Performance evaluation of EPCglobal Gen 2 protocol in wireless channel

Jae-Ryong Cha and Jae-Hyun Kim

School of Electrical and Computer Engineering
Ajou University, Suwon, Korea
E-mail: builder@ ajou.ac.kr and jkim@ajou.ac.kr

Abstract

[1]Radio frequency identification(RFID) system which is a simple form of ubiquitous sensor networks that are used to identify physical objects permits remote, non-line-of-sight, and automatic reading. In RFID system, when a reader sends the ID request command, if there are more than two tags' responses, their responses will collide on the RF communication link, and thus can not be received by the reader. An effective system must avoid this collision by using the anti-collision algorithm.

In this paper, we focus on the performance evaluation of EPCglobal Generation 2 protocol from the viewpoint of the anti-collision algorithm in an erroneous environment.

Introduction

Reliable identification of multiple objects is especially challenging if any objects are present at the same time. Several technologies are available, but they all have limitations. For example, bar code is the most pervasive technology used today, but reading them requires a line of sight between the reader device and the tag, manual, and close-ranging scanning. But Radio frequency identification(RFID) system which is a simple form of ubiquitous sensor networks that are used to identify physical objects permits remote, non-line-of-sight, and automatic reading. Instead of sensing environmental conditions, RFID system identifies the unique tags' ID or detailed information saved in them attached to objects[1],[2].

Passive RFID system generally consists of a reader and many tags. A reader interrogates tags for their ID or detailed information through an RF communication link, and contains internal storage, processing power, and so on. Tags get processing power through RF communication link from the reader and use this energy to power any on-tag computations. A reader in RFID system broadcasts the request message to the tags. Upon receiving the message, all tags send the response back to the reader. If only one tag responds, the reader receives just one response. But if there is more than one tag response, their responses will collide on the RF communication channel, and thus cannot be received by the reader. This generally is referred to as "Tag-collision" problem. An effective system must resolve this problem by using the anti-collision algorithm because the ability to identify many tags simultaneously is crucial for many applications [1]-[4].

In this paper, we focus on the performance evaluation of EPCglobal Generation 2(Gen 2) protocol which is a single global protocol for passive RFID system in ultra high frequency (UHF) band. For the simulator, all node models and process models were newly built based on the Gen 2 protocol. We also propose two scenarios using QueryAdjust command and QueryRep command when the collision occurs.

This paper is organized as follows. In section II we give some general insights on the Gen 2 protocol and in section III we describe our proposed scenarios. In section IV, we describe our OPNET Gen 2 model and section V shows the bit error model of our simulation. In section VI, we present simulation results and conclusions are given in section VII.

EPCglobal Gen 2 Protocol Overview

In this section we give a brief overview of Gen 2 protocol. In Gen 2 protocol, readers manage tag populations using the three operations shown in Figure 1. Each of these operations comprises one or more commands. The operations are defined as follows:

a) **Select :** The process by which a reader selects a tag population for inventory and access. Readers may use one or more Select commands to select a particular tag population prior to inventory.

b) **Inventory :** The process by which a reader identifies tags. A reader begins an inventory round by

[1] This work was supported by the second stage of Brain Korea 21 (BK21) Project in 2006.
This research is partially supported by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Science and Technology(MOST) 21st Century Frontier RD Program in Korea.

transmitting a Query command in one of four sessions. One or more tags may reply. The reader detects a single tag reply and requests the protocol control(PC), electronic product code(EPC), and cyclic redundancy check(CRC-16) from the tag. An inventory round operates in one and only one session at a time.

c) **Access :** The process by which a reader transacts with(reads from or writes to) individual tags. An individual tag must be uniquely identified prior to access. Access comprises multiple commands.

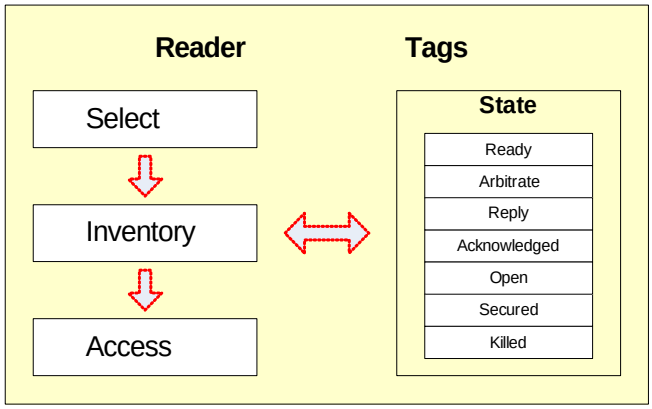

Figure 1. Reader/tag operations and tag state

In this paper, we consider the Selection process and Inventory process for tag identification.

Figure 2 shows the state diagram of Gen 2 protocol for the Inventory process. For tag reading, Gen 2 uses several inventory command sets which include Query, QueryAdjust, QueryRep, ACK, and NAK. The Query command initiates an inventory round and decides which tags should participate in the round (where "inventory round" is defined as the period between successive Query commands). The Query command contains a slot-count parameter Q. Upon receiving a Query command, participating tags pick a random value in the range (0, 2^Q-1) and load this value into their SC (slot counter). Tags that pick a zero shall transition to the Reply state and reply immediately. Tags that pick a nonzero value transition to the Arbitrate state and await a QueryAdjust or QueryRep command. Assuming that a single tag replies, the algorithm proceeds as follows:

a) The tag backscatters a 16 bit random number(RN16) as it enters Reply state.

b) The reader acknowledges the tag with an ACK containing this same RN16.

c) The acknowledged tag transitions to the Acknowledged state, backscattering its PC, EPC, and CRC-16.

d) The reader issues a QueryAdjust or QueryRep, causing the identified tag to invert its inventoried flag (i.e. A→B or B→A) and transition to the Ready state, and potentially causing another tag to initiate a query-response dialog with the reader, starting in step (a), above.

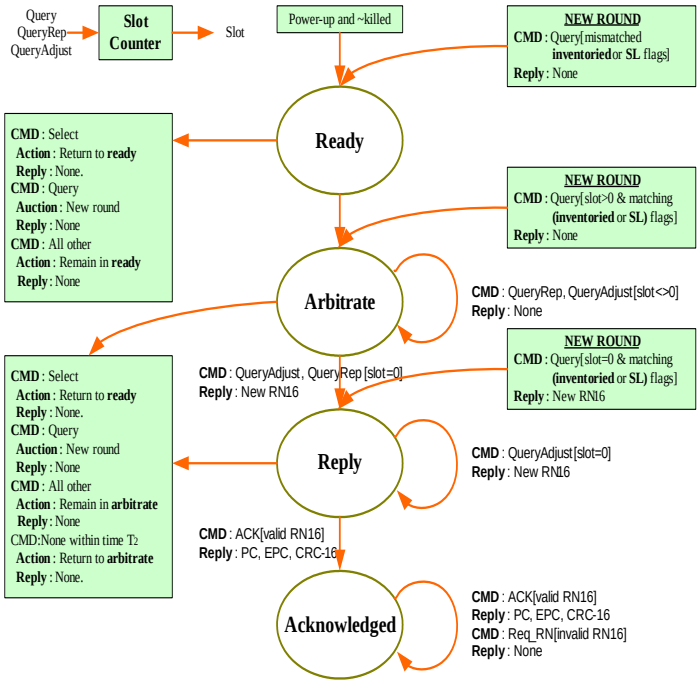

Figure 2. State diagram of Gen 2 protocol

If multiple tags reply in step (a) the reader, by detecting and resolving collisions at the waveform level, can resolve an RN16 from one of the tags, and the reader can ACK the resolved tag. Unresolved tags receive erroneous RN16s and return to the Arbitrate state without backscattering their PC, EPC, and CRC-16. If the reader sends a valid ACK (i.e. an ACK containing the correct RN16) to the tag in the Acknowledged state, the tag shall re-backscatter its PC, EPC, and CRC-16. At any point the reader may issue a NAK, all tags in the inventory round return to arbitrate without changing their inventoried flag.

After issuing a Query command to initiate an inventory round, the reader typically issues one or more QueryAdjust or QueryRep commands. Tags that receive a QueryAdjust command first adjust Q value (increment, decrement, or leave unchanged), then pick a new random value and load this value into their SC. Tags that pick a nonzero value transition to the Arbitrate state and await a QueryAdjust or QueryRep command.

Tags in the Arbitrate state decrement their SC every time they receive a QueryRep, transitioning to the Reply state and backscattering an RN16 when their SC reaches 0000h. Tags whose SC reached 0000h, who replied, and who were not acknowledged shall return to the Arbitrate state with a slot value of 0000h and shall decrement this slot value from 0000h to 7FFFh at the next QueryRep, thereby effectively preventing subsequent replies until the tag loads a new random value into its SC. Tags shall reply at least once in (0, 2^{Q}-1) QueryRep commands[5].

Gen 2 scenarios for the simulation

In this paper, we consider two scenarios shown in Figure 3. The first proposed scenario, TYPE 1, uses the QueryAdjust command when there is the collision after the reader's ID request command. Therefore all the tags receiving the QueryAdjust command from the reader select their new SC. All other procedures comply with the basic rules of Gen 2 protocol. The second scenario, TYPE 2, employs the QueryRep command in which all the tags receiving QueryRep command decrement their SC by 1.

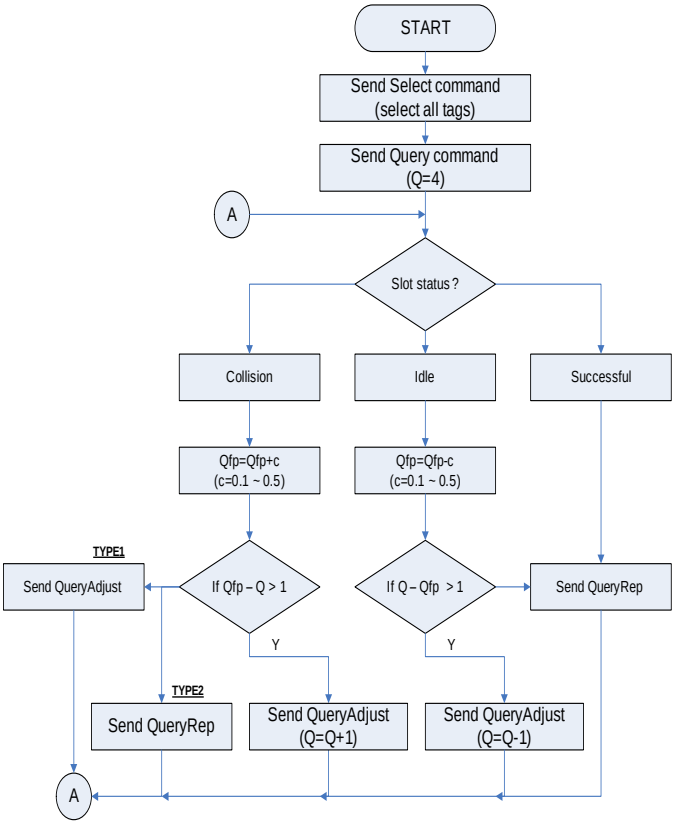

Figure 3. Procedure of the proposed scenarios

We also applied the Q-selection algorithm recommended by Gen 2 protocol. A reader uses for setting the slot-count parameter Q in a Query command. Qfp is a floating-point representation of Q; a reader rounds Qfp to an integer value and uses it as a criterion for determining whether to send the QueryAdjust command. The granularity, c value, used in Q-selection algorithm is 0.5 when 1<Q<5, 0.3 when 6<Q<10, and 0.1 when 11<Q<15[5].

OPNET Gen2 Model

For the simulation, we created all nodes in OPNET simulator and employed system parameters shown in Table 1.

Table 1. Simulation parameters

Gen 2 Parameters	Values
Preamble	87.5 μs
R->T Data Rate	80 KHz
T-> R Data Rate	320KHz
Tari	12.5 μs
RTcal	18.75 μs
TRcal	25 μs

Figure 4 shows the process model of a reader we created. The description of each state is as follows.

INIT state: In this state, the state variables used in the entire process are initialized.

SELECT state : In SELECT state, a reader selects the specific groups of tags and sets the inventory flag based on the criteria.

Query state : In Query state, a reader selects one of the designated groups and starts the inventory round.

IDLE state : In IDLE state, a reader waits an incoming event. The event can be either an incoming packet or the expiration of the clock timer.

REP_IDLE state : A reader moves to REP_IDLE state when the clock timer is expired. In REP_IDLE state, a reader transfers QueryRep commands.

REP_COLL state : A reader moves to REP_COLL state when the collision occurs in the communication link. In REP_COLL state, a reader can transfer either the QueryRep command or QueryAdjust command depending on the criteria.

REP_PROC state : A reader moves to REP_PROC state when the packet arrives without collision. In REP_PROC state, a reader proceeds with the following processes. If the reader receives RN16 from a tag without collision, the reader transfers the ACK command which lets the tag send its EPC. If the reader receives EPC, then it saves EPC before sending the QueryRep command.

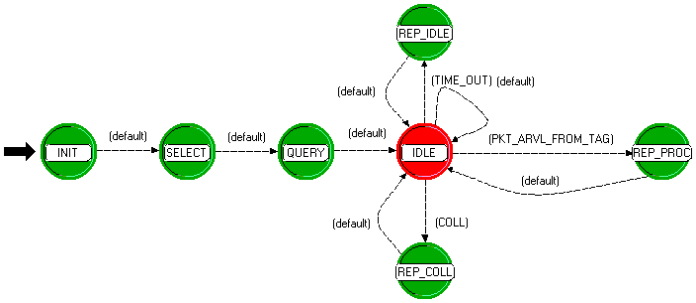

Figure 4. Process model of a reader

Figure 5 shows the process model of a tag we built. The description of each state is as follows.

INIT state: In this state, the state variables used in the entire process are initialized.

IDLE state: In IDLE state, a tag waits for the packets incoming from the reader.

RX state : A tag moves to RX state when the packet arrives from the reader without collision. A tag responds to the reader's commands based on the criteria.

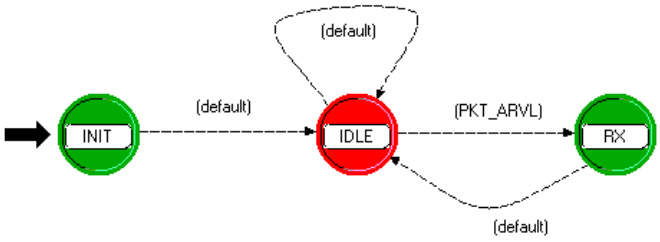

Figure 5. Process model of a tag

Bit-Error Model

In this paper, we employed the basic error model, dbu_error, used in bus link error model in OPNET simulator. Although the communication link between a reader and tags is wireless, we implemented the simulator based on the bus topology because of easy and quick implementation. The algorithm used by dbu_error to compute the number of bit errors in a packet utilizes the Equation (1) which is the probability that exactly k errors correspond to a number of different arrangements of the bits in the packet. After random number generation between 0 and 1, if the random number is less than this probability, yet higher than the probability of occurrence of the previous number of bit errors, then 1 is the number of bits errors allocated to the packet. The algorithm continues iterating in this manner until a value k is found for which the probability of k or fewer errors occurring is greater than the initial random number. Then the number of errors assigned to the packet is k.

$$P_k = \binom{N}{k} \cdot p^k \cdot (1-p)^{N-k}. \quad \textbf{(1)}$$

Where, N and p mean the packet length and the bit error rate(BER) respectively.

Simulation Results

We evaluated the performance of the Gen 2 protocol with an erroneous environment. The number of tags is from 20 to 200 by 20 and all simulations are run 20 times per scenario. The BER in communication link was configured as 0, 10^{-4}, 10^{-3}, and 10^{-2}. The metrics we used include Accuracy, Identification time, and Identification rate as follows.

Accuracy(%) : The ratio of the number of identified tags to the total number of tags

Identification time(ms) : Time taken to identify the total number of tags.

Identification rate(tags/sec): Number of tags identified per second.

Figure 6 shows the accuracy for the number of tags. In Figure 6, T1_3_e2 means that the type of the scenario is Type 1 in which scenario the range of the c value has 3 steps; the c value is 0.5 when 1<Q<5, 0.3 when 6<Q<10, or 0.1 when 11<Q<15, and the BER is 10^{-2}. Irrespective of the scenarios, when using the same BER the accuracy is similar. When the BER is 10^{-2}, the accuracy is about from 37% to 40%. In case that the BER is 10^{-3}, the accuracy is about 89% to 91%. Meanwhile in an environment that the BER is less than 10^{-4}, the accuracy is almost 100%.

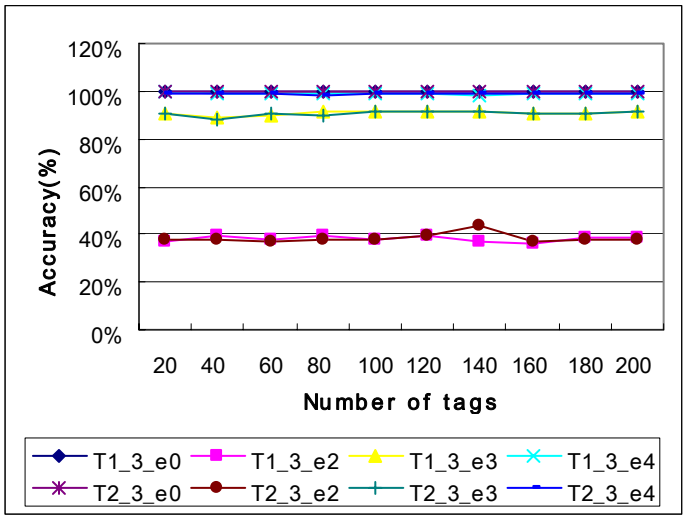

Figure 6. Accuracy vs. number of tags

Figure 7 represents identification time versus the number of tags. Each scenario shows similar performance, when the number of tags is small. However, in case that the number of tags is more than about 80, the performance of T1_3_e2 and T2_3_e2 is getting worse. When the number of tags is 200, T1_3_e2 is almost 2.5 times higher than the result of others(except for T2_3_e2).

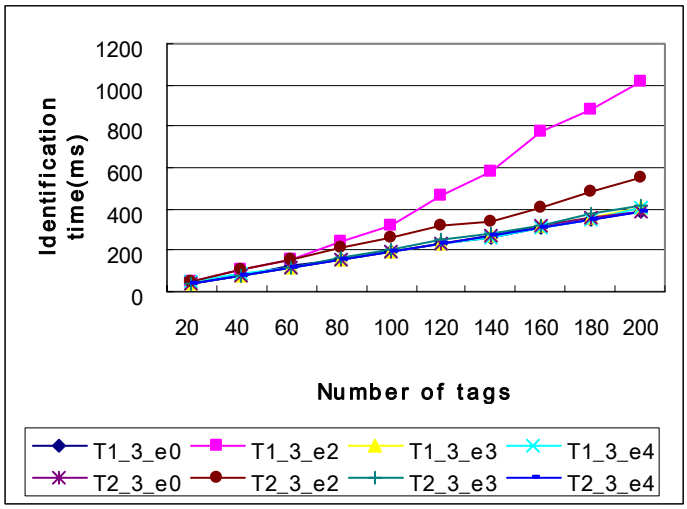

Figure 7. Identification time vs. number of tags

Figure 8 represents the identification rate for the types of scenarios. When the BER is less than 10^{-2}, the identification rate is around 500 tags(max. 535 tags) per second. In case that the BER is10^{-3}, T1_3_e2 identifies 197 tags per second and T2_3_e2 identifies 360 tags per second. It is also shown that regardless of the BER, the performance of Type 2 is always better than Type 1.

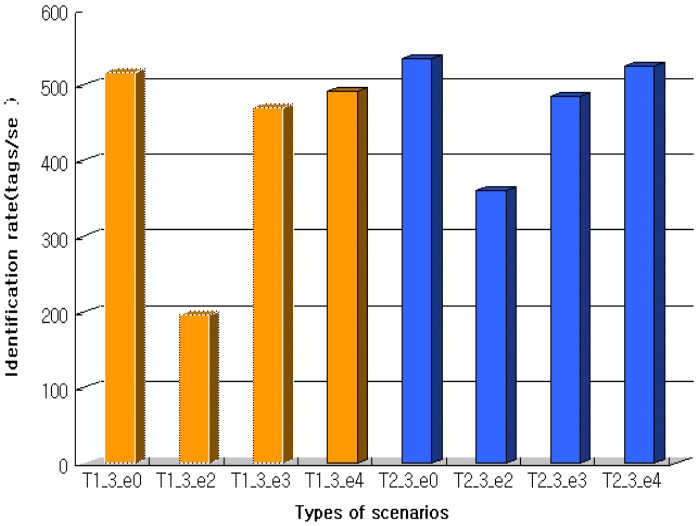

Figure 8. Identification rate vs. types of scenarios

Conclusion

In this paper, we evaluated the performance of Gen 2 protocol with an erroneous environment. The range of the number of tags is from 20 to 200 with incremental value being 20. The BER in communication link was configured as 0, 10^{-4}, 10^{-3}, and 10^{-2}. For the performance evaluation, we used the three metrics; Accuracy, Identification time, and Identification rate. According to the simulation, when the BER is 10^{-2}, the accuracy is about 40 % and when the BER is less than 10^{-3}, the accuracy is more than about 90%. Therefore, for the efficient communication in RFID system, BER less than at least 10^{-3} is needed. And, in an environment that the BER is less than 10^{-3}, a reader can identify about 500 tags per second. Finally, it is shown that regardless of the BER, Type 2 which is the algorithm using QueryRep command is always better than Type 1 which is the algorithm using QueryAdjust command.

References

[1]S. A. Weis, S. E. Sarma, R. L. Rivest, and D. W. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems," in *Proc*. IEEE SPC'03, Mar. 2003.

[2]M. Jaoccurt, A. Ehrsam, and U. Gehrig, "Contact-less Identification Device With Anti-collision Algorithm," in *Proc*. IEEE CSCC'99, Athens, Jul. 1999, pp. 269-273.

[3]K. Finkenzeller, RFID Handbook ; *Fundamentals and applications in Contact-less Smart Cards and Identification, Second Edition*, John Wiley & Sons Ltd, pp. 195-219, 2003.

[4]J. R. Cha and J. H. Kim, "Performance Evaluation of EPCglobal Generation 2 protocol in an RFID system," in *Proc*. KICS'06 fall, Jeju, Jul. 2006, p. 630.

[5]EPCglobal., Epc. *Radio-frequency identity protocols class-1 generation-2 uhf rfid protocol for communications at 860 mhz-960 mhz version 1.0.9.*, 2005.