$2*T_{pri}$
Delimiter	12.5 us
R=>T Preamble (RTP)	Delimiter + DATA0 + RTcal + TRcal
R=>T FrameSync	RTP – Trcal
T_1	Max(RTcal, $10*T_{pri}$)
T_2	$5*T_{pri}$
T_3	$5*T_{pri}$
EPC	96 bits
Q_0	4

(TARI: Reference time interval for a data-0 in reader-to-tag signaling; DATA0: Time interval for a data-0 in reader-to-tag signaling; DATA1: Time interval for a data-1 in reader-to-tag signaling; RTcal: Reader-to-tag calibration symbol; TRcal: Tag-to-reader calibration symbol; DR: Divide ratio; LF: Backscatter link frequency; T_{pri}: Link pulse-repetition interval; M: Number of subcarrier cycles per symbol in tag-to-reader direction; R=>T: Reader to tag; T=>R: Tag-to-reader)