Ütközésfeloldás RFID rendszerekben

TÓTH KATALIN, SCHULCZ RÓBERT, IMRE SÁNDOR

BME Híradástechnikai Tanszék, Mobil Távközlési Laboratórium
ktoth@mcl.hu, {schulcz, imre}@hit.bme.hu

Lektorált

Kulcsszavak: RFID, olvasók, transzponderek, ütközésfeloldás

Cikkünk az RFID környezetben felmerülő többes leolvasási problémák megoldására rendelkezésre álló ütközésfeloldó algoritmusok ismerteti kitérve az olvasók és a transzponderek ütközésére egyaránt. Továbbá az ISO 18000-3 Mode 1 ütközésfeloldásra használt algoritmusának szimulációs eredményeit is ismertetjük.

1. Az RFID rendszerek és a többes leolvasás problémaköre

Az RFID az angol Radio Frequency Identification, azaz a rádiófrekvenciás azonosítás kifejezés rövidítése. Ez a technológia automatikus azonosításra szolgál, jellegét tekintve a birtok alapú azonosítási rendszerek közé sorolható.

Az RFID rendszerek alapvetően legalább két eszközből állnak: egy azonosítóból és egy azonosítandóból. Az azonosító adatkapcsolatot kezdeményez az azonosítandóval, mely során mindkét vagy csak az egyik irányba adatátvitel történik. Ahogyan a technológia nevéből is kiderül, a kommunikáció rádiófrekvencián zajlik, ezért mind az azonosítónak, mind az azonosítottnak rádiós interfésszel kell rendelkeznie.

Az azonosítani kívánt objektumon (pl:. árucikk) egy transzpondert helyeznek el. Az azonosító egy RFID olvasó, amely képes olvasni és/vagy írni a transzpondert.

Egy az előbb vázolt egyszerű RFID rendszer kiegészülhet még alkalmazási környezettől függően vezérlő számítógéppel, mely több olvasó működésének összehangolását vezérli, valamint összeköttetést teremt az olvasók és a háttéradatbázis(ok) között. A háttéradatbázis funkcionalitását tekintve alkalmas az olvasó(k) érzékelési távolságán belüli transzponderekkel kapcsolatos információk tárolására; nem csak lekérdezhetőek, hanem az adatátvitel alatt írhatóak is. Az *1. ábra* egy RFID rendszer felépítését mutatja.

Az RFID alkalmazások az elláttási lánc teljes hosszán, azaz az egyes alkatrészek, alapanyagok előállításától, előkészítésétől egészen a végfelhasználókhoz való eljutásig jelen vannak. Éppen ezért gyakori, hogy egy raktári alkalmazás esetében másodpercenként több száz címkét kell leolvasni. Az alapvető követelmény az RFID rendszerek esetén a minél több címke minél rövidebb idő alatt történő leolvasása nagy megbízhatósággal, azaz ne maradjon ki egyetlen címke sem a leolvasásból. Előfodulhat az is, hogy nem a transzponderek ütköznek, hanem az olvasók. Ilyen környezet képzelhető el egy hipermarketben, ahol sok leolvasó kapu sorakozik egymás mellett a pénztáraknál. Ebben az esetben ütközés léphet fel az egyes olvasók között, hiszen két vagy több olvasó által lefedett térrészeknek lehetnek közös területei. Ha itt tartózkodik egy transzponder, akkor ütközés lép fel az olvasók között.

Az ütközések legszembetűnőbb negatív hatása a leolvasási sebesség csökkenése. Ennek kiküszöbölésére két módszer kínálkozik: a sávszélesség növelése és az ütközések eliminálása. A sávszélessége növelésének fizikai korlátja mellett szempont az is, hogy ennek a módszernek a kivitelezési költsége jóval nagyobb, mint egy beépített algoritmus futtatásának. Ezért a továbbiakban a különböző ütközésfeloldó algoritmusokat ismertetjük a legelterjedtebb szabványok esetén, továbbá bemutatunk néhány szimulációs eredményt is arra vonakozóan, milyen paraméterek befolyásolják a leolvasási időt.

RFID rendszerekben ütközésfeloldó eljárásnak nevezzük azt a technikát, vagy protokollt, amely lehetővé teszi az interferencia nélküli többszörös hozzáférést.

1. ábra RFID rendszer felépítése

2. Ütközésfeloldó algoritmusok olvasók ütközésére

Az olvasók ütközésének feloldására három algoritmus terjedt el: NAK-alapú, beacon-alapú megoldások, valamint a változtatható hatótávolságú olvasók alkalmazása.

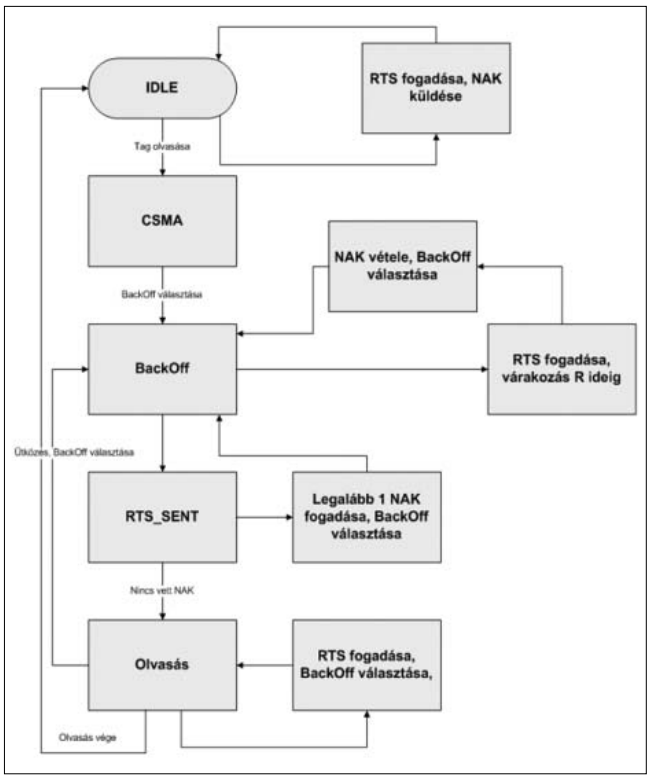
A *NAK-alapú ütközésfeloldás* feltétele, hogy minden olvasó saját vezérlőcsatornával rendelkezzen. E vezérlőcsatorna hatósugara elég nagy ahhoz, hogy az olvasó az összes, a környezetében lévő – az adatcsatornában interferenciát előidéző – olvasóról tudjon. (A NAK a Not Acknowledged, azaz nem nyugtázott kifejezés rövidítése.)

Ennek megfelelően az olvasó először egy RTS üzenetet küld szét a környezetében lévő összes olvasónak. Amelyik épp foglalt, egy NAK üzenetet küld vissza. Ha az olvasó legalább egy NAK-ot vesz, korlátozza az adatcsatornához való hozzáférését. Ezt a folyamatot szemlélteti a *2. ábra*.

A *beacon-alapú olvasók közötti ütközésfeloldás* feltételei a NAK-alapú megoldással azonosak: saját, nagy hatósugarú vezérlőcsatorna. Az eljárás szerint az olvasó – olvasás közben – periodikus beacon üzeneteket küld a vezérlőcsatornáján. Ha egy másik olvasó is olvasni szeretne, előbb ellenőrzi az utolsó beacon óta eltelt időt. Ha ez kisebb, mint egy megadott érték (vagyis a beacon periódusideje), az olvasás valószínűleg még folyamatban van, tehát várakozik, míg az előírt idő alatt nem jön újabb beacon. Az algoritmus működése a *3. ábrán* követhető nyomon.

Kézenfekvő és egyszerű megoldás az egyes olvasók hatósugarának adaptív változtatása. Az eljárás minden *n* olvasási ciklus után a felügyelt terület duplázásával, ütközés esetén pedig *p* valószínűséggel ennek csökkentésével optimalizálja a leolvasásokat. Ezzel gyakorlatilag egy SDMA (Space Division Multiple Access), azaz térosztásos többszörös hozzáférés valósul meg.

2. ábra NAK alapú ütközésfeloldás



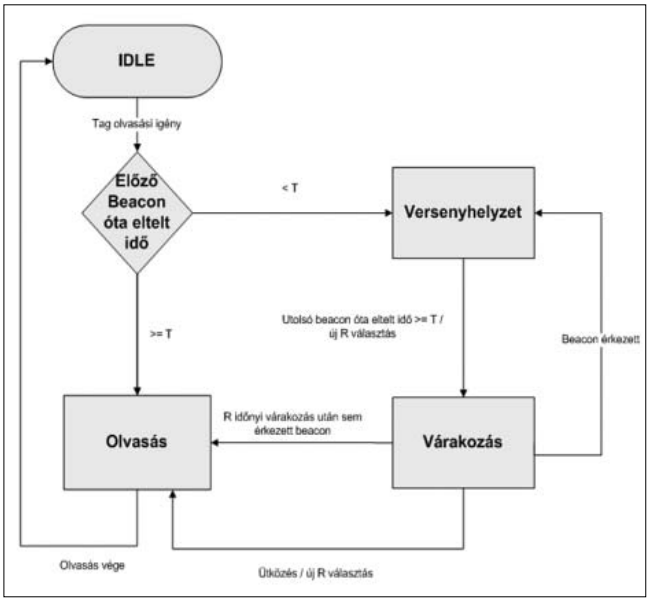
3. Ütközésfeloldó algoritmusok transzponderek ütközésére

Transzponderek ütközése úgy jöhet létre, hogy az olvasó hatósugarában számos tag van jelen. Olvasáskor a leolvasó egység egy request üzenetet broadcastol a tag-eknek. Amennyiben csak egyetlen tag tartózkodik a térben, az olvasó egyetlen választ vesz. Ugyanakkor minden tag visszaküldi a válaszát az olvasónak, ha több is van belőlük a rendszerben. Ez ütközést eredményez a rádiócsatornán, így válaszüzenetük nem lesz értelmezhető. Ennek kiküszöbölésére fejlesztettek ki számos ütközéselkerülő algoritmust, amelyeket az alábbiakban ismertetünk.

Az ütközések feloldása az esetek nagy részében valamilyen *többszörös közeghozzáférési feladat* megoldására vezethető vissza, így a következőkben ezek a – részben már más területekről ismert – algoritmusok kerülnek bemutatásra.

Az RFID rendszerekben is négyféle többszörös hozzáférési eljárást különböztetünk meg: térosztásos többszörös hozzáférés (SDMA, Space DMA), frekvencia-osztásos többszörös hozzáférés (FDMA, Frequency DMA), időosztásos többszörös hozzáférés (TDMA, Time DMA) és végül a kódosztásos többszörös hozzáférés (CDMA, Code DMA). Fontos megjegyezni, hogy néhány gyártó nem hozza nyilvánosságra saját ütközésfeloldó algoritmusait, míg mások a szabványban is lefektetett, elterjedt megoldásokat használják. Általánosságban elmondható, hogy az egyedi megoldások is ezen szabványokból táplálkoznak némi alkalmazás-specifikus kiterjesztés felhasználásával.

3. ábra Beacon alapú ütközésfeoldás



Az **SDMA** eljárás a rendelkezésre álló területet szeletekre osztva növeli meg a kapacitást.

Egyik lehetséges megoldás az olvasók hatótávolságának jelentős csökkentése, ezzel együtt sok olvasó és antenna elhelyezése a mezőben. Az így létrejövő tömbök nagyjából hézagmentesen fedik le a kívánt területet. Ennek eredményeképpen a szomszédos olvasók számára többszörösen is rendelkezésére áll a csatornakapacitás.

Egy másik lehetőség az *adaptív SDMA* megvalósítása, amikor elektronikusan vezérelt irányított antennát használunk az olvasóban, mely akármikor az aktuálisan kapcsolatban lévő transzponderre irányítható. Így az egyes tag-ek akár térbeli helyzetük, egymással és az olvasóval bezárt szögük alapján különböztethetők meg. Ilyen összeköttetések 850 MHz feletti RFID rendszerekben valósíthatók meg leginkább. Az eljárás legfőbb hátránya az összetett antenna és antennarendszerek kiépítése, mely csak speciális alkalmazásokban teremt neki létjogosultságot. Adaptív SDMA rendszerekben az elektronikusan vezérelt antenna iránya a fenti módon körforgásban is változtatható, így lépésről lépésre haladva minden transzponder beleesik a nyalábba.

FDMA felhasználása során egyidejűleg több tag számára is rendelkezésre áll egy-egy teljes értékű, más-más frekvenciájú vivő. RFID rendszerekben ez szabadon állítható, alharmonikus vivőket használó transzponderekkel valósítható meg. Mind a broadcast szinkroncsatorna, mind az energiaellátás egy alkalmasan választott f_a frekvencián kerül továbbításra. A transzponderek válaszukat a rendelkezésre álló néhány (f_1 - f_N) válaszcsatornán küldik vissza, így teljesen más frekvenciatartomány használható up- és down irányokra.

Backscatter és terhelésmodulált rendszerek számára egy rendelkezésre álló lehetőség több független vivőfrekvencia használata tag→olvasó irányban. Ezen eljárás hátránya szintén anyagi jellegű. Az olvasókba minden frekvenciához dedikált vevőket kell elhelyezni, ami jelentősen megnöveli az előállítási költségeket. Emiatt az FDMA használata csak egyes speciális alkalmazásokban ajánlott.

A **TDMA** azon eljárások összessége, melyek használata során a teljes rendelkezésre álló csatornakapacitás – a tag-ek valamilyen sorrendjét követve – az egymás után következő tag-ek számára teljes mértékben kiosztásra kerül. Széles körben elterjedt az alkalmazása többek közt a digitális mobil rádiórendszerekben. RFID rendszerekben ütközés feloldó algoritmusokban találkozunk vele. Megkülönböztetünk transzponder-vezérelt és adó/vevő-vezérelt eljárásokat.

A transzponder-vezérelt eljárások aszinkron módon működnek, mivel az olvasó nem szól bele az adatátvitel vezérlésébe (ezen az elven működik a későbbiekben tárgyalt ALOHA eljárás is). Tovább is csoportosíthatjuk az aszinkron módszereket: megkülönböztethetjük az eljárásokat aszerint is, hogy az olvasó küld-e egy „kikapcsol” jelzést a transzpondernek a sikeres adatátvitel után.

A transzponder-vezérelt eljárások nem elég flexibilisek, ugyanakkor eredendően lassúak. A legtöbb alkalmazás ezért az adó/vevő-vezérelt eljárásokat használja, mely során az olvasó vezérli az adatátvitelt. E módszerek szinkron eljárásoknak tekinthetők, hiszen minden transzpondert az olvasó ellenőriz és vezérel folyamatosan.

Működése során az olvasó mindig pontosan egy transzpondert választ ki a zónán belül, majd e két fél között megindul az adatátvitel (hitelesítés, írás/olvasás stb.). Kizárólag ezután történhet meg a kapcsolat bontása és az új összeköttetés létesítése egy másik transzponderrel. Ezek az eljárások is tovább csoportosíthatók polling, azaz lekérdezéses, valamint binary search, azaz bináris keresés elven működő algoritmusokká. Minden eljárás közös abban, hogy az egyes tag-eket egy egyedi azonosító alapján választja ki.

Lekérdezéses eljárások

A lekérdezéses eljárásnak a korrekt működéshez feltétlenül szüksége van a kommunikációban vélhetőleg részt vevő tagok azonosítójáról készült listára. A lekérdezéses folyamat során az olvasó egyenként végig lépked ezen a listán, egészen addig, amíg egy transzponder a megfelelő szériaszámmal válaszol. Ez a módszer elég időigényes, csak kis számú transzponder esetén alkalmazható hatékonyan.

A lekérdezés elvén működő eljárások közül a következőket fogjuk részletesen ismertetni: bináris keresés, dinamikus bináris keresés, réselt bináris fa és a bitről bitre bináris fa.

A *bináris keresés* az egyik legelterjedtebb és legflexibilisebb algoritmusok egyike. Működése során a transzpondert egy olyan csoportból választja ki, melyek azonosítóit szándékosan ütközésbe hozza egymással úgy, hogy a tag-ek egyszerre küldik vissza sorszámaikat az olvasó kérésére válaszolva. Ehhez szükséges, hogy az olvasó képes legyen az egyes bitpozíciók pontos detektálására, melyet a helyes kódolás megválasztásával ér el (pl. Manchester-kódolás).

Az algoritmus előre definiált interakciók (parancsok és válaszok) sorozatát tartalmazza, mely segítségével képes kapcsolatot teremteni az olvasó és a tag-ek között azzal a céllal, hogy közülük majd egyet adatátvitel céljából kiválasszon. A tényleges működéshez egy parancskészletet definiál REQUEST, SELECT, READ_ DATA ÉS UNSLECT utasításokkal. Ezen kívül minden transzponder egy egyedi sorozatszámmal rendelkezik. A könnyebb megértés érdekében az algoritmus működését egy olyan példával szemléltetjük, melyben 8 bites ID-k szerepelnek, ezzel 256 különböző tag egyediségét garantálva.

A 8 bites sorozatszámokat használó transzponderek a hexadecimális számrendszerben a 00-FFh, a decimális számrendszerben a 0-255, míg a bináris számrendszerben a 00000000-11111111 azonosítókkal jellemezhetők. Az algoritmus első lépéseként az olvasó egy REQUEST parancsot küld broadcast üzenetként (<=11111111) argumentummal. A 11111111b-es azono-

sító a legnagyobb értékű a 8 bites sorozatszámok között. Ez azt jelenti, hogy a zónában lévő minden tag válaszolni fog erre a kérésre, hiszen csak kisebb, vagy egyenlő lehet az azonosítójuk, mint 11111111b.

A precíz bitszinkron elengedhetetlen az algoritmus működéséhez, ennek biztosítása kritikus az ütközések elkerülése és felismerése szempontjából.

Az *1. táblázatban* látható, hogy a 0., a 4. és a 6. bitpozícióban ütköznek a sorozatszámok, mivel ezeken a helyeken a különböző bitértékek szuperponálódnak. Az ütközés az olvasó számára azt jelenti, hogy legalább két transzponder tartózkodik a zónában. A vett jel a következő lenne: 1X1X001X.

A 6-os bitpozíció a legmagasabb értékű a sorozatszámon belül, ahol még ütközés történt az első iteráció során. Ez azt jelenti, hogy legalább egy transzponder található az SNR > 11000000b és az SNR < 10111111b tartományokban. Ahhoz, hogy egyetlen transzpondert legyünk képesek kiválasztani az egyelőre ismeretlen számú csoportból, limitálni kell a lehetséges sorszámtartományt a következő iterációk során.

A folyamatot kezdjük a < 10111111b tartománnyal, vagyis a 6. pozíciót 0-nak rögzítjük, és figyelmen kívül hagyjuk az alacsonyabb tartományt, az adott bitértékek 1-esre állításával.

Amint az olvasó kiadta a REQUEST (<=10111111) parancsot, minden – a feltételt kielégítő – transzponder saját sorszámával válaszol. A példában ezek az 1, 2 és 3-as számú transzponderek. Ebből megállapítható, hogy legalább két transzponder lesz jelen a második iterációban is. A vett bitsorozat (101X001X) még mindig négy lehetséges sorozatszámot takar. Az ütközések ismételt jelenléte a 2. iteráció során újabb szűkítést igényel a 3. lépésben.

Ekkor az olvasó a REQUEST (<=10101111) parancsot adja ki a *2. táblázat* szerinti szabályok alapján. Ezt a feltételt csak a 2-es transzponder teljesíti a 10100011 sorozatszámmal. Mivel sikerült egyetlen valós sorozatszámot kiválasztani, további iterációra nincs szükség. Jelenleg tehát kommunikációra vagyunk képesek a 2-es számú transzponderrel. Írhatjuk, olvashatjuk, anélkül hogy a többi tag-gel interferenciába kerülne. Az adatátvitel végeztével egy UNSELECT kiadásával elnémíthatjuk az ismételt REQUEST-ig így biztos inaktív tag-et. A maradék tag-ek kiválasztásához szükséges iterációk száma egyre csökken, ahogy sorban válnak UNSELECED állapotúvá a transzponderek.

Az iterációk átlagos *L* száma, mely megmondja hány lépésben detektálható egyetlen transzponder egy n tagből álló csoportban könnyen számolható:

$$L(N) = ld(N) + 1 = \frac{\log(N)}{\log(2)} + 1$$

Amennyiben egyetlen transzponder tartózkodik a zónában, az iterációk száma természetesen egy, ha ennél több, *L* könnyen megnőhet.

A bináris keresés algoritmus legfőbb negatívuma a rendkívül pazarló működése volt, hiszen mind a keresési/lekérdezési kritériumok, mind a sorozatszám teljes hosszában átvitelre került minden lépésben. Ezt küszöböli ki a dinamikus bináris keresés algoritmus: a teljes sorozatszámok átvitele helyett most azoknak csak egy részét, X-nél vágott hányadukat visszük át. X jelöli azt a legmagasabb bitpozíciót, ahol az előző iteráció során ütközés volt. Az olvasó tehát csak (N-X) ismert részét küldi el a REQUEST parancs argumentumaként, majd leállítja az adást. Minden transzponder, mely sorozatszáma kezdete egyezik (N-X)-szel, válaszol az (X-1)-től 0-ig tartó, fennmaradó bitek átvitelével.

A *réselt bináris fa algoritmus* működése a következő: amennyiben az i. időrésben ütközés történik, minden transzponder, mely nem vett részt benne, vár annak feloldásáig. Az ütköző tag-ek véletlenszerűen két csoportba (nullás, egyes) kerülnek szétválasztásra. Az első (nullás) csoportban lévő tag-ek az (i+1). időrésben adnak újra, míg a másik csoportban lévő transzponderek várnak ennek sikeres végrehajtásáig. Amennyiben az (i+1). időrés üres, vagy sikeres átvitel történt benne, úgy a második csoport elemei az (i+2). időrésben adnak újra. Amennyiben az (i+1). időrés ütközést tartalmaz, úgy az egész eljárás újra megismétlődik.

A szükséges iterációk száma *n* számú tag esetén a következő egyenlet alapján alakul:

$$I_{SBT} = 1 + \sum_{k=2}^{n} \binom{n}{k} \frac{2(k-1)(-1)^k}{[1 - p^k - (1-p)^k]}$$

Végül a *bitről bitre bináris fa típusú keresést* mutatjuk be az ALOHA típusú ütközéselkerülő algoritmusok ismertetése előtt. A bitről-bitre algoritmus induláskor bekéri a transzponderek azonosítójának 1. bitjét. Miután a tag-ek elküldték a 0-t, vagy az 1-et, az olvasó megnézi, volt-e ütközés, mielőtt bekéri a következő bitet. Amennyiben volt, két csoportba osztja az ütköző tag-eket, majd kiválasztja az egyiket, és csak tőlük bekéri a következő bitpozíció értékét.

Az algoritmus e lépéseket ismétli, míg a teljes – sorozatszámokból felépített – bináris fát be nem járja. Ha ütközés nélkül veszi a k. bitet, annak értékét eltárolja a memóriájában. Ellenkező esetben, ütközés esetén a k. bitet 0-ként menti el, majd inaktív állapotba küldi az összes tag-et, melyek k. bitje 1. Ebben az állapotban a transzponderek ideiglenesen letiltják saját azonosítójuk küldését egészen addig, míg egy tag teljesen azonosításra nem került. Az algoritmus addig ismétli magát, míg ezzel minden tag-et azonosít.

A szükséges iterációk (I_{BBT}) száma: $I_{BBT} = n \times j$.

Transzponder 1	10 11 00 10
Transzponder 2	10 10 00 11
Transzponder 3	10 11 00 11
Transzponder 4	11 10 00 11

1. és 2. táblázat

Parancs	1. iteráció	n. iteráció
REQUEST > RANGE	0	Bit(x) = 1, Bit (0-tól (X-1)-ig) = 0
REQUEST < RANGE	SNRmax	Bit(x) = 0, Bit (0-tól (X-1)-ig) = 1

ALOHA-típusú eljárások

A transzponder által vezérelt ütközésfeoldó eljárások ismertetése során az ALOHA-típusú eljárásokról lesz szó.

Az ALOHA eljárás működésének alapját az képezi, hogy amint egy adatcsomag készen áll a küldésre, az nyomban továbbításra is kerül.

Ez a módszer általában csak olvasható transzponderekkel együtt használatos, azok alacsony csomagmérete miatt (jellemzően azonosítók). Ez az adat ciklikus sorozatban kerül küldésre, mely hosszú adások közti szünetekkel jár együtt. Ugyanakkor az egyes tagek által használt ismétlési idők nagyban különböznek egymástól. Ez azt jelenti, hogy nagy eséllyel különböző időpillanatokban lépnek fel az egyes tag-ek adatátviteli igénnyel, kicsi a torlódás/ütközés esélye.

A terhelés természetesen összhangban van az egyes időpillanatokban egyidejűleg adó transzponderek számával. Az átlagos *G* terhelés egy *T* megfigyelési időtartamra vett átlag, mely könnyen számolható az adatcsomag τ átviteli idejéből:

$$G = \sum_{l}^{n} \frac{\tau_l}{T} r_l$$

ahol *n* a rendszerben lévő transzponderek száma, és r_l = 0, 1, 2, ... az l. transzponder által átvitt adatcsomagok száma. Az *s*-sel jelzett áteresztőképesség értéke 1 a hibamentes átvitel idejére (ütközésmentes esetben), valamint 0 minden más esetben (például nincs adatátvitel, torlódás miatt nem továbbítható adatcsomag hiba nélkül).

Az átlagos *S* kibocsátás a felkínált *G* terhelésből számítható: $S = G \cdot e^{-2G}$.

Ha megvizsgáljuk az *S* kibocsátást a felkínált *G* terhelés függvényében, azt találjuk, hogy *G* = 0,5-nél *S*-nek egy maximuma van, mely értéke 18,4%.

Kisebb terhelés esetén a csatorna az idő nagy részében üres lenne; nagyobb terhelés az ütköző transzponderek számának rohamos növekedéséhez vezetne. Ilyen formán a csatorna kapacitásának 80%-a kihasználatlannak látszik. Az ALOHA egyszerű implementációja viszont lehetővé teszi annak ütközés feloldó algoritmusként való használatát. Ezen eljárást használják digitális hírhálózatokban, mint például a csomagkapcsolt rádió, mely világszerte kedvelt kommunikációs forma a rádióamatőrök között.

A sikeres átvitel *q* valószínűsége az átlagos *G* terhelésből és az *S* kibocsátásból számolható az alábbi képlet alapján:

$$q = \frac{S}{G} = e^{-2G}$$

A *k* számú adatcsomag *T* idő alatt sikeres átvitelének *p*(*k*) valószínűsége könnyen számolható az átvitelhez szükséges *r* idő, valamint az átlagos *G* terhelésből. A *p*(*k*) eloszlás Poisson-jellegű:

$$p(k) = \frac{(G\frac{T}{\tau})^k}{k!} e^{-(G\frac{T}{\tau})}$$

Az ALOHA relatív alacsony teljesítményének növelésére fejlesztették ki a **réselt ALOHA** (S-ALOHA, Slotted ALOHA) eljárást. A tag-ek, amennyiben ezt az algoritmust használják, csak meghatározott időpillanatokban kezdeményezhetnek adást (szinkronpontok, időrések). A tag-ek szinkronizációja az olvasó feladata, ami következtében ez egy sztochasztikus, adó/vevő-vezérelt TDMA ütközésfeloldó eljárás. Ezt használva ütközés fele akkora periódusban következhet csak be, mint ALOHA-t használva.

Azonos méretű adatcsomagokat (és ezáltal azonos átviteli időt) feltételezve, ALOHA használata során ütközés lép fel, amennyiben két transzponder $T < 2\tau$ időintervallumon belül küld. Réselt ALOHA és a szinkronpontok használatával ez az intervallum $T = \tau$ -ra csökken le. A réselt ALOHA áteresztőképessége ezen megfontolások alapján számolható: $S = G \cdot e^{-G}$.

A réselt ALOHA rendszerek S kibocsátása maximummal rendelkezik *G*=1 terhelésnél. Ez azt jelenti, hogy pontosan annyi transzponder van a zónában, ahány szabad időrés. Ha túl sok transzpondert helyezünk a térbe, az átbocsátóképesség hamar 0 közelébe esik le, legrosszabb esetben végtelen számú próbálkozás után sem leszünk képesek sorozatszámot detektálni. Ez a jelenség elkerülhető megfelelő számú időrés meghatározásával, ugyanakkor várható az ütközésfeloldó algoritmus teljesítményének csökkenése. Ennek oka az, hogy a rendszernek állandóan figyelnie kell minden elérhető transzponderre a zónában a szeletek teljes időtartama alatt akkor is, ha csak egy transzponder tartózkodik a térben.

A **dinamikus réselt ALOHA** ezen próbál segíteni az időrések számának dinamikus megválasztásával.

Egyik lehetőség erre a rendelkezésre álló időrések számának átvitele, melyet a transzponderek a REQUEST parancs argumentumaként kapnak meg: várakozó üzemmódban az olvasó ciklikus REQUEST parancsokat broadcastol, melyet egy vagy két időrés követ a lehetséges transzponderek számára. Ha a transzponderek nagy száma jelenti a szűk keresztmetszetet az időrésekben, minden REQUEST üzenettel megnő ezek száma, amíg egy egyedi transzponder tisztán detektálhatóvá válik. Ezt követően az olvasó egy BREAK üzenetet küld ki, mely hatására a detektálton kívüli transzponderek blokkolt állapotba kerülnek.

4. SimTag szimuláció

Most, hogy mind az olvasók, mind a traszponderek ütközéseit feloldó alkalmazható algoritmusokat ismertettük, érdemes megvizsgálni, hogy a gyakorlatban milyen paraméterek befolyásolják a többes leolvasás idejét, hatékonyságát. Ehhez egy ISO által kibocsátott szabvány szerinti ütközésfeloldó algoritmus szimulációs eredményeit mutatjuk be.

Az **ISO 18000-3 MODE 1** alapértelmezésben nem definiál ütközésmenedzsmentet. Ez a működés egy sima ALOHA eljárásnak feleltethető meg, mely során az ütköző tag-ek valamikor a későbbiekben újraadnak. A szimulációt a SimTag ütközésfeloldó protokollok szimu-

lációjára kifejlesztett programjával végeztük: mindenre kiterjedő paraméterkínálatával számos különböző szimuláció futtatását teszi lehetővé.

A szimulációs eredmények értelmezésének kulcsa az algoritmus *Main és Extended módja* közti különbségek alapos ismerete. A két eljárás közötti szembetűnő különbség az időrések kiosztásában van. Míg a Main mód fix 16 időrés/kör paraméterrel dolgozik, addig az Extended mód adaptív módon, dinamikusan, a tag-ek és az ütközések számának függvényében állítja ezt a paramétert működés közben. A tartomány határai az előzetes konfiguráció során adhatók meg.

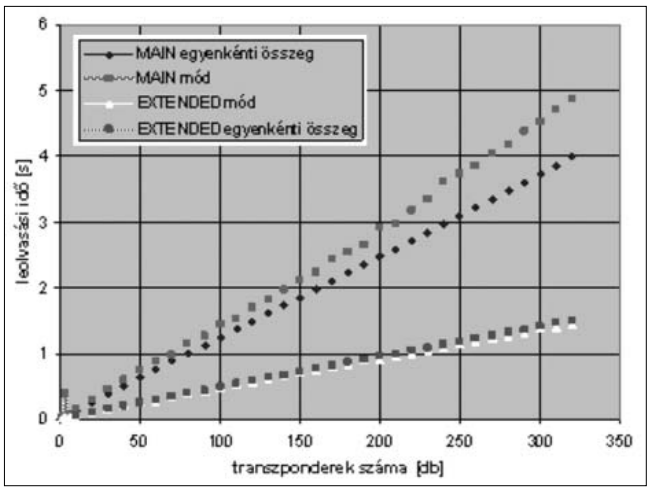
A szimulációkat két fő területre koncentrálva futtattuk le: a teljes leolvasási időt vizsgálva, illetve az időrések számának változtatásával futtatva a programot.

A teljes leolvasási idő vizsgálata

A teljes leolvasási idők szimulációja során megvizsgáltuk, hogyan függ a populáció méretétől az abszolút leolvasási idő rögzített paraméterek mellett Main, illetve Extended módok használata esetén.

Első lépésként egy tesztet futtattunk Main módban, mely 1-től 350-es populációig vizsgálta a transzponderenkénti leolvasási időket. Ezt összegeztük, később – mint egy idealista – viszonyítási alapként szolgált: mekkora lenne a teljes leolvasási idő, ha ez kizárólag a transzponderenkénti leolvasási időkből állna össze, semmiféle járulékos időt nem tartalmazna.

Ezzel szemben áll egy másik teszt eredménye, mely – Main, valamint Extended módokban lefuttatva a szimulátort – tartalmazza 6 teljes leolvasás átlagolt eredményét a transzponderek növekvő populációját tekintve. A teszt eredménye 1-től 320-as populációig 10-es lépésekben egyenként 6 mérés átlagát tartalmazza.



4. ábra
A leolvasási idők alakulása a transzponderek számának függvényében MAIN és EXTENDED módban

Az eredmények egymásra helyezett grafikonjai *(4. ábra)* jól mutatják az egyes algoritmusok közötti alapvető különbségeket.

A transzponderenkénti összeg jó közelítéssel egy lineáris görbe, hiszen a Main módban leolvasott transzponderek összege néhány tíz transzponder felett nem ingadozik nagy mértékben. Ehhez képest a valódi ütközésfeloldó algoritmus (Main módban) szintén lineáris, de meredekebb jellegű görbe. A járulékos kommunikációs idők tehát egyre nagyobb mértékben rakódnak rá. Ennek oka a Main mód működésében keresendő. Az eljárás fix, 16 időrést használ körönként. Ez a nem dinamikus, adaptív algoritmus egyre rosszabb hatásfokkal működik, ahogy a populáció mérete nő.

Ennél lényegesen jobb, gyorsabb működésű az Extended mód. 300-as populációval 1,36 mp alatt végez, míg a Main mód 4,5 mp alatt teszi ezt. Az előbbi algoritmus előnye annál nagyobb, minél nagyobb a populáció, éppen ezért nagy számú tag-eket használó alkalmazásokban érdemes implementálni az Extended módot. Az algoritmus adaptív, dinamikusan változtatja a körönkénti időrések számát (természetesen az előzetes konfigurációtól függően), így sokkal flexibilisebb, képes alkalmazkodni nem csak a transzponderek számához, hanem az ütközések arányához is.

Időrésparaméterek hatásvizsgálata

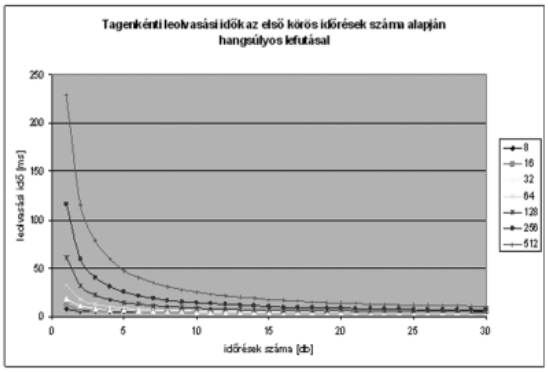
Az időrésparaméterek hatásvizsgálatának szemléltetése előtt érdemes megjegyezni, hogy a szimuláció annak köszönhetően végezhető el, hogy az Extended mód három paraméteren keresztül kínál fel lehetőséget az időrések számának befolyásolására. A kezdő, a minimális, valamint a maximális időrések számával tág határok között konfigurálható az ütközésfeloldó algoritmus, melynek eredményeit a következőkben ismertetem.

A *kezdő* – vagyis az első körben kiosztásra kerülő – *időrések száma* alapjaiban befolyásolhatja a leolvasási időket. Hogy ez miként függ a zónában lévő transzponderek számától, azt az alábbi szimuláció ismerteti.

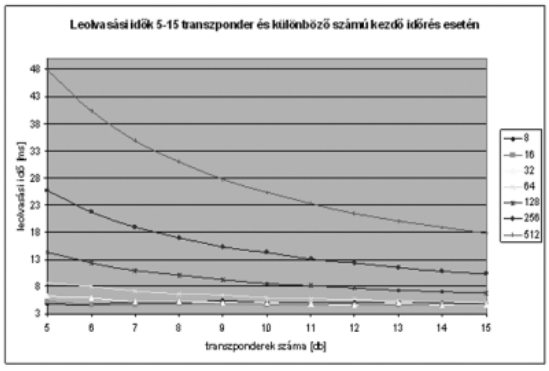
Első lépésként hét szimulációt futtattunk 8, 16, 32, 64, 128, 256, valamint 512 kezdő időrés használatával. A tesztek 320-as populációval készültek, egyenként 25-25 mérés átlagaként. Az összesített eredményeket a következő oldalon az *5-8. ábrák* szemléltetik.

Látható, hogy a több időréses első kör hátránnyal indul kis számú transzponder esetén, ami természetes is, hiszen rengeteg időrés marad kihasználatlanul ez esetben, ami megnöveli az átviteli időt. Ez a hátrány azonban elfogy 150-200-as populációnál, mely felett a több időréses leolvasások jobban teljesítenek. Ugrásszerű a növekedés 512 időrést használata esetén, de a 8-hoz képest már a 64-es időrés szám is szembetűnő növekedést jelent. Az időrések számát tehát célszerű az adott alkalmazáshoz igazítani, hogy a kihasználatlan időrések ne rontsák az algoritmus teljesítményét, ezzel pedig az olvasás időszükségletét. Fontos még egyszer megjegyezni, ezek az eredmények az adott populációt alkotó tag-ek egyenkénti leolvasási idejére vonatkoznak, nem pedig a teljes időre.

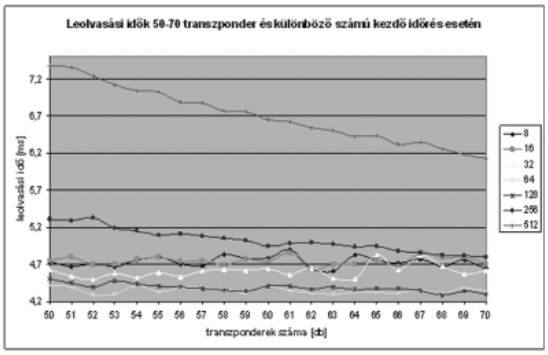
A körönkénti *minimálisan kiosztandó időrések száma* szintén erős befolyásoló tényező. Túlzottan magas értékével rengeteg kihasználatlan időrés keletkezik, alacsonyra állítása ugyanakkor sok ütközést eredményez. Pontos beállítása szintén szimuláció segítségével történhet a rendszerépítés ezen fázisában.



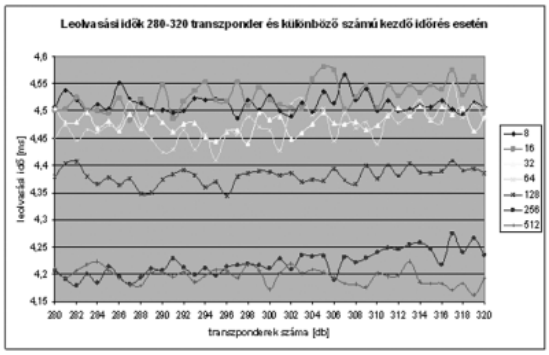
5. ábra
A tag-enkénti leolvasási idők alakulása az első körös időrések száma alapján hangsúlyos lefutással



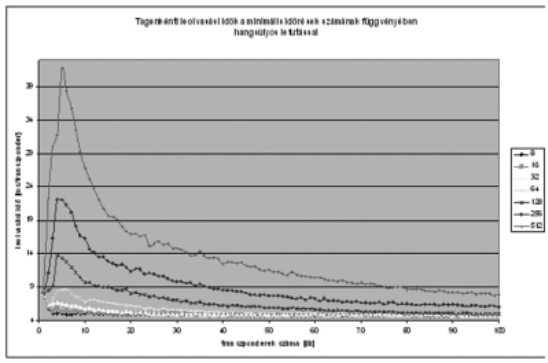
6. ábra
A leolvasási idők alakulása 5-15 transzponderre (kezdő időrések száma szerint)



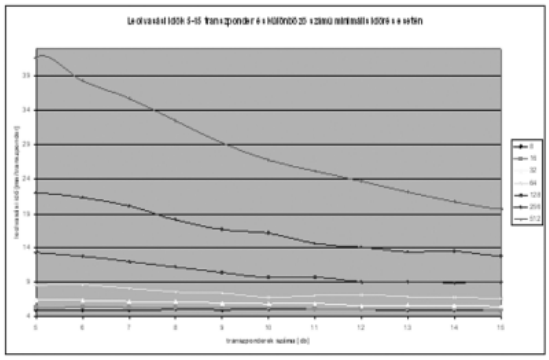
7. ábra
A leolvasási idők alakulása 50-70 transzponderre (kezdő időrések száma szerint)



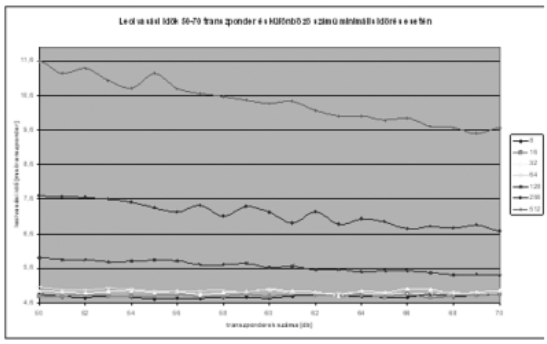
8. ábra
A leolvasási idők alakulása 280-320 transzponderre (kezdő időrések száma szerint)



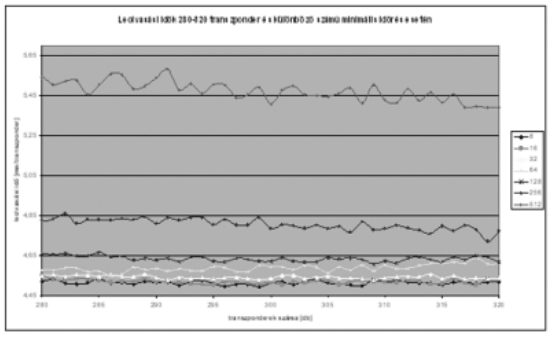
9. ábra
A leolvasási idők alakulása a transzponderek számának függvényében a minimális időrések száma szerint, hangsúlyos végértékkel



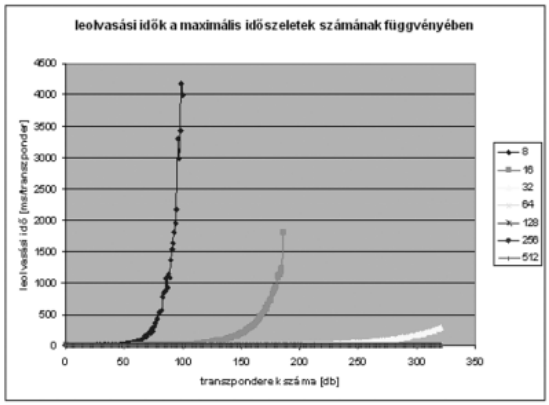
10. ábra
A leolvasási idők alakulása 5-15 transzponderre (minimálisan kiosztandó időrések száma szerint)



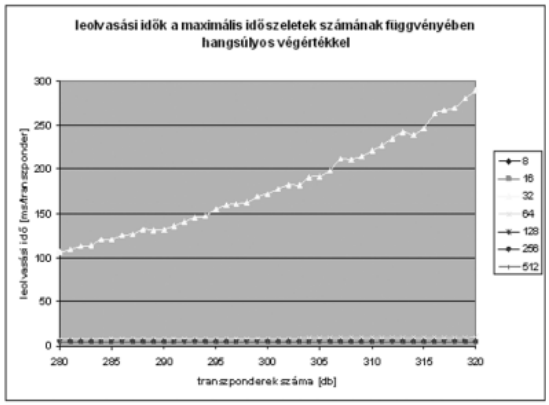
11. ábra
A leolvasási idők alakulása 50-70 transzponderre (minimálisan kiosztandó időrések száma szerint)



12. ábra
A leolvasási idők alakulása 280-320 transzponderre (minimálisan kiosztandó időrések száma szerint)



13. ábra
A leolvasási idők alakulása a transzponderek számának függvényében a maximális időrések száma szerint



14. ábra
A leolvasási idők alakulása a transzponderek számának függvényében a maximális időrések száma szerint hangsúlyos végértékkel

A *következő ábrák (9-12.)* a minimális időrések számának hatásait illusztrálják, annak különböző értékei esetén. Szintén 320 transzponder adatainak átlagából készültek a grafikonok. A vizsgált tartományban nagy meglepetés nincs: több időrés rosszabbul teljesít, a leggyorsabb leolvasást a legkevesebb időrés használata adja. Figyelem: ez csak egy alsó korlát az adaptív algoritmus számára. Az eljárás maga dönt arról, hogy e keretek között milyen számú időrést választ. Nem meglepő tehát, hogy a legjobb eredmény akkor adódik, ha az algoritmus szabad kezet kap.

A *maximálisan kiosztandó időrések számát* vizsgáló szimuláció hasonló az előzőhöz, megvizsgálja mi történik, ha az időrések számát meghatározó algoritmus korlátok közé szorul.

Várható, hogy igen leromlik a teljesítmény kis számú maximális időrés használatával, nagy számú korlát esetén viszont az algoritmus maga dönt a jó választásról. A szimuláció eredményei a *13-14. ábrán* láthatók.

A grafikonok igazolják a feltételezést: 8, valamint 16 időrés használata 50, illetve 150 transzponder felett gyakorlatilag soha véget nem érő leolvasást eredményez. A minimális leolvasási időket több száz transzponder esetén a 64, 128, 256, valamint 512-es beállítások adják, az ennél kevesebb időrést használó tesztekben az idők exponenciális jelleggel nőnek a transzponderek számának növelésével. Az algoritmus tehát optimálisan működik alaphelyzetben, ha tág határok közt tartjuk a működési területét, ezzel elérve a saját maga általi optimális választást, mely a leghatékonyabb működést eredményezi.

A szimuláció eredményeiből világosan látszik, hogy nagyban befolyásolja a kiosztott időrések száma, illetve a rendszerben lévő transzpoderek száma a leolvasási időt. Nyilvánvalóan minél több a transzponder, annál nagyobb az ütközés valószínűsége, és így növekszik a leolvasási idő, illetve minél több időrést oszt ki az algoritmus az egyes transzpondereknek, annál többször lesz kihasználatlan a csatorna, illetve megtalálható az az optimális időrés populációnként, ahogyan a legjobb hatásfokkal futhat az algoritmus.

Fontos megjegyezni, hogy a szoftver működését a Gemplus és a TagSys mérnökei tesztelték, eredményeiket saját protokolljaikban is felhasználják. A szimuláció tényleges rendszerépítési- és méretezési kérdésekre is választ ad, ugyanakkor általánosságában is tárgyalja a felmerülő problémakört.

5. ISO és EPCglobal szabványok az RFID területén

Végül egy nagyon rövid, nem átfogó leírást adunk az RFID szabványokkal kapcsolatban.

A két legjelentősebb ellátási láncokkal kapcsolatos szabvány kidolgozója az ISO (International Standarization Organization) és az EPCglobal Inc. A legtöbb nemzeti és ipari szabvány ezekre épül. Az ISO 18000-es számú szabványa foglalkozik. Az EPC szabványok nevüket az Electronic Product Code-ról, azaz az elektronikus termékkódról kapták, mellyel minden terméket egyedileg lehet azonmosítani, az EPC eszközöket pedig osztályokba és generációkba sorolták: A Class1 besorolás passzív, csak olvasható, úgynevezett backscatter tag-eket takar egyszer írható, nem felejtő memóriával. A Gen2 generáció pedig a 860-960 MHz-es működési sávra és a 96-256 bites azonosítóformátumra utal.

6. Összefoglalás

A bemutatott ütközésfeloldó algoritmusok jó kiindulási alapként szolgálnak a jövőben még gyorsabb és hatékonyabb többes leolvasási algoritmusok fejlesztésére. Természetesen alkalmazási környezettől függ, hogy milyen algoritmust célszerű használni a speciális igényeknek megfelelően. Az azonban elengedhetetlenül fontos, hogy megfeleljenek a kidolgozott eljárások valamelyik szabványnak, hiszen csak így biztosítható a széleskörű felhasználhatóság. Mivel az EPCglobal Gen2-es szabványát átadták az ISO-nak, hogy az a 18000-es szabványcsalád része lehessen, célszerű ennek alapján kidolgozni, fejleszteni, esetleg javítani a meglévő ütközésfeloldó algoritmusokat.

Ha kellően megbízható és jó teljesítőképességű algoritmus készül el, akkor azt a későbbiekben széles körben lehet használni kezdve egy hipermarketben történő vásárlást követő fizetéstől egy raktárbázis tartalmának lekérdezéséig. Ehhez nagyon jó eszközként szolgálnak a külöféle szimulációs szoftverek, melyekkel akár teljes rendszerek megtervezése is lehetséges. A cél minden RFID-vel kapcsolatos fejlesztés során az, hogy mindenki által használható, együttműködő rendszerek jöjjenek létre, várhatóan tehát ez fogja vezérelni a jövőben a többes leolvasási fejlesztéseket is.

A kutatást a Mobil Innovációs Laboratóriumban végeztük a Nemzeti Kutatási és Technológiai Hivatal Jedlik Ányos programjában a rádiófrekvenciás azonosítás ipari továbbfejlesztési lehetőségeinek témájában.

Irodalom

[1] K. Finkenzeller,
RFID Handbook,
Swadlincote: Wiley & Sons Ltd. 2003.

[2] S. M. Birari,
Mitigating the Reader Collision Problem in RFID Networks with Mobile Readers (Februar 2006) [Online].
http://www.it.iitb.ac.in/~shailesh/ThirdStage.ppt

[3] S. W. Lee,
„A Multiple Access Algorithm for Passive RFID tags,"
Thesis, School of Electrical and Electronic Engineering,
College of Engineering, Yonsei University,
Seoul, South Korea, 2005.

[4] BME-HIT, RFID kutatócsoport:
RFID rendszerek felhasználása,
http://rfid.answare.hu