نشریه علمی "پدافند الکترونیکی و سایبری"
سال هشتم، شماره ۴، زمستان ۱۳۹۹، ص ۱۲۸-۱۱۹

علمی- پژوهشی

ارائه یک پروتکل ضدتصادم در سامانه‌های RFID

بهرام زینلی[۱]، حسین بهرامگیری[۲*]
۱- کارشناس ارشد و ۲- استادیار، دانشگاه صنعتی مالک اشتر

(دریافت: ۱۳۹۸/۱۰/۰۶، پذیرش: ۱۳۹۹/۰۵/۱۵)

چکیده

امکان دست‌یابی آسان و سریع به اطلاعات، با استفاده از سامانه‌های RFID (Radio Frequency IDentification)، سبب توسعه روزافزون این فناوری، در کاربردهای مختلف از جمله کاربردهای نظامی گردیده است. این فناوری شرایط را برای دستیابی سریع به آگاهی وضعیتی در یک محیط عملیاتی یا پشتیبانی فراهم می‌آورد؛ ولی از سوی دیگر با حضور عوامل مختلف و متعدد در محیط، با چالش تصادم در فرایند شناسایی همراه است. تصادم برچسب‌ها، در سامانه‌های RFID اجتناب‌ناپذیر است و بر عملکرد سامانه‌های RFID به‌ویژه در محیط‌هایی با تعداد زیادی برچسب، تأثیر فراوانی دارد. در این مقاله، یک پروتکل ضدتصادم که ترکیبی از روش‌های ALOHA و درختی است، ارائه می‌شود. در پروتکل ارائه‌شده هر قاب و شکاف زمانی، دارای پیشوندهایی هستند و در هر شکاف زمانی، برچسب‌هایی که شناسه آن‌ها با پیشوندهای قاب و شکاف زمانی مطابقت داشته باشد، پاسخ خود را برای برچسب‌خوان ارسال می‌کنند. در این پروتکل با استفاده از روش کدگذاری منچستر، موقعیت بیت‌های تصادمی مشخص‌شده و در ادامه برچسب‌های تصادمی به زیرگروه‌هایی مجزا تقسیم می‌شوند تا میزان تصادم در فرایند شناسایی برچسب‌ها کاهش یابد. نتایج شبیه‌سازی نشان می‌دهد زمان شناسایی و تعداد بیت‌های ارسالی برچسب‌ها در پروتکل پیشنهادی کمتر از پروتکل‌های ضدتصادم معمول است و این موضوع سبب بهبود عملکرد پروتکل پیشنهادی گردیده است.

کلیدواژه‌ها : سامانه RFID، پروتکل ضدتصادم، شناسایی برچسب، پروتکل مبتنی درخت، ALOHA

۱- مقدمه

سامانه RFID یک فناوری برای خودکار کردن شناسایی تجهیزات، کالاها، حیوانات و انسان‌ها است. این سامانه‌ها علاوه بر شناسایی خودکار، سبب افزایش سرعت، کاهش خطای ورود اطلاعات و تأمین امنیت می‌شوند. هر سامانه RFID از چندین برچسب، یک یا چند برچسب‌خوان و یک سرویس‌دهنده نهایی تشکیل می‌شود. برچسب یکی از مؤلفه‌های اصلی سامانه‌های RFID است که به فرد، کالا یا هر وسیله دیگری متصل می‌گردد. برچسب‌ها از نظر منبع انرژی به سه گروه برچسب‌های فعال، غیرفعال و نیمه‌فعال تقسیم می‌شوند. دیگر مؤلفه اصلی سامانه‌های RFID، برچسب‌خوان است که پل ارتباطی بین برچسب و سرویس‌دهنده نهایی است. برچسب‌خوان سیگنال‌های دریافتی از برچسب را پردازش کرده و متناسب با آن، پیامی را برای سرویس‌دهنده نهایی ارسال می‌کند. همچنین برچسب‌خوان وظیفه تأمین انرژی برای برچسب‌های غیرفعال، نیمه‌فعال و ارسال اطلاعات برای برچسب‌ها را نیز بر عهده دارد.

اطلاعات جمع‌آوری‌شده توسط برچسب‌خوان برای تحلیل و پردازش به سرویس‌دهنده نهایی که یکی دیگر از مؤلفه‌های سامانه RFID است، منتقل می‌شود. در سرویس‌دهنده نهایی یک پایگاه داده و یک بخش پردازشی وجود دارد، که در پایگاه داده برخی از اطلاعات برچسب‌ها، نظیر کلید محرمانه، شناسه یکتا و شناسه پویای[۱] برچسب، ذخیره شده است. در بخش پردازشی، عملیات محاسباتی مطابق با پروتکل سامانه RFID انجام می‌شود و در نهایت پاسخ مورد نظر برای برچسب‌خوان ارسال می‌شود. در این مقاله سامانه RFID، شامل یک برچسب‌خوان و چندین برچسب به همراه یک سرویس‌دهنده نهایی در نظر گرفته شده است که متداول‌ترین نوع سامانه‌های RFID است. برچسب موجود در این سامانه از نوع برچسب‌های غیرفعال در نظر گرفته شده است که ارزان قیمت‌ترین نوع برچسب‌ها در بازار هستند. همچنین در این مقاله برچسب‌خوان و سرویس‌دهنده نهایی برای راحتی یک موجودیت در نظر گرفته شده است.

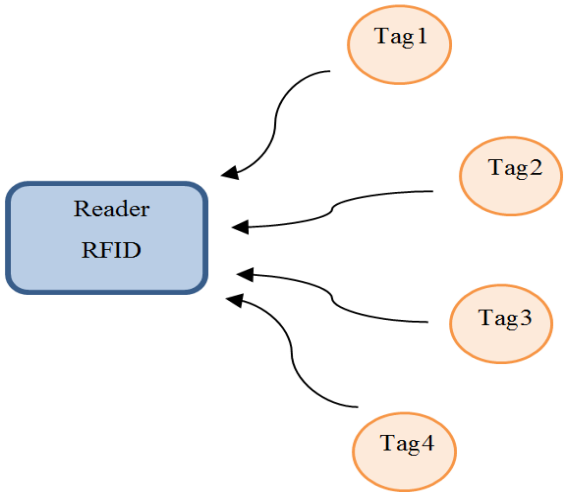
سامانه‌های RFID با وجود تمام مزایا و کاربردهایی که دارند ولی دارای مسائل و چالش‌هایی نیز هستند. یکی از این مسائل و چالش‌ها، مسئله تصادم[۲] یا برخورد برچسب‌ها در سامانه‌های RFID است. تصادم برچسب‌ها زمانی رخ می‌دهد که در یک

*رایانامه نویسنده مسئول: bahramiri@mut.ac.ir

[1] Dynamic Identifier
[2] Collision

سامانه RFID، چندین برچسب به‌طور هم‌زمان بخواهند اطلاعات خود را از طریق امواج رادیویی، به یک برچسب‌خوان مشترک ارسال کنند. در این حالت، سامانه RFID دچار اخلال می‌شود، زیرا اطلاعات برچسب‌ها به‌درستی توسط برچسب‌خوان دریافت نمی‌شود و در نتیجه برچسب‌خوان توانایی شناسایی اطلاعات هیچ‌کدام از برچسب‌ها را ندارد. شکل (۱) تصادم برچسب‌ها را در یک سامانه RFID نمایش می‌دهد. در سامانه‌های RFID، تصادم برچسب‌ها اجتناب‌ناپذیرند و باعث تأخیر در فرایند شناسایی و اتلاف انرژی می‌شوند. تاکنون پروتکل‌های ضدتصادم زیادی برای حل این مشکل ارائه شده است که به سه گروه پروتکل‌های ضدتصادم مبتنی بر ALOHA، درخت[1] و ترکیبی[2] تقسیم می‌شوند.

در این مقاله، در بخش دوم به بررسی پروتکل‌های ضدتصادم و برخی از کارهای انجام‌شده در این حوزه پرداخته می‌شود. در بخش سوم، پروتکل ضدتصادم پیشنهادی و شیوه رفع تصادم برچسب‌ها معرفی می‌شود. در بخش چهارم و پنجم به‌ترتیب الگوریتم برآورد تعداد برچسب و کدگذاری منچستر که در پروتکل پیشنهادی استفاده شده شرح داده می‌شود. در بخش ششم پروتکل پیشنهادی با مثالی توصیف گردیده و در نهایت در بخش هفتم جمع‌بندی مقاله انجام شده است.



شکل (۱): تصادم برچسب‌ها

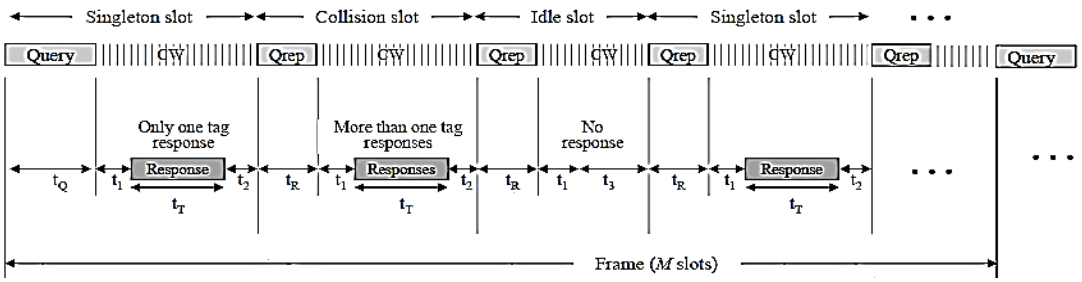
۲- مروری بر تحقیقات پیشین

پروتکل‌های ضدتصادم مبتنی بر روش ALOHA یکی از مهم‌ترین پروتکل‌های ضدتصادم، در سامانه‌های RFID هستند که از نوع پروتکل‌های احتمالاتی[3] هستند. در این نوع پروتکل‌ها، برچسب‌ها در صورت درخواست برچسب‌خوان، پاسخ خود را با انتخاب یک زمان تصادفی برای برچسب‌خوان ارسال می‌کنند. تاکنون پروتکل‌های ضدتصادم مبتنی بر ALOHA زیادی معرفی شده است که از مشهورترین آن‌ها می‌توان پروتکل ضدتصادم ALOHA محض[4] (PA)، ALOHA برهه‌ای[5] (SA)، FSA[6] و DFSA[7] نام برد [۱]. پروتکل ضدتصادم PA به‌دلیل وجود تصادم‌های جزئی فراوان که یک یا چند بیت از اطلاعات برچسب‌ها با یکدیگر برخورد می‌کنند، کارایی پایینی دارد. برای حل تصادم‌های جزئی در پروتکل PA، پروتکل ضدتصادم SA ارائه گردید که در آن از شکاف‌های زمانی استفاده شده است. در پروتکل SA، برچسب‌ها در ابتدای هر شکاف زمانی مجاز به ارسال اطلاعات هستند. در این حالت تصادم تنها در ابتدای هر شکاف زمانی رخ می‌دهد و به همین دلیل کارایی آن از پروتکل PA بهتر است. در پروتکل SA به‌دلیل عدم محدودیت در تعداد شکاف زمانی، زمان شناسایی برچسب‌ها افزایش می‌یابد. در پروتکل‌های FSA و DFSA برای کاهش زمان شناسایی برچسب‌ها، از قاب زمانی استفاده می‌شود که اندازه قاب پارامتری بسیار با اهمیت برای این پروتکل‌ها است؛ زیرا اگر اندازه قاب کمتر از تعداد برچسب‌ها باشد، تعداد شکاف‌های تصادمی افزایش می‌یابد و اگر اندازه قاب خیلی بیشتر از تعداد برچسب‌ها باشد، تعداد شکاف‌های بیکار افزایش می‌یابد که هر دو حالت برای سامانه RFID نامطلوب است و زمان اجرای پروتکل را افزایش می‌دهد.

پروتکل‌های ضدتصادم مبتنی بر درخت، از نوع پروتکل‌های قطعی[8] هستند و به این دلیل به آن‌ها قطعی گفته می‌شود که شناسایی برچسب‌ها در زمانی محدود، قطعی است. در تمام پروتکل‌های مختلف درختی، فرض می‌شود تصادم جزئی رخ ندهد و تصادم‌ها به‌صورت هم‌زمان و در همه بیت‌های شناسه برچسب رخ دهد. در پروتکل‌های درختی، طول شناسه برچسب‌ها بسیار اهمیت دارد، زیرا ممکن است تمامی بیت‌های شناسه در فرایند شناسایی برچسب‌ها، بررسی شوند. در این نوع پروتکل‌ها، پروتکلی کارایی بیشتری دارد که طول درخت کمتری جهت شناسایی شناسه‌ها داشته باشد. در این روش، برچسب‌های تصادمی در برچسب‌خوان به زیرگروه‌هایی مجزا تقسیم می‌شوند و ایجاد زیرگروه تا زمانی که حداکثر یک برچسب در هر شاخه درخت وجود داشته باشد، ادامه می‌یابد. پروتکل QT[9] در [۲] شناخته‌شده‌ترین پروتکل مبتنی بر درخت است.

[1] Tree
[2] Hybrid

[3] Probabilistic
[4] Pure ALOHA
[5] Slotted ALOHA
[6] Framed Slotted ALOHA
[7] Dynamic Framed Slotted ALOHA
[8] Deterministic
[9] Query Tree



شکل (۲): مدت زمان‌های ارسال در ارتباط برچسب‌خوان و برچسب [۹]

از دیگر پروتکل‌های درختی می‌توان پروتکل‌های BS[1] [۳]، CT[2] [۴]، OQTT[3] [۵]، CwT و QwT [۶] نام برد. همچنین در [۷] دو پروتکل مبتنی بر روش QT با نام‌های BQ-MDPM[4] و MDPM[5] ارائه شده که هر دو عملکردی به مراتب بهتر از روش QT دارند.

دیگر نوع پروتکل‌های ضدتصادم، پروتکل‌های ترکیبی هستند. در ساختار این پروتکل‌ها، از هر دو گروه پروتکل‌های ALOHA و درختی استفاده می‌شود. در این نوع پروتکل‌ها، بخشی از تصادم به وجود آمده در برچسب‌خوان با استفاده از پروتکل‌های ضدتصادم درختی و بخشی دیگر با استفاده از پروتکل‌های ضدتصادم ALOHA برطرف می‌شود. پروتکل‌های ترکیبی عموماً کارایی بهتری نسبت به پروتکل‌های درختی و ALOHA دارند ولی ساختار آن‌ها نسبت به پروتکل‌های درختی و ALOHA پیچیده‌تر است. در [۸] دو پروتکل ضدتصادم ترکیبی ATSA[6] و ImATSA[7]، در [۹] پروتکل ضدتصادم MCT[8]، در [۱۰] پروتکل ضدتصادم NQA[9] ارائه شده است.

۳- پروتکل پیشنهادی

پروتکل پیشنهادی یک پروتکل از نوع پروتکل‌های ضدتصادم ترکیبی است. شیوه انتقال اطلاعات بین برچسب‌خوان و برچسب‌ها در پروتکل پیشنهادی، مطابق شکل (۲) که بر اساس استاندارد EPC Global C1G2 است، انجام می‌شود. در طی فرایند شناسایی پروتکل ضدتصادم پیشنهادی، برچسب‌خوان یک صف پرس‌وجوی Q را برای ثبت مقادیر پیشوند قاب و اندازه قاب که به ترتیب با نمادهای pre و F نشان داده می‌شود، ایجاد می‌کند. در آغاز هر قاب، برچسب‌خوان مقادیر pre و F را از صف پرس‌وجوی Q استخراج و سپس فرمان $Query(pre,F)$ را برای برچسب‌های موجود در محدوده خود ارسال می‌کند. در هر قاب به‌جز شکاف اول، برچسب‌خوان از فرمان *QueryRep(spre)* استفاده می‌کند که *spre* نماد پیشوند شکاف ارسالی است. در قسمت برچسب، هر برچسب ابتدا وضعیت خود را در حالت غیرفعال قرار می‌دهد، و سپس حداکثر دو عمل مقایسه‌ای زیر را انجام می‌دهد.

مقایسه پیشوند قاب: هر برچسب بعد از دریافت فرمان $Query(pre,F)$، شناسه خود را با پیشوند pre مقایسه می‌کند. اگر شناسه برچسب، با پیشوند pre یکسان باشد، برچسب در حالت فعال قرار می‌گیرد. در این حالت، اگر $F=1$ باشد برچسب فوراً بیت‌های باقیمانده شناسه خود را به‌عنوان پاسخ برای برچسب‌خوان ارسال می‌کند در غیر این صورت برچسب، مقدار پیشوند شکاف خود را، با استفاده از P بیت بعد از پیشوند pre که $P=\log_2 F$ است، محاسبه می‌کند. مقدار پیشوند شکاف محاسبه شده توسط برچسب، با نماد $tpre$ نمایش داده می‌شود. اگر مقدار $tpre$ برابر صفر باشد، برچسب فوراً تعداد بیت‌های باقی‌مانده شناسه خود را به‌عنوان پاسخ، برای برچسب‌خوان ارسال می‌کند. اگر این دو حالت برای برچسب رخ ندهد، برچسب در انتظار فرمان *QueryRep(spre)* خواهد ماند و فرایند مقایسه پیشوند شکاف را اجرا می‌کند.

مقایسه پیشوند شکاف: بعد از دریافت فرمان *QueryRep(spre)*، برچسب فعالی که پاسخی برای برچسب‌خوان ارسال نکرده است، مقدار $tpre$ خود را با مقدار *spre* دریافتی مقایسه می‌کند. اگر $tpre=spre$ باشد برچسب تعداد بیت‌های باقی‌مانده شناسه خود را در همان شکاف برای برچسب‌خوان ارسال می‌کند.

در آغاز فرایند پروتکل در قسمت برچسب‌خوان، صف پرس‌وجو به‌صورت $Q=\{(\varepsilon,F_0)\}$ است، که ε یک دنباله تهی و F_0 طول قاب اولیه است. طول قاب اولیه، توانی از دو است، یعنی اگر تعداد برچسب‌های اولیه برابر n باشد، طول قاب اولیه برابر $F_0=2^{\lfloor \log_2 n \rfloor}$ خواهد بود که $\lfloor\ \rfloor$ قسمت صحیح مقدار را نمایش می‌دهد. در ابتدای پروتکل همه برچسب‌ها با دریافت مقدار ε در حالت فعال قرار می‌گیرند و سپس با توجه به مقدار F_0 پاسخ خود را ارسال یا منتظر دریافت فرمان *QueryRep(spre)* خواهند ماند.

[1] Binary Search
[2] Collision Tree
[3] Optimal Query Tracking Tree
[4] Bit Query Modified Dual Prefixes Matching
[5] Modified Dual Prefixes Matching
[6] Assigned Tree Slotted ALOHA
[7] Improved ATSA
[8] M-ary Collision Tree
[9] Nested Q-Algorithm

در حالت کلی، عملکرد پروتکل ضدتصادم پیشنهادی در قسمت برچسب‌خوان به پنج گام تقسیم می‌شود. اگر صف پرس‌وجو در برچسب‌خوان به‌صورت $Q=\{(pre_1,F_1),(pre_2,F_2),...,(pre_l,F_l)\}$ باشد، پنج گام به‌صورت زیر است:

گام (۱): برچسب‌خوان فرمان پرس‌وجوی $Query(pre_1,F_1)$ را برای برچسب‌ها ارسال می‌کند.

گام (۲): برچسب‌خوان صف پرس‌وجوی Q را به‌صورت $Q=\{(pre_2,F_2),...,(pre_l,F_l)\}$ به‌روزرسانی می‌کند. مقادیر C_{idle}، C_{succ} و C_{coll} که به ترتیب نمایانگر تعداد شکاف‌های خالی، موفق و تصادمی است، برابر صفر قرار می‌دهد. همچنین برچسب‌خوان مقدار شمارنده شکاف را که با $slot$ نمایش داده می‌شود برابر صفر قرار می‌دهد.

گام (۳): بعد از دریافت پاسخ برچسب‌ها توسط برچسب‌خوان، یکی از حالت‌های زیر رخ می‌دهد:

- اگر یک پاسخ دریافت شده باشد. برچسب‌خوان آن برچسب را شناسایی کرده و مقدار C_{succ} را یک واحد افزایش می‌دهد.
- اگر برچسب‌خوان چندین پاسخ دریافت کند، یک تصادم رخ می‌دهد. در این حالت با توجه به این‌که شناسه‌های دریافتی با استفاده از کدگذاری منچستر توسط برچسب‌ها ارسال می‌شوند، موقعیت بیت‌های تصادمی در برچسب‌خوان مشخص می‌شوند. اگر دنباله دریافتی به‌صورت $b_1b_2....b_{c_1-1}b_{c_1}b_{c_1+1}...b_{c_2-1}b_{c_2}b_{c_2+1}...b_y$ باشد و b_1 اولین بیت دریافتی، b_{c_1} و b_{c_2} به‌ترتیب اولین و دومین بیت تصادم در دنباله دریافتی و b_y آخرین بیت دنباله دریافتی باشد، آنگاه برچسب‌خوان با توجه بیت‌های تصادمی دو مقدار $\{(pre_1 \| spre \| b_1b_2....b_{c_1-1} \| 0 \| b_{c_1+1}...b_{c_2-1})\}$ و $\{(pre_1 \| spre \| b_1b_2....b_{c_1-1} \| 1 \| b_{c_1+1}...b_{c_2-1})\}$ را در صف پرس‌وجوی Q قرار می‌دهد و C_{coll} را یک واحد افزایش می‌دهد. برای سادگی $b_1b_2...b_{c_1-1} \| 0 \| b_{c_1+1}..b_{c_2-1}$ و $b_1b_2...b_{c_1-1} \| 1 \| b_{c_1+1}..b_{c_2-1}$ به‌ترتیب با $manch_0$ و $manch_1$ نمایش داده می‌شود.
- اگر برچسب‌خوان هیچ پاسخی دریافت نکند، برچسب‌خوان تنها مقدار C_{idle} را یک واحد افزایش می‌دهد.

گام (۴): در پایان قاب، اگر $slot \neq F_1-1$ باشد، برچسب‌خوان مقادیر را به‌صورت $slot=slot+1$ و $spre=de2bi(slot)_P$ تنظیم می‌کند و سپس فرمان $QueryRep(spre)$ را انتشار می‌دهد و در ادامه گام (۳) را اجرا می‌کند. تابع $de2bi(\ )_P$ مقدار دسیمال $slot$ را به یک دنباله باینری با طول P بیت تبدیل می‌کند. در پایان قاب، اگر $slot=F_1-1$ باشد، برچسب‌خوان با توجه به مقادیر C_{idle}، C_{succ} و C_{coll} تعداد برچسب‌های تصادمی را با استفاده از الگوریتم برآورد تعداد برچسب، تخمین می‌زند و اندازه قاب پیشوندهای بدون اندازه قاب ذخیره‌شده در صف Q را محاسبه می‌کند. اگر اندازه قاب محاسبه شده با F_{new} نمایش داده شود و صف پرس‌وجوی Q به‌صورت زیر باشد:

$$\begin{aligned}Q=\{&(pre_2,F_2),...,(pre_l,F_l),(pre_1 \| spre_1 \| manch_0),\\&(pre_1 \| spre_1 \| manch_1),(pre_1 \| spre_2 \| manch_0),\\&(pre_1 \| spre_2 \| manch_1),...,(pre_1 \| spre_t \| manch_0),\\&(pre_1 \| spre_t \| manch_1)\}\end{aligned}$$

آنگاه صف پرس‌وجوی Q به‌صورت زیر به‌روزرسانی می‌شود:

$$\begin{aligned}Q=\{&(pre_2,F_2),...,(pre_l,F_l),(pre_1 \| spre_1 \| manch_0,F_{new}),\\&(pre_1 \| spre_1 \| manch_1,F_{new}),(pre_1 \| spre_2 \| manch_0,F_{new}),\\&(pre_1 \| spre_2 \| manch_1,F_{new}),...,(pre_1 \| spre_t \| manch_0,F_{new}),\\&(pre_1 \| spre_t \| manch_1,F_{new})\}\end{aligned}$$

اندازه قاب جدید (F_{new}) به این صورت محاسبه می‌شود، اگر $C_{coll}=1$ آنگاه $F_{new}=1$ است، در غیر این‌صورت با استفاده از مقادیر C_{idle}، C_{succ} و $C'_{coll}=C_{coll}-1$، طول قاب برابر $F=2^{\lfloor \log_2 \hat{n}/C'coll \rfloor}$ خواهد بود که مقدار $\hat{n}$ برابر تعداد برچسب‌های تخمینی از الگوریتم برآورد تعداد برچسب است.

گام (۵): اگر صف پرس‌وجوی Q تهی نیست، گام (۲) اجرا شود، در غیر این‌صورت فرایند شناسایی برچسب‌ها، به پایان رسیده است.

۴- الگوریتم برآورد تعداد برچسب

عملکرد پروتکل‌های ضدتصادمی، برای تعیین اندازه قاب نیاز به تخمین تعداد برچسب‌های موجود در محدوده برچسب‌خوان دارند که این موضوع تا حد زیادی به الگوریتم برآورد تعداد برچسب، وابسته است. برای تخمین تعداد برچسب‌های موجود در محدوده برچسب‌خوان در پروتکل پیشنهادی از الگوریتم برآورد تعداد برچسب ILCM[1] در [۱۱] استفاده شده است. در الگوریتم برآورد تعداد برچسب ILCM، تعداد برچسب‌های تخمینی، با استفاده از رابطه زیر محاسبه می‌شود:

$$\hat{n}=kC_{succ}+l \qquad (۱)$$

که $\hat{n}$ تعداد برچسب‌های تخمین زده شده است و k و l با استفاده از روابط (۲) و (۳) محاسبه می‌شوند:

$$k=\frac{C'_{coll}}{(4.344N-16.28)+(\frac{F}{-2.282-0.273N})C'_{coll}+0.2407\ln(F+42.56)} \qquad (۲)$$

[1] Improved Linearized Combinatorial Model

$$l = (1.2592 + 1.513F)\tan(1.234N^{-0.9907}C'_{coll}) \quad (۳)$$

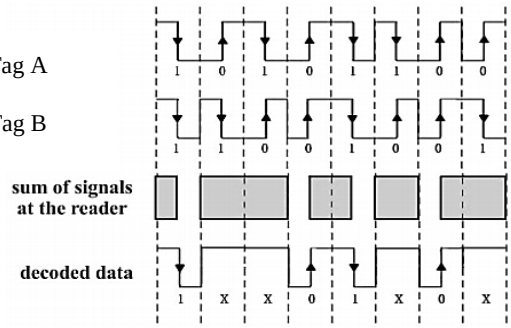
در روابط (۲) و (۳)، F نمایش‌دهنده اندازه قاب است.

الگوریتم‌های برآورد تعداد برچسب ILCM در مقایسه با الگوریتم برآورد تعداد برچسب vogt در [۱۲] که در پروتکل‌های ATSA و ImATSA استفاده شده است، از لحاظ تأخیر زمانی، مشابه یکدیگر رفتار می‌کنند، ولی میزان پیچیدگی محاسباتی الگوریتم ILCM در مقایسه با الگوریتم vogt کمتر است. میزان پیچیدگی محاسباتی الگوریتم ILCM برابر $O(1)$ است در حالی که میزان پیچیدگی محاسباتی الگوریتم vogt برابر $O(n)$ است. به‌دلیل همین کاهش پیچیدگی، میزان انرژی مصرفی در برچسب‌خوانی که از الگوریتم ILCM استفاده می‌کند نسبت به برچسب‌خوانی که از الگوریتم vogt استفاده می‌کند، کاهش می‌یابد [۱۱].

۵- کدگذاری منچستر

کدگذاری منچستر که در پروتکل پیشنهادی مورد استفاده قرار گرفت، روشی برای انتقال داده است. در این نوع کدگذاری، در وسط هر دوره بیتی، یک گذار وجود دارد که به‌عنوان محتوای بیت در نظر گرفته می‌شود. اگر گذار بالارونده باشد نشان‌دهنده مقدار بیت صفر است و اگر گذار پایین‌رونده باشد، نشان‌دهنده مقدار بیت یک است. در سامانه‌های RFID، هر برچسب سیگنال‌های پاسخ خود را، بر اساس کدگذاری منچستر ارسال می‌کند. بنابراین اگر دو (یا بیشتر) برچسب به‌طور هم‌زمان، دنباله بیت‌های داده مختلف خود را ارسال کنند، پس از آن، گذارهای پایین‌رونده و بالارونده هر بیت دریافتی همدیگر را خنثی می‌کنند و در نتیجه یک خطا در آن بیت به وجود می‌آید. برچسب‌خوان با استفاده از شیوه کدگذاری، توانایی شناسایی بیت‌های تصادمی و غیر تصادمی را خواهد داشت.

شکل (۳) مثالی از کد منچستر را نمایش می‌دهد که در آن شناسه‌های برچسب A و B به ترتیب برابر 10101100 و 11001001 است. زمانی که دو برچسب A و B به‌طور هم‌زمان شناسه‌های خود را با استفاده از کد منچستر ارسال می‌کنند، سیگنال دریافتی توسط برچسب‌خوان به‌صورت 1xx01x0x خواهد بود، که مقدار x بیت تصادمی را نمایش می‌دهد. در این مثال موقعیت بیت‌های تصادمی، بیت‌های دوم، سوم، ششم و هشتم است. پروتکل‌های ضدتصادم با استفاده از موقعیت بیت‌های تصادمی، برچسب‌ها را به‌صورت هوشمندانه‌ای به زیرگروه‌هایی مجزا تقسیم می‌کنند که این تقسیم‌بندی سبب کاهش زمان شناسایی برچسب‌ها می‌شود [۵]. در این مقاله برای پروتکل‌های ImATSA و OQTT نیز از کدگذاری منچستر استفاده می‌شود.



شکل (۳): مثال کد منچستر [۹]

۶- مثال پروتکل پیشنهادی

در این قسمت چگونگی عملکرد پروتکل پیشنهادی با مثالی توضیح داده می‌شود. فرض کنید برچسب‌های موجود در محدوده برچسب‌خوان، T1، T2، T3، T4، T5، T6، T7، T8 و T9 باشند که شناسه هر یک در جدول (۱) نمایش داده شده است. قاب اولیه در این مثال برابر $F_0 = 4$ فرض شده است و صف پرس‌وجو در ابتدای فرایند شناسایی به‌صورت $Q = \{(\varepsilon, 4)\}$ است.

جدول (۱): شناسه برچسب‌ها در مثال پروتکل پیشنهادی

نام برچسب	شناسه برچسب
T1	‘001001001111’
T2	‘001001010011’
T3	‘001001011101’
T4	‘001001011110’
T5	‘001011010110’
T6	‘100110011001’
T7	‘111100100111’
T8	‘111110100100’
T9	‘111110100111’

فرایند انجام پروتکل ضدتصادم پیشنهادی در این مثال به‌صورت زیر است:

۱) برچسب‌خوان فرمان $Query(\varepsilon, 4)$ برای برچسب‌های موجود در محدوده خود ارسال می‌کند.

۲) برچسب‌خوان صف پرس‌وجو را به‌صورت $Q = \{\}$ به‌روزرسانی می‌کند و مقادیر را برابر $C_{idle} = 0$، $C_{succ} = 0$، $C_{coll} = 0$ و $slot = 0$ تنظیم می‌کند.

۳) برچسب‌ها با دریافت دستور $Query(\varepsilon, 4)$، ابتدا شناسه خود را با پیشوند قاب ε مقایسه می‌کنند. به‌دلیل این‌که ε پیشوند قاب اولیه است، همه برچسب‌ها در این حالت در وضعیت فعال قرار می‌گیرند. برچسب‌ها با استفاده از P بیت اول شناسه خود، مقدار $tpre$ را محاسبه می‌کنند که $P = log_2 4 = 2$ است. بنابراین $tpre$ برچسب‌ها به ترتیب برابر ’00‘، ’00‘، ’00‘، ’00‘، ’00‘، ’10‘، ’11‘، ’11‘و ’11‘

است. برچسب‌های T1، T2، T3، T4 و T5 به‌دلیل این‌که *tpre* برابر صفر دارند، بلافاصله پاسخ خود را برای برچسب‌خوان ارسال می‌کنند.

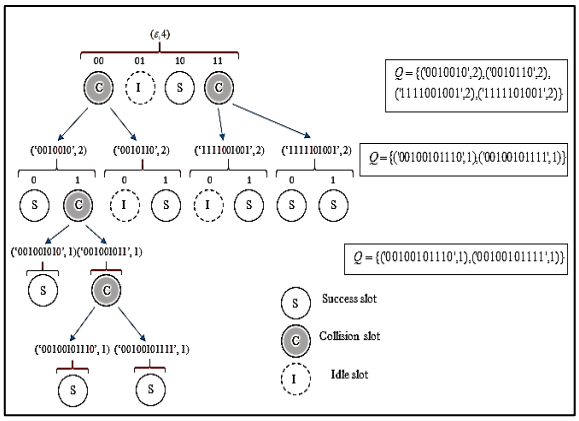
۴) به‌دلیل اینکه در شکاف اول قاب، تصادم رخ می‌دهد، برچسب‌خوان با توجه به موقعیت بیت اول و دوم تصادم، مقادیر را به‌صورت $Q=\{'0010\underline{0}10','0010\underline{1}10'\}$، $C_{coll}=1$، $slot=1$ ذخیره می‌کند و سپس فرمان *QueryRep*('01') را برای شروع شکاف دومِ قاب ارسال می‌کند.

۵) برچسب‌هایی که *tpre* برابر '01' دارند، بعد از دریافت فرمان *QueryRep*('01') پاسخ خود را برای برچسب‌خوان، ارسال می‌کنند، که در این مثال، چنین برچسبی وجود ندارد، در نتیجه برچسب‌خوان هیچ پاسخی دریافت نمی‌کند. در این حالت برچسب‌خوان مقادیر را به‌صورت $C_{idle}=1$ و $slot=2$ تغییر می‌دهد و سپس فرمان *QueryRep*('10') را برای شروع شکاف سوم قاب ارسال می‌کند.

۶) بعد از دریافت فرمان *QueryRep*('10') توسط برچسب‌ها تنها برچسب T6 پاسخ خود را ارسال می‌کند. در این شکاف، تغییرات به‌صورت $C_{succ}=1$ و $slot=3$ انجام می‌شود. سپس برچسب‌خوان فرمان *QueryRep*('11') را برای شروع شکاف چهارم قاب ارسال می‌کند.

۷) در چهارمین قاب بعد از دریافت فرمان *QueryRep*('11') برچسب‌های T7، T8 و T9 که *tpre* برابر '11' دارند، پاسخ خود را برای برچسب‌خوان ارسال می‌کنند. در این شکاف با توجه به اینکه بیش از دو برچسب پاسخ خود را ارسال کرده‌اند، تصادم رخ می‌دهد. برچسب‌خوان با توجه به اولین و دومین بیت تصادم، تعداد شکاف‌های تصادمی را برابر $C_{coll}=2$ و صف پرس‌وجو را به‌صورت $Q=\{'0010\underline{0}10','0010\underline{1}10','1111\underline{0}01001','1111\underline{1}01001'\}$ تغییر می‌دهد. در این گام مقدار شمارنده شکاف برابر $slot=4-1$ است که نشان‌دهنده آخرین شکاف قاب است. با توجه به اینکه $C_{coll}=2$ است، بنابراین برچسب‌خوان با استفاده از مقادیر C_{idle}، C_{succ} و $C'_{coll}=C_{coll}-1$ و به کمک الگوریتم ILCM تعداد برچسب‌های پاسخ‌دهنده را تخمین می‌زند. اگرتعداد برچسب‌های تخمینی برابر $\hat{n}$ باشد، آنگاه اندازهقاب بعدی برابر $F=2^{\lfloor \log_2 \hat{n}/C'_{coll} \rfloor}=2$ خواهد بود. در نتیجه اندازه قاب پیشوندهای موجود در صف Q که فاقد اندازه قاب هستند، برابر 2 خواهد بود. صف پرس‌وجوی Q پس از قرار دادن اندازه قاب به‌صورت $Q=\{('0010\underline{0}10',2),('0010\underline{1}10',2),('1111\underline{0}01001',2),('1111\underline{1}01001',2)\}$ به‌روزرسانی می‌شود.

۸) برچسب‌خوان با استفاده از زیرگروه‌های ایجادشده در صف پرس‌وجو فرایند پردازشی پروتکل پیشنهادی را ادامه می‌دهد و در صورت ایجاد تصادم در هر یک از زیرگروه‌ها، زیرگروه‌های دیگری را ایجاد می‌کند. این فرایند با خالی شدن صف پرس‌وجو به پایان می‌رسد. شکل (۴) فرایند شناسایی برچسب‌های موجود در این مثال را نمایش می‌دهد.



شکل (۴): فرایند شناسایی برچسب‌ها با پروتکل پیشنهادی

۷- شبیه‌سازی

پروتکل پیشنهادی با استفاده از نرم‌افزار متلب شبیه‌سازی شده است. در این شبیه‌سازی، سامانه RFID شامل یک برچسب‌خوان و n برچسب غیرفعال در محدوده برچسب‌خوان فرض شده است که هر برچسب $i\in\{1,\ldots,n\}$ دارای یک شناسه منحصر به فرد ۹۶ بیتی است و سرعت تبادل اطلاعات برابر $DR=128kbps$ است. کانال ارتباطی بین برچسب‌خوان و برچسب‌ها، بدون نویز و ایده‌آل در نظر گرفته شده است و پاسخ برچسب‌ها به برچسب‌خوان بدون در نظر گرفتن فاصله برچسب‌ها از برچسب‌خوان به‌طور هم‌زمان فرض شده است.

پارامترهای زمانی ارتباط بین برچسب‌خوان و برچسب‌ها در این سامانه RFID بر اساس شکل (۲) که مبتنی بر استاندارد EPC Global C1G2 است، انجام شده است. در این شکل فرمان *Query* برای ایجاد یک قاب و در بردارنده اطلاعاتی نظیر اندازه قاب و پیشوند قاب است. فرمان *QueryRep* تنها شامل اطلاعات پیشوند هر شکاف است. t_Q و t_R به ترتیب نمایش‌دهنده زمان ارسال فرمان‌های *Query* و *QueryRep* از برچسب‌خوان به برچسب‌ها است. t_T مدت زمانی است که برچسب نیاز دارد تا پاسخ برچسب‌خوان را ارسال کند. t_1 مدت زمان انتقال فرمان برچسب‌خوان تا پاسخ برچسب است. t_2 مدت زمان ارسال پاسخ برچسب تا فرمان بعدی برچسب‌خوان است. t_3 مدت زمان انتظار برچسب‌خوان بعد از زمان t_1 در صورتی که هیچ برچسبی، پاسخی برای برچسب‌خوان ارسال نکند. برای سادگی بیشتر در

شبیه‌سازی این زمان‌بندی‌ها به‌صورت روابط زیر انجام شده است:

$$T_Q = t_Q + t_1 \quad (۴)$$

$$T_{QR} = t_R + t_1 \quad (۵)$$

$$T_{succ} = T_{coll} = t_T + t_2 \quad (۶)$$

$$T_{idle} = t_3 \quad (۷)$$

برای همه پروتکل‌ها، بر اساس [۸]، $t_2 = 75us$ و $t_3 = 30us$ در نظر گرفته شده است. طول فرمان‌های *Query* و *QueryRep* برای پروتکل‌های ATSA، ImATSA و پروتکل پیشنهادی به ترتیب برابر $L_{pre} = 22$ و $L_{spre} = 4$ بیت و برای پروتکل‌های QT و OQTT با توجه به اینکه نیازی به ارسال طول قاب ندارند، $L_{pre} = 18$ در نظر گرفته شده است. لازم به ذکر است که پروتکل‌های QT و OQTT نیازی به فرمان *QueryRep* ندارند. مقادیر پارامترهای مختلف زمانی در جدول (۲) نمایش داده شده است و k نماد تعداد بیت‌های ارسال شده در هر شکاف زمانی است. همچنین طول قاب اولیه در این شبیه‌سازی برای پروتکل پیشنهادی و پروتکل‌های ATSA و ImATSA برابر $F_0 = 128$ در نظر گرفته شده است.

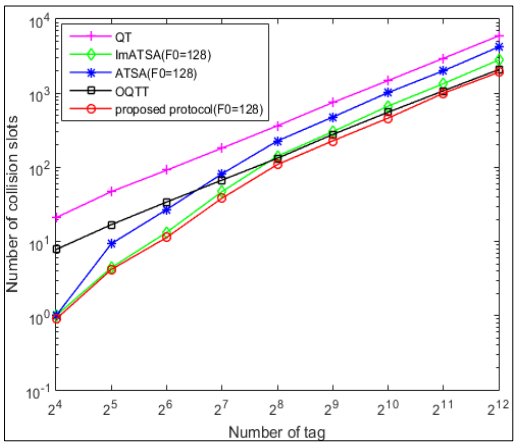
در این شبیه‌سازی، ابتدا به مقایسه تعداد شکاف‌های تصادمی، بیکار و تعداد کل شکاف‌ها پرداخته می‌شود و سپس به مقایسه زمان شناسایی برچسب‌ها و تعداد بیت‌های ارسالی بین برچسب‌خوان و برچسب پرداخته می‌شود. در شبیه‌سازی پروتکل OQTT به‌دلیل تعداد کم شکاف‌های لازم در الگوریتم تخمین بیت (BEA[1]) در [۵]، صرف‌نظر شده است. نتایج به‌دست‌آمده در این شبیه‌سازی از میانگین ۲۰ بار اجرا با شناسه‌های مختلف به دست آمده است. شناسه‌های مورد استفاده در این شبیه‌سازی عددهای تصادفی ۹۶ بیتی هستند که کمک نرم‌افزار Matlab تولید شده‌اند.

جدول (۲): پارامترهای زمانی شکاف‌ها (us) (مبتنی بر [۸])

پروتکل	T_Q	T_{QR}	$T_{succ} = T_{coll}$	T_{idle}
ATSA	$(22 + L_{pre}) / DR$	$(4 + L_{spre}) / DR$	$75 + k / DR$	30
ImATSA	$(22 + L_{pre}) / DR$	$(4 + L_{spre}) / DR$	$75 + k / DR$	30
QT	$(18 + L_{pre}) / DR$	—	$75 + k / DR$	30
OQTT	$(18 + L_{pre}) / DR$	—	$75 + k / DR$	30
پروتکل پیشنهادی	$(22 + L_{pre}) / DR$	$(4 + L_{spre}) / DR$	$75 + k / DR$	30

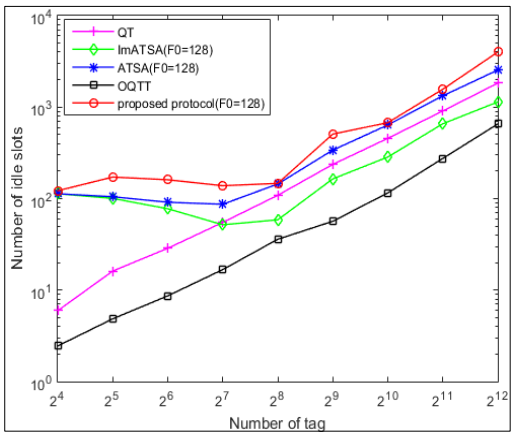
شکل (۵) تعداد شکاف‌های تصادمی را در پروتکل پیشنهادی و سایر پروتکل‌ها نمایش می‌دهد. همان‌طور که مشاهده می‌شود، پروتکل QT بیشترین تعداد شکاف تصادمی را نسبت به پروتکل‌های دیگر دارد که دلیل آن استفاده از درخت باینری برای تقسیم برچسب‌های تصادمی، به دو زیرگروه است. به همین دلیل بسیاری از برچسب‌های تصادمی در شکاف‌های بعدی نیز با یکدیگر تصادم دارند. پروتکل‌های پیشنهادی، ATSA و ImATSA، برچسب‌های تصادمی را به زیرگروه‌هایی مجزا تقسیم کرده، سپس با استفاده از روش ALOHA به شناسایی برچسب‌های هر زیرگروه می‌پردازند. پروتکل‌های پیشنهادی، ATSA و ImATSA در $n < 128$ به‌دلیل بزرگ بودن قاب اولیه نسبت به تعداد برچسب موجود در محدوده برچسب‌خوان، شکاف‌های تصادمی کمی ایجاد می‌کنند. پروتکل OQTT نیز در ابتدا با استفاده از الگوریتم BEA، تعدادی زیرگروه تشکیل می‌دهد و سپس با استفاده از روش درخت باینری به شناسایی برچسب‌های هر زیرگروه می‌پردازد. دلیل کمتر بودن تعداد شکاف‌های تصادمی در پروتکل پیشنهادی نسبت به پروتکل‌های ATSA و ImATSA به‌دلیل استفاده از درخت باینری در اولین بیت تصادم و ایجاد دو زیرگروه در صف پرس‌وجو است.

شکل (۶) تعداد شکاف‌های بیکار در پروتکل پیشنهادی و سایر پروتکل‌ها را نمایش می‌دهد. همان‌طور که مشاهده می‌شود، تعداد شکاف‌های بیکار در پروتکل پیشنهادی از سایر پروتکل‌ها بیشتر است. زیاد بودن تعداد شکاف‌های بیکار در پروتکل پیشنهادی نسبت به دیگر پروتکل‌ها به این دلیل است که گاهی مواقع تصادم رخ داده با تقسیم دو زیرگروه در بیت اول تصادم و ایجاد یک شکاف برای بیت دوم تصادم برای هر کدام از این زیرگروه‌ها رفع می‌شود، ولی پروتکل پیشنهادی برای بیت دوم تصادم یک قاب با تعداد حداقل دو شکاف ایجاد می‌کند که تنها یکی از این شکاف‌ها موفق و بقیه شکاف‌ها بیکار هستند. به همین علت تعداد شکاف‌های بیکار در پروتکل پیشنهادی زیاد است. پروتکل‌های پیشنهادی، ATSA و ImATSA در $n < 128$ به‌دلیل بزرگ بودن اندازه قاب اولیه نسبت به تعداد برچسب‌های موجود در محدوده برچسب‌خوان، شکاف‌های بیکار زیادی تولید می‌کنند. پروتکل OQTT به‌دلیل استفاده از الگوریتم BEA در ابتدای شناسایی برچسب‌ها، دارای کمترین تعداد شکاف بیکار است.



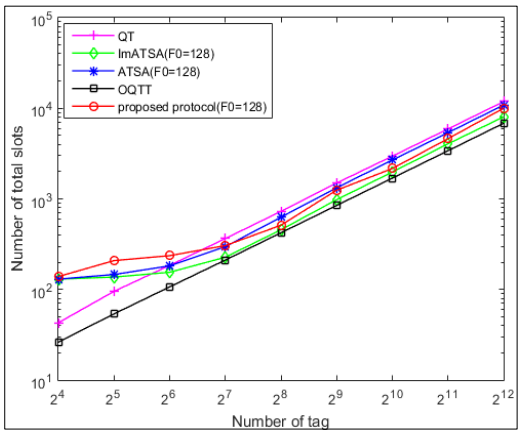
شکل (۵): مقایسه تعداد شکاف‌های تصادمی پروتکل‌ها

[1] Bit Estimation Algorithm



شکل (۶): مقایسه تعداد شکاف‌های بیکار پروتکل‌ها

شکل (۷) تعداد کل شکاف‌های پروتکل پیشنهادی را با دیگر پروتکل‌ها نمایش می‌دهد. تعداد کل شکاف‌ها برای هر پروتکل برابر مجموع تعداد شکاف‌های تصادمی، موفق و بیکار است. دلیل زیاد بودن تعداد کل شکاف‌ها در پروتکل پیشنهادی و پروتکل‌های ATSA و ImATSA برای $n<128$ به‌دلیل بزرگ بودن اندازه قاب اولیه نسبت به تعداد برچسب‌ها است. تعداد کل شکاف‌ها در پروتکل پیشنهادی نسبت به پروتکل‌های ImATSA و OQTT برای $n>128$ بیشتر است که دلیل آن زیاد بودن تعداد شکاف‌های بیکار در پروتکل پیشنهادی است. در شکل (۷) کمترین تعداد شکاف را، پروتکل OQTT دارد که به‌دلیل استفاده از الگوریتم BEA است.

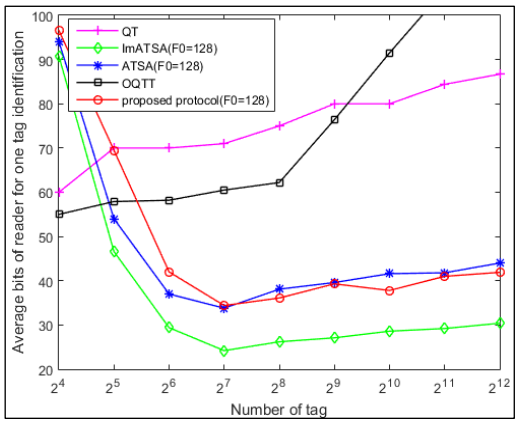


شکل (۷): مقایسه تعداد کل شکاف‌ها در پروتکل‌ها

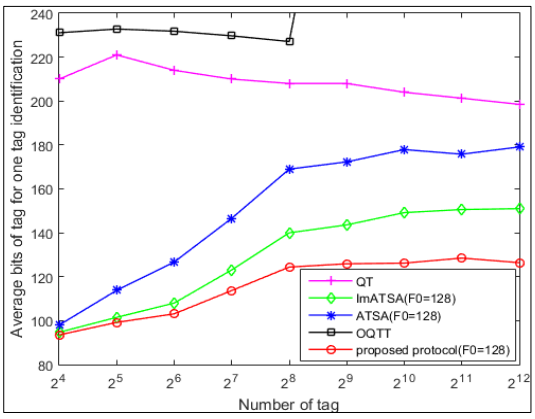
شکل (۸) تعداد بیت‌های انتقالی برچسب‌خوانِ پروتکل پیشنهادی را، با سایر پروتکل‌ها برای شناسایی یک برچسب نمایش می‌دهد. همان‌طور که مشاهده می‌شود، برچسب‌خوانِ پروتکل پیشنهادی، تعداد بیت‌های کمتری را نسبت به دیگر پروتکل‌ها به‌جز پروتکل ImATSA ارسال می‌کند. البته لازم به ذکر است یکی از اهداف ارائه پروتکل پیشنهادی، کاهش تعداد بیت‌های ارسالی توسط برچسب است که سبب افزایش ارسال تعداد بیت‌های شناسه، توسط برچسب‌خوان می‌شود. به همین دلیل، در این نمودار تعداد بیت‌های ارسالی برچسب‌خوانِ پروتکل پیشنهادی از پروتکل ImATSA بیشتر است. میزان بیت‌های ارسالی در پروتکل پیشنهادی، برای $n>128$ از پروتکل‌های ATSA و QT کمتر و از پروتکل‌های OQTT و ImATSA بیشتر است. برای پروتکل‌های پیشنهادی، ATSA و ImATSA در $n\leq128$ به‌دلیل بزرگ بودن اندازه قاب اولیه نسبت به تعداد برچسب‌ها، بیت‌های ارسالی برچسب‌خوان، از سایر پروتکل‌ها بیشتر است.

برای بهبود عملکرد برچسب‌خوانِ پروتکل پیشنهادی، با توجه به اینکه افزایش تعداد بیت‌های ارسالی باعث افزایش مقدار انرژی می‌شود، از الگوریتم ILCM استفاده شد، تا مقدار انرژی مصرفی در برچسب‌خوان کاهش یابد. همان‌طور که قبلاً ذکر شد، ساختار الگوریتم ILCM پیچیدگی کمتری نسبت به الگوریتم vogt به‌کاررفته در پروتکل‌های ATSA و ImATSA دارد. در شکل (۸) پروتکل OQTT به‌دلیل استفاده از الگوریتم BEA در $n>128$، تعداد بیت‌های ارسالی آن به‌صورت خطی افزایش یافته است.

شکل (۹) تعداد بیت‌های انتقالی برچسب‌ها در پروتکل پیشنهادی را، با سایر پروتکل‌ها برای شناسایی یک برچسب، نمایش می‌دهد. همان‌طور که در این شکل مشاهده می‌شود، تعداد بیت‌های انتقالی برچسب‌ها در پروتکل پیشنهادی از سایر پروتکل‌ها کمتر است، که یکی از مهم‌ترین ویژگی‌های این پروتکل نسبت به سایر پروتکل‌ها است. با توجه به محدودیت انرژی در برچسب‌های غیرفعال، هر چقدر تعداد بیت‌های انتقالی برچسب‌ها به برچسب‌خوان کمتر باشد، انرژی مصرفی برچسب، کاهش می‌یابد. در پروتکل پیشنهادی تعداد بیت‌های ارسالی برچسب‌ها نسبت به پروتکل ImATSA که نزدیک‌ترین تعقیب کننده پروتکل پیشنهادی است، ۱۵/۴۲٪ کاهش یافته است. دلیل افزایش ناگهانی تعداد بیت‌های انتقالی برچسب در $n>128$ برای پروتکل OQTT، استفاده از الگوریتم BEA در ابتدای پروتکل است.

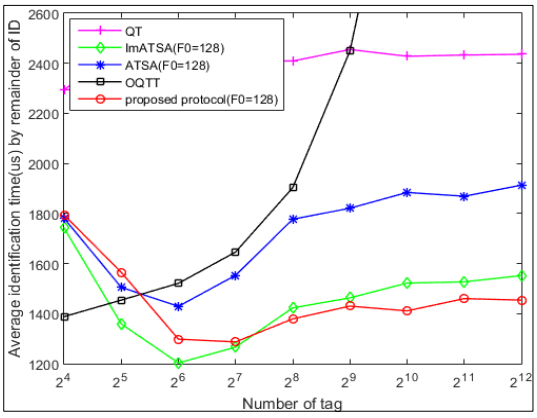


شکل (۸): تعداد بیت‌های انتقالی برچسب‌خوان برای شناسایی یک برچسب



شکل (۹): تعداد بیت‌های انتقالی برچسب برای شناسایی یک برچسب

شکل (۱۰) میانگین زمان شناسایی یک برچسب در پروتکل پیشنهادی و سایر پروتکل‌ها را نمایش می‌دهد. مقادیر میانگین زمان شناسایی پروتکل‌ها، با استفاده از جدول (۲) محاسبه شده است و مدت زمان انجام الگوریتم BEA در پروتکل OQTT برای n برچسب با توجه به [۵] برابر $0.0025n$ زمان یک شکاف موفق در نظر گرفته شده است. همان‌طور که در شکل مشاهده می‌شود، میانگین زمان شناسایی پروتکل پیشنهادی، برای $n>128$ کمتر از پروتکل‌های دیگر است که یکی از مهم‌ترین ویژگی‌های پروتکل پیشنهادی است. برای $n<128$ میانگین زمان شناسایی یک برچسب به‌دلیل بزرگ بودن اندازه قاب اولیه، از برخی پروتکل‌ها بیشتر است. در مجموع هر چقدر زمان شناسایی برچسب‌ها در یک پروتکل کمتر باشد، نشان‌دهنده بهتر بودن پروتکل از سایر پروتکل‌ها است. همان‌طور که مشاهده می‌شود پروتکل پیشنهادی در زمان‌هایی که تعداد برچسب‌ها از اندازه قاب اولیه بیشتر هستند، کمترین زمان شناسایی را دارد.



شکل (۱۰): میانگین زمان شناسایی یک برچسب با ارسال باقیمانده شناسه برچسب

۸- نتیجه‌گیری

در این گزارش یک پروتکل ضدتصادم ترکیبی که سبب کاهش زمان شناسایی برچسب‌ها و کاهش تعداد بیت‌های ارسالی برچسب‌ها می‌شود، ارائه گردید. پروتکل پیشنهادی نسبت به نزدیک‌ترین تعقیب‌کننده خود یعنی پروتکل ImATSA، از نظر تعداد بیت‌های ارسالی برچسب‌ها، ۱۵/۴۲٪ کاهش یافته که این بهبود سبب کاهش زمان شناسایی و کاهش مصرف انرژی در برچسب‌های غیرفعال که دارای محدودیت انرژی هستند، می‌شود. همچنین به‌دلیل استفاده از الگوریتم برآورد تعداد برچسب ILCM پیچیدگی محاسباتی در برچسب‌خوانِ پروتکل پیشنهادی نسبت به برچسب‌خوانِ پروتکل‌های ATSA و ImATSA کاهش یافته است که این کاهش پیچیدگی محاسباتی، سبب کاهش مصرف انرژی در برچسب‌خوانِ سامانه RFID می‌شود.

عملکرد پروتکل پیشنهادی در محیط‌هایی که تعداد برچسب‌ها موجود با اندازه قاب اولیه تناسب ندارد سبب ایجاد شکاف‌های تصادمی یا بیکار زیادی می‌شود. برای رفع این مسئله نیاز به ارائه یک الگوریتم برآورد تعداد برچسب مناسب برای قاب اولیه با کمترین شکاف تصادمی یا بیکار است. همچنین ارائه یک الگوریتم برآورد تعداد برچسب برای تعیین اندازه دیگر قاب‌ها با تولید حداقل شکاف خالی سبب بهبود عملکرد پروتکل پیشنهادی می‌شود. با ارائه این دو الگوریتم، زمان شناسایی برچسب‌ها و انرژی مصرفی در پروتکل پیشنهادی کاهش می‌یابد. ارائه این دو الگوریتم و پیاده‌سازی سخت‌افزاری آن برای کارهای آینده پیشنهاد می‌شود.

۹- مراجع

[1] A. Rennane, H. Saadi, R. Touhami, and M. C. Yagoub, "{Rennane, 2012 #1}," in Microelectronics (ICM), 2012 24th International Conference on, IEEE, pp. 1-4, 2012.

[2] C. Law, K. Lee, and K.-Y. Siu, "Efficient memoryless protocol for tag identification," in Proceedings of the 4th international workshop on Discrete algorithms and methods for mobile computing and communications, pp. 75-84, 2000.

[3] Q. Zhihong and W. Xue, "An overview of anti-collision protocols for radio frequency identification devices," China Communications, vol. 11, no. 11, pp. 44-59, 2014.

[4] Y. Qing, L. Jiancheng, and W. Hongyi, "A collecting collision tree protocol for RFID tag identification with capture effect," International Journal of Distributed Sensor Networks, vol. 11, no. 9, p. 673253, 2015.

[5] Y.-C. Lai, L.-Y. Hsiao, H.-J. Chen, C.-N. Lai, and J.-W. Lin, "A novel query tree protocol with bit tracking in RFID tag identification," IEEE Transactions on Mobile Computing, vol. 12, no. 10, pp. 2063-2075, 2012.

[6] H. Landaluce, A. Perallos, E. Onieva, L. Arjona, and L. Bengtsson, "An energy and identification time decreasing procedure for memoryless RFID tag anticollision protocols," IEEE Transactions on

[10] X. Wang, L. T. Yang, H. Li, M. Lin, J. Han, and B. O. Apduhan, "NQA: A nested anti-collision algorithm for RFID systems," ACM Transactions on Embedded Computing Systems (TECS), vol. 18, no. 4, pp. 1-21, 2019.

[11] P. Šolić, J. Radić, and N. Rožić, "Energy efficient tag estimation method for ALOHA-based RFID systems," IEEE Sensors Journal, vol. 14, no. 10, pp. 3637-3647, 2014.

[12] H. Vogt, "Efficient object identification with passive RFID tags," in International Conference on Pervasive Computing, Springer, pp. 98-113, 2002.

Wireless Communications, vol. 15, no. 6, pp. 4234-4247, 2016.

[7] J. Su, Y. Chen, Z. Sheng, Z. Huang, and A. X. Liu, "From M-ary Query to Bit Query: a new strategy for efficient large-scale RFID identification," IEEE Transactions on Communications, 2020.

[8] L. Zhang, J. Zhang, and X. Tang, "Assigned tree slotted aloha RFID tag anti-collision protocols," IEEE transactions on wireless communications, vol. 12, no. 11, pp. 5493-5505, 2013.

[9] L. Zhang, W. Xiang, X. Tang, Q. Li, and Q. Yan, "A time-and energy-aware collision tree protocol for efficient large-scale RFID tag identification," IEEE Transactions on Industrial Informatics, vol. 14, no. 6, pp. 2406-2417, 2017.

Journal of Electronical & Cyber Defence
Vol. 8, No. 4, 2021, Serial No. 32

Introduction of An Anti-Collision Protocol in RFID Systems

B. Zeynali, H. Bahramgiri*

*Malek-ashtar University of Thechnology

(Received: 27/12/2019, Accepted: 05/08/2020)

ABSTRACT

The ability to easily and quickly access information through radio frequency identification (RFID) systems has led to the ever-expanding use of this technology in a variety of usages such as tactical applications. This technology provides the conditions for better situational awareness in tactical environments or logistics. But, on the other hand, the process of identification in a scenario with different and numerous actors, faces the collision challenge. The collision of tags in RFID systems is inevitable, and it affects the performance of RFID systems, especially in environments with a large number of tags. In this report, a hybrid (of ALOHA and tree-based) anti-collision protocol is proposed. In the provided protocol, each frame and slot have prefixes, and in each slot, tags whose identities match the prefixes of the frame and the slot, send their responses to the reader. In this protocol, using the Manchester Coding method, the position of the collision bits is specified, and then collision tags are split into separate subgroups to reduce the collision rate in the identification process. The simulation results of this protocol indicate that compared to other benchmark protocols, the identification time and the bit number of transferred tags in the proposed method are lower, which improve its performance.

Keywords: RFID System, Anti_Collision Protocol, Tag Identification, Tree-Based Protocol, ALOHA

* Corresponding Author Email: bahramgiri@mut.ac.ir