ĐÁNH GIÁ HIỆU SUẤT CỦA CÁC GIAO THỨC CHỐNG XUNG ĐỘT KHÔNG BỘ NHỚ DỰA TRÊN CÂY TRUY VẤN TRONG HỆ THỐNG RFID

Nguyễn Đức Nhật Quang[1*], Phạm Thị Thúy Sang[2], Lê Văn Hòa[3], Võ Viết Minh Nhật[4]

[1] Khoa Điện, Điện tử và Công nghệ vật liệu, Trường Đại học Khoa học, Đại học Huế

[2] Trung tâm Công nghệ thông tin tỉnh Thừa Thiên Huế

[3] Trường Du lịch, Đại học Huế

[4] Viện Khảo thí và Bảo đảm chất lượng giáo dục, Đại học Huế

*Email: ndnquang@hueuni.edu.vn

Ngày nhận bài: 8/01/2024; ngày hoàn thành phản biện: 8/02/2024; ngày duyệt đăng: 5/3/2024

TÓM TẮT

Nhận dạng bằng tần số vô tuyến (Radio Frequency IDentification - RFID) được sử dụng rộng rãi trong nhiều lĩnh vực, như giám sát công nghiệp, quản lý chuỗi cung ứng, theo dõi dòng nguyên liệu, v.v. Trong một hệ thống RFID thụ động, xung đột thẻ là nguyên nhân chính ảnh hưởng đến hiệu suất của toàn hệ thống. Xung đột thẻ xảy ra khi có nhiều thẻ phản hồi cùng lúc đến đầu đọc. Điều này khiến đầu đọc không thể nhận dạng được thẻ. Bài báo phân tích và đánh giá các giao thức chống xung đột không bộ nhớ dựa trên cây truy vấn (Query Tree - QT). Kết quả mô phỏng cho thấy giao thức QT và các biến thể của nó khá hiệu quả trong nhận dạng thẻ trong đó một số biến thể QT có ưu điểm vượt trội. Kết quả phân tích dựa trên các tiêu chí đánh giá hiệu năng khác nhau cũng cho thấy rằng, để nâng cao hiệu suất của một tiêu chí thì phải chịu thiệt hại đối với một số tiêu chí khác. Một giải pháp thỏa hiệp do đó cần được xem xét.

Từ khóa: Nhận dạng bằng tần số vô tuyến, chống xung đột, cây truy vấn, không bộ nhớ, đánh giá hiệu năng

1. MỞ ĐẦU

Công nghệ RFID đóng một vai trò quan trọng trong Internet of Things (IoT) [1], vì các đối tượng được gắn thẻ RFID có thể được nhận dạng duy nhất. Một hệ thống RFID bao gồm các đầu đọc (reader), thẻ (tag) và máy chủ (host). Khi khoảng cách giữa thẻ và đầu đọc đủ gần, tức nằm trong vùng truy vấn của đầu đọc, chúng có thể giao tiếp với nhau. Đầu đọc bắt đầu một thủ tục truy vấn (hoặc nhận dạng) bằng cách gửi một lệnh (command) truy vấn yêu cầu các thẻ phản hồi. Một thẻ có thể thuộc loại chủ động

(active), nếu nó có nguồn cung cấp năng lượng riêng, hoặc bị động (passive), nếu nó không có nguồn năng lượng nào. Đối với thẻ thụ động, nó sẽ tận dụng năng lượng từ xung truy vấn của đầu đọc để gửi phản hồi ID của nó. Máy chủ lưu lại kết quả truy vấn thẻ cho các xử lý sau này.

Do các thẻ hoạt động độc lập và không có kênh trao đổi thông tin nào khác giữa chúng, chúng có thể phản hồi yêu cầu của một đầu đọc đồng thời, kết quả là xung đột tín hiệu sẽ xảy ra và đầu đọc không thể nhận dạng được thẻ nào. Làm thế nào để giảm xung đột tín hiệu đọc thẻ và tăng tốc quy trình truy vấn do đó là rất quan trọng. Bài báo này sẽ trình bày và phân tích các giải pháp chống xung đột tín hiệu đọc thẻ.

Đã có một số giao thức chống xung đột được đề xuất để giảm xung đột tín hiệu thẻ, mà có thể được phân loại thành hai loại: giao thức dựa trên ALOHA và giao thức dựa trên cây (tree). Nguyên tắc của giao thức dựa trên ALOHA [2] là, khi nhận được yêu cầu truy vấn của đầu đọc, mỗi thẻ chọn một cách độc lập thời gian trễ (back-off time) ngẫu nhiên để phản hồi ID của nó cho đầu đọc. Nếu không có xung đột xảy ra trong quá trình phản hồi, ID của thẻ được xem được nhận dạng thành công và được đầu đọc xác nhận. Thẻ có ID được nhận dạng sẽ ngừng phản hồi với đầu đọc. Một thẻ chưa được nhận dạng sẽ liên tục chọn thời gian chờ ngẫu nhiên và phản hồi bằng ID của nó cho đến khi nó được nhận dạng và được xác nhận bởi đầu đọc. Trong các giao thức dựa trên cây [3], một nhóm thẻ được chia tách thành các nhóm con theo yêu cầu của đầu đọc và/hoặc thực trạng xung đột tín hiệu thẻ. Việc chia tách sẽ tiếp tục cho đến khi chỉ có một thẻ trong nhóm con được nhận dạng thành công.

Bài báo này tập trung vào giao thức QT [4], là một giao thức đơn giản (plain) và không bộ nhớ (memoryless). Không bộ nhớ vì thẻ không sử dụng bộ nhớ để lưu trạng thái giao thức; Đơn giản vì QT không sử dụng các kỹ thuật đặc biệt, như theo dõi bit (bit-tracking) [5], sửa ID (ID-revising) [6] và nhận dạng lại (re-identification). Dựa trên giao thức QT, các giao thức biến thể của giao thức QT được phân tích và đánh giá. Các thử nghiệm mô phỏng và so sánh với QT với các biến thể của nó cũng được tiến hành. Các tiêu chí đánh giá hiệu năng (metrics) là dựa trên số chu kỳ xung đột (collision), số chu kỳ nhàn rỗi (idle), độ trễ (delay) nhận dạng và chi phí truyền thông để nhận dạng thành công toàn bộ thẻ. Kết quả cho thấy rằng QT và các biến thể của nó là khá hiệu quả trong việc truy vấn thẻ, trong đó các biến thể của QT vượt trội hơn QT về một số tiêu chí hiệu năng. Tuy nhiên, để vượt trội được của một vài tiêu chí này, các biến thể của QT đều phải chịu thiệt hại hiệu năng của một số tiêu chí khác. Giải pháp thỏa hiệp do đó cần được tính đến.

Phần còn lại của bài báo được tổ chức như sau. Giao thức QT được mô tả trong Phần II. Một số biến thể của giao thức QT được phân tích trong Phần III. Mô phỏng và so sánh hiệu năng giữa giao thức QT và các biến thể của nó trong Phần IV. Cuối cùng, kết luận được rút ra trong Phần V.

2. GIAO THỨC CÂY TRUY VẤN

Giao thức cây truy vấn hay giao thức QT là một giao thức không bộ nhớ, được đánh giá khá hiệu quả trong hệ thống RFID. Trong giao thức này, đầu đọc gửi một truy vấn đến các thẻ và thẻ nào có tiền tố ID phù hợp sẽ trả lời ID của chúng [4]. Phản hồi từ thẻ phụ thuộc trực tiếp vào truy vấn hiện tại và bỏ qua lịch sử giao tiếp trước đó, nên được gọi là không bộ nhớ. Các thẻ sử dụng trong giao thức QT chỉ yêu cầu phần cứng đơn giản vì chúng chỉ so sánh truy vấn của đầu đọc với ID của chính chúng và phản hồi nếu trùng khớp. Thuật toán 1 mô tả cách thức giao thức QT hoạt động.

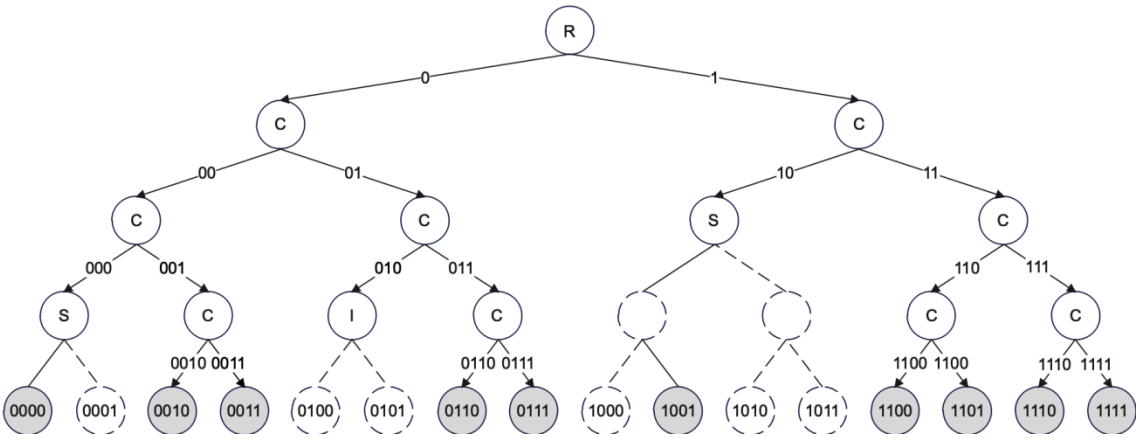
Thuật toán 1. QT

```
Input: Các thẻ trong vùng đọc của đầu đọc;
Output: ID_list;                    // danh sách ID các thẻ đọc thành công
        idle, success, collision;   // các biến đếm số vòng nhàn rỗi (idle),
                                    // thành công (success) và xung đột (collision)
1.  idle ← 0; success ← 0; collision ← 0;
2.  ID_list ← ∅;
3.  L ← ∅;                          // danh sách các thẻ trả lời tại mỗi vòng truy vấn
4.  Push(Q,1); Push(Q,0);           // initialize Q
5.  while (Q != ∅)
6.      q ← Pop(Q);                 // Pop up chuỗi truy vấn q từ ngăn xếp S
7.      L ← Broadcast(q);           // Broardcast prefix q đến các thẻ;
                                    // Kết quả là danh sách thẻ có ID prefix khớp với q
8.      if |L| == 0 then            // Không có thẻ nào trả lời (idle)

9.          idle ← idle + 1;
10.     elseif |L| == 1 then        // Chỉ một thẻ trả lời (success)
11.         success ← success + 1;
12.         ID_list.append(q+L[0]); // thêm thẻ đọc thành công (q+L[0]) vào ID_list
13.     else
14.         collision ← collision + 1;
15.         Push(Q, q1);            // tách nhánh bằng cách push chuỗi truy vấn q1
            Push(Q, q0);            và q0 vào ngăn xếp S
16.     end
17.   end
18. end
```

Quá trình nhận dạng thẻ của giao thức QT gồm nhiều vòng lặp. Trong mỗi vòng, đầu đọc gửi một truy vấn và các thẻ có tiền tố ID khớp với truy vấn hiện tại sẽ phản hồi ID của chúng ở dạng nhị phân. Nếu xảy ra xung đột, đầu đọc tạo ra hai truy vấn mới bằng cách nối thêm truy vấn (tiền tố) q với một bit nhị phân 0 hoặc 1. Các truy vấn mới này được đặt vào ngăn xếp LIFO (Last In, First Out). Trường hợp này tạo ra khe (slot) xung đột. Nếu không có phản hồi nào, đầu đọc biết rằng không có thẻ nào có tiền tố như

yêu cầu và truy vấn đó bị hủy. Trường hợp này tạo ra khe nhàn rỗi. Nếu chỉ có một thẻ phản hồi, thẻ đó được nhận dạng và trường hợp này tạo ra khe thành công. Bằng cách mở rộng tiền tố truy vấn cho đến khi chỉ còn một thẻ có ID khớp, thuật toán có thể xác định các thẻ còn lại. Quá trình nhận dạng được hoàn tất khi ngăn xếp LIFO rỗng. Hình 1 minh họa hoạt động giao thức QT khi đọc 10 thẻ. Với độ dài ID thẻ là k = 4 bit. Tổng cộng, đầu đọc sử dụng 21 vòng để đọc 10 thẻ trong hình.



Hình 1. Ví dụ mô tả hoạt động của giao thức QT, trong đó các hình tròn xám tương ứng với các thẻ có trong vùng truy vấn, hình tròn với chữ S, C và I lần lượt là trường hợp truy vấn thành công, xung đột và nhàn rỗi.

Giao thức QT có ưu điểm là đơn giản và giúp đầu đọc nhận dạng được hết toàn bộ các thẻ trong vùng truy vấn. Tuy nhiên, nhược điểm chính của QT là số vòng xung đột và nhàn rỗi cao, và cần thời gian dài để truy vấn hết các thẻ.

3. CÁC BIẾN THỂ CỦA GIAO THỨC QT

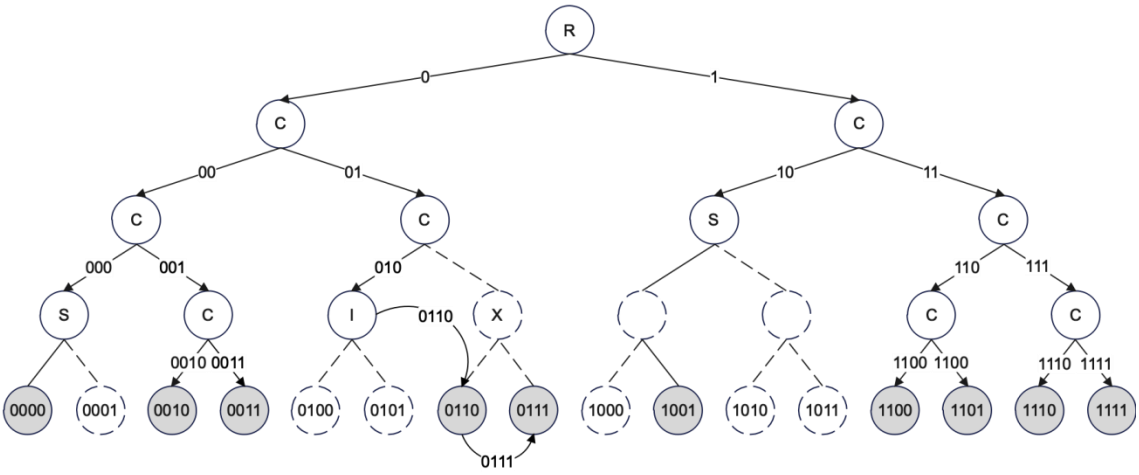
Để khắc phục nhược điểm của QT, một số biến thể của nó được đề xuất sau đây.

3.1. Giao thức Query Tree with Shortcutting (QTsc)

Giao thức QTsc [4] là một mở rộng của QT nhằm làm giảm số lượng truy vấn, tránh các truy vấn tạo ra xung đột và loại bỏ các truy vấn dư thừa. Ý tưởng lối tắt (shortcuting) của QTsc dựa trên suy luận sau: nếu chuỗi truy vấn "q" tạo ra xung đột và chuỗi "q0" dẫn đến không có phản hồi (idle) thì QTsc sẽ bỏ qua chuỗi "q1" vì chắc chắn sẽ tạo ra xung đột. Hai chuỗi truy vấn "q10" và "q11" sẽ được đưa vào ngăn xếp.

Quy trình nhận dạng tiến hành từng vòng lặp truy vấn cho đến khi ngăn xếp rỗng. Khi đó, tất cả các thẻ được nhận dạng thành công và quy trình nhận dạng của QTsc kết thúc. Với ví dụ như Hình 1, QTsc sẽ bỏ qua truy vấn với tiền tố "011" vì chắc chắn sẽ tạo ra xung đột và truy vấn trực tiếp với "0110" và "0111" như trong Hình 2.

Thuật toán 2 mô tả điểm cải tiến của QTsc so với QT (Thuật toán 1). QTsc bổ sung điều kiện khi chuỗi "q" tạo ra xung đột và chuỗi "q0" dẫn đến nhàn rỗi (từ dòng 08 đến 13). Trong trường hợp này, "q" sẽ được gán bằng giá trị tiếp theo trong hàng đợi Q (được thực hiện bởi hàm Pop(Q)), và sau đó, hai giá trị mới, "1" và "0", sẽ được thêm vào cuối hàng đợi Q (được thực hiện bởi hai hàm Push(Q,1) và Push(Q,0)).



Hình 2. Ví dụ mô tả hoạt động của giao thức QTsc.

Thuật toán 2. QTsc
Input: Các thẻ trong vùng đọc của đầu đọc;
Output: ID_list; // danh sách ID các thẻ đọc thành công
idle, success, collision; // các biến đếm số vòng nhàn rỗi (idle),
// thành công (success) và xung đột (collision)

```
1.  idle ← 0; success ← 0; collision ← 0;
2.  ID_list ← ∅;
3.  L ← ∅;                          // danh sách các thẻ trả lời tại mỗi vòng truy vấn
4.  Push(Q,1); Push(Q,0);           // initialize Q
5.  while (Q != ∅)
6.      q ← Pop(Q);                 // Pop up chuỗi truy vấn q từ ngăn xếp S
7.      L ← Broadcast(q);           // Broardcast prefix q đến các thẻ;
                                    // Kết quả là danh sách thẻ có ID prefix khớp với q
8.      if |L| == 0 then            // Không có thẻ nào trả lời (idle)
9.          idle ← idle + 1;
10.         if last_collision and q == last_q0 then // Khi q tạo ra xung đột và q0 tạo ra
            idle
11.             q ← Pop(Q);         // bỏ qua last_q1
12.             Push(Q,1); Push(Q,0); // nhảy đến q1 và q0
13.         end
14.     elseif |L| == 1 then        // Chỉ một thẻ trả lời (success)
15.         success ← success + 1;
```

```
16.         ID_list.append(q+L[0]);   // thêm thẻ đọc thành công (q+L[0]) vào ID_list
17.       else
18.         collision ← collision + 1;
19.         Push(Q, q1);              // tách nhánh bằng cách push chuỗi truy vấn q1
            Push(Q, q0);              và q0 vào ngăn xếp S
20.       end
21.     end
22. end
```

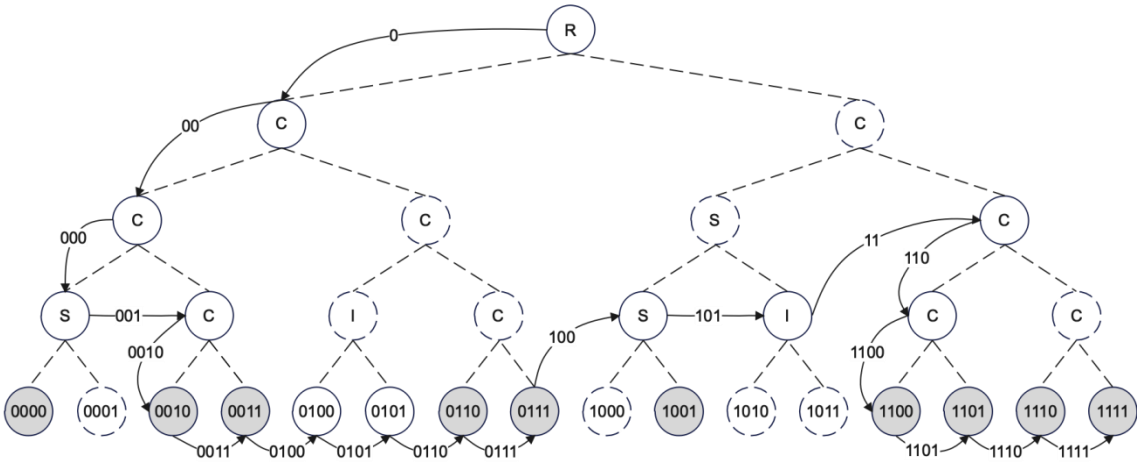
Giao thức QTsc có ưu điểm là đơn giản, đầu đọc nhận dạng hết các thẻ trong vùng truy vấn và bỏ qua được các truy vấn chắn chắn dẫn đến xung đột. Tuy nhiên QTsc vẫn còn những nhược điểm như: số vòng nhàn rỗi cao, thời gian truy vấn dài.

3.2. Giao thức Improve Query Tree (IQT)

Giao thức IQT là một phiên bản cải tiến của QT đã được đề cập ở phần II. Điểm khác biệt của nó so với QT là khi đầu đọc phát hiện một bit xung đột, nó sẽ gửi một tín hiệu đến tất cả thẻ để yêu cầu dừng việc phản hồi [7]. Điều này dẫn đến số bit phản hồi từ thẻ giảm. Ưu điểm của IQT là đơn giản; đầu đọc nhận dạng hết toàn bộ thẻ trong vùng truy vấn với số bit gửi đi giảm đáng kể nếu xung đột xảy ra sớm tại các bit đầu của chuỗi phản hồi. Tuy nhiên, nhược điểm của IQT là số vòng xung đột và nhàn rỗi cao; đồng thời cần thời gian dài để truy vấn hết các thẻ.

3.3. Giao thức Smart Trend Traversal (STT)

Giao thức STT là một phiên bản cải tiến của QT. STT giải quyết xung đột gây ra bởi các thẻ có mã định danh nhị phân được phân phối không đồng đều. Khác với đa số các biến thể của QT, STT thực hiện truy vấn theo chiều rộng, thay vì theo chiều sâu [8][9]. Khi một truy vấn tại một nút cho kết quả thành công, điều đó cho thấy giao thức nằm ở mức phù hợp, giao thức di chuyển theo chiều ngang và lấy chuỗi nhị phân ở nút tiếp theo theo thứ tự chiều rộng làm chuỗi truy vấn tiếp theo. Khi kết quả truy vấn ở trạng thái nhàn rỗi, điều đó cho thấy rằng giao thức được đặt ở mức quá thấp và nếu nút hiện tại là nút con bên phải thì giao thức sẽ di chuyển lên dọc theo cây. Ngược lại, nếu truy vấn dẫn đến xung đột thì giao thức ở mức quá cao và sẽ di chuyển xuống dọc theo cây. Trong nhiều trường hợp, STT di chuyển theo cấp độ chính xác của cây truy vấn nhị phân để giải quyết vấn đề xung đột.



Hình 3. Ví dụ mô tả hoạt động của giao thức STT.

Hình 3 mô tả một ví dụ minh họa về hoạt động của STT với tập thẻ cần truy vấn như Hình 1, trong đó các mũi tên cho thấy các bước tuần tự trong quá trình truy vấn. Với việc bỏ qua các vòng truy vấn xung đột như "01", "011", "1", "111", STT sử dụng 19 vòng truy vấn để đọc 10 thẻ, tránh được 2 vòng không cần thiết so với QT.

Tùy thuộc vào mật độ và phân bố ID thẻ, trong nhiều trường hợp, STT di chuyển ở các cấp độ chính xác của cây truy vấn để tránh xung đột thẻ, nhiều truy vấn dẫn đến xung đột có thể được bỏ qua, do đó STT có thể đạt được hiệu suất cao hơn. Tuy nhiên, khi xung đột xảy ra trong STT, cây truy vấn vẫn sử dụng phương pháp như QT để giải quyết xung đột.

4. ĐÁNH GIÁ HIỆU SUẤT CÁC BIẾN THỂ GIAO THỨC QUERY TREE

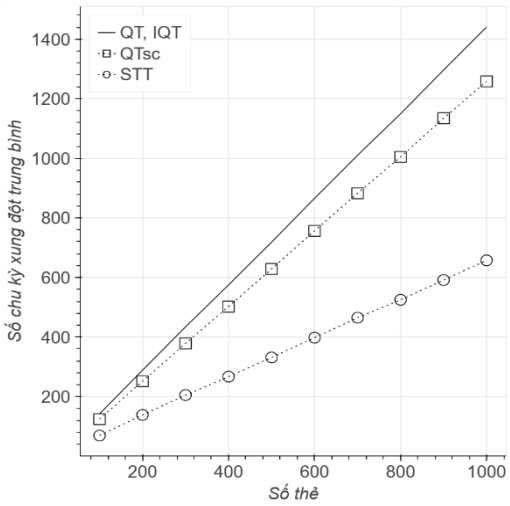
Trong phần này, chúng tôi tiến hành đánh giá hiệu suất của QT với các biến thể QTsc, IQT và STT dựa trên mô phỏng bằng Python 3.11.4. Để đo lường hiệu suất nhận dạng thẻ trong các giao thức dựa trên cây, chúng tôi xem xét các tiêu chí sau:

- ***Số chu kỳ xung đột***: Xung đột làm trì hoãn việc nhận dạng và tăng mức tiêu thụ năng lượng của thẻ.
- ***Số chu kỳ nhàn rỗi***: Nhàn rỗi là một yếu tố gây ra độ trễ nhận dạng.
- ***Chi phí truyền thông thẻ***: Số bit trung bình được truyền bởi các thẻ trong quá trình truy vấn. Điều này ảnh hưởng đến lượng điện năng tiêu thụ. Do thẻ không sử dụng nguồn nuôi trong nên yếu tố này phải ở mức thấp.

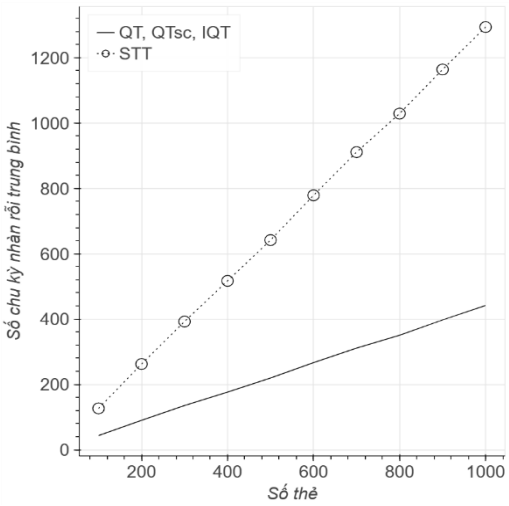
Chúng tôi mô phỏng một hệ thống RFID trong đó một đầu đọc giao tiếp với n = 100, 200, 300,..., 1000 thẻ nằm trong vùng đọc của nó. Các thẻ có độ dài ID bằng k = 96 bit theo tiêu chuẩn của EPCglobal Network [10] và phân phối đồng đều. Chúng tôi thực

hiện 100 lần mô phỏng cho mỗi trường hợp số thẻ nêu trên (n) và sử dụng kết quả trung bình để đánh giá hiệu suất.

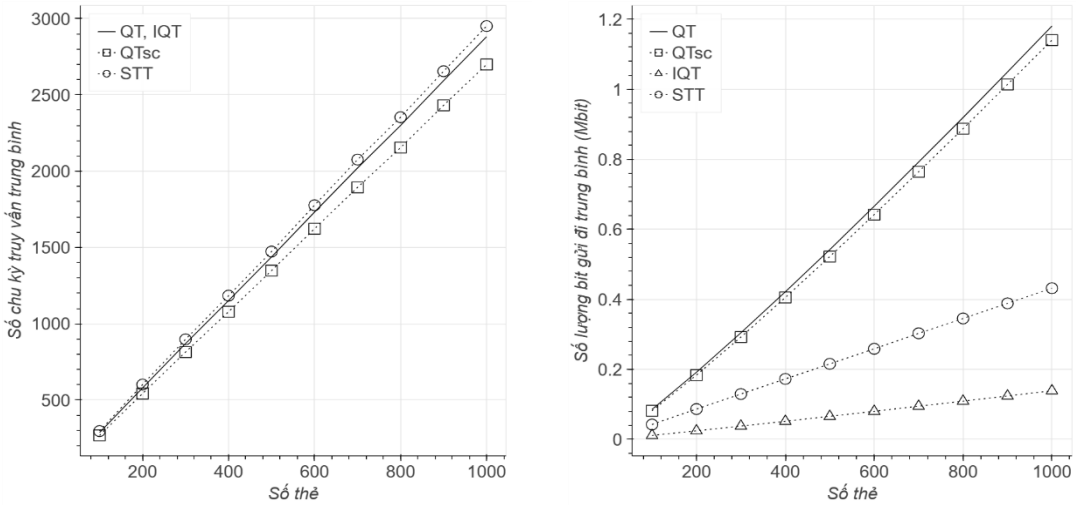
Hình 4 thể hiện kết quả mô phỏng thu được bằng cách thay đổi số thẻ trong vùng truy vấn sau 100 lần thử nghiệm. Nhìn chung, IQT có cùng phương pháp giải quyết các truy vấn xung đột và nhàn rỗi tương tự với QT nên đồ thị của 2 giao thức này ở Hình 4(a)(b)(c) giống hệt nhau. Hình 4(a) cho thấy xung đột đọc thẻ xảy ra thường xuyên hơn khi số lượng thẻ trong vùng truy vấn tăng lên đối với tất cả giao thức, đặc biệt với QT và IQT. QTsc thể hiện rõ ưu điểm của giao thức này khi tránh được các truy vấn chắc chắn dẫn đến xung đột bằng phương pháp lối tắt. Tính toán dựa vào kết quả mô phỏng cho thấy QTsc giảm được khoảng 15% số lượng truy vấn xung đột so với QT. Trong khi đó, nhờ vào phương pháp truy vấn theo chiều rộng thay vì theo chiều sâu như các giao thức còn lại, STT giảm được số lượng lớn truy vấn xung đột ở các nút cha (khoảng 45% so với QT). Tuy nhiên, STT lại bộc lộ nhược điểm khi truy vấn các thẻ có ID phân bố đồng đều. Lúc này, chính vì phương pháp truy vấn theo chiều rộng nên STT tăng đáng kể số lượng truy vấn nhàn rỗi ở các nút lá (khoảng 290% so với QT) như trong Hình 4(b). Hình 4(c) cho thấy QTsc có hiệu suất tốt nhất đối với các thẻ phân bố đồng đều. Ngược lại, STT không phù hợp với kiểu phân bố ID thẻ này. Mặc dù số lượng truy vấn nhàn rỗi ở STT rất lớn nhưng không có thẻ nào phản hồi, kết hợp với việc tránh được số lượng lớn truy vấn xung đột ở các nút cha nên STT tránh được các phản hồi của số lượng lớn các thẻ xung đột. Điều này đồng nghĩa với việc số lượng bit gửi đi ở đầu đọc và phản hồi từ thẻ được giảm đáng kể (giảm 3 lần so với QT). Tuy không sử dụng các cơ chế tránh xung đột như QTsc và STT, giao thức IQT vẫn có thể giảm số lượng bit truyền bằng cách gửi một tín hiệu đến tất cả thẻ để yêu cầu dừng việc phản hồi ngay sau khi phát hiện một bit xung đột. Hình 4(d) cho thấy số lượng bit truyền của giao thức IQT được giảm rõ rệt đối với số lượng thẻ khác nhau. Điều này dẫn đến hiệu quả tiết kiệm năng lượng được cải thiện.



(a) số chu kỳ xung đột trung bình



(b) số chu kỳ nhàn rỗi trung bình



(c) số chu kỳ truy vấn trung bình (d) số lượng bit gửi đi trung bình

Hình 4. So sánh hiệu suất của các giao thức QT, QTsc, IQT và STT với số lượng thẻ thay đổi.

5. KẾT LUẬN

Bài báo này đã trình bày và phân tích của các thuật toán chống xung đột tín hiệu đọc thẻ không bộ nhớ trong hệ thống RFID thụ động vì chúng ít phức tạp hơn khi triển khai thẻ và đầu đọc. Bài báo đã tập trung vào giao thức cây truy vấn (QT) và các biến thể của nó. Thông qua việc mô phỏng và so sánh hiệu năng giữa giao thức QT và các biến thể của nó trên các thẻ có ID phân bố đồng đều, hiệu năng đã được đánh giá dựa trên số chu kỳ xung đột, số chu kỳ nhàn rỗi và chi phí truyền thông để nhận dạng thành công toàn bộ thẻ trong vùng truy vấn.

Kết quả cho thấy rằng sự phân bố ID, độ dài và mật độ của thẻ là những yếu tố ảnh hưởng đến hiệu suất của hệ thống vì đây là những thông tin ánh xạ gốc của cây cần được giải quyết bằng thuật toán. Điều này mở ra hướng nghiên cứu mới về việc tối ưu hóa hiệu suất của hệ thống RFID, đặc biệt là trong việc giảm xung đột đọc thẻ và tăng tốc độ truy vấn. Tuy nhiên, cần thực hiện thêm nghiên cứu để xác định hiệu quả của các giải pháp này trong các môi trường thực tế và với các loại thẻ khác nhau.

TÀI LIỆU THAM KHẢO

[1] RFID-Journal, "That Internet of Things," 2017. Available online: http://www.rfidjournal.com/articles/.

[2] F. Schoute (2010). "Dynamic Frame Length Aloha," IEEE Transactions on Communications, vol. 31, pp. 565-568.

[3] G. Peeters and B. Houdt (2010). "Interference Cancellation Tree Algorithms with k-Signal Memory Locations," IEEE Transactions on Communications, vol. 58, pp. 3056-3061.

[4] C. Law, K. Lee, and K. Siu (2000). "Efficient memoryless protocol for tag identification," in Proceedings of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, Boston, MA, USA.

[5] W. Zhu, J. Gong, M. Li, J. Cao, Z. He and R. Xie (2023). "Multiple Resolution Bit Tracking for Continuous Reliable RFID Tag Identification," in IEEE Transactions on Mobile Computing, vol. 22, no. 10, pp. 6071-6085.

[6] W. Zhu, C. Huang and C. Ma (2022). "Analytic Hierarchy Process Based Compatibility Measurement for RFID Protocols," 2022 18th International Conference on Mobility, Sensing and Networking (MSN), Guangzhou, China, pp. 708-712.

[7] Zhou, Feng & Chen, Chunhong & Jin, Dawei & Huang, Chenling & Min, Hao (2004). "Evaluating and Optimizing Power Consumption of Anti-Collision Protocols for Applications in RFID Systems," Energy Procedia.

[8] L. Pan and H. Wu (2011). "Smart Trend-Traversal Protocol for RFID Tag Arbitration," IEEE Transactions on Wireless Communications, vol. 10, pp. 3565–3569.

[9] X. Wang, J. Liu, Y. Wang, X. Chen and L. Chen (2021). "Efficient Tag Grouping via Collision Reconciliation and Data Compression," in IEEE Transactions on Mobile Computing, vol. 20, no. 5, pp. 1817-1831.

[10] K. Finkenzeller (2003), "RFID Handbook: Fundamentals and Applications in Contactless Smart Cards and Identification," John Wiley & Sons.

PERFORMANCE EVALUATION OF QUERY TREE-BASED MEMORYLESS ANTI-COLLISION PROTOCOLS IN RFID SYSTEM

Nguyen Duc Nhat Quang[1*], Pham Thi Thuy Sang[2], Le Van Hoa[3], Vo Viet Minh Nhat[4]

[1] Faculty of Electronics, Electrical Engineering and Material Technology, University of Sciences, Hue University

[2] Hue Center of Information and Technology

[3] School of Hospitality and Tourism, Hue University

[4] Institute for Educational Testing and Quality Assurance, Hue University

*Email: ndnquang@hueuni.edu.vn

ABSTRACT

Radio Frequency IDentification (RFID) is widely used in many fields, such as industrial surveillance, supply chain management, material flow tracking, etc. Tag collisions are the primary factor influencing the overall performance of a passive RFID system. When multiple tags respond to the reader's queries simultaneously, this is known as a tag collision. As a result, the reader is unable to identify the card. This paper analyzes and evaluates memoryless collision prevention protocols based on Query Tree (QT). The simulation's findings demonstrate the effectiveness of the QT protocol and its variants in tag identification, where some QT variants offer particularly significant advantages. Results from analyses based on various performance evaluation criteria also indicate that harm to one criterion must occur in order to increase the performance of another. Thus, a compromise solution must be taken into account.

Keywords: Radio Frequency IDentification, anti-collision, query tree, memoryless, performance evaluation

Nguyễn Đức Nhật Quang sinh ngày 08/10/1992 tại Thừa Thiên Huế. Năm 2015, ông tốt nghiệp kỹ sư chuyên ngành Điện tử - Viễn thông, Trường Đại học Khoa học, Đại học Huế. Năm 2020, ông nhận bằng thạc sĩ chuyên ngành Khoa học máy tính và Kỹ thuật thông tin (CSIE) tại Trường Đại học Quốc gia Thành Công (NCKU), Đài Loan. Hiện nay, ông đang công tác tại Khoa Điện, Điện tử và Công nghệ vật liệu, Trường Đại học Khoa học, Đại học Huế.

Lĩnh vực nghiên cứu: Thiết kế vi mạch số, Trí thông minh nhân tạo (AI), Internet vạn vật kết nối (IoT), Hệ thống nhúng.

Phạm Thị Thúy Sang sinh ngày 23/3/1998 tại Quảng Trị. Năm 2020, bà tốt nghiệp cử nhân chuyên ngành Công nghệ thông tin, Trường Đại học Khoa học, Đại học Huế. Hiện nay, bà đang công tác tại Trung tâm Công nghệ Thông tin tỉnh Thừa Thiên Huế (Hue CIT).

Lĩnh vực nghiên cứu: Công nghệ phần mềm.

Lê Văn Hòa sinh ngày 30/7/1985 tại Huế. Ông nhận bằng Tiến sĩ Khoa học máy tính năm 2020 tại Trường Đại học Khoa học, Đại học Huế. Hiện công tác tại Khoa Quản lý sự kiện và công nghệ truyền thông, Trường Du lịch – Đại học Huế.

Lĩnh vực nghiên cứu: Mạng máy tính, Internet vạn vật kết nối (IoT), Mạng chuyển mạch chùm quang, Đa dạng chất lượng dịch vụ (QoS), du lịch thông minh, công nghệ RFID.

Võ Viết Minh Nhật nhận bằng Tiến sĩ về Công nghệ thông tin (Tin học nhận thức) tại Đại học Quebec ở Montreal, Canada vào năm 2007. Ông được phong phó giáo sư vào năm 2016 tại Đại học Huế, Việt Nam.

Lĩnh vực nghiên cứu: mạng chuyển mạch gói/chùm quang, hệ thống cảm biến/RFID không dây, thông minh bầy đàn và tính toán tiến hóa.