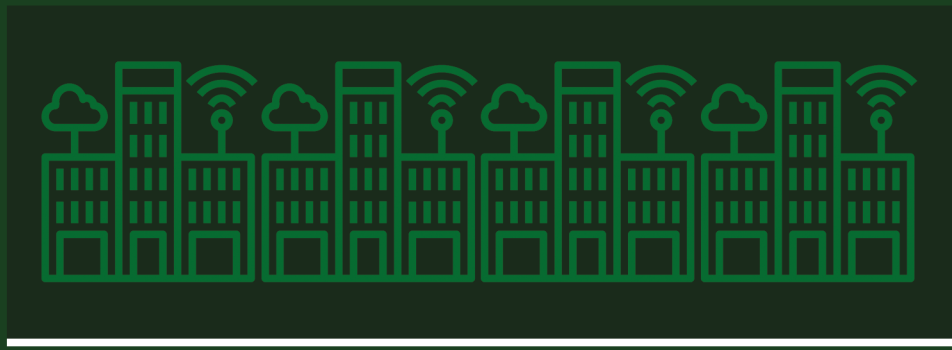
OKOSVÁROS-TECHNOLÓGIÁK XIII.

Vezetéknélküli szenzorhálózatok: az elmélettől a gyakorlatig



Fehér Gábor – Vida Rolland – Jakab László

Nemzeti
Közszolgálati Egyetem
Budapest

SZÉCHENYI 2020

Magyarország Kormánya

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

OKOSVÁROS-TECHNOLÓGIÁK
A technológia fejlődésének irányai és hatása
XIII. kötet

Sorozatszerkesztő:
Sallai Gyula

Fehér Gábor – Vida Rolland – Jakab László

VEZETÉKNÉLKÜLI SZENZORHÁLÓZATOK: AZ ELMÉLETTŐL A GYAKORLATIG

Nemzeti Közszolgálati Egyetem
Közigazgatási Továbbképzési Intézet
Budapest, 2020

A kötet a Nemzeti Közszolgálati Egyetem **KÖFOP-2.1.2-VEKOP-15-2016-00001 „A jó kormányzást megalapozó közszolgálat-fejlesztés”** projektje keretében, a Budapesti Műszaki és Gazdaságtudományi Egyetem Villamosmérnöki és Informatikai Karán létesült **„Okos város – okos közigazgatás”** kutatóműhelyben (2017/162/BME-VIK) készült.

Szerkesztő:
Vida Rolland, egyetemi docens, BME-VIK

Szakmai lektor:
Sallai Gyula, professor emeritus, BME-VIK

A kézirat lezárásának dátuma:
2018. március 31.

TARTALOM

1. BEVEZETÉS

Az ezt megelőző, „Szenzorok és kommunikációs technológiáik" című kismonográfiában főként az érzékelés tárgyát képező fizikai jelenségekkel, illetve az egyedi érzékelőkkel foglalkoztunk. Bemutattuk az egyedi szenzorok felépítését, működését, és az általuk használt különböző vezetékes és vezeték nélküli kommunikációs technológiákat. Ebben a kismonográfiában az egyedi szenzorok helyett az egymással együttműködő szenzor csomópontok által alkotott *vezeték nélküli szenzorhálózatok (WSN, Wireless Sensor Networks)* sajátságait, érdekesebb aspektusait mutatjuk be.

Elemezzük az egyugrásos és több-ugrásos hálózatok közötti különbségeket, meghatározzuk azt, hogy mit is értünk idővezérelt, eseményvezérelt, illetve lekérdezés alapú működési modellen. Bemutatjuk, hogy hogyan tudjuk minél hatékonyabban kihasználni az egyes szenzorok energiatartalékait úgy, hogy minél kiegyensúlyozottabb legyen a terhelés a hálózaton belül és minél hosszabb legyen a hálózat élettartama. Ehhez persze definiáljuk azt is, hogy mit is jelent a hálózat élettartama, milyen különböző meghatározásokat lehet figyelembe venni ezzel kapcsolatban.

Beszélünk a különböző alvásvezérlő algoritmusokról és a közeghozzáférést szabályozó protokollokról. Elemezzük azt is, hogy hogyan lehet a vezeték nélküli szenzorhálózatok topológiáját változtatni úgy, hogy a több-ugrásos rádiós kommunikáció folyamatosan biztosított legyen. Bemutatunk a vezeték nélküli szenzorhálózatok speciális igényeire szabott néhány útvonalválasztó és adat-aggregációs megoldást. Elemezzük azt, hogy milyen előnyökkel és hátrányokkal jár a szenzorok és a bázisállomások esetleges mozgatása, különböző mozgás modellek, illetve mobilitási stratégiák esetében. Mindemellett röviden bemutatunk néhány olyan szimulációs keretrendszert és teszthálózatot, melyek segítségével valósághűen lehet modellezni és tesztelni a vezeték nélküli szenzorhálózatokra szabott új kommunikációs protokollok működését és azok hatékonyságát.

A kismonográfia második felében, az elméleti bevezető után egy konkrét gyakorlati esettanulmányként részletesen bemutatjuk a LoRaWAN hálózatokat, mint a napjainkban használt talán legnépszerűbb vezeték nélküli szenzorhálózati megoldást. Előbb beszélünk a szabadalmaztatott, speciális modulációra épülő LoRa rádióról, bemutatjuk milyen kiterjedésű hálózatokat lehet kialakítani különböző környezeti paraméterek függvényében, mennyire más egy vidéki, sík területen kialakított hálózat egy városi, magas épületeket tartalmazó területen kiépített LoRaWAN hálózattól, és hogyan lehet modellezni a rádiós jelterjedést, illetve az átvitel minőségét befolyásoló hatásokat a LoRa rádió speciális esetére.

Elemezzük azt is, hogy mennyire befolyásolja a LoRa által használt ALOHA típusú, időosztás nélküli közeghozzáférési megoldás a hálózat lehetséges maximális méretét, milyen terjedési időkkel lehet számolni, és milyen következményei vannak az úgynevezett 1%-os csatornahasználati korlátozásnak. Ezek után bemutatjuk a

LoRaWAN hálózatok építő elemeit, a LoRaWAN végpontokat, az átjárókat, a hálózati szervert és az alkalmazás szervert. Ismertetjük a LoRaWAN adatcsomagok felépítését és az adatküldések ütemezését is. Röviden érintjük a LoRaWAN hálózatok biztonsági kérdéseit is, de a szenzorhálózatok biztonságával egy következő kismonográfia fog részletesen foglalkozni.

A szenzorhálózatok területével nagyon sok kutatócsoport foglalkozott, ennek megfelelően a monográfia irodalomjegyzéke is több mint 50 tételt tartalmaz, melyek között számos saját cikk is megtalálható.

A Szerzők[1]

[1] *Dr. Vida Rolland*, a BME Villamosmérnöki és Informatikai Kara Távközlési és Médiainformatikai Tanszékének egyetemi docense, a Nagysebességű Hálózatok (HSN) Laboratórium vezetője. PhD fokozatát a párizsi Université Pierre et Marie Curie egyetemen szerezte 2003-ban. Kutatási területei közé tartoznak a szenzorhálózatok, a járműhálózatok, a Tárgyak Internete és ezek alkalmazásai az okosváros-rendszerekben. Az BME-n futó Okos város villamosmérnöki mellékspecializáció felelőse. Tagja az IEEE Smart Cities Steering Committee-nek, az IEEE Sensors Council Administrative Committee-nek, az IEEE Sensors 2019 és IEEE Sensors 2020 nemzetközi konferenciák tudományos programbizottságának elnöke, az IEEE International Smart Cities Conference 2020 nemzetközi tudományos konferencia társelnöke, és az IEEE Sensors Letters tudományos folyóirat szerkesztője.

Dr. Fehér Gábor, a BME Villamosmérnöki és Informatikai Kara Távközlési és Médiainformatikai Tanszékének egyetemi docense, tanszékvezetőhelyettes. PhD fokozatát a BME-n 2004-ben szerezte. Kutatási területe a Tárgyak Internete, szenzorhálózatok kommunikációs technológiái, illetve a hálózat- és információbiztonság.

Prof. Dr. Jakab László, a BME Villamosmérnöki és Informatikai Karának egyetemi tanára, dékánja (2016-2019), az Okos város - okos közigazgatás kutatóműhely szakmai vezetője. Villamosmérnök diplomáját a BME-n szerezte meg 1981-ben. A műszaki tudomány kandidátusa 1992-ben, habilitált doktor 2013-ban, az MTA doktora címet pedig 2014-ben szerezte meg. Jelenleg az Elektronikai Technológia tanszék munkatársa.

2. SZENZORHÁLÓZATI ARCHITEKTÚRÁK

Egy vezeték nélküli szenzorhálózat ideális felépítése nagyon sok szemponttól függ. Fontos az, hogy mekkora a megfigyelni kívánt terület, illetve az, hogy a területen belül hogyan lehet telepíteni a szenzorokat, illetve a bázisállomás(oka)t. Be lehet-e menni a területre például, és szisztematikusan tudjuk-e telepíteni a szenzorokat a rendelkezésre álló villanyoszlopokra például, vagy a parkolóhelyek aszfaltcsíkjaiba, vagy különböző okok miatt (például egy veszélyes, radioaktív környezet vagy egy földrengés utáni összeomlott városrész) csak a szenzorok „véletlenszerű" kiszórása biztosítható, mondjuk egy repülőből vagy egy drónból kidobva azokat az érintett terület fölött. A kialakítandó vezeték nélküli szenzorhálózat architektúráját nagymértékben befolyásolja az is, hogy milyen rádiós kommunikációs technológiát kívánunk alkalmazni, szükség van-e esetleg átjátszók beiktatására a szenzorok és a bázisállomások közé, vagy biztosítható az, hogy a szenzorok közvetlenül elküldhessék adataikat a bázisállomások felé, akár nagy kommunikációs távolságok áthidalásával is.

Fontos szempont az is, hogy milyen követelményei vannak az alkalmazásnak, mely a szenzor csomópontok méréseit használja majd fel, szükség van-e például valós idejű adatbegyűjtésre, feldolgozásra és szükség esetén beavatkozásra, vagy csak jóval hosszabb időintervallumban kell gondolkodni az adatbegyűjtést és a feldolgozást illetően. Szintén az alkalmazás igényei szabják meg, hogy mennyire kell megbízhatónak lennie a szenzoroknak, illetve a szenzorhálózaton belüli kommunikációnak. Ha nagymértékű megbízhatóság az elvárás, akkor a hálózati architektúra kialakításánál ezt figyelembe kell venni, például nagyobb számú szenzor redundáns telepítésével, az alvásvezérlő algoritmusok megfelelő paraméterezésével, vagy tárolást és ismétlést *(store and forward)* végző eszközök beiktatásával, melyek szükség esetén újra tudják küldeni a szenzorok mérési adatait.

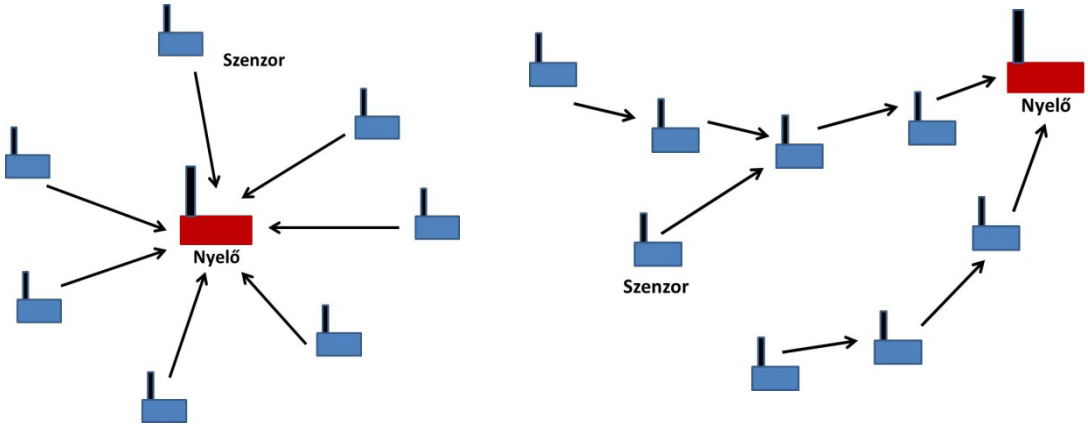
Szintén alakíthatja a hálózati architektúrát az, hogy milyen elvárásaink vannak az energiafelhasználást, illetve az ehhez kapcsolódó üzemeltetési és karbantartási költségeket illetően. Sok esetben az a cél, hogy a telepített vezeték nélküli szenzorhálózat éveken keresztül emberi felügyelet és beavatkozás nélkül működhessen, megbízhatóan és az alkalmazás elvárásainak megfelelően. Ehhez viszont az kell, hogy a legszűkösebb erőforrást jelentő energiával megfelelő módon gazdálkodjanak. Alapvetően befolyásolhatja például a szenzorhálózatok architektúráját az, ha az alkalmazott szenzorok képesek valamilyen „energia-aratási" *(energy harvesting)* megoldással, napenergiából, hőenergiából vagy például mozgási energiából feltölteni időről időre az energiakészleteiket.

A következőkben röviden bemutatjuk, hogy milyen lehetőségek vannak egy vezeték nélküli szenzorhálózat architektúrájának a kialakításában, és milyen előnyei, valamint hátrányai vannak a különböző lehetséges megoldásoknak.

2.1. EGYUGRÁSOS ÉS TÖBBUGRÁSOS KOMMUNIKÁCIÓ

Egy vezeték nélküli szenzorhálózat kialakításánál nagyon fontos szempont az, hogy milyen méretű a terület, melyet meg szeretnénk figyelni, hogy hány érzékelőt és hány bázisállomást szeretnénk kihelyezni, illetve az, hogy milyen rádiós kommunikációs megoldást akarunk vagy tudunk használni. A kommunikációs architektúrák tekintetében két különböző megoldást szoktak megemlíteni (2.1. ábra).

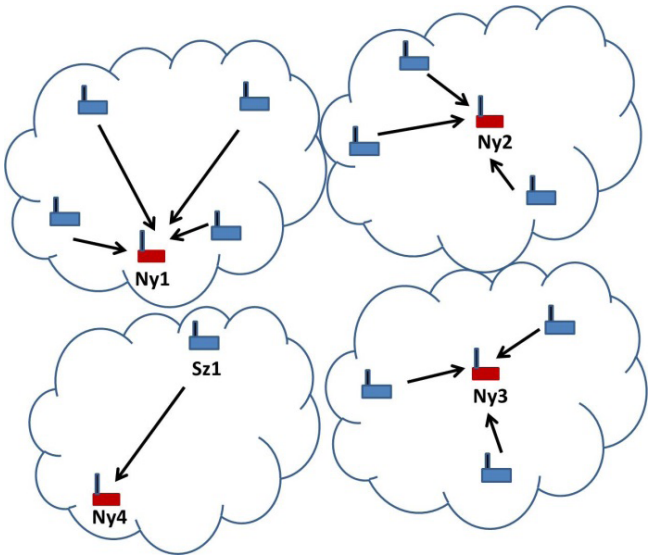
A legegyszerűbb megoldás az, ha egyetlen bázisállomást helyezünk ki, lehetőleg a megfigyelt terület közepére, és a szenzorok közvetlenül ennek a bázisállomásnak küldik a méréseik eredményét. Ezt a megoldást hívjuk *„egyugrásos" (single-hop)* hálózatnak. Előnye, hogy nagyon egyszerű és – viszonylag – megbízható, hiszen minden szenzor saját maga szervezi az adatküldését, és nem függ a környezetében levő többi szenzortól. További előny, hogy a bázisközelében levő szenzorok kis energiafogyasztással tudnak kommunikálni, feltéve, ha állítható a rádiós interfészen a küldési teljesítmény. Ugyanakkor azonban a bázisállomástól távol elhelyezkedő szenzorok jóval több energiát használnak fel a küldésre, így azok hamarabb lemerülnek. Erre megoldás lehet az, ha az egyetlen bázisállomást időközönként elmozgatjuk, kiegyenlítve ezáltal valamennyire az energiafogyasztást a hálózaton belül. Erre a megoldásra visszatérünk majd részletesebben a mobil bázisállomásokat bemutató 3.1-es alfejezetben.



2.1. ábra: Egyugrásos és többugrásos kommunikáció *(saját ábra)*

Szintén orvosolhatja a távoli szenzorok nagy energiaigényét az, ha nem egy, hanem több bázisállomást tudunk kihelyezni a területen belül (2.2. ábra). Ebben az esetben minden szenzor eldöntheti, valamilyen algoritmus vagy protokoll segítségével, hogy melyik bázisállomásnak fogja majd küldeni az adatait. Alapesetben nyilvánvalóan a hozzá fizikailag legközelebb álló bázisállomást fogja választani, de bizonyos esetekben más szempontokat is figyelembe vehet, például a bázisállomások terheltségét vagy az egyes bázisállomásokkal való kommunikáció megbízhatóságát (mely nem feltétlenül csak a fizikai távolságtól függ).

Példaként láthatjuk, hogy a 2.2. ábrán négy különböző bázisállomás van elhelyezve a területen (*Ny1,.., Ny4*), a szenzorok pedig eldönthetik, hogy melyik bázisállomásnak küldik majd az adataikat. Alapvetően a fizikai távolság a meghatározó, de a bázisállomások terheltsége is fontos lehet. Az ábrán pont erre az esetre látunk egy példát, hiszen az *Sz1* szenzorhoz legközelebb az *Ny1* bázisállomás áll, viszont az már eléggé terhelt, hiszen 4 másik szenzor is neki küldi az adatait. Ezért az *Sz1* szenzor úgy dönt, hogy inkább a tőle távolabb levő *Ny4* bázisállomás felé küldi a saját méréseinek eredményét.



2.2 ábra: Egyugrásos kommunikáció több bázisállomás esetén *(saját ábra)*

Az egyugrásos hálózatokat korábban jellemzően a kisebb területek lefedésénél alkalmazták. Az *LPWAN (Low Power Wide Area Network)* rádiós technológiák (például LoRa vagy SigFox) megjelenésével azonban lehetővé vált nagy, akár több tíz kilométer átmérőjű területek lefedése is közvetlen egyugrásos kommunikációval a szenzorok és a bázisállomás között. Ezeknél a technológiáknál viszont nagyon komoly korlátozások vannak az átvihető adatok mennyiségét illetően, illetve az átviteli sebesség is nagyon alacsony. Ennek megfelelően az LPWAN technológia csak olyan alkalmazások esetén használható, amelyeknél ezek a korlátozások elfogadhatóak.

A *többugrásos (multi-hop)* kommunikáció (2.1. ábra) egy megoldást jelenthet olyan esetekre, ahol a megfigyelni kívánt terület túl nagy, és a vezeték nélküli szenzorok egy része túl távol van ahhoz, hogy közvetlenül kommunikálhasson a bázisállomással. Ilyen helyzetben a távoli szenzorok a bázisállomáshoz közelebb álló szenzorok segítségével érhetik el a bázisállomást, egy többugrásos útvonal mentén, valamilyen útvonalválasztó algoritmus segítségével. A szenzorhálózatok speciális igényeire kidolgozott útvonalválasztó megoldásokra a következő alfejezetben még visszatérünk.

A többugrásos kommunikáció hátránya, hogy viszonylag nehézkes a szenzorok együttműködését hatékonyan megszervezni, hiszen a szenzorok az energiahatékonyságot szem előtt tartva az idő nagy részében alvó állapotban vannak, így pedig nem lesznek képesek más szenzorok adatait átvenni és továbbküldeni a kialakított útvonal mentén. Ennek a feladatnak a kihívásaira külön kitérünk majd az alvásvezérlő algoritmusokról szóló 2.6. alfejezetben.

Mindemellett szintén problémás lehet egy többugrásos hálózat esetén az is, hogy a csomagvesztésnek viszonylag nagy lehet az esélye az útvonal mentén, hiszen egy vezeték nélküli kommunikáció esetén mindig kérdéses a csatornahozzáférés, a lehetséges rádiós interferenciák, a többutas terjedés vagy a nagymértékű csillapítás megfelelő kezelése nagyobb távolságok áthidalása esetén. Minél hosszabb tehát az útvonal, azaz minél több ugrásból áll, annál inkább elképzelhető, hogy egy adott csomag sérül vagy elveszik valamelyik ugrás során.

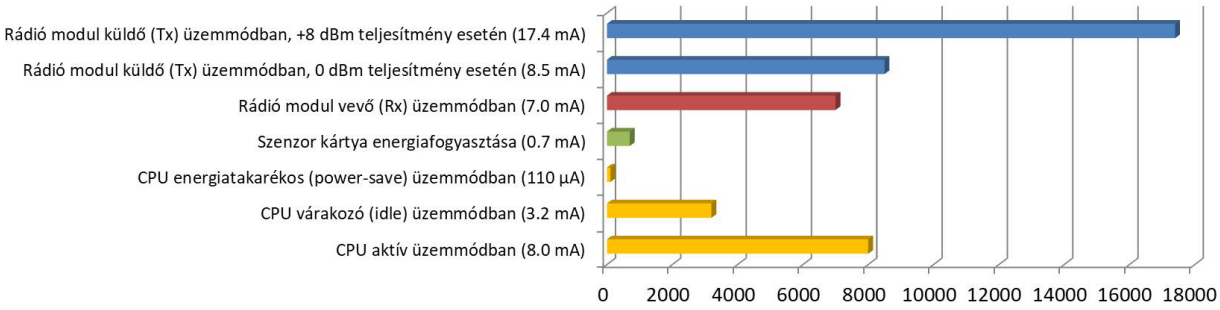
Ami viszont a hálózatban levő szenzorok energiafogyasztását illeti [Pesovic2010], a többugrásos hálózatok pont ellenkező módon viselkednek, mint azok a hálózatok, ahol mindenki közvetlenül a bázisállomással kommunikál. Több ugrás esetén a távoli szenzorok energiafogyasztása alacsony, mivel csak egy másik közeli szenzorhoz kell eljuttatni a csomagot, az pedig majd gondoskodik a továbbküldésről. Ehhez természetesen itt is kell az, hogy állítható legyen a szenzorok küldési teljesítménye. Másfelől viszont a bázisállomás közelében levő szenzorok hamarabb lemerülhetnek, hiszen az összes távoli szenzor forgalmának az elvezetésében részt kell venniük.

Természetesen ez a terhelés is enyhíthető azzal, ha több bázisállomást telepítünk a hálózatba, és minden szenzor a hozzá legközelebb álló, azaz a legkevesebb lépésben megközelíthető bázisállomáshoz küldi a saját adatait. Másfelől egy bázisállomás esetén is elképzelhető az, hogy a bázisállomást időről időre elmozdítjuk a hálózaton belül csak azért, hogy a közelében elhelyezkedő szenzorokat mentesítsük egy idő után a több-ugrásos kommunikációs útvonalak kialakításának következtében rájuk nehezedő terheléstől. De amint már mondtuk, a bázisállomások mobilitását részletező 3.1-es alfejezetben még részletesebben visszatérünk erre a kérdésre.

2.2. ENERGIATAKARÉKOSSÁG

A szenzorok energiatartalékai, a kis méretükből és alacsony árukból adódóan, általában erősen korlátozottak. Másfelől viszont a vezeték nélküli szenzorhálózatokkal kapcsolatosan gyakori elvárás az autonóm, felügyelet és emberi beavatkozás nélküli működés. A kezdeti telepítés után nem elfogadott az, hogy a szenzorokat működtető elemeket valaki időről időre lecserélje, vagy az akkumulátorokat feltöltse. Egy aszfaltba épített szenzort például, mely érzékeli a felette parkoló autókat vagy számolja az áthaladó forgalmat, nem célszerű néhány hetente-havonta kiemelni és elemet cserélni benne, ez túl költséges és időigényes lenne. Sőt, sok esetben (az emberi egészségre káros környezetbe telepített szenzorok esetén például – lásd sugárzás mérése egy atomreaktor bizonyos területein) az elemcsere nem is lenne megoldható. Ezért kiemelten fontos, hogy a szenzorok energiahatékonyan működjenek [Rault2014] [Heinzelman1999], akár 5-10 éven keresztül is ugyanarról az apró elemről, beavatkozás nélkül.

A szenzor a működésének különböző fázisaiban használja az energiakészletét: energia szükséges magához az érzékeléshez, a mért adatok tárolásához vagy akár azok előfeldolgozásához, a legtöbb energiát viszont kétségtelenül a rádiós kommunikáció igényli. A 2.3. ábra ezt szemlélteti egy Mica2 szenzor csomópont esetén, melynek energiafogyasztását a PowerTOSSIM szimulátor segítségével modellezték [Shnayder2004].



2.3 ábra: Egy Mica2 szenzor energiafogyasztása, µA-ben

(saját ábra, a [Shnayder 2004]-ben található eredmények alapján)

Ennek tudatában az energiahatékony működést célzó megoldások jelentős része a kommunikációs folyamat optimalizálásával próbálkozik. Lehet energiát spórolni azzal, ha csökkentjük a küldések gyakoriságát, feltéve, hogy az alkalmazás igényei megengedik ezt. De csökkenthetjük az adatok mennyiségét is, aggregálva a mérési eredményeket már az adott szenzornál, vagy akár a többugrásos útvonal mentén, bizonyos több erőforrással rendelkező dedikált aggregációs pontoknál. De csökkenthető a kommunikációs távolság is, azaz a szenzor és bázisállomás közötti távolság egyugrásos hálózat esetén, vagy az útvonal hossza többugrásos hálózatoknál. Ezekre a megoldásokra részletesebben is kitérünk a következő alfejezetekben.

A szenzorhálózat energiahatékonyságának egyik gyakran használt mérőszáma a hálózat élettartama – minél hatékonyabb egy adott megoldás, annál hosszabb élettartamot lehet a hálózat számára garantálni. De mit is jelent pontosan egy vezeték nélküli szenzorhálózat élettartama? Erre a kérdésre, bármennyire is furcsának tűnik, nem egyértelmű és nem egyszerű a válasz, ezért egy külön alfejezetet szentelünk neki.

2.3. A SZENZORHÁLÓZATOK ÉLETTARTAMA

Egy vezeték nélküli szenzorhálózat élettartamát sokan sokféleképpen definiálták. Tulajdonképpen elmondhatjuk azt, hogy az adott alkalmazás igényei szabják meg azt, hogy meddig is tekinthető egy adott hálózat „élőnek", „életben lévőnek". Talán legegyszerűbben úgy fogalmazhatnák ezt meg, hogy egy adott alkalmazás szempontjából egy szenzorhálózat addig tekinthető élőnek, ameddig a hálózat el tudja látni az érzékelési és adatküldési feladatát, az alkalmazás igényeinek megfelelően. Az egyes alkalmazások igényei viszont nagyon változóak lehetnek, ezért a hálózat élettartama is többféleképpen értelmezhető.

Bizonyos alkalmazások megkövetelik, hogy az összes telepített szenzor rendeltetésszerűen működjön, és a telepítéskor definiált módon és rendszerességgel küldjön mérési adatokat az adott érzékelni kívánt jelenségről. Ha tehát egyetlen szenzor is meghibásodik, vagy lemerül az akkumulátora, megrongálják vagy esetleg ellopják, akkor a teljes hálózatot „halottnak" tekintik, hiszen a hálózat már nem tudja azt a szintű szolgáltatást nyújtani, amit elvárnak tőle. Az ilyen jellegű alkalmazások viszont elég ritkák, hiszen a szenzorokról egy alapvető feltételezés az, hogy kicsik, olcsók, és emiatt sokat lehet belőlük telepíteni, megfelelően redundáns módon, hogy egy-egy szenzor kiesése esetén az alkalmazás igényeit továbbra is ki lehessen elégíteni. A vezeték nélküli szenzorhálózatok filozófiájába valamilyen szinten bele van tehát kódolva a meghibásodás, vagy az egyes szenzorok akkumulátorainak a lemerülése, az a fajta abszolút, 100%-os rendelkezésre állás tehát általában nem elvárás.

Sokkal elterjedtebb az a megközelítés, amelyik megengedi a szenzorok egy bizonyos százalékának a kiesését, de meghatároz egy adott korlátot, küszöbértéket, ami alatt már nem tekinti megfelelő mértékűnek a hálózat által nyújtott szolgáltatást, és inkább „halottnak" nyilvánítja a hálózatot. Ez a küszöbérték lehet például az, hogy a szenzorok legalább 90%-a életben legyen és rendeltetésszerűen működjön, bár itt a 90%-os értéket csak egy példának kell tekinteni. A pontos érték nyilvánvalóan nagyon sok mindentől függ, az alkalmazás igényeitől például, de attól is, hogy az eredeti hálózat telepítésénél milyen mértékű redundanciát alkalmaztak. Minél nagyobb a redundancia, annál több szenzor eshet ki a hálózatból úgy, hogy az alkalmazás elvárásait továbbra is ki lehet szolgálni.

Mindazonáltal, mivel a legtöbb esetben a szenzorok telepítése nem teljesen egyenletes a megfigyelt területen belül, az se mindegy, hogy pontosan mely szenzorok esnek ki, merülnek le a hálózatban. Bizonyos szenzorok fontosabbak lehetnek a hálózat egésze szempontjából, mint mások, hiszen lehetnek olyan szenzorok, melyeket pótolhatnak szomszédos érzékelők, más szenzorokat viszont nem. És ebből következik egy másik megközelítés is

a szenzorhálózat élettartamát illetően, amely nem az egyes szenzorok rendeltetésszerű működéséhez köti mindezt, hanem az érzékelési feladat ellátásához. Ennek megfelelően lehet úgy is tekinteni például a hálózat élettartamára, mint az olyan időszakra ameddig a teljes megfigyelni kívánt terület le van fedve legalább egy szenzor által, azaz nincsenek fehér foltok a lefedettségi térképen (az érzékelést illetően), illetve nincsenek olyan események, melyeket legalább egy szenzor ne detektált volna.

Mindemellett persze a teljes lefedettséget és/vagy a minden egyes esemény megbízható érzékelését megkövetelő megközelítés se feltétlenül jellemző minden esetre, minden alkalmazásra. Ha például egy városban több száz szenzort helyeztünk ki különböző utcákban a hőmérséklet vagy a páratartalom mérésére, de egy adott szenzor meghibásodik vagy lemerül, akkor megtörténhet, hogy egy adott utcából nem kapunk majd mérési eredményeket. Az érzékelési lefedettség nem lesz tehát teljes, de ettől függetlenül a hálózat továbbra is használható, hiszen a több szenzor mérése alapján is fel lehet állítani a város hőtérképét, a hiányzó adatot pedig akár interpolálhatjuk is a többi szenzor mérési eredményeiből.

Ebből kifolyólag tehát az érzékelési lefedettségre vonatkozó élettartam definíciók se követelnek meg általában teljes lefedettséget, hanem inkább azt kérik, hogy a megfigyelni kívánt terület egy „jelentős" része, mondjuk 90%-a, legyen lefedve szenzorok által, és ebben az esetben a hálózatot élőnek, megfelelően működőnek tartják. A lefedettségi százalék se feltétlenül egy jó mérőszám minden esetben, hiszen nagyon nem mindegy, hogy az a 10 százaléknyi lefedetlen terület hol helyezkedik el. Lehet, hogy ez a lefedetlen rész a megfigyelt terület szélein található, ahol amúgy se feltétlenül olyan fontosak a mérési eredmények, vagy ahol amúgy se nagyon történnek olyan események, melyeket monitorozni kellene. Másfelől viszont az is megtörténhet, hogy éppen a megfigyelt terület közepe válik lefedetlenné, ahonnan nagyon fontos lenne adatokat gyűjteni, vagy ahol nagyon sok olyan esemény történik, amit monitorozni kell.

Az érzékelési lefedettséget különben alapvetően másként kell értelmezni a megfigyelt fizikai paraméter vagy esemény függvényében. A szenzorok bizonyos gyakorisággal mérhetnek egy adott paramétert, és bizonyos gyakorisággal elküldik a mért adataikat. Egy hőmérsékletérzékelésnél viszont viszonylag lassan változik a mért érték, ezért ha ki is marad néhány mérés, vagy elveszik egy üzenet, a kieső adatot könnyen lehet azért pótolni, akár egy egyszerű interpolálással.

Vannak azonban olyan monitorozott jelenségek is, melyek pontszerűen jelennek meg, majd nagyon gyorsan megszűnnek. Ilyen például a mozgásérzékelés, vagy akár a zajérzékelés is. Itt gyakorlatilag folyamatos érzékelésre lenne szükség a teljes területen, ha minden eseményt biztosan észlelni szeretnénk. Ezt a folyamatos érzékelést a szenzorok viszont általában nem képesek biztosítani, hiszen nincs meg hozzá a szükséges energiatartalékuk.

A helyzetet könnyíti valamelyest, ha redundánsan, több szenzort helyezünk ki egymáshoz viszonylag közel úgy, hogy a szenzorok bármelyike képes legyen érzékelni ugyanazokat az eseményeket. Ezután pedig a szenzorok valamilyen alvásvezérlő algoritmus szerint periodikus időközönként váltják egymást, azaz a szenzorok nagy része folyamatosan alvó állapotban tud maradni, energiát spórolva, de mindig akad legalább egy szenzor, amelyik érzékeli a környezetet. Ezzel a megoldással jelentősen csökkenthető a szenzorok energiafelhasználása, de a folyamatos monitorozás így is nehezen biztosítható hosszútávon. Ezekben az esetekben a gondot az jelenti, hogy egy kieső mérési adat nagyon nehezen, vagy egyáltalán nem regenerálható a korábbi, illetve a későbbi eredményekből. Egy egyszerű példával élve, ha egy szenzor 10 másodpercenként felébred, érzékeli a zajszintet, majd elmegy aludni, egyáltalán nem biztos az, hogy egy nagyon magas zajszintet generáló eseményt érzékelni tud majd akkor, ha ez az esemény nagyon rövid ideig tart.

Az események érzékelése nagyon fontos célja lehet különböző szenzorhálózati alkalmazásoknak, ennek köszönhetően pedig vannak olyan helyzetek is, amikor a hálózat élettartamát az események teljes mértékű érzékeléséhez kötik. Egyszerűbben megfogalmazva, a vezeték nélküli szenzorhálózattal kapcsolatban elvárás az, hogy minden egyes eseményt érzékeljen. Ha egyetlen olyan esemény is történik, melyet különböző okok miatt a szenzorok nem érzékeltek, akkor a hálózatot „halottnak" tekintik, hiszen nem teljesítette az elvárt szolgáltatást. Ez azonban ismét egy elég „merev" definíció, az alkalmazások többségében azért elfogadható az, ha egy-egy eseményt néha nem detektálunk, feltéve, hogy az események jelentős részét azért sikerül megfelelő módon érzékelni.

Végül érdemes arról is szót ejteni, hogy a vezeték nélküli szenzorhálózatok élettartamát nem feltétlenül csak az érzékelés minősége vagy hatékonysága határozhatja meg, hanem a kommunikációs képesség biztosítása is. A szenzorhálózat ugyanis hiába képes megfelelő minőségben érzékelni a kívánt jelenségeket vagy eseményeket, ha a mérési adatokat nem képes eljuttatni a bázisállomásokig, akkor a hálózatot nem igazán tekinthetjük élőnek, hiszen nem látja el rendeltetésszerűen a feladatát.

Egyugrásos hálózatok esetén ez azt jelenti, hogy a bázisállomástól távol levő szenzorok bár még nem merültek le teljesen, és képesek például érzékelni a kívánt jelenséget, de annyi energiájuk már nincs, hogy az alkalmazás igényeinek megfelelő gyakorisággal eljuttassák a bázisállomáshoz a mérési eredményeiket. Többugrásos hálózatok esetén, mint említettük, a bázisállomáshoz közel levő szenzorokon van a nagyobb teher, hiszen a hálózat összes többi szenzorjának a mérési eredményeit továbbítaniuk kell a bázisállomás felé, ezért ezek a közeli szenzorok merülnek majd le leghamarabb. Ha pedig ez bekövetkezik, akkor megtörténhet az, hogy bár a bázisállomástól távol levő érzékelők rendeltetésszerűen működnek, mindenki mér és bizonyos időközönként egy több-ugrásos útvonalválasztó algoritmus segítségével el is indítja az adatait a bázisállomás felé, ezek soha nem érkeznek meg a célállomáshoz, mert a bázisállomás közelében lévő területen nem marad olyan szenzor, amely továbbküldhetné ezeket az adatokat. Ebben az esetben is természetesen mondhatjuk azt, hogy a szenzorhálózat nem működik rendeltetésszerűen, tehát halottnak tekinthető.

De mi is legyen a teendő egy „halott" hálózattal, és miért is fontos az, hogy hogyan is definiáltuk a hálózat élettartamát? Egyfelől, ha a hálózat meghal, akkor általában be kell avatkozni, és újra életre kell kelteni. Ez többféleképpen történhet. Egyrészt lehetséges az, hogy egyszerűen megkeressük a megrongálódott vagy lemerült szenzorokat, és kicseréljük vagy feltöltjük azokat olyan mértékben, hogy az alkalmazás minőségbéli igényeit ismét ki tudjuk szolgálni. Ez a beavatkozás nyilvánvalóan igen költséges lehet, úgy anyagilag, mint emberi erőforrásokat illetően. Bizonyos esetekben pedig nem is lenne lehetséges.

Egy másik lehetőség az, hogy az eredeti telepítésnél, a szükségesnél jóval nagyobb redundanciával telepítjük ki a szenzorokat, de alapjáraton azoknak csak egy részét használjuk. Ha viszont egy idő után a szenzorok egy része elkezd lemerülni, akkor a redundáns szenzorok akár automatikusan is aktivizálhatóak, visszaállítva az elvárt minőségi szolgáltatást. Ennek a megközelítésnek az a hátránya, hogy több kihelyezett szenzor esetén a kezdeti telepítés, azaz a *CAPEX (Capital Expenditure)* költségek nagyobbak lesznek, viszont jelentős megtakarítás érhető el az üzemeltetési, azaz *OPEX (Operational Expenditure)* költségeken.

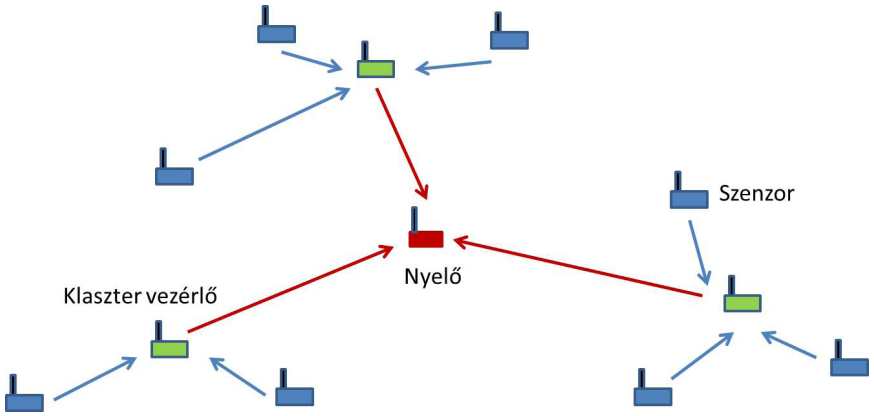
Másfelől a hálózat élettartama bizonyos esetekben csak egy mérőszámot jelent, amelynek segítségével a kutatók megpróbálják a különböző kommunikációs protokollokat, útvonalválasztó algoritmusokat, alvásvezérlő-mechanizmusokat, illetve ezeknek az energiahatékonyságát összehasonlítani egymással. Minél hosszabb élettartamot képes biztosítani egy adott algoritmus egy vezeték nélküli szenzorhálózat számára, az alkalmazás igényeinek megfelelő szintű kiszolgálása mellett, az adott megoldás annál hatékonyabbnak, jobbnak értékelhető.

2.4. TOPOLÓGIA-KONTROLL

Ahogy azt már említettük, a vezeték nélküli szenzorhálózatok fizikai topológiája sok esetben véletlenszerűen alakul ki, és nem egy jól megtervezett, akár egy adott alkalmazás igényeihez illeszkedő, célzott telepítés eredményeképpen. Egy földrengés vagy árvíz esetén például helikopterből dobhatnak ki szenzorokat az összedőlt vagy elmosott épületekből álló területre, esetleges túlélők után kutatva. Egy ilyen véletlenszerű telepítés után a szenzorok előbb megpróbálják feltérképezni a környezetüket, esetleg kapcsolatba lépnek más, közelükben levő szenzorokkal, majd egy ad-hoc jellegű hálózatot építenek ki egymás között. A fizikai kapcsolatok (rádiós linkek) tehát a szomszédos szenzor csomópontok között véletlenszerűen jönnek létre. Fontos azonban egy logikai topológia kialakítása is, különös tekintettel a skálázhatóságra. Erre több lehetséges megoldás létezik.

Egy *elosztott (distributed)* vagy *lapos (flat)* struktúra esetén nincs kifejezetten kialakított logikai struktúra, minden csomópont egyenrangú a hálózatban, és minden szenzor csomópontnak részt kell vennie a hálózat vezérlésében. Mivel minden szenzor csak a saját rádiós hatósugarán belül található szomszédos csomópontokat ismeri, időről időre terjesztenie kell a saját útválasztó tábláját, hírt adva a többi csomópontról is. Ez a megoldás egy több ezer szenzor csomópontból álló hálózat esetén nyilvánvalóan nem skálázható.

Egy másik lehetőség, hogy a szenzorhálózatot úgynevezett *klaszterekre* [Abbasi2007] bontjuk úgy, hogy minden szenzor csomópont legalább egy klaszterben szerepeljen (2.4 ábra). Minden klaszternek lesz egy vezérlője *(cluster head)*, azaz egy olyan állomás, amely vezérli a klaszteren belüli összes többi szenzor csomópont kommunikációját. Ezen felül a klasztervezérlők egymással is kommunikálnak, akár egy hierarchikus klaszter struktúrát is kialakítva, ha ez szükséges.



2.4. ábra: Kommunikáció egy klaszterekre bontott hálózatban *(saját ábra)*

A klaszterek optimális kialakítása nem mindig egy egyszerű feladat, hiszen már az is kérdéses, hogy pontosan hány klasztert is érdemes kialakítani egy adott fizikai topológia esetén, illetve hogy egy adott szenzor csomópontnak melyik klaszterhez éri meg leginkább csatlakoznia. Nyilvánvalóan egy adott klasztervezérlő és a szenzor közötti távolság egy nagyon fontos szempont, de a klaszterek minél egyenletesebb terheltségét is érdemes figyelembe venni.

Fontos megjegyezni, hogy a klasztervezérlők jóval nagyobb terhelést kapnak, mint egy egyszerű szenzor csomópont, ezért a klasztervezérlők kilétére kiemelt figyelmet kell fordítani. Az egyik lehetőség az, hogy a klasztervezérlők dedikált csomópontok legyenek, amelyek a többi szenzor csomóponthoz képest több erőforrással, leginkább nagyobb energiatartalékkal rendelkeznek. Ez megvalósulhat úgy, hogy ezek a klasztervezérlők például folyamatosan csatlakoztatva vannak az elektromos hálózatra, de az is lehetséges, hogy a hálózaton belül csak a klasztervezérlők rendelkezzenek valamilyen alternatív energiaforrással (például napelemmel). A korlátozott energiatartalékokkal rendelkező „egyszerű" szenzorok erőforrásait tehát kíméljük, cserében viszont plusz feladatokat adunk a napelemekkel rendelkező klasztervezérlőknek.

Megtörténhet viszont az is, hogy nincsenek a hálózatban plusz erőforrásokkal rendelkező dedikált klasztervezérlők. Ebben az esetben érdemes lehet a klasztervezérlő szerepet váltogatni a csomópontok között, valamilyen „vetésforgó", „round-robin" megoldással például. Ez azonban akár a klaszterek újraosztásával is járhat, hiszen ha egy klaszteren belül megváltozik a klasztervezérlő kiléte, akkor előfordulhat az, hogy egy bizonyos csomópont számára már túl távoli lesz az új klasztervezérlő, és inkább egy másik klaszterbe csatlakozna.

A klaszterek periodikus újraosztását különben több tényező is indokolhatja. Egyfelől figyelembe kell venni a klasztervezérlő és a csomópontok közötti távolságok szerinti optimalizálást, amit már említettünk.

Másfelől a szenzorok vagy akár a klaszterfejek mobilitása is befolyásolhatja a topológiát, függetlenül attól, hogy ez a mobilitás a szenzor saját erőforrásainak felhasználásával valósul-e meg, vagy a szenzor egyszerűen egy mozgó tárgyra, személyre vagy állatra van rögzítve.

Harmadrészt az is megtörténhet, hogy a monitorozni kívánt események eloszlása a hálózatban nem egyenletes, ami az „eseménydús" területen levő klaszterek jóval nagyobb terheléséhez vezet. Ilyenkor érdemes lehet áttervezni a topológiát úgy, hogy a sűrű eseményekkel terhelt területen jóval kisebbek legyenek a klaszterek, több klasztervezérlővel, ahol pedig kevés az esemény és ezáltal kevesebb a jelenteni szükséges adat, ott nagyobb klasztereket alakítsunk ki, kevesebb klasztervezérlővel. És mivel az események gyakorisága, a megfigyelt jelenségtől és az alkalmazás igényeitől függően időben és térben dinamikusan változhat, érdemes lehet a klaszterhatárokat periodikus időközönként újra optimalizálni, vagy egy adott paraméter valamilyen küszöbszintjéhez kötni.

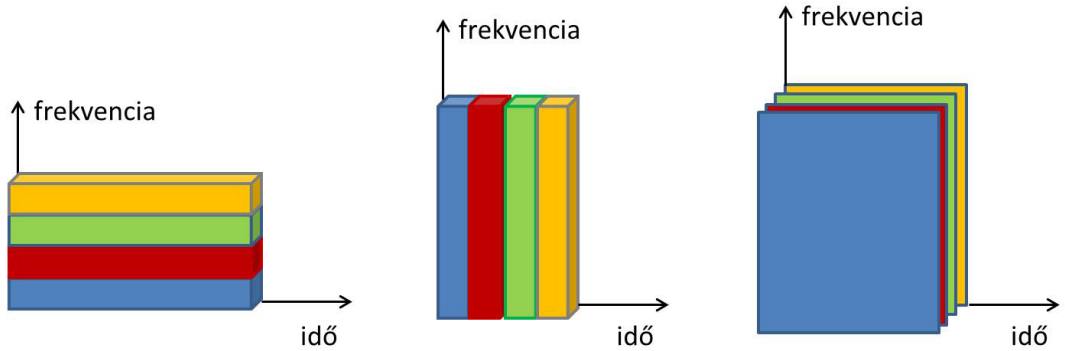
Érdemes ugyanakkor azt is megjegyezni, hogy a klaszterhatárok, illetve a klasztervezérlők kilétének folyamatos újraszámolása valamilyen komplex optimalizálási feladat eredményeképpen jelentős számítási kapacitást, illetve komoly jelzésforgalmat igényelhet. Szükséges lehet tehát azt is mérlegelni, hogy megéri-e, vagy milyen sűrűn éri meg újrakonfigurálni a topológiát, mi az a nyereség, ami elérhető az új konfigurációval ahhoz a plusz terheléshez képest, amit az újrakonfiguráció lebonyolítása okoz.

2.5. KÖZEGHOZZÁFÉRÉS VEZÉRLÉSE

Amikor nem egyedülálló szenzorokról beszélünk, hanem egy adott fizikai területen elhelyezett, egymás rádiós hatósugarában levő szenzor csomópontokról, melyek bizonyos feltételek mellett egy ad hoc jellegű vezeték nélküli szenzorhálózatot alakítanak ki egymás között, akkor feltétlenül beszélni kell arról is, hogy mindez hogyan valósul meg a rádiós erőforrások felhasználását illetően. Egyszerűbben megfogalmazva, ha az adott hálózatban több szenzor is ugyanazokat a rádiós erőforrásokat, ugyanazokat a frekvenciatartományokat, csatornákat szeretné használni, akkor a kérdés az, hogy ezt a konfliktushelyzetet hogyan lehet hatékonyan kezelni. A választ erre az adatkapcsolati rétegben működő protokollok adják meg.

A közeghozzáférés vagy *csatornahozzáférés vezérlése (MAC, Medium Access Control)* az adatkapcsolati réteg egyik legfontosabb feladata, a keretképzés, hibadetektálás és javítás, illetve az esetleges forgalomszabályozás mellett. A közeghozzáférés vezérlése leginkább *üzenetszórásos (broadcast)* csatornák esetén létfontosságú, itt ugyanis több eszköz szeretné párhuzamosan használni ugyanazt a rádiós csatornát, és valamilyen módon szabályozni kell azt, hogy ezt hogyan, milyen sorrendben tehessék meg az egyes eszközök.

Alapvetően két csatornakiosztási stratégiát különböztethetünk meg, statikus és dinamikus stratégiákat. A statikus módszereknél valamilyen fix algoritmus alapján rendelnek hozzá az egyes eszközökhöz dedikált erőforrásokat. Ebben a kategóriában az egyik lehetőség a *frekvenciaosztásos multiplexelés (FDM, Frequency Division Multiplexing)*, azaz lehetőleg minden kommunikáló szenzornak külön frekvenciaszeletet dedikálnak úgy, hogy az egymás közvetlen környezetében levő szenzorok ne zavarják egymást, az adatküldéseik ne interferáljanak (2.5 ábra). A kiosztható frekvenciatartomány/csatornaszám viszont természetesen korlátozott, ezért ugyanazt a csatornát több szenzor is használhatja akkor, ha egymás rádiós hatósugarán kívül esnek.



a) FDM *b) TDM* *c) CDM*

2.5 ábra: Multiplexelési megoldások *(saját ábra)*

Egy másik statikus módszer az *időosztásos multiplexálás (TDM, Time Division Multiplexing)*, melynek során több szenzor is ugyanazon a frekvencián/csatornán fog majd kommunikálni, de dedikált időszeleteken, így kerülve el az ütközéseket. Ennek a megoldásnak az előnye az, hogy akár egy nagyobb frekvenciatartományon, több csatornát összefogva (*channel bonding*) nagyobb átviteli kapacitás biztosítható, ám ennek a nagy frekvenciatartománynak a használata időben korlátozott.

Figyelembe véve a szenzorhálózatok adatforgalmának speciális jellemzőit, az időosztásos multiplexelésnek mindenképpen megvan az az előnye, hogy a szenzorok csak nagyon rövid ideig foglalják el a csatornát, ami jól megfelel a szenzorok működési modelljének. Egy olyan szenzornak ugyanis, mely az idő jelentős részében alszik, néha-néha felébred, esetleg mér és esetleg elküldi a mérési adatait (bár ez se mindig szükséges, csak az alkalmazás igényeit kielégítő mértékben), gyakorlatilag felesleges egy külön dedikált csatornát fenntartani.

A harmadik statikus módszer az úgynevezett *kódosztásos multiplexelés (CDM, Code Division Multiplexing)*, amikor az egyes szenzor csomópontok a teljes frekvenciatartományban és a teljes időintervallumban használhatják a csatornát, a küldéseikhez viszont dedikált kódokat rendelünk. A fogadó fél a kód birtokában tudja majd értelmezni, visszafejteni az adatokat, ezzel párhuzamosan pedig a csatornán jelen levő összes többi adatot, melyet egy másik kóddal kódoltak, csak zajként fogja érzékelni. Bizonyos korlátozott számú kód használata esetén ez a megoldás

egy nagyon hatékony közeghozzáférést biztosít, az algoritmus komplexitása viszont valamilyen szintű plusz terhelést jelent a csomópontok számára, mely egy korlátozott erőforrásokkal rendelkező szenzor esetén fontos lehet.

Mindemellett természetesen az is elképzelhető, sőt, gyakran alkalmazott megoldás, hogy ezeket a technikákat egymás mellett, egymással párhuzamosan alkalmazzuk. Lehetséges például az, hogy az egymás közelében elhelyezkedő szenzor csomópontokat valamilyen algoritmus alapján (akár véletlenszerűen, akár valamilyen alkalmazás logika alapján) csoportokra osztjuk, minden egyes csoport egy-egy külön frekvenciát kap majd a kommunikációra, a csoporttagok pedig időszeletekre osztva férhetnek hozzá az adott rádiós csatornához. Ez tehát egy lehetőség arra, hogy a frekvenciaosztásos (FDM) és az időosztásos (TDM) multiplexelést kombináljuk. De ugyanígy lehetséges az is, hogy egy adott rádiós csatornán több szenzor csomópont kommunikációját kódosztással különböztessük meg, kombinálva így a frekvenciaosztásos és a kódosztásos multiplexelést. Sőt, az is lehetséges elvileg, hogy a szenzor csomópontokat csoportokra bontsuk, minden csoporthoz egy egyedi rádiós csatornát társítsunk, a csoporton belül alcsoportokat hozzunk létre, és minden alcsoporthoz időszeleteket allokáljunk, az adott csatornán és az adott időszeletben pedig több szenzor is kommunikálhasson különböző kódokat használva. Ez tehát egy olyan megoldás lenne, ahol a frekvenciaosztásos, az időosztásos és a kódosztásos multiplexelést is párhuzamosan alkalmaznánk.

A fentebb említett statikus megoldások legfontosabb hátránya az, hogy sok eszköz, illetve nem egyenletes forgalom esetén a rádiós csatorna kihasználtsága drasztikusan lecsökken. Ezt tetézi az, hogy vezeték nélküli szenzorhálózatok esetén a szenzor csomópontok az idő nagy részében alvó állapotban vannak. A statikus csatornahozzáférési megoldások viszont nem feltétlenül vannak összehangolva az alvásvezérlő algoritmusokkal, nem veszik figyelembe azt, hogy melyik szenzor csomópont mikor van alvó üzemmódban. Így megtörténhet az, hogy egy adott szenzor rendelkezésére bocsájtanak erőforrásokat, azaz dedikált rádiós csatornát, időszeleteket vagy kódokat, akkor is, ha az adott szenzor épp nem tudja ezeket felhasználni, mert alvó üzemmódban van. Az ilyen fajta kiosztás nyilvánvalóan nem hatékony, sőt, pazarolja az amúgy is szűkös erőforrásokat. Ezzel szemben a dinamikus módszereknél a változó igényeknek megfelelően oszthatjuk ki a csatorna hozzáférés jogát, ehhez viszont természetesen ismernünk kell ezeket a változó igényeket. A dinamikus csatornakiosztásnál tehát minden esetben létezik egy lekérdezési (*polling*) fázis is.

A közeghozzáférés vezérlésére kidolgozott megoldások hatékonysága kiemelt fontosságú minden vezeték nélküli kommunikációs technológia esetén, de a szenzorhálózatok esetében néhány speciális követelményt is figyelembe kell venni ezek tervezésénél. Így például számolni kell azzal, hogy a csomópontok aktív részvétele csak az idő kis töredékében biztosítható, hiszen a cél az, hogy az egyes szenzorok minél hosszabb ideig lehessenek alvó állapotban, így takarékoskodva az energiatartalékaikkal (alvás-ébrenlét ütemezés). Másfelől a frekvenciagenerátorok pontossága általában elég csekély, így az időosztásos technikák nem igazán hatékonyak. Végül pedig érdemes olyan megoldást javasolni, amelyik egyszerűen implementálható, alacsony komplexitású.

Az egyik legismertebb szenzorhálózati MAC protokoll az *S-MAC (Sensor Medium Access Control)* [YE2002] [YE2004]. Az S-MAC által használt modell néhány előfeltétellel él a hálózat működését illetően. Egyfelől arra épít, hogy sok, apró, ad hoc módon telepített csomópont számára szeretnénk többugrásos kommunikációt biztosítani, mely kommunikáció nem csak a bázisállomás felé, hanem csomópont párok között is történhet. Ez különben nem feltétlenül jellemző a szenzorhálózati alkalmazások többségénél, hiszen a forgalmi modellek általában a *pont-többpont (P2MP, Point-to-Multipoint),* azaz a bázisállomástól a szenzorok felé, és a *többpont-pont (MP2P, Multipoint-to-Point),* azaz a szenzoroktól a bázisállomás felé irányuló kommunikációra redukálódnak. Másfelől az

S-MAC protokoll azt is feltételezi, hogy az egész telepített szenzorhálózat egyetlen alkalmazás céljait szeretné dedikáltan kiszolgálni (az internet nem egy ilyen hálózat például), valamint azt is, hogy előfordulnak hosszú, tétlen időszakok, amikor a szenzorok alvó üzemmódban vannak. Végül az S-MAC azt is feltételezi, hogy az alkalmazás, amelyet ki szeretnénk szolgálni, elvisel bizonyos szintű késleltetést (például nem valós idejű felügyeleti rendszereket szeretnénk kiszolgálni).

Az S-MAC energiahatékony működésének egyik fontos pillére a periodikus figyelés és alvás. Ha a szenzorok nem érzékelnek eseményeket, a küldendő adatforgalom is várhatóan kicsi lesz majd, ezért nem szükséges az összes szenzornak aktívnak maradnia. Az S-MAC megengedi tehát, hogy a szenzor csomópontok periodikusan alvó üzemmódba kapcsoljanak; ilyenkor az eszköz kikapcsolja a rádióját, és egy *időzítőt (wake-up timer)* állít be a felébredéshez. Egy teljes aktív-inaktív ciklus képez egy keretet, melyben az aktív időtartam fix, az inaktív pedig változó hosszúságú. Az aktív időtartam hossza a fizikai és a MAC réteg paramétereitől függ, az inaktív periódus hosszával pedig állítható a keret mérete, illetve az úgynevezett *duty-cycle*, azaz az aktív és az inaktív periódus aránya.

Minden szenzor csomópont megválaszthatja a saját aktív/inaktív ütemezését, de ugyanakkor szinkronizálódnia is kell a szomszédos csomópontokhoz. Az egyes csomópontok üzenetszórással juttatják el a saját ütemezésüket közvetlen szomszédjaikhoz, kommunikálni pedig így bármely két csomópont képes lesz egymással, akkor is, ha az ütemezésük különböző. Ha tehát A csomópont szeretne küldeni B-nek, megvárja míg B aktív állapotba került. Ha viszont egyszerre többen is szeretnének B-nek küldeni, akkor versengés indul a csatornáért az IEEE 802.11-ből ismert *RTS-CTS (Request to Send – Clear to Send)* mechanizmus segítségével.

Ennek a mechanizmusnak a lényege az, hogy a hagyományos vezetékes hálózatokkal ellentétben, ahol adás előtt bele tudunk hallgatni a csatornába, és érzékelhetjük azt, ha valaki éppen foglalja azt a csatornát, egy vezeték nélküli környezetben erre nincs lehetőség, egyfelől az úgynevezett rejtett állomás problémája, másfelől az úgynevezett látható állomás problémája miatt. A rejtett állomás problémája abból adódik, hogy belehallgatva a rádiós csatornába csak azt tudom érzékelni, hogy a saját rádiós hatósugaramon belül nincsen egyetlen olyan másik csomópont se, amelyik abban a pillanatban foglalná a csatornát. Ettől függetlenül megtörténhet viszont az, hogy számomra egy rejtett, távoli állomás mégis foglalja azt a csatornát, és ha az a másik csomópont, melynek küldeni szeretnénk, hallja ezt a számunka távoli csomópontot is, akkor ütközés következhet be a csatornán.

Ezzel szemben a látható állomás problémája éppen az imént vázolt forgatókönyvnek az ellentétét jelenti. Ha ugyanis belehallgatva a csatornába azt érzékeljük, hogy a csatorna foglalt, az nem jelenti feltétlenül azt, hogy nem lenne szabad nekünk is küldeni ugyanarra a csatornára, hiszen lehet, hogy a két adás címzettje egymástól távol van, és a két kommunikáció működhetne akár párhuzamosan is, csomagütközés nélkül.

A versenyhelyzetet, és az imént említett két problémát úgy lehet feloldani tehát, hogy küldés előtt a küldő csomópont megkérdezi a célzott vevőt arról, hogy adhat-e, azaz nincs-e folyamatban olyan adatküldés melyet egy új, párhuzamos adatküldés veszélyeztetne. Erre a célra szolgál a *Request-to-Send (RTS)* üzenet. Ha pedig a megcélzott vevő nem látja ennek akadályát, és képes fogadni az adatcsomagot, akkor egy *Clear-to-Send (CTS)* üzenettel válaszol, megadva a zöld lámpát az adatok indításához.

Megtörténhet ugyanakkor az, hogy több csomópont is párhuzamosan szeretne adni, mindenki elküldi a saját RTS üzenetét, melyekre párhuzamosan több csomópont is válaszolhatna CTS üzenettel. Ebben a helyzetben viszont az a csomópont nyeri a versenyt, amelyik először küldte el a CTS csomagját. Ezt a csomagot meghallva, a többi csomópont a kialakított adatkapcsolat lezárásáig felhagy az addigi ütemezésével, elkerülve azt, hogy esetleg zavarja a létrejött kommunikációt. A megoldás előnye az, hogy nincs szükség klaszterképzésre és koordinációra a klasztervezérlők között (mint ahogy azt az előző alfejezetben bemutattuk), a csomópontok elosztott módon,

egymás között tudják kezelni a kialakult versenyhelyzetet. A megoldás hátránya viszont az, hogy a csomópontok periodikus alvása miatt a késleltetés nagyon nagy is lehet, különösen multi-hop kommunikációnál egy nagy kiterjedésű hálózaton.

2.6. ALVÁS-ÉBRENLÉT ÜTEMEZÉS

Mint azt már több ízben is említettük korábban, a szenzorok egyik legszűkösebb és ugyanakkor legfontosabb erőforrása az energiatartalékuk, hiszen az elemek, akkumulátorok sok esetben nem cserélhetőek vagy nem újratölthetőek. A szenzorok egyik működési alapelve tehát, hogy az idő jelentős részében alvó állapotban vannak, és csak nagyon rövid időszakokra ébrednek fel, mérés, illetve kommunikáció céljából.

Maga az alvás is lehet különben különböző „mélységű". Létezik olyan állapot, amikor az adott szenzor nem mér, nem figyeli a környezetét, viszont a rádiós csatornát igen, lehetőséget adva arra, hogy szükség esetén más szenzor adatát továbbküldhesse a bázisállomás felé. Ezzel ellentétben lehet olyan alvási üzemmód is, amikor a rádiós interfészt kapcsolja ki, hiszen ez használja a legtöbb energiát, még akkor is, ha maga a szenzor nem küld semmiféle adatot, sőt, nem is feltétlenül hall semmilyen kommunikációt a rádiós csatornán, de maga a csatorna monitorozásnak is már komoly energiaigénye van. Emellett viszont a szenzor érzékelési üzemmódban maradhat, akár folyamatosan is, ha az adott jelenség monitorozása nem igazán meríti az energiatartalékait.

Ezen felül létezhet olyan üzemmód, amikor a szenzor csomópont nem is mér, nem is hallgatja a rádiós csatornát, de azért távolról felébreszthető szükség esetén, a bázisállomás vagy egy szomszédos szenzor által, illetve olyan üzemmód is lehetséges, amikor ez a távoli felébresztés sem lehetséges.

Ha csak egy adott szenzorról beszélünk, akkor ezt az alvás-ébrenlét ütemezést nem kell senkivel összehangolnia, saját maga dönthet arról, hogy mikor van ébren, mikor és milyen hosszú időre megy aludni, illetve arról, hogy ez az alvó üzemmód mit is takar tulajdonképpen. Ezzel szemben viszont, ha nem egy egyedi szenzorról, hanem egy vezeték nélküli szenzorhálózatról beszélünk, akkor a hálózatot alkotó szenzor csomópontok alvás-ébrenlét ütemezése már nem lehet egy egyéni döntés következménye, hanem a szomszédos szenzoroknak egymással összehangolva kell az alvási időszakokról dönteniük.

Érdemes megjegyezni, hogy egyfelől a monitorozási feladat, másfelől pedig a rádiós kommunikáció biztosítása más és más alvás-ütemezési stratégiákat kíván. A kiindulási állapot mindenképpen az, hogy a szenzorok az adott területen belül redundánsan vannak kihelyezve. A fizikai jelenségek monitorozásánál az az elgondolás, hogy az adott területen, több egymáshoz közel álló szenzor is képes ugyanazt a jelenséget, ugyanazokat az eseményeket észlelni. Ezért ezek közül a szenzorok közül elég az is, ha csak egy van ébren egy adott pillanatban, a többi csomópont lehet alvó üzemmódban. Nyilvánvalóan ezt az érzékelési feladatot a szenzorok egymással koordinálva látják majd el, az aktuális erőforrásaik és az alkalmazás igényeinek megfelelően. Itt tehát az a cél, hogy lehetőleg mindig csak egy szenzor legyen ébren az adott területen.

Ezzel szemben, ha a rádiós kommunikáció biztosítását nézzük egy többugrásos hálózatban, ott éppen az a cél, hogy egyszerre legyen ébren több olyan szenzor csomópont is, amelyik a forrás csomópont és a bázisállomás közötti útvonalon helyezkedik el, hogy a forrás csomópont által mért adatokat eljuttathassák a bázisállomáshoz. Ha ez nem biztosítható, akkor az adatok, bár elindulnak a forrás szenzortól, elakadnak valahol útközben, amikor egy adott ugrásnál nincs egyetlen ébren lévő szenzor sem, amelyik átvehetné, továbbküldhetné a csomagot. Ez

persze kezelhető úgy, hogy az adott szenzor csomópont, ahol elakadt a csomag, eltárolja azt, amíg fel nem ébred valaki a közelében, akinek továbbküldhető. Mivel a szenzorok általában kisméretű, néhány bájtnyi adatot küldenek csak, egy ilyen csomag átmeneti tárolása nem feltétlenül okoz gondot, még akkor is, ha a szenzor csomópontok memória kapacitása is általában erősen korlátozott. Kérdés azonban, hogy mi történik akkor, ha a csomagot tároló csomópont maga is elmegy alvó üzemmódba, mire a szomszédságában levő csomópontok felébrednek. Ahhoz, hogy ez a megoldás működőképes legyen, valamilyen szintű együttműködést, elosztott alvás-ébrenlét ütemezést feltételez a csomópontok között.

A vezeték nélküli szenzorhálózatok igényeire szabott alvás-ébrenlét ütemezést illetően több algoritmust is javasoltak. Ezek nagy része azonban az idővezérelt hálózatokra alkalmazható csak, ahol bizonyos jól kiszámolható időközönként kell a szenzoroknak jelentést küldeniük a bázisállomás felé. Ilyenkor jól ütemezhető az, hogy mikor kell az egyes szenzoroknak mérniük, illetve éber állapotban lenniük ahhoz, hogy a rádiós interfészen továbbíthassák a szomszédos csomópontok csomagjait a bázisállomás felé, az alkalmazás igényeinek megfelelően.

Ezzel szemben sokkal nehezebb ezt a kérdést egy esemény-vezérelt hálózatban kezelni, ahol teljesen véletlenszerűen, vagy legalábbis nehezen előrelátható módon fognak bizonyos események bekövetkezni, melyeket a szenzoroknak egyfelől érzékelniük kell, másfelől pedig a hálózatnak képesnek kell lennie arra is, hogy akár valós időben is biztosítsa a csomagok eljuttatását a bázisállomás felé, még egy többugrásos hálózat esetén is. Az ilyen dinamikus alvás-ébrenlét ütemezési kihívásokra viszonylag kevés megoldás született a mai napig. (Az idővezérelt, eseményvezérelt, illetve lekérdezés alapú hálózatokat a 2.8. alfejezetben még részletesen tárgyaljuk.)

Az alvás-ébrenlét ütemezését persze tovább bonyolítja, ha a topológia dinamikusan változik, vagy azért, mert megrongálódnak vagy lemerülnek szenzorok, vagy például azért, mert a szenzorok vagy a bázisállomás elmozognak a hálózaton belül. Ha ugyanis bármilyen módon megváltozik a hálózati topológia, az magával hozhatja a hálózaton belül a kommunikációs útvonalak változását is, ami pedig azzal jár majd, hogy az egyes csomópontok más-más szomszédokkal kell majd egyeztetniük az alvás-ébrenlét ütemezését illetően is.

2.7. ÚTVONALVÁLASZTÁS

Mint azt már említettük, egy nagyobb területet lefedő többugrásos vezeték nélküli szenzorhálózatban szükség van egy hatékony útvonalválasztó algoritmusra is, melynek segítségével a szenzorok mérési adatait eljuttathatjuk a bázisállomásig. A vezeték nélküli szenzorhálózatok azonban sok szempontból különböznek egy hagyományos vezetékes, de akár egy átlagos vezeték nélküli hálózattól is, ezért a hagyományos útvonalválasztó algoritmusok a legtöbb esetben nem alkalmazhatók. Íme néhány specifikum, amit figyelembe kell venni.

Egy vezeték nélküli szenzorhálózat telepítését illetően sok esetben egy véletlenszerűen szerveződő ad hoc hálózatnak tekinthető, a telepítés után viszont maguk a csomópontok általában statikusak, nem mozognak. A topológia, melyre az útvonalválasztó algoritmust építjük, ennek ellenére változhat, egyfelől a lemerülő, meghibásodó szenzoroknak, másrészt az alkalmazott alvásvezérlő algoritmusnak köszönhetően.

Másfelől a vezeték nélküli szenzorhálózatok igényeire szabott útvonalválasztó algoritmusnak robusztusnak kell lennie egy vagy több szenzor csomópont meghibásodására vagy lemerülésére is. Ezt nagyrészt redundáns telepítéssel szokták elérni, azaz úgy és annyi szenzort helyeznek el a területen, hogy szükség esetén minden szenzor környezetében legyen legalább egy másik szenzor, mely át tudja venni a szerepét, akár ideiglenesen, akár

végleg, úgy az érzékelést, mint a kommunikációt tekintve is. Ha azonban a szenzor csomópontok egy nagyobb százaléka merül le, akkor egy idő után a hálózat nem lesz már képes megfelelő módon ellátni a feladatát, vagy azért mert egy bizonyos területet nem fednek már le aktív szenzorok, és így az ott történő eseményeket nem tudjuk érzékelni, vagy azért, mert a még aktív szenzorok nem találnak egyetlen érvényes útvonalat sem a bázisállomással történő kommunikációhoz. Ilyenkor a szenzorhálózat részleges vagy teljes újratelepítése szükséges.

Szintén a szenzorhálózatokra jellemző, hogy a kommunikáció, mint azt már említettük, javarészt multipont-pont vagy pont-multipont jellegű, azaz a szenzorok egy (vagy több) bázisállomásnak szeretnék továbbküldeni az adataikat, illetve a bázisállomások a szenzorok felé küldhetnek különböző utasításokat, lekérdezéseket. Az esetek túlnyomó többségében nem szükséges viszont útvonalakat építeni vagy tárolni tetszőleges forrás – cél csomópont párok között, mint azt a hagyományos útvonalválasztó algoritmusok teszik, hiszen két szenzor csomópont alapesetben nem akar egymással kommunikálni, az alkalmazásoknak nincsenek ilyen jellegű igényeik.

A továbbiakban, a teljesség igénye nélkül, röviden bemutatunk néhány olyan útvonalválasztó algoritmust, melyet kifejezetten a vezeték nélküli szenzorhálózatok speciális igényeire fejlesztettek ki.

2.7.1. Diffúziós protokollok lapos topológiához

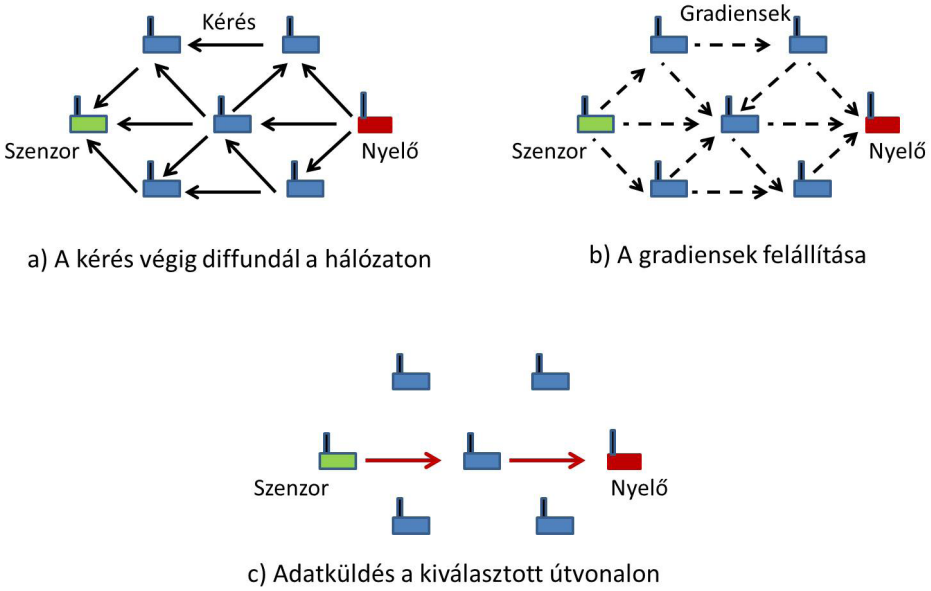
A korábbi alfejezetekben említett lapos (flat) topológia esetén a szenzor csomópontok egyenrangúak, egymással együttműködnek az útvonalválasztási feladat ellátásában. A kategória egyik ismert megoldása a *SPIN (Sensor Protocol for Information via Negotiation)* [Vidhi2014], melynek a lényege az, hogy minden csomópont minden adatot szétterjeszt az összes többi szenzor csomópont számára, így az információ bármelyik csomóponttól azonnal lekérdezhető. Az alapötlet az, hogy az egymáshoz közel elhelyezkedő szenzorok nagyon hasonló, vagy akár azonos adatokkal rendelkeznek, ezért elég csak azokat az adatokat terjeszteni, amelyek még nincsenek meg a szomszédoknál. Az adatokhoz leírók (meta-adatok) társulnak, első fázisban ezeket cserélik ki egymás között a csomópontok, így második fázisban elég csak a nem redundáns meta-adatokhoz tartozó valós adatokat kicserélni.

A protokoll működése röviden a következőképpen írható le: ha egy szenzor csomópont új adat birtokába kerül, üzenetszórással meghirdeti annak meta-adatát; ha egy szomszédos csomópont érdeklődik az adat iránt, azt egy adatkérő üzenettel jelzi, a forrás csomópont pedig válaszul elküldi a kért adatot; amikor a szomszéd megkapja ezt, ő is elkezdi hirdetni a saját szomszédjai között. Az üzenetküldések eredményeként egy idő után az adat az egész hálózatban szétterjed.

Egy másik ismert megoldás lapos topológia esetére az *irányított diffúzió (Directed Diffusion, DD)* [Intanagonwiwat 2000], mely egy adatcentrikus és alkalmazástudatos protokoll. A szenzorok mérési adatait egy attribútum–érték párossal írja le, a különböző szenzor forrásoktól származó adatokat pedig aggregálja a hálózaton belül, útban a bázisállomás felé (2.6. ábra). Amikor a bázisállomásnak információra van szüksége, egy kérést küld a hálózatba üzenetszórással. A kérés végig „diffundál" a hálózaton, minden szenzor csomópont továbbítja azt szomszédjainak. A csomópontok ezután gradienseket állítanak fel az attribútum–érték pár, illetve a küldő szomszéd irányát figyelembe véve. A gradiens nagysága a szomszédoktól függően más-más lehet. Ezek után az adatküldés a szenzoroktól a bázisállomás felé történik, a legnagyobb gradiens irányában.

A cél egy olyan hatékony fa kiépítése, melynek mentén az adatok aggregálhatóak. Az adataggregáció hatékonysága függ majd a csomópontok elhelyezkedésétől, számától, a hálózati kommunikációs topológiától és az alkalmazott eseménymodelltől is. Ha például egy adott eseményt egy adott területen belül több egymáshoz

közeli szenzor is érzékelt, akkor az erről szóló jelentésük könnyen aggregálható, útban a bázisállomás felé. Ha viszont több diszkrét szenzor a hálózat különböző pontjairól indít el jelentéseket a bázisállomás felé, megeshet, hogy a jelentések útvonalai csak a bázisállomás közvetlen környezetében találkoznak, így az aggregálás kevésbé lesz hatékony.



a) A kérés végig diffundál a hálózaton

b) A gradiensek felállítása

c) Adatküldés a kiválasztott útvonalon

2.6. ábra: Az adatküldés különböző fázisai az irányított diffúzióban *(saját ábra)*

Az irányított diffúzió egy variánsa az úgynevezett *pletyka útválasztás (gossiping)* [Blywis2010], melynél elárasztás helyett csak azokhoz a csomópontokhoz irányítjuk a bázisállomás által küldött lekérdezést, amelyek az adott eseményt megfigyelték, vagy információt szereztek róla. Ha egy szenzor csomópont érzékel egy eseményt, azt hozzáadja egy lokális eseménytáblához és létrehoz egy úgynevezett „ügynök" csomagot, mely a hálózatban utazva terjeszti az információt az adott eseményről. Ha ezek után egy kérés érkezik a bázisállomástól az adott eseményt illetően, akkor üzenetszórás helyett bármelyik csomópont megválaszolhatja azt, feltéve, ha ismeri az utat a forráshoz, azaz járt nála az adott ügynök csomag.

A kérések és ügynökök élettartama a hálózaton belül egy állítható paraméter. Az ügynökök útvonalának meghatározása azonban nem triviális, és ez nagyban befolyásolhatja a protokoll hatékonyságát. Érdemes megjegyezni, hogy a pletykáláson alapuló útválasztás csak akkor igazán működőképes, ha az események száma kicsi, hiszen sok esemény esetén az eseménytáblák karbantartása túl költséges, főleg akkor, ha az érdeklődés az események iránt viszont alacsony.

2.7.2. *Adaptív klaszter protokoll hierarchikus topológiához*

Az elosztott, lapos topológiára épülő útválasztás alternatívája a hierarchikus klaszter topológiák használata. Az ebbe a kategóriába tartozó útvonalválasztó algoritmusok közül talán a legismertebb a *LEACH (Low Energy Adaptive Clustering Hierarchy)* [Heinzelman2000] protokoll, melynél a klasztervezérlőket véletlenszerűen választják, és időről

időre változtatják őket, az energiahasználat kiegyensúlyozása végett. A klasztervezérlő a szenzor csomópontoktól érkező adatokat aggregálja és tömöríti, majd így továbbítja a bázisállomás felé. A klaszteren belüli és a klaszterek közötti ütközések elkerülésére időosztásos, illetve kódosztásos többszöri hozzáférést (TDMA/CDMA) alkalmaz a közeghozzáférés vezérlésére. Az adatok gyűjtése centralizált és periodikus. Az inicializációs fázisban minden csomópont sorsol magának egy véletlen számot, és ha ez a szám egy bizonyos küszöbérték alatt van, akkor abból a csomópontból klasztervezérlő lesz. Minden klasztervezérlő hirdeti magát; akiből pedig a sorsolás alapján nem lett rögtön klasztervezérlő, csatlakozik valamelyik klaszterhez, ahhoz, melynek hirdetését megkapta, például a vett jel erősségét véve alapul.

A megoldás hátránya, hogy egyfelől nem egyértelmű, hogy a csomópontok milyen valószínűséggel váljanak klasztervezérlőkké, azaz hogyan történjen a küszöbérték megválasztása. Másfelől az is probléma lehet, hogy a klasztervezérlők térbeli eloszlása a véletlen sorsolásnak köszönhetően nem feltétlenül lesz egyenletes, és akár az is előfordulhat, hogy néhány csomópont rádiós környezetében nem lesz egyetlen klasztervezérlő sem. Egy lehetséges kiterjesztése a megoldásnak az, ha egy adott csomópont az aktuális energiaszintjétől teszi függővé, hogy milyen valószínűséggel deklarálja magát klasztervezérlőnek.

2.8. ESEMÉNY-, IDŐ- ÉS LEKÉRDEZÉS ALAPÚ ADATKÜLDÉSI STRATÉGIÁK

A szenzorhálózatok működésének megértéséhez fontos szétválasztani egy adott szenzor két alapvető funkcióját: egyfelől magát az érzékelést, másfelől az adatok továbbküldését egy bázisállomáson keresztül egy adatfeldolgozó központba.

Az érzékelés gyakorisága függ a mért paraméter változékonyságától, illetve az alkalmazás igényeitől. Egy ultrahangos távolságmérésen alapuló parkoló szenzornak például nem szükséges másodpercenként mérnie, hogy parkol-e felette jelenleg egy autó vagy sem, hiszen egy napon belül egy adott parkolóhelyen általában 10-15 parkolási esemény történik csak, és egy adott parkolás akár több óráig is eltarthat. Ha tehát a szenzor energiatartalékait akarjuk kímélni, akkor bőven elegendő mondjuk csak 5-10 másodpercenként mérni, hiszen ezzel is biztosítható, hogy minden egyes parkolási eseményt külön-külön érzékelni tudjunk. Ha viszont az aszfaltba épített szenzor nem a parkolást érzékeli, hanem egy adott forgalmi sávban számolja a felette elhaladó autókat, például a környéken levő közlekedési lámpák forgalomhoz illesztett vezérlésének céljából, akkor szükséges lehet másodpercenként többször is mérni, hogy pontos mérési adatokkal rendelkezhessünk.

A szenzor csomópontok méréseinek gyakorisága sokszor nincs feltétlenül összhangban a kommunikáció gyakoriságával, hiszen az alkalmazás igényeitől függően az adatokat lehet aggregálva, csak egy későbbi időpontban továbbküldeni a bázisállomás felé, függetlenül attól, hogy milyen sűrűn történt maga a mérés. Az előbbi példával élve, ha az alkalmazás célja a különböző útvonalak terheltségének a monitorozása különböző napszakokban, akkor elegendő lehet csak óránként, vagy naponta néhány alkalommal jelentenie a szenzornak. Ha viszont egy navigációs alkalmazásnak valós idejű információra van szüksége az optimális útvonalak tervezéséhez, akkor szükséges lehet a jóval sűrűbb kommunikáció.

Az üzenetküldés gyakorisága tehát alapvetően független az érzékelés gyakoriságától, viszont az alkalmazás igényei nagyban befolyásolják. Ennek megfelelően három különböző adatküldési stratégiát [Tilak2002] különböztethetünk meg: *eseményvezérelt*, *idővezérelt*, illetve *lekérdezés alapú* működést.

Idővezérelt (time-driven) működés esetén a szenzor periodikus időközönként elküldi mérési eredményeit a bázisállomásnak, függetlenül a mért értékektől. Egy tipikus példa ilyen megoldásra egy meteorológiai szenzor egy okos városban, mely előre meghatározott időközönként küldi el az általa mért hőmérséklet, páratartalom vagy légnyomás adatokat.

Bizonyos esetekben azonban nincs szükség folyamatos adatküldésre, csak akkor kell a bázisállomást értesíteni, ha valami szokatlan, a megszokottól eltérő „esemény" történik. Ezt a megoldást hívja a szakirodalom *eseményvezérelt (event-driven)* működésnek. Konkrét példaként említhetjük a már érintett parkoló szenzorok esetét: ameddig nem történik változás egy parkolóhely állapotában (folyamatosan szabad vagy folyamatosan foglalt) nem szükséges értesíteni a bázisállomást, vagy legalábbis nem olyan sűrűn, mint egy forgalomszámláló szenzor esetén. Ha viszont egy szabad helyet elfoglal egy autó, vagy egy foglalt hely felszabadul, akkor erről az „eseményről" rögtön értesíteni kell a bázisállomást.

Egy másik tipikus példa az, ha talajnedvességet mérünk, mondjuk egy okos parkban; alapesetben a talaj nedvességtartalma megfelelő, a szenzornak nem kell semmit küldenie a bázisállomás felé. Ha viszont a nedvességtartalom egy bizonyos szint alá csökken, azaz túlságosan száraz a talaj, akkor erről az eseményről értesítést kell küldeni, hogy adott esetben be lehessen indítani az öntözőberendezést.

Az eseményvezérelt működés előnye, hogy általában jóval energiatakarékosabb, mint az idővezérelt megoldás. Hátránya viszont, hogy kevésbé megbízható. Ha huzamosabb ideig nem kapunk értesítést egy adott szenzortól, ez lehet annak a jele, hogy nem történt semmilyen jelentést érdemlő esemény, de az is lehet, hogy a szenzor lemerült, vagy meghibásodott, és ezért nem küld semmit. Ennek a kiküszöbölésére érdemes az eseményvezérelt és idővezérelt működést ötvözni, azaz alapesetben a szenzor csak az előre definiált események bekövetkeztekor jelez, viszont periodikusan (de jóval ritkábban, mint egy idővezérelt működésnél) küld azért egy-egy üzenetet, jelezve hogy továbbra is rendeltetésszerűen működik.

Egy olyan ötvözése is elképzelhető az eseményvezérelt és az idővezérelt működésnek, melyben az idővezérelt működés paraméterei változnak dinamikusan annak függvényében, hogy a rendszeresen lejelentett értékek elérik-e az egy adott eseményhez tartozó küszöbértéket vagy sem. Hogy egy példát mondjunk erre, egy nukleáris erőművön belül, illetve annak környezetében a szenzorok folyamatosan, idővezérelt módon mérhetik a radioaktivitás mértékét, de sem magát a mérést, sem az adatok küldését alapesetben nem szükséges percenként megtenniük, hiszen normális, rendeltetésszerű működés esetén ezek az értékek nagyon alacsonyak lesznek, és alapvetően nem változnak. Ez tehát egy hagyományos idővezérelt működést jelent, viszonylag nagy időintervallumokkal. Ha viszont bekövetkezik valamilyen probléma az erőmű működésében, ami esetleg magával hozza a radioaktivitás mértékének emelkedését, akkor – ha a jelentett értékek meghaladnak egy bizonyos küszöbértéket –, a hálózat automatikusan egy sokkal sűrűbb érzékelési és kommunikációs módra fog váltani. A szenzorhálózat továbbra is idővezérelt módon fog működni, de az időintervallumok csökkennek a bekövetkezett esemény hatására. Ha pedig később a radioaktivitás szintje ismét normalizálódik, vissza lehet majd térni a nagyobb időintervallumokra.

A fenti két megoldással ellentétben a vezeték nélküli szenzorhálózat kommunikációját lehet *lekérdezések alapján* is vezérelni (*query-driven network*). Ebben az esetben a szenzorok telepítve vannak, bizonyos időközökben mérhetnek, de jelentéseket nem küldenek feltétlenül a bázisállomás felé, spórolva az energiatartalékaikkal. Viszont, ha az alkalmazás valamelyik modulja, vagy egy felhasználó megfogalmaz egy lekérdezést a hálózat felé, akkor a lekérdezésben meghatározott szenzorok válaszolni fognak erre.

2.9. ADATOK AGGREGÁLÁSA

Mint már említettük, a szenzorok energiahatékonyságát jelentősen növelhetjük azzal, ha a mérési adatokat a csomóponton vagy a hálózaton belül megpróbáljuk aggregálni. Ez bizonyos esetekben nyílván adatvesztéssel is jár, pontosabban az adatok térbeli vagy időbeli felbontása nem lesz olyan részletes, de ha ezzel együtt az alkalmazás igényeit ki tudjuk elégíteni, akkor az aggregálás javasolt [Krishnamachari2002].

Az időbeli aggregáció minden egyes szenzor csomóponton külön biztosítható. Ennek lényege az, hogy bár a szenzor sűrűn mintavételezi a megfigyelt környezeti paramétert, a mérési eredményeket nem küldi tovább azonnal a bázisállomás felé, hanem azokat helyben feldolgozza, és valamilyen aggregált értéket (például átlag, minimum vagy maximum érték, variancia) továbbít csak, jóval ritkább időközönként. Tipikusan mondjuk egy hőmérséklet adat nem szokott érdekelni minket olyan nagyon, az aggregált értékek kifejezőbbek lehetnek.

A térbeli aggregációnál ezzel szemben az egymáshoz térben közel elhelyezkedő, és emiatt nagyrészt redundáns mérési eredményeket produkáló szenzorok adatait aggregáljuk. A redundanciára szükség van ahhoz, hogy a hálózat megbízhatóan működjön néhány szenzor kiesése esetén is, a redundáns szenzorok méréseit viszont célszerű aggregálni a kommunikációs útvonalak mentén, ezzel csökkentve a felesleges adatforgalmat. A redundáns szenzorok itt abban is segíthetnek különben, hogy az esetlegesen meghibásodott szenzorok szokatlan, az átlagtól jelentős mértékben eltérő mérési eredményeit fel tudjuk ismerni (*outlier detection*). Ebben az esetben ezeket az eredményeket, melyek egy adott szenzor hibás működéséből adódnak, természetesen ki fogjuk majd szűrni, nem vesszük bele az aggregációba, hiszen az jelentősen torzítaná, mondjuk az átlag értékét.

Egy hierarchikusan klaszterezett szenzorhálózat esetén az aggregációt célszerű lehet a klasztervezérlőkben elvégezni, egy lapos, többugrásos hálózatban viszont az aggregáció megtörténhet minden egyes elágazási pontban a bázisállomást és a szenzorokat összekötő feszítőfa mentén. Ilyenkor az adatbegyűjtési folyamatot célszerű úgy szervezni, hogy a bázisállomástól legtávolabb eső csomópontok kezdjék el először az adatküldést, és az útvonalak belső csomópontjain elhelyezkedő szenzorok csak akkor küldjenek, ha a rajtuk keresztül kommunikáló összes szenzortól megérkezett már a (részben) aggregált adat, azt feldolgozták, és a saját mérési eredményüket is beillesztették az aggregátumba. Mindez azonban sokszor nem egy triviálisan kezelhető feladat, főleg, ha az alkalmazásnak komoly szolgáltatásminőségi követelményei vannak, például a késleltetést illetően.

A szakirodalom rendkívül bőséges az adataggregációs megoldásokat illetően [Rajagopalan2006]. A különböző javaslatok általában valamilyen kompromisszumot próbálnak elérni a hálózat élettartama, a késleltetés, az adatok pontossága és az energiafogyasztás között. Ezen megoldások részletes bemutatása azonban meghaladja e kismonográfia kereteit.

3. MOBILITÁS SZENZORHÁLÓZATOKBAN

Amikor a vezeték nélküli szenzorhálózatok reflektorfénybe kerültek néhány évvel ezelőtt, az alkalmazások és a kutatások nagy része szinte kizárólag a statikus környezetekre fókuszált. Ennek ellenére hamar felismerték, hogy a szenzor csomópontok vagy a bázisállomás mobilitásának előnyei lehetnek, melyeket érdemes lehet megvizsgálni. A szenzor csomópontokról általában azt feltételezzük, hogy az energiatartalékaik túlságosan korlátozottak ahhoz, hogy önállóan mozogjanak.

Számos lehetőség van azonban arra, hogy a szenzorok ne saját energiájukat használják mozgásra: a szenzorokat viselhetik állatok, emberek, rögzítve lehetnek járművekre, de lebeghetnek is a víz felszínén, például monitorozva egy olajfolt terjedését. Ezekben az esetekben a szenzor csomópontok gyakorlatilag ellenőrizetlenül mozognak, a szenzorhálózat topológiája pedig ennek megfelelően nagyon dinamikusan változhat, ami megnehezíti a mérési adatok folyamatos rendelkezésre állásának biztosítását, vagy azok hatékony begyűjtését. Másfelől a kontrollált szenzor mobilitás, melynek során bizonyos szenzorok valamilyen konkrét algoritmusnak megfelelően mozognak, sok szempontból hasznos lehet: megszüntethetjük a lefedetlen területeket, közelről követhetjük a mozgó eseményeket, vagy összeköthetjük a gyéren lefedett területeken található csomópontokat a hálózat többi részével.

A szenzorokkal ellentétben viszont a bázisállomásoknak sokkal nagyobb energiatartalékaik lehetnek, az esetleg lemerülő akkumulátorok cseréje vagy újratöltése is sokkal könnyebben megvalósítható, ezért általában elfogadhatóbb a bázisállomások mobilitását feltételezni. A mobil bázisállomások használatának számos előnye lehet. Többugrásos hálózatokban a bázisállomás időközönkénti elmozdulása nyomán újrakonfigurálódhatnak az adatgyűjtési útvonalak, ami egy kiegyensúlyozottabb energiafogyasztást eredményez a hálózat egészében. Egyugrásos hálózatok esetében a távoli szenzorok energiaforrásai merülnek le leghamarabb, a bázisállomás mozgatásával viszont a kommunikációs távolságok lerövidíthetők. Ez a mozgatás lehet véletlenszerű, de vannak olyan megoldások is, melyek a szenzorok aktuális energiatöltöttsége alapján igyekeznek mentesíteni azokat a területeket, ahol kevesebb a rendelkezésre álló energia.

Mozgó szenzor csomópontok vagy bázisállomások esetén több fontos szempontot is szem előtt kell tartani. Hogyan ismertetik például a mobil elemek új fizikai elhelyezkedésüket a többi csomóponttal energiahatékony módon? Hogyan szervezzük át a kommunikációs útvonalakat az új pozíciókat figyelembe véve? Hogyan befolyásolja az elemek mobilitása más mechanizmusok (például klaszterezés, alvás-vezérlés vagy adataggregáció) hatékonyságát? A következőkben röviden bemutatunk néhány szenzor és bázisállomás mozgatási stratégiát, azok előnyeit és hátrányait.

3.1. BÁZISÁLLOMÁS MOBILITÁS

A bázisállomás mozgatása többféleképpen is elképzelhető. A legegyszerűbb megoldás a véletlenszerű mozgatás, amikor a bázisállomás például véletlenszerűen választott irányvektorok mentén véletlenszerűen választott ideig mozog, majd irányt változtat, ismét véletlenszerűen választott paramétereket használva. A *Data MULE (Mobile*

Ubiquitous LAN Extensions) [Shah2003] megoldás például mozgó adathordozó „öszvéreket" használ, azaz olyan bázisállomásokat, melyek véletlenszerűen bolyonganak egy gyéren lefedett területen, összegyűjtik a szenzoroktól a mért adataikat, amikor elhaladnak mellettük, azokat tárolják, és ha egy vezetékes kapcsolattal rendelkező átjátszó csomópont közelébe kerülnek, akkor leadják a hordozott adatcsomagokat.

Ezzel a megoldással összegyűjthetőek az adatok akkor is, ha a hálózat teljes rádiós konnektivitása nem biztosított. Ráadásul, ha az adatgyűjtést több adathordozó öszvér is segíti párhuzamosan, akkor a megoldás megbízhatósága, robusztussága is biztosított. A javaslat hátránya viszont az, hogy kiszámíthatatlan mikor érkezik egy adott szenzor közelébe valamelyik „öszvér", ezért valós idejű alkalmazások esetén nem igazán használható.

Egy hasonló megoldás a *SENMA (Sensor Networks with Mobile Agents)* [Tong2003], ahol egy mobil ágens (például egy drón) véletlenszerűen repked egy szenzorokkal lefedett terület felett, és akkor gyűjti be a szenzorok adatai, amikor a rádiós kommunikációs viszonyok a legmegfelelőbbek. De léteznek olyan megoldások is, melyek a járókelők okostelefonjait használják a szenzorok adatainak begyűjtésére [Jayaraman2007], csak abban az esetben persze, ha ezek a felhasználók előzetesen telepítettek a telefonjukra egy erre képessé tevő alkalmazást. Az adatgyűjtő „öszvérekkel" vagy a repkedő drónokkal ellentétben persze a járókelők nem „véletlenszerűen" mozognak, hanem egy bizonyos célállomás felé haladnak valamilyen útvonal mentén. A célállomás kiválasztását a járókelők személyes indítékai befolyásolják, a szenzorhálózati alkalmazás erre nincs hatással.

Jósolható mozgás esetén a bázisállomás egy előre meghatározott útvonal mentén mozog, általában egyenletes sebességgel [Chakrabarti2003]. A szenzorok ismerik a bázisállomás útvonalát, és előre tudják, mikor fog a hozzájuk legközelebb álló ponton áthaladni, így megpróbálják erre az időpontra időzíteni a csomagküldésüket, ezzel is csökkentve az energiafelhasználást. Egy jó példa lehet erre a tömegközlekedési járművek (buszok, villamosok) használata a szenzor adatok begyűjtésére egy okos városban. Hiszen ebben az esetben ismerjük a buszok és villamosok útvonalát, menetrendjét, sőt, jó esetben az aktuális pozíciójukat és átlagos haladási sebességüket is, ha azok fel vannak szerelve például a Futár rendszerben is használt műholdas navigációs technológiával és rádiós kommunikációs megoldásokkal.

Vannak azonban olyan megoldások is, ahol az állomás egy körkörös útvonal mentén halad, de a szenzorok nem csak akkor küldenek adatokat, amikor a bázisállomás a legközelebb van hozzájuk, hanem folyamatosan, akkor is, ha a bázisállomás épp a hálózat átellenes részen van [Luo2005]. Ilyenkor persze egy többugrásos útvonal mentén éri el a csomag a bázisállomást. Ennek a megoldásnak a célja nem a kommunikációs távolságok csökkentése, hanem az, hogy a hálózat energiaterhelése minél kiegyensúlyozottabb legyen. A megoldás hatékonyságát összehasonlították a terület közepére kihelyezett statikus bázisállomás, illetve a véletlenszerűen mozgó bázisállomás esetével is. Az eredmények azt mutatták, hogy a bázisállomás véletlen mozgatásával már jelentősen csökkenthető a legterheltebb szenzor energiafogyasztása, de a bázisállomásnak a terület perifériáján történő körkörös, jósolható mozgatásával tovább javíthatjuk a terheléseloszlást, ez pedig a hálózat élettartamának jelentős, akár 500%-os növekedését is jelentheti. A megoldás hátránya ugyanakkor, hogyha a bázisállomás nem a hálózat közepén, hanem annak a perifériáján helyezkedik el, akkor egy adott szenzor üzenete akár kétszer olyan távolságra kell, hogy eljusson, ami időkritikus alkalmazásoknál nem mindig elfogadható. Egy felső korlát a késleltetésre azonban könnyedén adható, ellentétben a véletlenszerűen mozgó bázisállomás esetével.

Egy további optimalizálása a fenti javaslatnak az ellenőrzött mobilitás az olyan esetekre, amikor, bár a bázisállomás egy előre meghatározott útvonal mentén halad, de a sebessége nem egyenletes, hanem külső körülmények által befolyásolt, melyek előre nem jósolhatóak. Az *AIMMS (Autonomous Intelligent Mobile Micro-Server)*

[Kansal–Somasundara2004] [Kansal–Rahimi2004] rendszerben a bázisállomás a sebességét az aktuális területen beküldésre váró adatok mennyiségétől teszi függővé, azaz lelassít ott, ahol a környező szenzoroknak sok küldendő adatuk van, vagy viszonylag rossz a rádiós csatorna minősége, és felgyorsít ott, ahol kevés adatot kell begyűjtenie. Mivel azonban a szenzorok nem tudják előre, mikor fog megérkezni a bázisállomás a környezetükbe, a bázisállomás feladata a szenzorok értesítése jelenlétéről, és mérési adataik elkérése. Ez gyakorlatilag megfelel a korábban bemutatott lekérdezés alapú hálózatok működési modelljének.

Az eddig bemutatott megoldásoktól teljesen eltérnek a bázisállomás adaptív mozgatását célzó stratégiák, melyeknél a cél az, hogy a bázisállomást folyamatosan a hálózat optimális energiafogyasztását jelentő pozíció felé mozgassuk, figyelembe véve a hálózatban aktuálisan jelen levő eseményeket. Ha a hálózatban egy adott pillanatban csak azok a szenzorok akarnak kommunikálni, melyek érzékeltek egy bizonyos eseményt, akkor érdemes lehet a bázisállomást közel vinni ezekhez a szenzorokhoz, ezzel is csökkentve a kommunikációs útvonalak hosszát, növelve az energiahatékonyságot és a kommunikáció megbízhatóságát.

Ha azonban több párhuzamos esemény is megjelenik a hálózatban, nem egyértelmű, hogyan válasszuk ki a bázisállomás optimális pozícióját. Egy megoldás erre az lehetne, hogy a hálózat teljes energiafogyasztását minimalizáljuk, figyelembe véve az aktuális események térbeli eloszlását. Mivel egy esemény jelentéséhez szükséges energia arányos az esemény és az állomás közötti távolsággal, ez azt jelentené, hogy meg kell keresni azt a pozíciót, amely esetében az átlagos távolság a párhuzamos eseményekhez minimális.

Ennek a megoldásnak a hátránya az, hogy bár a teljes energiafogyasztást minimalizáljuk, az egyes szenzorokra jutó terhelés nagyon kiegyensúlyozatlan lehet. Ezért egy másik megoldás az lehet, hogy a legterheltebb szenzor energiafogyasztásának minimalizálását célozzuk meg, azaz a bázisállomás és az események közötti maximális távolságot minimalizáljuk. Nem életszerű viszont azt feltételezni, hogy a bázisállomás közvetlenül az optimális pozícióba tud mozogni, inkább csak elindul, „lép egyet" e pont felé egy adott időintervallumban. Az események viszont megjelennek és eltűnnek, illetve akár mozoghatnak is a térben, ezért a feladat az, hogy folyamatosan újraszámoljuk az optimális pozíciót, és a bázisállomás közelítsen efelé, amennyire lehet [Vincze2007].

3.2. SZENZOR MOBILITÁS

A szenzorok mobilitására általában úgy tekintenek, mint egy plusz dimenzió, mely jelentősen megnehezíti a szenzorok telepítését és menedzselését. Másfelől azonban a mobilitásra képes szenzorok a hálózatot rugalmasabbá, újrakonfigurálhatóvá teszik, és új lehetőségeket nyitnak a szenzorhálózatok optimális működtetésére.

Alapvetően két esetet különböztetünk meg a szenzorok mobilitásánál. Ha a szenzorok csak bizonyos külső körülmények miatt mozognak (például lebegnek a vízen vagy a levegőben), akkor a kérdés nem az, hogy hogyan mozgassuk őket, hanem hogy hogyan kezeljük ezt a természetes mobilitást úgy, hogy a hálózat hatékonyan működhessen. Másfelől viszont ha van lehetőség arra, hogy a szenzorok mozgását közvetlenül vagy közvetetten befolyásoljuk, akkor ezt a lehetőséget össze kell tudni hangolni a hálózat működését javító többi mechanizmussal.

Amikor irányított mobilitásról beszélünk, akkor több lehetséges cél is felmerülhet. A mobil szenzorok segítségével biztosíthatjuk a megfigyelt terület lefedettségét érzékelési és kommunikációs szempontból is, de adaptívan reagálhatunk a környezeti változásokra is. Külön érdemes megemlíteni a heterogén hálózatok esetét is, ahol csak néhány szenzor képes mozogni, hiszen ezek adott esetben drágább, nagyobb és több energiát igénylő eszközök lehetnek.

Egy lehetőség például az, hogy ezeknek a mobil szenzoroknak többletfeladatokat határozunk meg. Ha például a területet lefedjük nagyszámú kis, olcsó, statikus szenzorral, akkor, ha a mért adatok egy bizonyos érdekes eseményre utalnak a hálózat bizonyos részein, a mobil szenzorokat utasíthatjuk, hogy menjenek arra a területre és végezzenek pontosabb méréseket [Trajcevski2004] [Tseng2005] vagy készítsenek képeket [Gerla2003]. De vannak olyan elosztott szenzorhálózatokat használó megoldások is [Krishna2004], melyeknek célja több mozgó célpont észlelése és követése. Miután a szenzorok észlelik az adott célpontot, eldöntik, hogy elkezdik-e követni azt, vagy egy helyben maradnak. Ez a döntés függhet a célpont fontosságától, de a szenzorok közötti koordinációs mechanizmustól is.

4. SZENZORHÁLÓZATOK MODELLEZÉSE

A vezeték nélküli szenzorhálózatok új kommunikációs algoritmusainak a kidolgozásánál fontos, hogy nagyméretű szenzorhálózatok működését tudjuk modellezni, figyelembe véve a szenzor csomópontok speciális feltételeit. Mivel a hagyományos, általános hálózati szimulációs eszközök nem igazán alkalmasak erre a célra, az utóbbi években dedikált WSN szimulációs szoftvereket is kidolgoztak. Bármennyire is hasznos azonban egy szimulációs megoldás, a valós kommunikációs viszonyokat és a szenzorok valós energiafogyasztását nem mindig egyszerű megfelelően modellezni. Nagyméretű hálózatok kísérleti célú telepítése viszont sok pénzbe kerülhet, ezért vált fontossá olyan teszthálózatok kiépítése, melyeken keresztül akár több száz valós, telepített szenzorhoz lehet távolról hozzáférni, ezeket a csomópontokat fel lehet programozni a kívánt kísérleti algoritmussal, majd valós méréseket lehet futtatni valós szenzorokon, valós körülmények között. A továbbiakban röviden bemutatunk néhány WSN szimulációs környezetet és teszthálózatot, melyek nagy népszerűségnek örvendenek a témával foglalkozó kutatók körében.

4.1. SZIMULÁCIÓS ESZKÖZÖK

Az egyik legismertebb a *TOSSIM* szimulációs környezet, amely a *TinyOS* operációs rendszert használó szenzor csomópontok működésének elemzésére alkalmas [TOSSIM2013]. Ez egy diszkrét esemény szimulátor: futás közben az eseményeket egy idő szerint sorrendezett sorból olvassa ki, majd hajtja végre. Két programozói interfészt támogat, Python és C++ nyelveken. Pythonban lehetséges egy futó szimulációval dinamikusan kommunikálni, elemezéseket végezni lépésről lépésre, mint egy nyomkövető, hibakereső programmal. A Python emulátor viszont nagyon lassú lehet bizonyos esetekben, ezért használják a C++ interfészt is. A kód fordítása a két nyelv között általában nagyon egyszerűen megoldható.

Bár a TOSSIM egy nagyon precíz szimulációs környezet, számos hátránya is van, főleg ha nagyobb méretű hálózatok működését szeretnénk elemezni. Alapvetően a TOSSIM maximum 1000 csomópontos homogén hálózatokat tud szimulálni, bár ez a TinyOS operációs rendszer kötöttségéből adódik, miszerint egy hálózaton belül minden csomóponton ugyanannak a kódnak kell futnia. Másfelől a csomópontokon futó alkalmazás számára allokált memória – függetlenül a hálózat méretétől – fix méretű, ezért ha nagyobb hálózatokat akarunk emulálni, akkor egyszerűbb, kompaktabb alkalmazásokat kell írnunk. További hátrány, hogy a bit-szintű granularitásnak köszönhetően a szimulációs idő nagyon megnőhet a hálózat méretével; egy 1000 csomópontos hálózat esetében akár hetekig is futhat egy adott szimuláció. Léteznek azonban olyan „kiskapuk" is [Li2012], melyek segítségével nagyméretű és heterogén hálózatok szimulációja is hatékonnyá tehető a TOSSIM környezetben.

Vezeték nélküli szenzorhálózatok szimulációja megoldható a talán legismertebb, *ns-2 (Network Simulator 2)* hálózati szimulátorban is [Issariyakul2011], bár nem erre a célra tervezték. Az ns-2 egy objektum orientált, diszkrét esemény szimulátor, és szintén két nyelvet használ: *C++* és *OTcl (Object oriented Tool Command Language)*. C++ -ban kell megírni a különböző protokollokat és kiterjeszteni a szimulációs könyvtárakat, míg OTcl szkriptekkel

lehet a szimulátort konfigurálni, megadni a hálózati topológiát, a szimulációs forgatókönyveket, illetve megjeleníteni a szimulációs eredményeket. Ami a szenzorhálózatok szimulációját illeti, az ns-2 támogat különböző rádiós protokollokat, mint az *IEEE 802.11 (WiFi)* vagy az *IEEE 802.15.4 (Zigbee)*, de sok hiányossága is van. Nem lehet modellezni például ns-2-ben az érzékelési folyamatot; az energia modell, a csomag formátum és a közeghozzáférési protokollok pedig teljesen különböznek a valós életbeli szenzorhálózati megoldásoktól.

Az *ns-3* [ns3 2017], mely szintén egy nagyon ismert és sokak által használt szimulációs környezet, nem igazán tekinthető az ns-2 kiterjesztett verziójának, hanem inkább egy teljesen új szimulációs eszköz, mely nem is kompatibilis az ns-2-vel. Ami a szenzorhálózatokat illeti, az ns-3 tartalmaz már olyan modulokat, melyek a 802.15.4, a *6LoWPAN (IPv6 over Low power Wireless Personal Area Networks)* [Montenegro2007] vagy az *RPL (IPv6 Routing for Low-Power and Lossy Networks)* [Winter2012] protokollokat implementálják.

Ezen kívül számos más szimuláció környezet létezik a szenzorhálózatok kommunikációjának modellezésére, mint például az *OMNeT++* környezetre épülő *Castalia* [Rastegarnia2011], a *J-SIM* [Sobeih2005], a *SENSE* [Chen2005], az *sQualNet* [Varshney2007] vagy a *Shawn* [Fekete2007]. Létezik számos áttekintő cikk is a szenzorhálózatok szimulációs környezeteit illetően, például [Nayyar2015], de jelen tanulmányban nincs lehetőség a terület ennél részletesebb ismertetésére.

4.2. TESZTHÁLÓZATOK

Egy hatékony, gyors, és könnyen kezelhető szimulációs eszköz kétségkívül nagy segítséget jelenthet a szenzorhálózatokkal foglalkozó kutatóknak és fejlesztőknek, de számos olyan környezeti paraméter befolyásolhatja a szenzorok működését, melynek valósághű modellezése sok esetben szinte megoldhatatlan. A másik lehetőség tehát az, hogy egy nyílt hozzáférésű teszthálózat valós szenzor csomópontjaira töltjük fel a javasolni kívánt alkalmazást vagy protokollt, így tesztelve annak helyes és hatékony működését.

Ilyen, kifejezetten szenzorhálózatokra tervezett teszthálózatból szerencsére egyre több van. A *WISEBED* [Coulson2012] például egy nagyméretű WSN teszthálózat, melyet 9 európai egyetem és kutatóintézet közösen hozott létre. A különböző helyszíneken telepített hálózat különböző gyártók különböző funkcionalitással rendelkező eszközeit képes integrálni egy közös middleware réteg segítségével és támogatja a Contiki, a TinyOS és az iSense operációs rendszereket futtató csomópontokat is.

Az *IoT-LAB (korábbi nevén SensLAB)* [Burindes–Roziers2011] [FIT/IoT-LAB] egy több mint 2700 szenzor csomópontot tartalmazó teszthálózat, mely hat franciaországi helyszínen (Inria Grenoble, Inria Lille, ICube Strasbourg, Inria Saclay, Inria Rennes és Institut Mindes-Telecom Paris) van elosztott módon telepítve (4.1. ábra). Az IoT-LAB biztosítja a hálózati csomópontok teljes kontrollját, illetve közvetlen hozzáférést nyújt az átjátszó csomópontokhoz, melyekhez a szenzorok kapcsolódnak. Lehetővé válik ezáltal a szenzorok energiafogyasztásának monitorozása, illetve különböző hálózati metrikák (mint a végponttól-végpontig késleltetés, az áteresztő képesség vagy az overhead) megfigyelése. A kísérleti protokollok telepítése gyors és egyszerű, csakúgy, mint az eredmények begyűjtése és feldolgozása. A *Contiki* és a *TinyOS* mellett a *FreeRTOS* operációs rendszert is támogatja, a hálózathoz pedig egy webes felületen keresztül lehet hozzáférni.

4.1. ábra: Az IoT-Lab egyik alhálózata (https://www.iot-lab.info/workshop-iot-oct2015/)

Az egyik legismertebb európai okos város projekt a *Smart Santander* volt, mely 2010 és 2013 között futott, de a kiépített érzékelési infrastruktúra továbbra is működik, a több ezer telepített szenzor (hőmérséklet, légszennyezés, zaj, fény, és parkolás mérésére) pedig továbbra is üzemel, és szolgáltatja a városüzemeltetés és a városlakók számára a több százezer mérési eredményt naponta. Viszont kevesen tudják talán, hogy ez a telepített szenzorhálózat teszthálózatként is használható [SmartSantander2014]. Minden egyes szenzor csomópont ugyanis egy *Libelium Waspmote Expansion Radio Board*-ot tartalmaz (4.2. ábra), mely gyakorlatilag két rádiós egységet tartalmaz, melyek mindegyike 2.4 GHz-en kommunikál.

Az egyik rádiós egységen egy *DigiMesh* nevű módosított IEEE 802.15.4 protokoll fut, egy egyszerű útválasztó algoritmussal. Ezt az egységet használják az okos város alkalmazások által használt szenzor adatok begyűjtésére és a jelzésforgalomra is, illetve ezen az interfészen valósul meg a csomópontok flashelése is, ha egy módosított kódot szeretnénk futtatni rajtuk.

4.2. ábra: Libelium Waspmote Expansion Radio Board

A másik rádiós egységen viszont egy szabványos 802.15.4 protokoll fut, ezt lehet használni tesztelésre, azaz bárki írhat és futtathat teszt alkalmazásokat ezen a nagyméretű telepített szenzorhálózaton úgy, hogy az nem zavarja a Smart Santander alkalmazások rendeltetésszerű működését. Az egyetlen kivételt a parkolási eseményeket érzékelő, aszfaltba süllyesztett szenzorok jelentik, melyeknél nem megengedett tesztalkalmazások futtatása a korlátozott energiatartalékok miatt, illetve a mobil szenzorként működő rendőrautók, melyeken biztonsági okokból nem engedélyezték a második rádiós modul használatát és tesztalkalmazások futtatását.

További ismert szenzorhálózatos tesztrendszerek a Harvard egyetemen fejlesztett *MoteLab* [Allen2005], az uppsalai egyetemen fejlesztett nomadikus *Sensei* [Rensfelt2010] teszthálózat, vagy a Smart Santander projekthez hasonlóan a Massachussets-i Cambridgeben működő *CitySense* [Allen2008] városi teszthálózat.

5. A LORAWAN HÁLÓZATOK

A *LoRa (Low Power Radio)* rádiós technológia bemutatásával már a „Szenzorok és kommunikációs technológiáik" kismonográfiában [Jakab2018] is foglalkoztunk, azonban kevés szó esett arról, hogy hogyan tudjuk ezeket a rádiós modulokat hálózatba szervezni. Jelen fejezetben egy *LoRaWAN (LoRa Wide Area Network)* hálózatot és annak működését mutatjuk be felhasználói és szolgáltatói oldalról.

A LoRaWAN hálózatok szerveződésének bemutatása előtt érdemes megismerni, hogy mit várhatunk egyáltalán ezektől a hálózatoktól, amelyek a LoRa rádiós technológiára épülnek. Alacsony fogyasztású és nagy kiterjedésű hálózatról beszélünk, így érdemes megvizsgálni, hogy mit is jelentenek a gyakorlatban ezek a kifejezések a LoRa esetében.

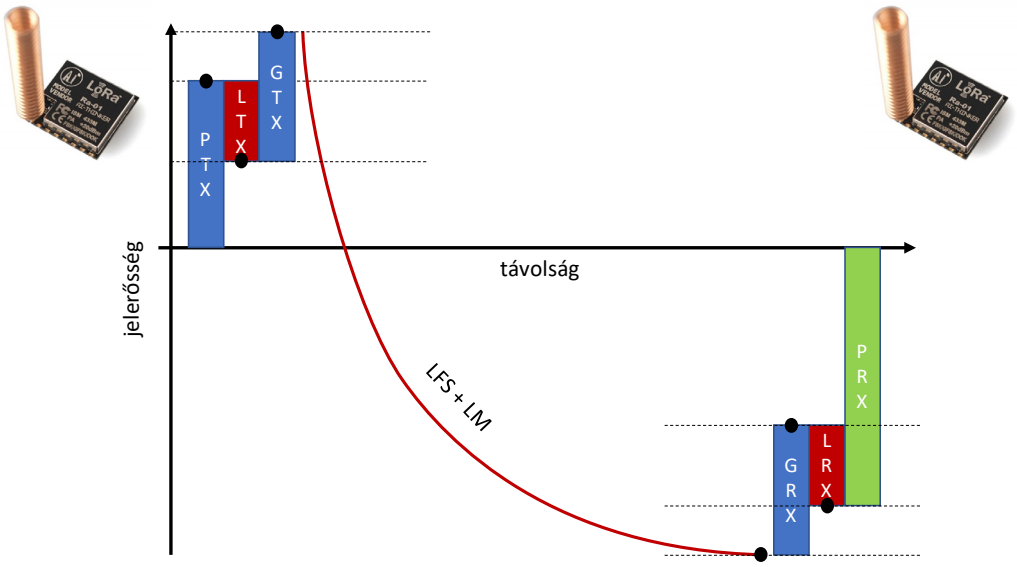
5.1. A LORA RÁDIÓRA ÉPÜLŐ HÁLÓZAT KITERJEDÉSE

Maga a LoRa az alacsony energiafelhasználású rádió modul, mégis igen nagy, több kilométeres nagyságrendű kapcsolatokat tud kiépíteni, köszönhetően a modulációs technológiának. Az alacsony energiafelhasználás egyik mércéje a *kapcsolat költségvetésének (link budget)* vizsgálata. Ez a mérce megmutatja, hogy mennyi és milyen veszteségek és nyereségek érhetik a rádiós átvitelt, miközben egyik pontról a másikra jut. Minden nyereség és veszteség bele van számítva, az eszközökön található antennától és vezetékektől kezdve a terjedés közbeni veszteségeken át.

A link budget fontosabb elemei a következőek:

- küldő kimeneti teljesítmény (P_{TX});
- küldő antenna nyereség (G_{TX});
- küldő oldali veszteségek, amelyeket az elektronika, kábelezés okoz (L_{TX});
- veszteség a rádiós térben (L_{FS});
- veszteségek a rádiós terjedés hatásai miatt (L_M), mint például interferencia;
- vevő oldali antenna nyereség (G_{RX});
- vevő oldali veszteségek (L_{RX}), az elektronika, áramköri jelterjedés miatt.

Összeadva ezeket a nyereségeket és veszteségeket, megkapjuk a *vevőhöz érkezett teljesítményt (PRX)*. Amennyiben ez a teljesítmény még mindig nagyobb, mint a *vevő érzékenysége (SRX)*, úgy a kommunikáció sikeres, azonban amennyiben alatta van, úgy nem tud a két egység egymással kommunikálni. Mint látható, a kommunikáció sikerére tehát nagyon sok tényező hat (5.1. ábra). Egy részét ezeknek a műszaki kialakítás adja, másik részét viszont az áthidalandó kapcsolat fizikai paraméterei.



5.1. ábra: Link Budget értelmezése *(saját ábra)*

Amennyiben meg szeretnénk becsülni, hogy egy ilyen kapcsolat maximálisan milyen távolságra tud eljutni, akkor tulajdonképpen nekünk a veszteségek maximális mértékét kell meghatároznunk a rádiós térben. A nyereség sem lehet ugyanakkor tetszőleges. Egy része hatóságilag szabályozva van, másik része pedig a fizikai jellemzők, illetve költségek miatt fog egy értéknél nem magasabbra menni. A hatósági szabályozás jelentősen eltérhet földrészek között, jellemzőn más paramétereket használnak az Amerikában, mint Európában.

Nézzünk meg egy tipikus értékkészletet a fenti paraméterekhez. A küldő oldali teljesítményt a hatóság, illetve az eszköz tápellátása korlátozza. Utóbbit most tekintsük úgy, hogy megfelelő mennyiségű energia áll rendelkezésre, így csak a hatósági korlát számít. Európában ez a korlátozás a kliens-szerver, azaz felfelé menő irány viszonylatában maximum 14 dBm. Csak összehasonlításképpen, az USA területén 20 dBm. Az antennákat tekintve általában a költségek szabnak határt a magas értékeknek. Tipikusan egy már jónak nevezhető antenna páros 9 dBi nyereséget nyújt. A költségterv pozitív oldalára írjuk még a vevő érzékenységét is. Ez az érzékenység változik gyártóról gyártóra, de természetesen az is számít, hogy milyen modulációt, illetve a moduláción belül milyen paramétereket használunk. Az igen gyakran alkalmazott, Semtech által gyártott SX1276 rádiós modul esetében ez akár -148 dBm.

Ezeket a nyereségeket összeadva és a vevő oldali nyereségek pozitívan számítva, hiszen ez a vételi képességet mutatja, a mostani példában a 173 dBm értéket kapjuk. A link budget ugyanakkor olyan veszteségeket is tartalmaz, amelyek zajként jelentkeznek a vevő oldalon függetlenül az áthidalandó távolságtól. Ezek miatt a link budget értéke reálisan 160 dBm érték köré csökken. Ebből az értékből kell következtetnünk az áthidalható távolság mértékére.

5.1.1. LoRa kapcsolat a szabadban

A rádiós kapcsolat, miközben az éterben terjednek a rádiójelek, veszteséget szenved és nagy távolságban már csak a töredék teljesítmény érkezik meg. A veszteség megbecslésére többféle matematikai képlet létezik. A legegyszerűbb eset az, amikor az adó és a vevő közvetlen rálátással van egymásra. Ez a *Line of Sight (LOS)* elnevezéssel bír. Természetesen ez egy ideális körülmény és bizony szenzorhálózatok esetén, főleg városokban, igen ritkán tud teljesülni. Mindenesetre az így kalkulált eredmények egy felső becslést adnak a maximálisan áthidalható távolságnak. Az LOS környezetet figyelembe vevő számítások a *FSPL (Free-Space Path Loss)*, azaz a szabad térben keletkező veszteséget mutató eredmények. Tudnunk kell, hogy a számításokban a szükséges teljesítmény négyzetesen arányos a távolsággal, illetve a rádióhullámok frekvenciájával. Tulajdonképpen a képletben csak ez a két tényező szerepel. Az FSPL a következő képlettel írható le:

$$FSPL(dB) = 10\log_{10}\left(\frac{4\pi df}{c}\right)^2$$

ahol a *d* a távolságot jelöli méterben, illetve az *f* paraméter a frekvenciát Hertzben. A *c* pedig a vákuumban mért fénysebesség. A képlet kicsit barátságosabb formára hozható, illetve a mértékegységeket át lehet számolni kilométerre és megahertzre. Ez esetben az egyszerűsítés után már a következő képletet kapjuk:

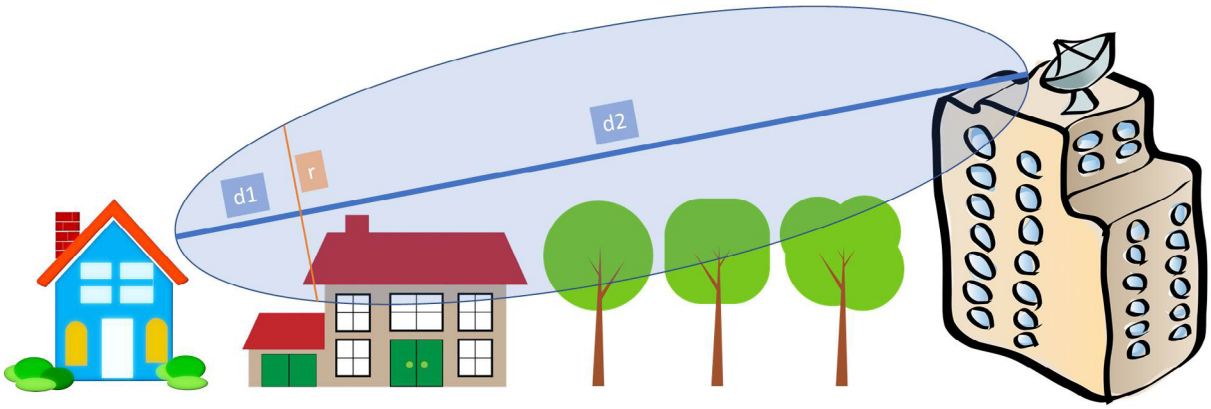
$$FSPL(dB) = 20\log_{10}(d) + 20\log_{10}(f) + 32.45$$

ahol a távolság (*d*), már kilométerben, a frekvencia (*f*) pedig már megahertzben van kifejezve.

A korábbi 160dBm értékből indulva, ha csak és kizárólag a szabad térben történő terjedés veszteségét figyelnénk, valamint az Európában leginkább használt 868 MHz ISM sávot tekintenénk, akkor a maximális távolságra 2750 km-t kapnánk. Sajnos azonban nem szabad csak ezt a paramétert tekinteni, ugyanis a rádióhullámok a terjedés közben interferálnak is, így további veszteség is jelentkezik. Ráadásul a rádiójelek a többutas terjedés miatt akár saját magukkal is interferenciát okozhatnak. Ez a jelenség a *fading*.

A fading hatása veszteségként fog jelentkezni az átvitelben. A veszteség becslésére további képletek és modellek léteznek. Az egyik gyakran használt modell, az úgynevezett *Rayleigh fading* model [Sklar1997]. A modellek alapján nagyjából 20-30 dB értékkel szokás számolni a fading jelenség miatt. Így a korábbi eredményünket is korrigálni kell ezzel a veszteséggel. Az így, 30dB fading veszteség értékkel kalkulált maximális távolság 87 km lesz. Ez még mindig hatalmas távolság, de ne felejtsük el, hogy feltételeztük, hogy az adó és a vevő közvetlen rálátással van egymásra.

A képet tovább rontja, hogy a közvetlen rálátásnak nem csak egy egyenes mentén kell teljesülnie. A kutatók a *Fresnel zónát* [Hristov2000] definiálták, ami egy ellipszoid formájú térrész a közvetett kapcsolat körül (5.2 ábra). A Fresnel zónának ugyancsak mentesnek kell lennie minden akadálytól, ahhoz, hogy ne számoljunk további veszteségekkel a terjedés során. Sőt, nem csak olyan természetes akadályokra kell gondolnunk, mint az épületek, növényzet, domborzat, hanem nagy távolságok esetén már figyelembe kell vennünk a Föld görbületét is.



5.2 ábra: A Fresnel zóna *(saját ábra)*

A Fresnel zónát egy adott pontban a kapcsolatra merőleges síkban értelmezett sugárral adjuk meg. A kiszámításához a következő képletet használjuk:

$$r = 17{,}31 \cdot \sqrt{\frac{d1 \cdot d2}{f \cdot d}}$$

ahol *d* az adó és vevő közötti távolság méterben, *d*1 és *d*2 az adott pont távolsága az egyik végtől és a másik végtől, végül pedig *f* a rádióadás frekvenciája megahertzben.

Ahhoz, hogy a képlet szerint a korábban kiszámolt 87 kilométeres távolságra képzelt Fresnel zónát tisztán tartsuk, úgy a zóna közepén, ahol a sugár a legnagyobb értéket mutatja, 86,66 méteres szabad területet szükséges hagyni. Ráadásul ebben a távolságban már jelentős a Föld görbülete is. Ahhoz, hogy közvetlen rálátásunk legyen két pontra ilyen távolságból, úgy, feltételezve, hogy az egyik pont a földfelszínén van, úgy a másik pontnak már 590 méter magasan kell lennie. Rövidebb távolságok esetén inkább a Fresnel zóna átmérője dominál, míg hosszabb távolságok esetén már a Föld görbülete számít jobban.

A Fresnel zóna esetében általában megengednek 20-40% akadályoztatást, ekkor ugyanis még nincs szignifikáns mértéke a veszteségnek. Mindazonáltal a korábbi számításokból látható tehát, hogy ez a távolság csak nagyon magasra kihelyezett antennák esetében fog működni.

5.1.2. LoRa kapcsolat városi környezetben

Városi környezetben ugyanakkor már máshogy kell számolnunk, mint a korábban leírtak, hiszen itt a leggyakrabban nem teljesül a közvetlen rálátás, az adó és vevő között több akadály is lehet, amely jelentősen befolyásolhatja a rádióhullámok terjedését.

Többféle matematikai modell létezik a városi környezet leírására. Az alábbiakban a LoRa esetében is jól alkalmazható *Okumura-Hata modellt* mutatjuk be. A modell a paraméterek állításával zsúfolt nagyvárosok és kevésbé sűrűn beépített elővárosok leírására alkalmas. Maga a modell az *Okumura modellre* [Okumura1968] épül, amit

Tokyo városában gyűjtött adatokra építettek. A *Hata modell* [Hata1980] egy továbbfejlesztett modell, amely tapasztalati megfigyeléseken alapszik. A megfigyelések alapján nagyvárosi környezetben a rádiós kapcsolat terjedési vesztesége (Path Loss, PL) a következő képlettel írható le:

$$PL(db) = 69{,}55 + 26{,}16 \cdot \log_{10} f - 13{,}2 \cdot \log_{10} hB - CH + (44{,}9 - 6{,}55 \cdot \log_{10} hB) \cdot \log_{10} d$$

ahol *f* a rádiós átviteli frekvencia megahertzben számolva, *hB* a vevő antenna magassága méterben, amely a kidolgozott modell működéséhez 30 méter és 200 méter közé esik, *CH* a küldő antenna magasságát jellemző faktor és *d* a távolság az adó és a vevő között.

Látható, hogy a képlet használatához még egy tényező kibontása szükséges, ez a küldő antenna magasságát jellemző tényező. A megfigyelések alapján ez az érték eltérő a város méretének függvényében. Közepes és kis városok esetén ezt a tényezőt a következő képlet írja le:

$$CH = 0{,}8 + (1{,}1 \cdot \log_{10} f - 0{,}7) \cdot hM - 1{,}56 \cdot \log_{10} f$$

ahol a *hM* paraméter a küldő antenna magassága méterben. A képlet akkor ad pontosabb becslést, ha ez a magasság 1-10 méter sávban helyezkedik el.

A fentiek alapján már számolható, hogy egy közepes, illetve nagyváros esetén az európai LoRa hálózatban használt 868 MHz frekvencián milyen veszteséget becslünk az adó és vevő között a távolság függvényében, adott antenna magasságokkal számolva.

A mostani példában számoljunk azzal, hogy a vevő antenna 40 méter magasan található, amely egy 10 emeletes ház tetejének felel meg. Az adó antenna legyen 2 méter magasságban, amely pedig egy egyszintes háznak felel meg. A paraméterek alapján először kiszámolandó a nagyvárosi környezetre jellemző antenna magasságát jellemző tényező, amely a behelyettesítések után a *CH* = 1,28 értéket adja. Ezt behelyettesítve a Hata modell veszteség képletébe, azt kapjuk, hogy:

$$PL(db) = 123 + 34.4 \cdot \log_{10} d$$

A korábbi számítások alapján azt mondtuk, hogy a terjedési veszteség a LoRa esetében, figyelembe véve a rádiós paramétereket és a frekvenciát, 160 dBm. Ezzel az értékkel számolva azt kapjuk, hogy közepes és nagy városokban az áthidalható távolság 11,9 km. Érdemes megjegyezni, hogy a valóságban találkozunk olyan eszközökkel is, amelyek esetében a link budget kisebb értéket mutat, jellemzően a kisebb antenna és a nagyobb, modulon belüli veszteségek miatt. Ilyen eszközök esetében a 150 dBm érték jobban közelíti a valóságot. Ugyanezeket a számításokat elvégezve, így már a távolságra csak 6,1 kilométert kapunk.

A Hata modellnek létezik egy külvárosokban, illetve elővárosokban alkalmazható változata is. Ebben a modellben azt korrigálják, hogy a külvárosokban általában alacsonyabb épületek találhatóak, az épületek falai vékonyabbak, elhelyezésük elszórtabb. Mindezek kisebb veszteséget eredményeznek, mint a közepes és nagyvárosi modell esetében. A megfigyelésre alapuló modellben a következő képlet írja le a közepes és nagyvárosi, illetve a külvárosi veszteségek különbségét:

$$LSU = LU - 2 \cdot \left(\log_{10} \frac{f}{28}\right)^2 - 5.4$$

ahol az *LSU* a külvárosi jelterjedési veszteség (Path **L**oss in **Su**burban Areas), míg az *LU* a fenti képletekkel számolt közepes és nagyvárosi veszteség (Path **L**oss in **U**rban Areas).

Az Európában használatos 868 MHz frekvencián azt kapjuk, hogy a két számítás között a különbség LSU = 9,84 dBm. Ennyivel lesz kevesebb a külvárosban a rádiós hullámok terjedési vesztesége, mint a közepes és nagyvárosok esetében. Elvégezve a távolságra vonatkozó számításokat, egy nagyon jó, maximálisan 160 dBm terjedési veszteség értéket elbíró kapcsolat esetében a távolság 23 kilométer, míg a reálisabb összeállítást tükröző 150 dBm esetében ez 11,7 kilométer.

Végül pedig a Hata modell számítása kapcsán meg kell, hogy jegyezzük, hogy azok kültérben elhelyezett antennákra vonatkoznak csak. Amennyiben az antennák, tipikusan az adó oldali antenna, a lakás belterületén van elhelyezve, úgy újabb és sajnos jelentős mértékű jelterjedési veszteséggel kell számolnunk. A tapasztalati mérések alapján ez a veszteség, amely az épület falain való átjutás eredménye, közepes és nagyvárosok esetén 20 dBm, míg külvárosok vagy elővárosok esetében 15 dBm mértékű. Praktikusan ez azt jelenti, hogy közepes és nagy városokban az áthidalható távolság a korábban bemutatott értékek negyedére csökken, míg elővárosok esetében közel harmadára. Praktikusan egy 150 dBm terjedési veszteséget elviselő kapcsolat a nagyvárosban 1,5 kilométer távolságot tud áthidalni, míg külvárosban 4,3 kilométert.

A leírtak alapján látható, hogy mit érthetünk nagy kiterjedésű hálózat esetén. A városokon kívüli környezetben, így például a precíziós mezőgazdasági alkalmazások esetében valóban nagy, több tíz kilométeres távolságokkal számolhatunk. Azonban, ahogy a városon belülre érünk és egyre több épület emelkedik a rádiós terjedés útjába, és jellemzően az okos város alkalmazások használják a rádiós csatornákat, úgy az áthidalható távolság lecsökken nagyjából 1,5 kilométer távolságra. Példaként, a modell alapján egy Budapest méretű nagyváros, amelynek 525 négyzetkilométer kiterjedéséből nagyjából 35 négyzetkilométer az igazi nagyvárosi környezet, lefedhető, még a gyengébb rádiós modulok esetén is, 26 külvárosi + 13 belvárosi antennával.

5.2. CSOMÓPONTOK SZÁMA A LORA RÁDIÓBÓL SZERVEZETT HÁLÓZATOKBAN

Egy hálózat kiterjedésével kapcsolatban azonban nem csak az fontos, hogy mekkora távolságokat tud áthidalni a rádiós kapcsolat, hanem az is, hogy mennyien használhatják együtt a hálózatot. A jelenlegi LoRa rádiós megoldások többsége az ALOHA típusú rádiós hozzáférést követi, ami praktikusan azt jelenti, hogy a rádiós modulok nem egyeztetnek az adás előtt, hanem amikor adni szeretnének, akkor elküldik a rádiós adást tekintet nélkül más adásokra. A párhuzamos rádiós adások egyrészt interferenciát okoznak, másrészt a vevő sem tudja párhuzamosan kiszolgálni ezeket az adásokat. Fontos tehát meghatározni, hogy hány rádiós adó osztozzon egy vevő állomáson. Szerencsére a szenzorok és alkalmazásaik jellegéből adódóan arra számíthatunk, hogy az adók viszonylag kis mennyiségű adatot fognak küldeni és a küldések között nagyobb alvási periódusok lesznek.

5.2.1. Terjedési idők a LoRa technológiára épülő hálózatokban

A LoRa technológia pontosan meghatározza a rádiós modulációt, azonban a küldendő adat moduláció előtti kódolása még több paraméter mentén eltérhet. Az egyik ilyen paraméter a *Spreading Factor (SF),* amely meghatározza, hogy egy adott elküldendő szimbólum hogyan képződjön le a moduláció során használt egységekre. A LoRa moduláció az átvitelhez *chirp* egységeket használ, amely egyszerűen fogalmazva egy áthúzás a kijelölt

frekvencia tartományban. A LoRa moduláció esetén a chirp sebessége az átvitel során használt frekvenciatartomány szélességétől függ csak, amely alap esetben 125 kHz. Továbbá LoRa esetében az összefüggés az, hogy a sávszélesség egyenlő a chirp küldési sebességgel, azaz 125 kHz sávszélesség esetén ez 125000 chirpet jelent másodpercenként. A spreading factor konkrétan azt mondja meg, hogy egy szimbólum hány chirp hosszú legyen. Az összefüggést a következő egyszerű képlet adja meg:

$$SF = \log_2 \frac{Rc}{Rs}$$

ahol Rc a chirp sebessége, míg Rs a szimbólum sebesség.

Fontos, hogy a LoRa technológia nem használja ki az összes lehetséges spreading factor értéket. Az európai használata során a 7 és 12 közötti spreading factor értékek használhatóak. Ez azt jelenti, hogy a leggyorsabb átvitel esetén, amelyet a 7-es spreading faktor nyújt, az átvitel kódolás során 128 chirp ír le egyetlen szimbólumot, meg a leglassabb átvitel esetén ennek harminckétszeresére, azaz 4096 chirp egységre van szükségünk. Ez egyben azt is jelenti, hogy a két sebesség között is nagyjából harminckétszeres különbség figyelhető meg.

A LoRa átvitel során az előbbieket kiegészítve még egy tényező befolyásolja a rádiós adások hosszát, ez pedig egy kódolási ráta. A LoRa rádiós átvitel esetében, annak érdekében, hogy az átvitt biteket épségben dekódolni lehessen a vevő oldalon, hibajavító kódolást használnak. A hibajavítás a *Forward Error Correction (FEC)* elven működik, azaz előre felkészülnek a rádiós átvitel során esetlegesen fellépő hibákra és úgy igazítják az elküldendő kódot, hogy amennyiben valahány bit az átvitel során megváltozna, úgy még azokat a vevő oldalon javítani lehessen. Ezt a redundanciát a *Coding Rate (CR)* fejezi ki. A LoRa átvitel során négyfajta CR étéket használhatunk: 4/5, 4/6, 4/7 és 4/8, az arányok a hasznos bitek és a ténylegesen átvitt bitek arányát mutatják. Azaz a 4/8 kódolási ráta esetén csak az átvitt bitek fele jelent valóban információt, a másik fél a hibajavítás miatt utazik. A leggyorsabb kódolási ráta esetén pedig az átvitt bitek nyolcvan százaléka lesz hasznos információ, a maradék húsz százalék a redundancia.

A fentiek alapján látható, hogy vannak gyorsabb és lassabb adatküldések is a LoRa rádió esetében. Felmerülhet a kérdés, hogy egyáltalán mi értelme van a lassabb adatküldésnek, hiszen ez mindenki számára rossz lehet, az információ lassabban érkezik meg, illetve a rádióadás hosszabb ideig foglalja a csatornát. A megoldás az, hogy ugyan az átvitel lassul, de egyben a vétel javul. A spreading factor egyben befolyásolja azt is, hogy az átvitel menynyire lehet zajos, úgy, hogy azt a vevő még dekódolni tudja. Azaz magasabb spreading factor tartományokban a vevő sokkal érzékenyebb.

Példaként egy 12-es spreading factor érték esetén a jel zaj viszony -20 dB lehet, ahol a negatív szám azt jelenti, hogy a vevő még akkor is működik, ha a jel már a zajszint alatt van. Ugyanez 10-es spreading factor esetén már csak -15 dB. Az érzékenyebb vevő nagyobb távolságok átvitelére lesz képes. A kódolási érték változtatásánál pedig a hibajavító képesség változik, amely kijavíthatja az interferencia miatt hibásan dekódolt adásokat. A LoRa rádiós megoldásban éppen ez a nagyszerű, hogy viszonylag széles tartományban választható meg, hogy egy rádiós kapcsolat mennyire legyen gyors vagy éppen mennyire legyen nagy távolságú és hibatűrő.

Az eddigi ismeretek alapján az is kiszámítható, hogy milyen sebességgel küld adatot egy LoRa rádiós modul, valamint ezzel együtt mennyi ideig foglalja az adott átvitel a rádiós csatornát. Ahhoz azonban, hogy pontosan meg tudjuk mondani, hogy egy adott méretű információ milyen gyorsan fog megérkezni a vevőhöz, és ezzel együtt mennyi ideig fogja foglalni az adott csatornát, még meg kell ismernünk az átküldendő adatok szerkezetét is.

A LoRa rádiós kapcsolaton az információ csomagokba (*packet*) van szervezve. Minden egyes rádiós csomag egy meghatározott jelsorozattal, úgynevezett preambulummal kezdődik. A preambulum hossza bájtban meghatározható a felhasználó által a 0 – 65535 tartományban. Tipikusan 8 bájt hosszú preambulumot szokás választani. Tudni kell még, hogy a preambulum átvitele során még 4,25 szimbólumnyi idő hozzáadódik az átvitelhez, vagyis hiába állítunk be 0 méretet a preambulumra, az minimum 4.25 jelből fog állni mégis. A preambulum része egyben egy szinkronizációt szolgáló szó érték is (*SyncWord*).

A preambulum után a legtöbb esetben egy csomag fejléc következik. A fejléc három információt hordoz: a csomag kódolási rátája, a csomag hossza, illetve, hogy a csomaghoz tartozik-e egy integritás védő CRC-16-os kód vagy sem. Ez a fejléc bizonyos esetekben elhagyható, amennyiben az említett értékek az átvitelen kívül egyeztetve lettek. Ezek után következik a hasznos teher része a csomagnak, vagyis a ténylegesen átvinni kívánt információ.

Az információ hossza nem tetszőleges, hanem a kódolás függvényében korlátozott. A gyorsnak számító hetes spreading factor esetében összesen 222 bájtnyi információ a megengedett, míg a leglassabb tizenkettes spreading factor érték esetében összesen 51 bájt küldhető. A csomag végén opcionálisan egy két bájtos CRC-16 integritás ellenőrző kód található. A különböző adatrészek azonos spreading factor értékkel vannak elküldve, ugyanakkor a kódolási ráta eltérő lehet a csomagon belül. A preambulum esetén ez nincs értelmezve, itt nem alkalmaznak redundanciát. A fejléc esetében a kódolási ráta mindig 4/8. A hasznos teher és a CRC-16 érték esetében a kódolási ráta a fejlécben beállított érték lesz (5.3. ábra).

5.3 ábra – LoRa keret felépítése

A leírtak alapján már kiszámítható pontosan, hogy egy LoRa rádiós kommunikáció az átküldött bájtok függvényében pontosan mennyi ideig tartozkodik az éterben. Erre a számításra nem csak azért van szükségünk, hogy meghatározzuk a teljes hálózat terhelhetőségét, hanem a később leírtak szerint a pontos időzítések kiszámításhoz is szükségünk lesz rá.

Az alábbi, 5.1. táblázatban pár példán keresztül bemutatjuk, hogy a gyakorlatban mit is jelentenek az egyes paraméter beállítások az átviteli sebességre és a levegőben töltött időre nézve. A táblázatban szereplő számítások mindegyikénél a sávszélességet a tipikusan előforduló 125 kilohertz értékre állítottuk, a preambulum választott mérete 8 bájt, és volt csomagfejléc és integritást védő CRC-16 kód.

	1.	2.	3.	4.	5.
Spreading factor (SF)	7	7	8	10	12
Coding Rate (CR)	4/5	4/5	4/5	4/6	4/8
Átvitt információ mérete	8	128	8	48	8
Számított hasznos sebesség	1773 bps	4756 bps	886.5 bps	468 bps	54 bps
Levegőben töltött idő	36,1 ms	215,3 ms	72,2 ms	821,2 ms	1187,8 ms

5.1 táblázat: Átviteli sebesség és levegőben töltött idő különböző paraméter beállítások esetén

Amint a táblázatból látható, a nagyobb spreading factor értékek esetében akár másodpercekig is eltarthat, amíg az információ a levegőben utazik. A levegőben eltöltött idő azért is fontos érték számunkra, mert ez alatt az idő alatt a csatorna foglalt, így más adók nem használhatják azt. Ezt a problémát könnyíti ugyanakkor, hogy a különböző spreading factorok használata esetén egymásra ortogonális kódolást használnak az átvitel során. Ez praktikusan azt jelenti, hogy hiába zajlik egy időben két különböző LoRa rádiós adás, ha azok különböző spreading factor értékeket használnak, akkor ezeket a vevő egy időben képes érzékelni. A LoRa vevőnek gyártott rádiós modulok valóban képesek az egyidejű adatok fogadására. Ráadásul további könnyítés, hogy a LoRa adás elküldésére nem is egy, hanem általában háromnál több csatorna áll rendelkezésre. Egy átlagos LoRa vevő, legalábbis azok, amelyek a Semtech által gyártott SX1301 rádiós modulra épülnek, egyszerre 8 olyan csatornát figyelnek, amelyen az adatküldést várják. A legtöbb vevőben ez a modul található meg, jelenleg nagyon kevés kivétel van ez alól.

Amikor két LoRa rádió modul egyszerre ad és ugyanazt a spreading factort használja, akkor bizony a két adás interferál egymással. Amennyiben a két adó között van annyi különbség, hogy jeleik vevőhöz érkezésekor az erősebb zajnak érezze a gyengébbik jelet, úgy az erősebb jel még dekódolható marad, de a gyengébb jel már nem fog eljutni a vevőhöz. Tipikusan egy, a vevőtől nagyobb távolságban lévő adó vagy egy olyan adó, amelynek rádiós jele csak sok akadályon keresztül jut a vevőhöz, csak akkor fog sikeres rádiós kapcsolatban adatot küldeni, ha a csatorna, amit használ, éppen szabad. Nagyon fontos tehát, hogy a sok potenciális adó ellenére legyen szabad csatorna, amin az eszközök kommunikálhatnak.

5.2.2. Az ETSI szabályozás

Pontosan azért, hogy elegendő szabad csatorna álljon rendelkezésre a kommunikáció létrehozására, léteznek nemzetközi előírások a még licensz nélkül használható *ISM (Industrial, Scientific and Medical)* sávokban is, amilyenben a LoRa rádiók működnek. Az előírásokat az *ETSI (The European Telecommunications Standards Institute)* határozza meg. Szabályozzák azt, hogy bizonyos csatornákon mekkora teljesítményt lehet használni, illetve azt is, hogy egy eszköz mennyi ideig veheti igénybe a csatornát adás céljára. Ez utóbbit úgy fejezik ki, hogy az idő hány százalékában lehet az eszköz adó állapotban. Az 5.2. táblázatban összefoglaltuk, hogy az ETSI által kiadott ERC REC 70-03 jelű ajánlásban hogyan definiálták azokat a csatornákat, amelyek érintik a LoRa rádiójelek európai átvitelét az általános felhasználás céljából, illetve a csatornák paramétereit.

A h1.7 jelű csatornában két szabályozás is érvényes, amennyiben az adó teljesítmény nem haladja meg az 5 mW értéket, úgy nincs korlátozás a hozzáférésben. A legnagyobb frekvenciaterületen tehát az 1 %-os csatorna használat van érvényben, ami azt jelenti, hogy amennyiben az adó 1 másodpercre adásra használhatja az adott csatornát, úgy a következő 99 másodpercben már nem használhatja azt. A korlátozás nem a teljes ISM tartományra érvényes, hanem külön csatornákon kell érvényesíteni. Így egy h1.4 csatornán történt adás után közvetlenül jöhet még egy h1.2, illetve h1.5, h1.6 és h1.7 csatornán történő adás is.

Csatorna jele	Alsó frekvencia	Felső frekvencia	Erősítés	Csatornahozzáférés
A1 h1.2	865 MHz	868 MHz	25 mW/14 dBm	1 %
A1 h1.4	868 MHz	868.6 MHz	25 mW/14dBm	1 %
A1 h1.5	868.7 MHz	869.2 MHz	25 mW /14 dBm	0.1 %
A1 h1.6	869.4 MHz	869.65 MHz	500 mW/27 dBm	10 %
A1 h1.7	869.7 MHz	870 MHz	5 mW/7 dBm	Nincs korlátozás
A1 h1.7	869.7 MHz	870 MHz	25 mW/14 dBm	1%

5.2 táblázat: LoRa csatornák paraméterei az ETSI szabályozás szerint

Azt, hogy milyen csatornákat használ egy adott LoRa kommunikáció, az is megszabja, hogy az adónak és a vevőnek ugyanazon a csatornán kell adnia, illetve figyelnie. Mint említettük, a gyakorlatban használt LoRa vevők egyszerre csak 8 csatornát képesek figyelni, így nem tetszőleges a csatornaválasztás az adónál.

A különböző LoRa eszköz gyártók és LoRa kapcsolat üzemeltetők által használt frekvenciák előre ismertek. Európában a gyakorlatban az alábbi frekvenciák használatosak: 868,1 MHz, 868,3 MHz, 868,5 MHz, 867,1 MHz, 867,3 MHz, 867,5 MHz, 867,7 MHz és 867,9 MHz. Mindegyik csatorna 125 KHz szélességű. Ezen kívül még két csatorna van speciális használattal. Az egyik, már felsorolt 868,3 MHz csatorna, ahol hetes spreading factor esetén akár dupla olyan széles, 250 KHz szélességű csatornát is használni lehet, amely megduplázza az átvitel sebességét. A másik pedig a 869,5 MHz csatorna, amely egy külön tartományban található, és ráadásul 10% csatornahasználat, valamint emelhető adóteljesítmény jellemzi. Ezt a csatornát azonban az adó és a vevő között a visszairányú kommunikációra használják általában. Így az európai LoRa átvitel során végeredményben egy vagy két frekvenciatartomány használható, alkalmazástól függően és jellemzően 1% a csatornahasználat korlátozása.

A távközlésben, amikor emberek beszélgetnek egymással már megfelelően kidolgozott modellek vannak arra, hogy mikor kezdeményeznek hívásokat és egy hívás ideje meddig tart. Így ma már egyszerűen lehet hálózatokat tervezni és méretezni. A gépek közötti kommunikációban, vezeték nélküli hálózatok esetén azonban még csak terjedőben vannak az alkalmazások, így egyelőre nehéz jó modellt találni.

Az 1% csatornahasználat mellett, ha azt néznénk, hogy minden eszköz maximálisan ki akarja használni a rendelkezésre álló csatornát, akkor összesen 4800 egység alkothatná a hálózatot maximum. Az eredménynél figyelembe vettük, hogy az egyes rádiós adások 6 különböző spreading factor segítségével küldhetik adásaikat, és ezt 8 különböző csatornán tehetik meg, amelyek nem zavarják egymást. Minden egyes spreading factorhoz csatornánként 100 adót képzeltünk, amik teljesen megtelítik az adott csatornát. Ez egy elméleti maximum, ami csak azzal a feltételezéssel helyes, ha elvárjuk, hogy ne legyen ütközés az adások között, azaz minden egyes adás sikeres legyen, illetve feltételezzük, hogy nincs más rádiós technológia az adott frekvenciatartományban, ami zavart okozhatna, illetve az adások megfelelően vannak ütemezve, hogy egymást se zavarják. Feltételezhetjük azonban, hogy a végpontok nem akarnak minden egyes lehetséges pillanatban adatot küldeni, így tulajdonképpen ennél jóval több csomópont is tartózkodhat egy hálózatban.

Egy másik számítási módszernél abból indulunk ki, hogy megnézzük, mekkora adatmennyiséget lehet a hálózatba küldeni adott idő alatt. Ha megvizsgáljuk a maximálisan átvihető adatok mennyiségét, akkor minden egyes spreading factor tartományra külön értéket fogunk kapni. A most leírt példában 8 bájtos adatcsomagokkal fogunk számolni, amely telemetriai és okos város alkalmazásoknál egy jellemző méret lehet adatgyűjtés és vezérlés esetén is. Az adatok küldésénél azonban sajnos ennél nagyobb adatátvitellel kell számolnunk. Mint majd később azt részletezzük, a hálózatba szerveződés miatt a hasznos teher mellett további információkat is át kell vinnünk. Ez majd a mi általunk bemutatott LoRaWAN protokoll esetén eléri a minimum 13 bájtot. Így 21 bájtos adatcsomagokkal fogunk számolni. Az 5.3 táblázat megmutatja, hogy egy csomag átvitele során az adott spreading factor használata mellett mennyi időt tölt el a levegőben az adatcsomag, illetve ezzel a megoldással 1 óra alatt mennyi információt lehet átküldeni.

A táblázat utolsó sorában azt is feltüntettük, hogy mennyire gyakori az, hogy az adott spreading factor értéket választja egy adott állomás. Az eloszlás nem egyenletes. A számításhoz a már ismertetett Hata modell lett felhasználva és nagyvárosi környezetben úgy tekintették, hogy a vevő körül egyenletes eloszlással vannak elhelyezve a csomópontok. A csomópontok pedig azt a spreading factor értéket választják, amely számukra a legalacsonyabb, tehát egyben leggyorsabb átvitelt biztosító érték, amelyen az adó és vevő közötti távolság áthidalható. Amint látható a nagyobb speading factor értékeken több adó található.

	SF7	SF8	SF9	SF10	SF11	SF12
1 csomag átviteli ideje	56.6 ms	102.9 ms	185.3 ms	370.7 ms	659.5 ms	1318.9 ms
1 óra alatt átvihető adatcsomagok száma	63631	34981	19423	9712	5459	2730
1 óra alatt átvihető adatmennyiség	497	273	152	76	43	21
1 adó által küldhető adatcsomagok száma 1 óra alatt	636	349	194	97	54	27
Gyakoriság Hata modell szerint	19 %	8 %	10 %	14 %	20 %	28 %

5.3 táblázat: Átviteli paraméterek különböző spreading factor értékek esetén

Amennyiben nem azt feltételezzük, hogy minden adó maximálisan ki akarja használni a hálózatot, úgy szükségünk lesz annak ismeretére, hogy egy adó adott időközönként, mondjuk óránként, mennyi üzenetet küld. Mint említettük ez alkalmazásonként nagyon eltérő lehet. Egy telemetriai gázmérő állás leolvasó elegendő lehet, ha naponta 1 jelentést küld. Egy villanyóra leolvasó ugyanakkor, ha a szolgáltató pontos képet szeretne kapni a felhasznált villamos áramról és annak ütemezéséről, akkor akár 5 percenként is küldhet információt. Míg egy utcai parkolás megoldásnál nagyjából 10-20 parkolási eseménnyel és egyben üzenettel számolhatunk naponta. A példában most tételezzük fel, hogy az alkalmazás 5 percenként küld egy üzenetet, azaz óránként 12 üzenettel számolunk. Ha minden adást ki akarunk szolgálni és erre a megfelelő ütemezőt is alkalmazni tudjuk, amely a ren-

delkezésre álló 8 európai csatornán ütközésmentesen osztja szét az adásokat, azaz az egyes adások egymással nem interferálnak, akkor összesen, spreading factor szerint növekvő sorrendben 42421, 23321, 12949, 6474, 3639 és 1820 végpontot, azaz összesen 90624 végpontot tudunk kiszolgálni. Azonban, ha feltételezzük az egyenletes eloszlást a vevő körül és ragaszkodunk a teljes mértékű kiszolgáláshoz, akkor a legnagyobb spreading factor értékből és a hozzá tartozó gyakoriságból kell kiindulni, miszerint összesen nagyjából 6500 végpont lehetséges csupán.

Láthatólag igen vegyes eredményeket kapunk a hálózat végpontjainak becsült számosságára, ha különböző feltételezésekből indulunk ki. Mivel ezek a hálózatok jelenleg még csak terjedőben vannak, így nehéz pontosan megfogalmazni a feltételezéseket is. A gyakorlat azonban azt mutatja, hogy a számításokkal ellentétben nem minden esetben ragaszkodunk a százszázalékos adatkézbesítéshez, elfogadható, ha az átküldött adatok egy része akár elveszik, amíg ez az adatveszteség nem jelentős. Ez növeli a végpontok számát az adott hálózatban.

Ugyanakkor azt is meg kell jegyeznünk, hogy a gyakorlatban alkalmazott ütemezés nem alkalmas arra, hogy az adók által küldött adatcsomagok interferenciamentesek legyenek, az adatok ütközni fognak az éterben, aminek hatására bizonyos adatküldések elveszhetnek. Az ütközések gyakoriságát csökkenti, ha csökkentjük a vevők számát. Végeredményben azt kapjuk, hogy az egy vevő által kiszolgálható adók száma tízezres nagyságrendben mérhető, amely megfelelően magas szám. A teljes hálózat természetesen több vevő alkalmazásával ennek többszörösét fogja kiszolgálni.

5.3. A LORA ESZKÖZ FOGYASZTÁSA

A LoRa eszközökre épülő hálózat méretét csak közvetett módon befolyásolja az egyes eszközök fogyasztása. A fogyasztás ugyanis az alkalmazást, alkalmazhatóságot fogja leginkább befolyásolni és így az alkalmazás elterjedése révén érinti a LoRa technológiára épülő hálózat méretét.

5.3.1. Az energiafogyasztás számítása

A fogyasztás kiszámítása során a már korábban ismertetett jellemzőkre fogunk építeni. Az eszközök, amelyek szenzorként és aktuátorként működhetnek, a rádiós kommunikáció mellett ellátják az érzékelés és beavatkozás tényleges feladatát. A korábbi kismonográfiában [Jakab2018] utaltunk rá, hogy a műveletek energia felhasználása sokféle lehet. Egy hőmérséklet kiolvasása például alig jár érzékelhető energia mennyiséggel, ugyanakkor egy olyan szenzor, amely például a mérés előtt hosszasan felfűti a mérő áramköröket, jelentős energiamennyiséget fogyaszt. Nehéz így általánosítani, de a rádiós kommunikáció a többi komponenshez viszonyítva jelentős energiát igényel, így mindenképpen foglalkozni kell ezzel a fogyasztással.

A rádiós modul fogyasztásával kapcsolatban a modul aktuális fogyasztása, valamint használatának rendszeressége és időtartama azok a paraméterek, amelyeket figyelembe kell vennünk. Az aktuális fogyasztás a modul adatlapjáról leolvasható. Több adatlap részletezi az értékeket aszerint, hogy melyik rádiós fázisban van éppen a modul. Megkülönböztetnek például olyan szakaszt, ahol a preambulumot ismeri fel az egység, illetve olyan szakaszt, amelyben már az adott adást veszi. Az egyszerűség kedvéért mi egy, a modulra átlagos használat mellett jellemző értékkel fogunk számolni. A legtöbb Európában használható LoRa rádiós modul implementáció az SX1276

modulra vagy az RN2843 modulra épül. Valójában azonban az RN2843 modulon belül is egy SX1276 chip található, illetve e mellé raktak egy saját mikrokontrollert. Az 5.4. táblázatban bemutatjuk a rádiós rész működése közben, adat fogadásra és adat küldésre jellemző fogyasztásokat a termékek adatlapja szerint, A rádiós adat küldésének mérésekor az Európában jellemző maximális teljesítményre vonatkozó adatot jelenítettük meg.

	SX1276	RN2843
Áramfelvétel rádiós adat fogadásánál	12 mA	14.2 mA
Áramfelvétel rádiós adat küldésénél	29 mA	39 mA
Áramfelvétel alvó állapotban	0.2 uA	1.8 uA

5.4 táblázat: Energiafogyasztás különböző LoRa rádiós modulok esetén

A fogyasztás mérésekor azt is tudnunk kell, hogy mennyi ideig történik maga a fogyasztás. A szenzorok esetében, mint már korábban is megállapítottuk, az a jellemző, hogy egy-egy mérést követően az adott eszköz sokáig alvó állapotban marad. Így elegendő a küldés idejével és gyakoriságával számolnunk.

Az aktuátorok esetében azonban más a helyzet. Amennyiben a beavatkozásnak azonnalinak kell lennie, úgy az eszköznek állandóan figyelnie kell a rádiós csatornát, utasításokat várva. Egy másfajta alkalmazás esetén nem feltétlenül kell azonnali beavatkozás, elegendő, ha az aktuátor időnként megvizsgálja, hogy érkezett-e számára üzenet, és ha igen, akkor teljesíti a kapott parancsot. Ebben az esetben, feltételezve az adó és vevő közötti szinkronizációt, elegendő, ha a rádiós modul csak időnként van bekapcsolva. A szinkronizációt biztosíthatja egy pontos óra, de elképzelhető az is, hogy az elküldött üzenetekhez időzítik az üzenetvétel időpontját. Könnyen megállapítható, hogy bár a 12/14,2 mA áramfelvétel nem igazán jelentős, egy 200 mAh teljesítményű gombelemet 14 óra alatt teljesen lemerít, míg egy 3000 mAh okostelefonhoz vagy a köznapi AA méretű elemekhez mérhető kapacitással is közel 9 nap alatt végez. Éppen ezért olyan alkalmazás, amely azonnali beavatkozásra ad lehetőséget és így megkívánja, hogy a rádió állandóan bekapcsolva legyen, csak állandó tápellátással hajtható.

A szenzorok esetére példaként tekintsünk egy olyan alkalmazást, amely mérési adatokat továbbít, majd a mérések után belehallgat a hálózatba, hogy érkezett-e számára utasítás. A méréseket és az adatküldést végezze 5 percenként el. Az egyszerűség kedvéért feltételezzük, hogy a rádiós csatorna megfigyelése mindössze 10 ms ideig tart. Ez alatt az idő alatt kell eldöntenie, hogy érkezik-e számára parancs vagy sem. Csak a rádiós modul fogyasztását vizsgáljuk, a szenzor többi részének a fogyasztását nem számoljuk.

A fogyasztás kiszámításához most is szükségünk van arra az időmennyiségre, amíg a rádiós modul aktív adatküldés, illetve adatfogadás állapotban. Mint azt korábban leírtuk a LoRa rádiós modulok esetén a küldés hosszának kiszámításánál nem mindegy, hogy milyen spreading factor értéket, valamint kódolási rátát választunk. A mostani számításnál az 5.1 táblázat kapcsán korábban már definiált értékek szerint fogunk számolni, azaz 8 bájtos preambulummal, 8 bájtos adatokkal, de összességében 21 bájtos adatcsomagokkal, illetve 4/5 kódolási rátával, miközben a spreading factor értéke 7 és 12 között változik. Az 5.5. táblázatban foglaljuk össze az eredményeket. Az eredményekből az látható, hogy amennyiben egy erősebb telep biztosítja az eszköz tápellátását, úgy elképzelhető, hogy valóban sok alkalmazás tudja igénybe venni a LoRa rádiós lehetőségeket, mert azok külső, állandó tápellátás nélkül is évekig üzemképesek maradnak.

	SF7	SF8	SF9	SF10	SF11	SF12
1 csomag átviteli ideje	*56.6 ms*	*102.9 ms*	*185.3 ms*	*370.7 ms*	*659.5 ms*	*1318.9 ms*
Áramfelvétel 1000 küldés esetén	*0.61 mAh*	*1.11 mAh*	*2.01 mAh*	*4.02 mAh*	*7.14 mAh*	*14.29 mAh*
Áramfelvétel 1000 vizsgálat esetén	*0.039 mAh*					
Élettartam 200 mAh esetén	*2.9 év*	*1.6 év*	*0.9 év*	*171 nap*	*97 nap*	*48.5 nap*
Élettartam 3000 mAh esetén	*43.7 év*	*24.7 év*	*13.9 év*	*7 év*	*4 év*	*2 év*

5.5 táblázat: Energiafogyasztás különböző spreading factor értékek esetén

5.3.2. LoRa rádiós technológia alkalmassága

Végeredményben elmondható, hogy a LoRa rádiós technológia segítségével valóban nagy, több kilométeres kiterjedésű hálózatot lehetet kiépíteni, méghozzá úgy, hogy egyetlen vevő állomás akár több tízezer végpontot tud kiszolgálni, melyek fogyasztása akár olyan alacsony, hogy szenzorként évekig üzemelhetnek állandó tápellátás hiányában is.

5.4. A LORAWAN HÁLÓZATI TECHNOLÓGIA

A LoRa rádiós technológia tehát alkalmas arra, hogy segítségével nagy kiterjedésű, sok felhasználós alacsony fogyasztású hálózatot építsünk. A LoRa szövetség által készített hálózatba szerveződés a LoRaWAN hálózati technológia [LoRaAlliance2017]. Szinte kivétel nélkül, amikor LoRa technológiára épülő nagy kiterjedésű hálózatról beszélünk, a való életben ezt a hálózatot értjük. Mindazonáltal megjegyezzük, hogy nem ez az egyetlen technológia, amely segítségével a LoRa rádiós modulok hálózatba szervezhetőek. Egy ígéretes próbálkozás a *RadioShuttle* hálózat [RadioShuttle2017], amely számos előnyt nyújt a LoRaWAN technológiával szemben, kihasználva azt, hogy a hálózatban több olyan elem is lehet, amely képes az internetre csatlakozni. Az ilyen jellegű megoldások egyelőre nem terjedtek el és a jelenlegi LoRaWAN lefedettség mellett kétséges, hogy annyi előnyt tudnának nyújtani, amely mellett megérné a mostani LoRaWAN technológiát leváltani.

5.4.1. LoRaWAN építő elemek

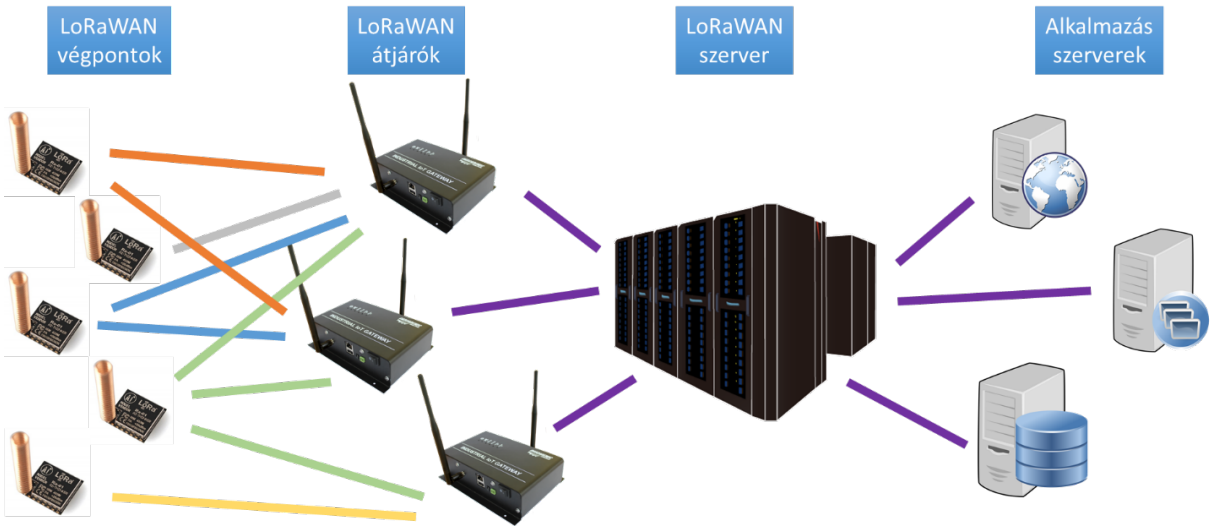
A LoRaWAN hálózatok csillag alapú hálózatok szuperpozíciójaként képzelhetőek el. A LoRa rádióval rendelkező egyszerűbb felépítésű végpontok elszórva helyezkednek el a hálózat területén. A tőlük érkező és az oda továbbítandó forgalmakat a *LoRa átjárók (gateway)* kezelik. Mint azt korábban írtuk, az átjáró rádiója eltér a végpontoknál

alkalmazott berendezésektől. Legfőbb képessége, hogy egyszerre, egy időben több csatornát is képes kezelni. Bár a rádiós modul többszöröződésével megoldható lenne tetszőleges számú csatorna kezelése, az európai gyakorlatban összesen 8 csatornát jelöltek ki, így ezek az átjárók általában ennek a 8 csatornának ez elérését biztosítják. Természetesen, ahogy arról szó volt, minden egyes csatornán 6 egymásra ortogonális kódkészlet érkezhet, azaz mindegyik csatorna képes dekódolni az Európában használatos 6 féle spreading factort.

Az átjárók a legtöbb esetben már nem rádiós kapcsolaton keresztül, hanem vezetékes hálózatban továbbítják és fogadják a végpontok forgalmát az internet felé. Az átjárókat hívják még ezen kívül *packet forwarder* elemeknek is, ugyanis fő feladatuk a csomag továbbítása.

Az internetes szakaszon túl található a *LoRa szerver* (5.4. ábra). Feladata, hogy koordinálja az átjárók munkáját, illetve összekösse a végpontokat és az alkalmazásokat vagy szolgáltatásokat. A végpontok koordinációja leginkább akkor fontos, amikor a végpont felé közvetítenek üzenetet, ugyanis egyáltalán nem mindegy, hogy melyik átjáró lesz kiválasztva az üzenet továbbítására a vezeték nélküli szakaszon. Nem biztos, hogy mindig pont azt az átjárót kell választani, amely a legközelebb található a végponthoz, ettől eltérő stratégiák is létezhetnek. A LoRa szerver az átjárók irányításán túl a kapcsolatot tartja az alkalmazás szerverekkel.

Eltérően a hagyományos IP hálózatoktól és például az ugyancsak LPWAN technológiára épülő NB-IoT hálózattól, a LoRaWAN hálózatban nincs IP cím szerinti címzés. Az eszközök és az adatokkal foglalkozó, adatgyűjtő és vezérlő alkalmazás egymással összeköttetésben van. Az eszközök csak az alkalmazás számára tudnak adatokat küldeni, illetve csak innen kaphatnak adatokat. Tehát a végpontok közötti közvetlen kommunikáció is ki van zárva. Természetesen elképzelhető olyan alkalmazás is, amely aztán a beérkezett adatokat már az adatok között szereplő IP címre továbbítja, mintegy alkalmazás szintű átjáró.



5.4 ábra: A LoRaWAN hálózat fő elemei *(saját ábra)*

5.4.2. A LoRaWAN biztonsága

Az IoT kapcsán ma sokszor emlegetik a biztonságot, pontosabban annak hiányát, amelynek oka részben az is, hogy ezek az eszközök általában szerényebb képességekkel és így egyben gyengébb védekező képességekkel rendelkeznek, mint általában a számítógépes hálózat más elemei. Sőt, nem túl vad becslések szerint az IoT eszközök száma jóval meghaladja majd a klasszikus hálózati eszközök, így PC-k, mobiltelefonok számát, és ezáltal még nagyobb potenciális veszélyt jelentenek.

A LoRaWAN hálózat önmagában egy zárt hálózatot alkot, amely az interneten keresztül kommunikál, de a hálózatba belépési pontként az internet felől csak az alkalmazások szolgálnak. Ennek következtében, hiába van sokmillió gyengébb képességű végpont a LoRaWAN hálózatban, egy megfelelően védett alkalmazás szerver védi az összes végpontot is. A másik potenciálisan nagy biztonsági problémát az adatszivárgás vagy adatlopás jelenti. Végső soron a végpontok forgalmát az interneten keresztül továbbítják, általában a LoRaWAN átjáró és a LoRaWAN szerver között a publikus interneten. Ez azonban a megfelelő titkosítás alkalmazásával nem jelent veszélyt, a megfelelően titkosított adatokhoz ugyanis nem férhet hozzá harmadik fél, és azokat meg sem képes változtatni úgy, hogy azt a kommunikáló felek ne vegyék észre. A hálózatba és az alkalmazáshoz, ugyancsak a titkosítás miatt, nem kerülhetnek be olyan elemek, amelyeket előzetesen nem hitelesítettek.

Ugyanakkor azt meg kell jegyezni, hogy a szolgáltatás megbénítás jellegű támadások továbbra is sikeresek lehetnek. A rádiós technológiák esetében ugyanis viszonylag egyszerűen véghezvihető az adott frekvenciatartomány megbénítása (*jamming*). A LoRa nem könnyű célpont, ugyanis, ahogyan azt be is mutattuk, igazán zajos átvitel esetén is képes lehet fenntartani a kommunikációt, de a megfelelő teljesítménnyel elnyomhatóak a küldési frekvenciák. Ez a fenyegetés adott LoRaWAN végpontokat érint, a hálózat többi részére és más hálózatokra nincs hatással.

5.4.3. LoRaWAN végpont

A LoRaWAN végpont (lásd 5.5 ábra) a LoRa rádiós technológiát használja kommunikációra. A kommunikáció paramétereit a hálózaton keresztül is lehet állítani. A végpontokat a LoRaWAN hálózat képes azonosítani és forgalmukat a megfelelő alkalmazáshoz továbbítani, illetve az onnan jövő forgalmat a végpontra küldeni. Mindezt a hálózat biztonságosan teszi meg, harmadik fél nem férhet az adatokhoz, illetve a végpontok nem megszemélyesíthetőek.

5.5 ábra: Wemos + LoRa Shield *(saját felvétel)*

A végpontok egy LoRaWAN hálózatban rendelkeznek saját azonosítóval (Device Address). Ez az azonosító mindössze 4 bájt méretű, ami azt is jelenti egyben, hogy 232, azaz körülbelül 4.3 milliárd különböző azonosítójú eszköz létezhet. Ennél a számnál azért nagyobb számú eszközre számít az IoT világ, hiszen már a 4 bájtos IP-cím sem volt elegendő. A problémát az oldja meg, hogy ennek a címnek csak a LoRaWAN hálózaton belül kell egyedinek lennie, egy másik LoRaWAN hálózatban használható ugyanez a cím. Az eszközök nem kommunikálnak közvetlenül más internetes eszközökkel, így a LoRaWAN hálózati kommunikációt csak a hálózaton belül kell azonosíthatóvá tenni. Sőt, valójában az eszköz címének nem is mind a 32 bitje használható fel azonosítási célra, ugyanis az cím első 7 bitje magát a hálózatot azonosítja. Ez mindenhol így van LoRaWAN hálózatokban. Az értelme az, hogy így az átjáró el tudja dönteni, hogy az éppen érkezett csomag a saját hálózatába tartozik vagy sem. Amennyiben nem a saját hálózatába tartozik, még mindig dönthet úgy, hogy továbbítja azt (roaming), ekkor a csomagban jelzett szolgáltató felé kell továbbítania. Vannak speciális címterületek is. Ilyen a 0000000 és 0000001 kezdetű címek (0x0 és 0x1 hexadecimális leírásban), amelyeket kísérleti címnek osztottak ki, ilyen címeket bárki használhat lokális hálózaton.

A LoRaWAN végpontok rendelkeznek egyedi címmel is. Ez egy 8 bájtos cím, amely valóban egyedi az összes hálózati elemet figyelembe véve. A cím egy EUI64 cím (*Extended Unique Identifier*), tartalmaz gyártó specifikus biteket és egyedi biteket. A hálózati forgalomban ugyanakkor a LoRaWAN eszköz ezt a címét nem használja, helyette ott a már bemutatott 32 bit méretű azonosító található.

A címeken kívül a LoRaWAN eszköz rendelkezik még kulcsokkal is, hiszen a forgalom a LoRaWAN hálózatban titkosítottan halad. A forgalom titkosításához két különböző kulcsot használ az eszköz. Mindkét kulcs 128 bit hosszú, amely megfelel a mai gyakorlatnak, miszerint nem tekintjük biztonságosnak azokat az algoritmusokat, amelyek 100 bit méret alatti kulcsokat használnak. A titkosításhoz és egyéb kriptográfiai műveletekhez pedig *AES (Advanced Encryption Standard)* algoritmust használja a LoRaWAN protokoll. Az AES titkosítás egy mai napig korszerű titkosító, amely kedvező paraméterekkel rendelkezik a gyorsaság és memóriahasználat területén és mindezidáig feltörhetetlennek bizonyult. Több más hálózati kommunikációs protokoll is nem véletlenül választja ugyanezt a titkosító algoritmust.

A két titkosító kulcs az *alkalmazás viszony kulcs (Application Session Key – AppSKey)*, illetve a *hálózati viszony kulcs (Network Session Key – NwkSKey)*.

Az *AppSKey kulcsot* kizárólag a végberendezés és az alkalmazás szerver ismerheti, amely a felhasználó területén fut. Sok esetben azonban, amikor a felhasználó nem futtat ilyen szerver funkciót, a LoRaWAN szerver átveheti ezt a feladatot és a felhasználó számára biztosíthatja a kulcs kezelését is. Ekkor a felhasználónak muszáj megbíznia a LoRaWAN hálózat üzemeltetőjében, hiszen ebben az esetben az üzemeltető manipulálni tudja az adatforgalmat. A kulcs célja, hogy biztosítsa az alkalmazás adatok titkosítását, nehogy harmadik fél felfedhesse azt. Ennek érdekében az átküldött adatok az AppSKey kulcs segítségével titkosítva vannak. Annak érdekében, hogy az adatokat ne lehessen megváltoztatni, az átküldött információ, valamint a hozzá tartozó fejlécek hitelesítve vannak. A hitelesítéshez a végpont, illetve a LoRaWAN szerver az *NwkSKey-t használja*. Ezt a kulcsot ismernie kell a LoRaWAN hálózati szervernek, különben nem tudja elvégezni a hitelesítést. A hitelesítés során a végpont egy 4 bájtos ellenőrző összeget készít szintén az AES algoritmus segítségével a NwkSKey kulcs felhasználásával. Ezt az ellenőrző összeget vizsgálja meg a LoRaWAN szerver és amennyiben korrektnek találja, úgy a csomagot továbbítja az alkalmazáshoz. A LoRaWAN átjáró, amely továbbítja a csomagot, nem ismeri ezt a kulcsot, így az minden csomagot továbbít, amely a hálózatába tartozik, még azokat is, amelyek később megbuknak a hitelesség ellenőrzésen.

Probléma lehet még az is, hogy a támadó visszajátszhat egy egyszer már elküldött, hitelesnek bizonyult csomagot a hálózatban. A nyílt rádiós hálózat lehetővé teszi ugyanis, hogy a támadó lehallgassa a kommunikációt, majd később változtatás nélkül visszajátssza azt. Annak érdekében, hogy ezt a támadást sikeresen megakadályozza a LoRaWAN hálózat, szükség van arra, hogy azonosítani lehessen azt a szituációt, amikor egy adatcsomag ismételten érkezik meg a LoRaWAN szerverhez. Ennek érdekében az adatcsomag fejlécében számlálók vannak elhelyezve, melyek külön-külön számolják a szerver által végpont felé, illetve a végponttól a szerver felé küldött adatcsomagokat. Amennyiben a számlálók értéke elmarad egy már beérkezett adatcsomag számlálójától, úgy azt a szerver, illetve végpont érvényteleníti és nem adja át az alkalmazás számára. Ebből viszont az is következik, hogy a megfelelő használat érdekében az AppSKey és NwkSkey kulcsokat minden egyes új viszonyban le kell cserélni. A későbbiekben ismertetjük a hálózathoz csatlakozási mechanizmust is, ahol látni fogjuk, hogy ez valóban meg is történik.

A NwkSKey esetében van még egy speciális felhasználási terület. Létezik ugyanis olyan forgalom is, amelyet nem az alkalmazás, hanem maga a LoRaWAN hálózati szerver küld a végpont felé. Ezek az üzenetek is természetesen titkosítottak és védettek, azonban ezt nem védheti az alkalmazás viszony kulcsa, hiszen azt a szerver nem ismeri. Így ezek az üzenetek az NwkSKey segítségével vannak titkosítva. Az üzenetek tartalma olyan végrehajtandó parancs, amely a rádiós kommunikációra vonatkozik, tehát nem kerül az alkalmazáshoz.

5.4.4. Végpont csatlakozása LoRaWAN hálózathoz

A LoRaWAN hálózati kommunikációhoz kulcsokra van szükség, ezeket a kulcsokat a végpontnak és a hálózati szervernek, illetve az alkalmazás szervernek ismernie kell. A hálózathoz csatlakozás egy olyan művelet, amely során ezek a kulcsok kioszthatóak a megfelelő hálózati elemek között. Fontos, hogy mivel az eszközökön nem feltétlenül tudjuk biztonságosan tárolni a titkosítás és hitelesítés során használt kulcsokat, ezért minden egyes eszköznek egyedi kulcsa van a kommunikációhoz.

A LoRaWAN hálózati szervernek annyi hálózati viszony kulcsot kell tárolnia, ahány aktív végpont van a hálózatában, míg az alkalmazás szervernek annyit alkalmazás viszony kulcsot, ahány végpont felhasználója van. Ezek adott esetben igen nagy számok tudnak lenni, tekintve az IoT alkalmazásokat. Ellenben így elérhető az, hogy amennyiben támadók egy eszközt kompromittálnak és megszerzik az aktuális viszony kulcsokat, úgy továbbra sem férnek hozzá a többi eszköz forgalmához, illetve nem tudnak új eszközöket a hálózatba regisztrálni.

A LoRaWAN protokoll kétféle csatlakozási mechanizmust támogat. Az egyik egy egyszerűbb csatlakozás és olyan esetekre ajánlják, amikor a másik, biztonságosabb csatlakozás valamiért nem érhető el, vagy azt a tesztelési fázisban nem érdemes még használni. Ez az *ABP (Activation By Personalization)* módszer. Ebben az esetben az alkalmazás viszony kulcs és a hálózati viszony kulcs el van tárolva a végpontban és a LoRaWAN szerverben, azok értéke viszonyról viszonyra nem változik meg. Ez egy egyszerű megoldás és mivel nem igényel kommunikációt a végpont és a szerver között, így nem kell egyetlen elküldött csatlakozó üzenet sem, hogy a végpont és a LoRaWAN szerver kommunikálni tudjon.

A megoldásnak azonban több hátránya is van. Egyrészt az összes eszközön külön be kell állítani ezeket a kulcsokat egyedire, amely okot adhat arra, hogy hibázzunk a kulcsok beállítása során vagy arra, hogy a kulcsokat biztonságos időn túl használjuk. A nagyobb probléma azonban az, hogy a statikusan beállított hálózati kulcs

esetében nem működik megfelelően a megismételt csomagokra épülő támadások kiszűrése. A LoRaWAN szerver ugyanis úgy ellenőrzi, hogy egy adott adatcsomag már szerepelt a hálózati forgalomban, hogy számlálók segítségével számolja a fel és le irányban közlekedő adatcsomagokat. Amennyiben a kulcs nem változik, úgy vagy a végpontnak még az esetleges üzemzavarok és kikapcsolások esetén is el kell tárolnia ezeket a számlálókat, vagy muszáj a LoRaWAN szerver oldalon a számlálókat minden egyes újraindításnál nullázni. Ez utóbbi művelet nem csak azért problémás, mert nehezebben elvégezhető, hanem azért is, mert az ismételt csomagok lehetősége biztonsági probléma. Végső soron így ezt a csatlakozási módszert csak tesztelésre érdemes használni, illetve olyan hálózatokban, ahol az adatbiztonság nem számít.

A másik csatlakozási módszer az *OTAA (Over The Air Activation)*. Ahogy a nevéből azt sejteni lehet, itt már lesz kommunikáció a végpont és a LoRaWAN szerver között a csatlakozás során. A módszer segítségével a megfelelő kulcsok települnek a megfelelő eszközökön. A végpont megkapja a frissen előállított alkalmazási viszony kulcsot (AppSKey) és hálózati viszony kulcsot (NwkSKey), illetve egy *eszköz címet (Device Address)* is.

Ugyanez az alkalmazási viszony kulcs előáll az alkalmazás szervernél is, illetve a LoRaWAN szerver megszerzi a hálózati viszony kulcsot, valamint ez az eszköz az, amely kiosztja az eszköz biztosan egyedi, hálózaton belüli címét. Mindehhez szükséges azonban, hogy a készülék és az alkalmazás hitelesíteni tudja egymást. Erre a célra van az eszköz 64 bites egyedi azonosítója, a korábban ismertetett EUI64 cím. Hasonlóan az eszköz címéhez, maga az alkalmazás is rendelkezik egyedi címmel, ami szintén egy 8 bájtos cím, ez az AppEUI. Ezt a címet ismernie kell a végpontnak is. A két azonosító nem titkosított, bár nincsenek hirdetve, de lehallgatással bárki megismerheti azokat.

A hitelesítéshez még szükséges egy titkos kulcs, ez pedig egy 128 bites kulcs lesz, az AppKey. A kulcs és a két azonosító használatával, az AES kriptográfia algoritmus segítségével a végpont és az alkalmazás szerver elvégzi a kulcsok kialakítását és kiosztását. A mai szolgáltatásoknál azonban egyelőre egy helyen van az alkalmazás szerver és a LoRaWAN szerver, így az NwkSkey és az AppSkey egy helyen keletkezik és emiatt nem jelent problémát azok elosztása a szerver oldalon.

A hálózati kommunikáció mindössze egy fel- és egy leirányú üzenetből áll a csatlakozás során. Az első üzenetet a végpont küldi a LoRaWAN hálózatba. Jelzi, hogy csatlakozni szeretne és elküldi a saját, illetve az alkalmazás azonosítóját, valamint egy 16 bites frissen előállított álvéletlen értéket (*DevNonce*). Az adatok egyelőre még nem lehetnek titkosítottak, különben a LoRaWAN szervernek nagyon nehéz dolga lenne, hogy megtalálja azt az alkalmazást, ami ezt kezeli. Titkosítás ugyan nincs, de a csatlakozás kérés hitelessége ellenőrizhető, ugyanis a végpont azt, hálózati viszony kulcs hiányában az AppKey segítségével előállítja.

Az átküldött paraméterek segítségével a megfelelő alkalmazás szerverhez kerül a kérés. Az alkalmazás szerver is előállít egy 24 bites véletlen értéket (*AppNonce*), illetve felhasználja a hálózat 24 bites bájtos azonosítóját is (*NetID*). A 24 bites azonosító első 7 bitje a hálózati prefix, a többi bit tetszőleges. Az azonosítók, illetve véletlen értékek segítségével a szerver legyártja a kulcsokat. A csatlakozási kérésre a szerver választ küld. A válaszban benne van a szerver által gyártott véletlen érték és a hálózati azonosító. Előbbi paramétert a végpont még biztosan nem ismeri. Ezen paraméterek segítségével a végpont már maga is legyárthatja a hálózati viszony kulcsot és az alkalmazás viszony kulcsot éppen úgy, ahogy azt a szerver oldal teszi. A helyes paramétereket felhasználva mindkét fél ugyanazokra a kulcsokra fog jutni.

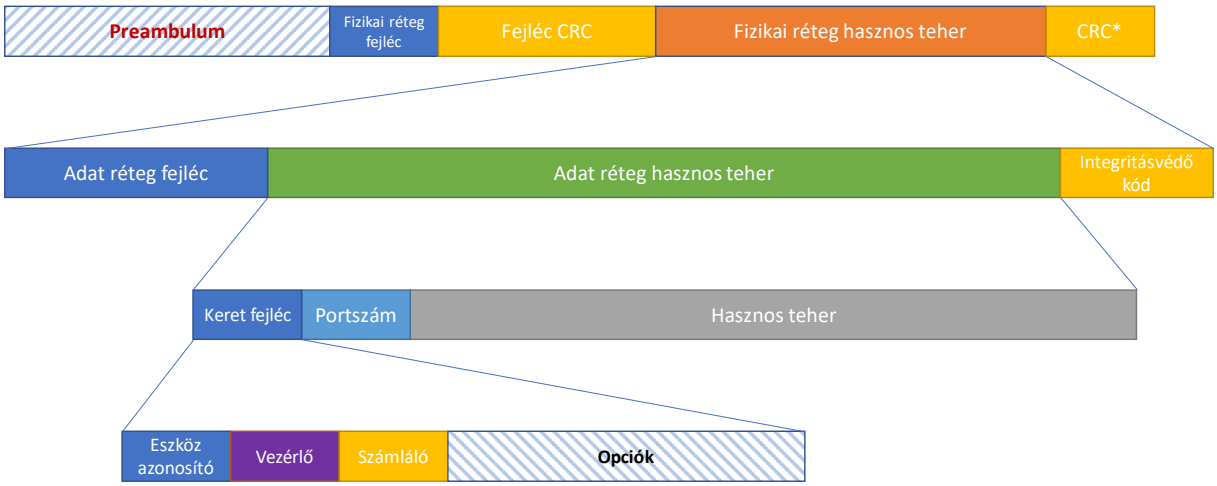
A szerver válasza már titkosított, a titkosításhoz az AppKey kulcsot használja. A csatlakozási kérésre küldött válaszban még több fontos információ is megtalálható, amely szükséges a kommunikációhoz. Van információ

a válasz üzenetek időzítéséről és sebességéről, illetve van információ a használható csatornák listájáról is. Ez utóbbi maximálisan 5 elemet tartalmaz. Európában a 868,1 MHz, 868,3 MHz és 868,5 MHz csatornák számítanak az alapértelmezett csatornáknak. Ezt egészítheti ki a LoRaWAN szerver még a saját hálózatán használt csatornákkal. A válasz üzenet időzítése és sebessége a LoRaWAN hálózatból a végpont felé küldött üzenetekre vonatkozik. A válasz üzenetek mindig ugyanolyan vagy lassabb sebességgel haladnak, mint az elküldött üzenetek.

Amint látható, az OTAA csatlakozási mód egyfelől biztonságosabb csatlakozást tesz lehetővé, mint az ABP módszer, másfelől a hálózati szerver a csatlakozás során megoszthat fontos paramétereket a végponttal.

5.4.5. *Üzenetek felépítése a LoRaWAN hálózatban*

A LoRa rádió esetében már ismertettük az a szerkezetet, amelyben az adat, mellette a preambulum, fejléc és integritásvédő kód a rádiós adásba kerül. A LoRaWAN hálózaton közlekedő csomagok kezeléséhez további információkra van szükség, így az adathoz további fejlécek és funkcionális kódok kapcsolódnak. Egy LoRaWAN hálózatban közlekedő adatcsomag szerkezetét az 5.6. ábra mutatja.



5.6 ábra: A LoRaWAN adatcsomag felépítése *(saját ábra)*

Az ábra legalján található a már megismert LoRa formátumú adatcsomag. Az adat részben (*PHYPayload*) a LoRaWAN hálózatban egy adatkapcsolat szintű keret kerül. Ez a keret tartalmaz külön fejlécet (*MHDR*), benne van az adatkapcsolat szintű tartalom (*MACPayload*) és tartalmaz egy kriptográfiai integritásvédő kódot (*MIC*). A fejléc mindössze egyetlen bájt, amely a csomag típusát adja meg, illetve egy verziószámot a csomag felépítésére nézve.

A típust tekintve 7 féle csomag létezik. Van csatlakozási kérelem és az erre érkező válasz. Van megerősítés nélküli (lásd 5.10-es ábra) és megerősítést megkövetelő adat és ezek léteznek fel és lefelé irányban is, illetve van újracsatlakozási kérelem. Mivel ezeket a típusokat 3 biten kódolja a LoRaWAN szerkezete, így van még lehetőség egy típusra, amelyet saját célra lehet felhasználni. A fejléc egyelőre még 3 jövőbeli célokra fenntartott bitet tartalmaz, majd marad még 2 bit a verziószám jelzésére. Egyelőre az első verziónál jár a protokoll. A kriptográfiai integritásvédő kódról már esett szó. Ez biztosítja, hogy amennyiben a csomagban változás történik a kézbesítés közben, úgy

az a vevő tudomására jusson. Az integritás védelem kiterjed a teljes csomag tartalmára. Előállításához a NwkSKey kulcsot használj a végpont, illetve a LoRaWAN szerver.

Az adatkapcsolat szintű tartalomban egy magasabb szintű keret van, amely a hálózati protokollok szerveződése alapján a hálózati szintnek felel meg. Itt már megtaláljuk a hasznos tartalmat (*FRMPayload*), de még itt is szükség van további fejlécekre (*FHDR*), illetve egy opcionális port jelölőre (*FPort*). A fejlécben található a küldő hálózati eszköz 4 bájos címe (*DevAddr*), amely szükséges a végpont azonosítására és a megfelelő titkosító kulcsok kiválasztására. Van itt egy 1 bájtos vezérlő kód (*FCtrl*), amely vezérlő biteket tartalmaz. Két vezérlő bit *ADR (Adaptive Data Rate)* és *ADRACKReq* foglalkozik az adatküldések optimalizálásával.

A LoRaWAN szerver folyamatosan monitorozza a végpontok kapcsolódási paramétereit és amennyiben lehetőség van gyorsabb kapcsolatra, úgy utasíthatja a végpontokat, hogy változtassanak paramétereiken. A parancsot azonban csak akkor küldi ki, ha arra a végpont valóban igényt tart és azt jelzi is az ADR bit beállításával. Mozgó végpontok esetén nem érdemes a hálózattól várni az optimalizálást, hiszen a változó vételi paraméterek mellett képtelen lesz a kapcsolatot optimalizálni. Éppen ezért ilyen esetekben a végpont nem kéri az optimalizálást.

Egy másik bit az optimalizálás kérésének *nyugtázására* vonatkozik. A végpont ugyanis nem mindig tudhatja biztosan, hogy az általa küldött adatcsomag valóban eljutott a LoRaWAN hálózatba. Ha ugyanis nem érkezik semmilyen visszajelzés a kapcsolat paramétereinek változtatására, akkor is gondolhatja a végpont, hogy minden rendben van, azért nem küld parancsokat a LoRaWAN szerver. Azért, hogy megbizonyodhasson arról, hogy valóban minden rendben van, az ADRACKReq bit beállításával kéri, hogy nyugtázzák az ADR kérését. Ezt a LoRaWAN szerver meg is fogja tenni válaszában.

A nyugtázás kérése nem szükséges minden egyes felfelé menő csomag esetében, a gyakorlatban sok, tipikusan 64 darab felfelé menő adatcsomag után kérik csak a nyugtázást. A vezérlő bitekhez tartozik még a nyugtázás (ACK) bit is. Mind a végpont, mind a LoRaWAN átjáró kérheti, hogy a rádiós átvitel sikerességéről nyugta érkezzen. A kérelem az adatcsomag típusával állítható be, a nyugtázás célját pedig az éppen tárgyalt bit szolgálja. Ha a végpont kéri a nyugtázást, akkor a LoRaWAN szerver közvetlen válaszában megküldi a nyugtát. Amennyiben viszont az átjárótól érkezik nyugtázási kérelem a végpont felé, úgy a végpont is a következő alkalommal köteles nyugtát küldeni, de erről a pillanatról tetszőlegesen dönthet. Általában az egyszerűbb végpontok ilyenkor azonnal nyugtát küldenek, míg a több számítási kapacitással rendelkező végpontok inkább kivárják vele a következő adatküldést és ott helyezik el egyben a nyugtát is.

Fontos, hogy nyugtázás minden kérésre csak egyszer történik meg, nincs ismétlés, ha a nyugtát szállító csomag esetleg elveszne. Amennyiben a küldő fél nyugtát kér, és az nem érkezik meg, akkor az elküldött adatcsomag újra elküldésre kerül egy meghatározott számú ismétlődés erejéig. Azonban, ha a többszöri adatküldési kísérletre sem érkezik meg a nyugtázás, akkor amennyiben a végpont próbálkozik, úgy dönthet úgy, hogy egy biztosabb kapcsolati paraméter beállítás mellett kísérletezik újra vagy feladja a próbálkozásokat. Amennyiben a LoRaWAN hálózat várja a nyugtát a végponttól és az nem érkezik meg többszöri próbálkozásra sem, úgy a LoRaWAN hálózat azt fogja gondolni, hogy a végpont már nem elérhető és nem fog ismét próbálkozni. Az ismételt adatküldések esetén érdemes lassabb, de egyben hosszabbtávú és biztonságosabb adatátvitelt választani.

A fejléc egy újabb bitje (*FPending*), amely csak a végpont felé irányuló forgalomban található meg, tájékoztatja a végpontot, hogy további adatok várnak rá a LoRaWAN hálózatban. A végpontnak ilyenkor egy újabb, akár egy hasznos adatot nem tartalmazó, de a hálózatba irányuló csomagot kell küldenie, hogy arra válaszként ismét jöhessen adat a hálózat felől.

A fejléc itteni részében még található 3 bit, amely az alább ismertetett opciók rész (*FOpts*) hosszát adja meg. A fejléc következő 2 bájtja számlálót tartalmaz. Mint arról már írtunk az adatcsomag visszajátszásos támadások kivédésének érdekében a végpont és a hálózat egy számlálóval számolja, hogy mennyi adatcsomag lett küldve külön-külön fel és le irányban. Amennyiben egy olyan szám érkezik, amely már egyszer kijátszott adatcsomagra utal, úgy a végpont vagy a LoRaWAN hálózat ezt a csomagot eldobja.

A számláló mindkét irányban 32 bites, azonban a fejlécben mindössze 16 bit áll a rendelkezésre, hogy az értékét jelezzék. Ez praktikusan nem jelent gondot, amikor a 16 bites számláló átfordul, a végpont vagy a hálózat frissíti az át nem vitt bitek értékét. A számlálók közül mindig csak az egyik utazik a csomagban természetesen, a végpont által küldött adatcsomagokban az a számláló van, amely a felfelé irányuló csomagokat számolja, míg a hálózattól érkező adatcsomagokban a lefelé irányuló csomagok számlálója található.

A 32 bites számláló még a titkosítás során is szerepet kap, így biztosítják, hogy minden egyes adatcsomag egyedi kulcsfolyammal legyen titkosítva. A LoRaWAN 1.1 szabvány ezen a területen újítást vezetett be. Mivel az alkalmazás szerver és a hálózati szerver funkciója jobban elkülönül, így szükséges, hogy a hálózat által, alkalmazást nem érintő csomagoknak külön számlálója legyen a végponthoz küldött csomagok esetén. Azt, hogy melyik számláló van éppen a csomagban, a következőkben ismertetett FPort érték mondja meg. Az OTAA csatlakozás után a számlálók értéke mindig nullázódik, azonban az ABP csatlakozás után ezeket az értékeket meg kell őrizni.

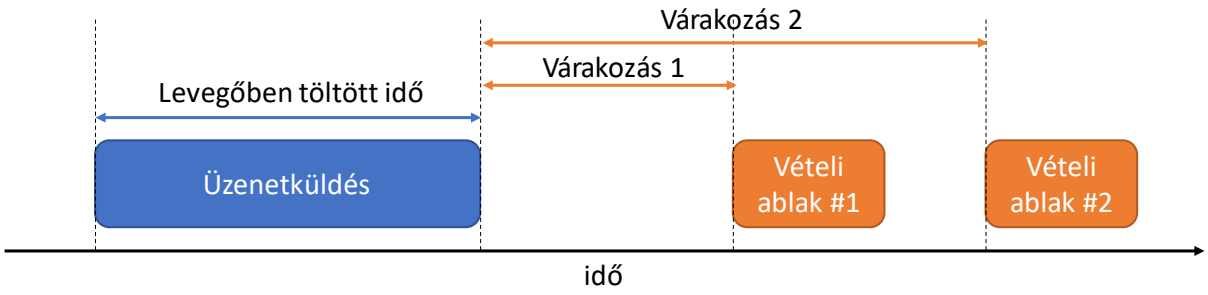
Végül a fejlécben egy opciókat tartalmazó *opcionális rész (FOpts)* található. Ez egy maximum 15 bájt hosszú rész, és arra a célra szolgál, hogy az adatkapcsolati paramétereket és az eszközök állapotát érintő parancsokat a LoRaWAN hálózat el tudja küldeni, illetve azt a végpont nyugtázhassa. A parancsok a végponton, illetve az alkalmazás szerveren futó alkalmazáshoz nem is jutnak el, a hálózati rész dolgozza fel azokat. Az adatcsomag fejléce ugyan védve van változtatások ellen, de nem titkosított, így az itt közölt parancsokat és információkat bárki lehallgathatja. A LoRaWAN protokoll éppen ezért majd egy másik, immár titkosított helyen is lehetőséget ad ugyanezeknek a parancsoknak az átvitelére. Egy adatcsomagon belül azonban mindkét helyen nem lehet parancs. A jelenleg definiált lehetséges parancsok a következőek: kapcsolat ellenőrzése, kapcsolat paramétereinek állítása, csatornahozzáférési idő állítása, vételi paraméterek és időzítés állítása, eszköz státuszának lekérdezése, csatorna paramétereinek módosítása. Van lehetőség hálózaton belül egyedi parancsok kiadására is.

Az adatcsomag fejléce és a tényleges adatmező között található az 1 bájt hosszú port megjelölés (*FPort*). Ez az alkalmazáson belül jelölheti meg az adatot. Az alkalmazás 1 és 223 közötti értéket választhat tetszőlegesen, a többi értéknek speciális jelentése van. A 0 port érték jelzi, hogy az adatmezőben nem alkalmazás adat található, hanem a már korábban ismertetett adatkapcsolatra vonatkozó parancsok vannak elhelyezve. A jelentősége az, hogy az itt küldött parancsok titkosítva vannak a NwkSKey kulcs segítségével. A 224 port érték tesztelő csomagot jelöl. A 225 vagy efölötti port értékek pedig egyelőre tartalékolva vannak későbbi felhasználás céljából. Végül az adatmező következik, amely az átvitel során minden esetben titkosítva van, bár a titkosító kulcs változhat.

5.4.6. LoRaWAN adatcsomag ütemezése

A végpontok adatcsomagjaikat tetszőleges időzítéssel küldhetik el, valójában teljesen függetlenek a többi végponttól. A LoRaWAN hálózat azonban már nem küldhet akármikor adatcsomagot a végpont felé, ugyanis a legtöbb IoT végpont az ideje nagy részében alszik, így félő, hogy az elküldött csomag nem érkezne meg.

A LoRaWAN végpontok között ezért 3 különböző típust különböztetünk meg. Az egyik az úgynevezett *Class A* végpont, amely kétirányú kommunikációra is képes, azonban az eszköz felé irányuló kommunikációt pontosan ütemezni kell annak érdekében, hogy azt a végpont megkapja (5.7. ábra). A végpont minden egyes elküldött üzenet után két vételi ablakot nyit meghatározott időpontokban, ahol várja, hogy a hálózat számára üzenetet küldjön. A vételi ablakokat ±20 mikroszekundum pontossággal kell nyitnia és legalább olyan hosszan nyitva kell tartania, hogy azon a rádiócsatorna hallgatásával legalább egy preambulumot meghallhasson, ha éppen adatot küldenek. Amennyiben nem észlel semmilyen számára küldött adást az első időablakban, akkor az első időablak zárása után majd egy második időablakot is meg kell nyitnia ugyanilyen céllal.



5.7 ábra: Vételi időablakok Class A üzemmód esetén *(saját ábra)*

A két időablakban általában eltérő az, hogy milyen közös átviteli paramétereket használ a végpont és a hálózat. Az első időablakban, ha máshogy nem parancsolja a hálózati szerver, úgy azon a csatornán és olyan spreading factorral kell a végpontnak az adást várnia, amilyen csatornán, valamint ahogy az adás kiment. Az első ablak általában a küldés befejezése után pontosan 1 másodperccel nyílik, de a hálózati szerver ezt is felülírhatja. A második időablak már, amennyiben a hálózat máshogy nem parancsolja, régió specifikus csatornán és átviteli paraméterekkel működik. Európában ez a 869,525 MHz csatorna, amely eltérő szabályozással rendelkezik az alapértelmezett csatornákhoz képest. Egyrészt itt az adó teljesítmény nagyobb mértékre emelhető, másrészt a csatornakihasználtság is engedékenyebb. Biztos, ami biztos a spreading factor 12 értékre kerül, amely a legnagyobb hatótávolságot és a legbiztonságosabb vételt biztosítja. Amennyiben a LoRaWAN hálózati átjáró mindkét időablakban ütemez adást, úgy pontosan ugyanazt az adatcsomagot kell megismételnie a második időablakban, mint amit az első időablakban küldött, csak természetesen más kapcsolati paraméterekkel.

A Class A végpontok esetében tehát csak akkor lehetséges a végpont felé történő adatküldés, amennyiben a végpont előtte a hálózati szerver irányába küld adatot. Ekkor azonban a szerver jelezheti, hogy nem csak egy, hanem több adat is várja a végpontot, így az üres adatok küldésével hozzájuthat az összes rá várakozó adathoz. Ez a fajta kommunikáció természetesen elsősorban akkor működik jól, ha a végpontnak van közlendője és az alkalmazásnak viszonylag kevés mondanivalója van a végpont számára. Tipikusan telemetriai mérések, ahol a szenzorok előre meghatározott program szerint küldik adataikat.

A LoRaWAN *Class B* eszközei ütemezett fogadó ablakkal rendelkeznek. Azon túl, hogy a működésük a Class A osztály szerint is megvalósul és a küldések után a megfelelő 2 vételi ablakot megnyitják, ráadásul ezek az eszközök még előre meghatározott időnként extra ablakokat nyitnak, hogy megnézzék, érkezett-e számukra adat a hálózat felől.

Ahhoz, hogy ezeket az ütemezett vételi ablakokat éppen a megfelelő pillanatban nyissák meg, az átjárók speciális jelzéseket (*Beacon*) küldenek. A végpontban a megfigyelésre nyitott vételi ablak a *Ping időrés (Ping slot)*. A Beacon üzenetek küldését az átjárók szinkronizálják egymáshoz, amelyhez a gyakorlatban GPS vevőket használnak. A Beacon üzenetek 128 másodperc távolságra követik egymást és azért is különlegesek, mert nem a már korábban ismertetett LoRa rádiós üzenet formátumot használják, hanem saját formátumuk van.

Formátumuk úgy van kialakítva, hogy elősegítse a vételüket még akkor is, amikor a végpontok csökkentett energiafelhasználással futnak. Egy Beacon periódus 4096 Ping időrésre van felosztva, amely 30 ms időréseket enged meg egyenként, úgy, hogy a Beacon üzenetek előtt egy bizonyos idővel már nem engednek időréseket, nehogy az ottani adás belelógjon a Beacon üzenetbe. A végpontok egy vagy több ilyen időrést kaphatnak meg, ekkor kell hallgató üzemmódba kapcsolniuk a rádiójukat.

A Ping időrések száma kettő hatványa és a legkisebb periódus kb. 1 másodperc, míg a legnagyobb 128 másodperc. Az egymást követő Beacon periódusokban nem mindig ugyanazok a Ping időrések jutnak a végpontnak, hanem azokat kriptográfiai algoritmusok segítségével álvéletlen módon jelölik ki. Az időrés meghatározásában az eszköz címe (Device Address) és az időszakot megelőző Beacon üzenet küldésének ideje játszik szerepet. A módszer lényege, hogy elkerüljék azokat a szituációkat, amikor egy végpont amiatt nem tudna Class B üzemmódban működni, mert valami más végpont vagy akár a hálózattól független elem állandóan zavarná éppen azt az időrést. A Beacon üzeneteket a végpontoknak így állandóan figyelniük kell, ez egyben segít abban is, hogy az óráik szinkronban legyenek. Amennyiben a végpont nem észlel egy Beacon üzenetet, úgy továbbra is fenntarthatja ezt a működését, de el kell kezdenie a Ping időréseket és a Beacon üzeneti időablakokat kiszélesítenie, keresve az előbb vagy később érkező üzeneteket. Ilyen esetben tehát, amennyiben a Beacon üzenetek nem szűntek meg, úgy a végpont képes lesz visszatalálni a megfelelő időzítéshez. A végpontoknál elvárás, hogy a szinkron elvesztése után még 2 óráig képesek legyenek saját óráikra támaszkodva a megfelelő ütemezéssel működni.

Minden végpont Class A üzemmódban indul, és ha üzemmódot szeretne váltani, akkor azt jeleznie kell a hálózat felé. A végpontnak előzetesen egy Beacon jelet kell fogadni. Amennyiben nincs ilyen jel, mert tegyük fel, a végpont közelében lévő átjáró nem szolgáltat ilyen jelet, akkor sajnos a végpont nem tud Class B üzemmódba kerülni. A keresés meggyorsítható, ha a végpont megkérdezi a hálózati szerver az időzítésről MAC parancson keresztül. Az üzemmód váltás után a végpont minden egyes hálózat felé küldött üzenetében jelzi, hogy immár Class B üzemmódban működik. A LoRaWAN hálózati szerver innen értesül az üzemmód váltásról. Amennyiben hosszabb időre megszűnik a Beacon üzenetek vétele, úgy a végpont visszakerül Class A üzemmódba. Amíg a végpont Class B üzemmódban van, addig MAC parancsok segítségével mind az eszköz, mind a hálózat érheti a Ping paraméterek megváltoztatását.

A Class B üzemmód további jelentősége, hogy segítségével megvalósítható *többesküldés (multicast)* átvitel is. A végpontok ilyenkor két címmel rendelkeznek, egyrészt egy saját címmel (Device Address), illetve egy vagy több multicast címmel. A multicast címeknél ugyanazt az algoritmust használják, mint az egyedi címek esetén ahhoz, hogy meghatározzák a Ping időrések kezdetét. A multicast lehetőség azonban több ponton is korlátozott az egyedi címhez képest. Egyrészt itt minden végpontnak a multicast címhez tartozó kulcsa közös. Ez korlátozott biztonságot nyújt, éppen ezért nem is engedik, hogy a multicast üzenetekben a hálózat vagy a végpontok a kapcsolatra vonatkozó parancsokat küldjenek. A multicast üzeneteket nem kérhetnek nyugtát és nem is hordozhatnak nyugtázást. A LoRaWAN protokoll nem nyújt támogatást a multicast csoportok menedzselésére, ez az alkalmazás feladata, mint ahogy a közös kulcs végpontokhoz történő eljuttatása is.

A Class B üzemmód tehát olyan szenzorok számára ajánlott, amelyek egyben adatot is várnak az alkalmazásuktól még úgy is, ha éppen nem küldenek adatot. A beavatkozás nem azonnali, széles skálán választható ugyan, de minimum 1 másodperc késleltetéssel számolni kell. Ezt az üzemmódot ugyanakkor még mindig használhatják azok az eszközök, amelyek saját akkumulátorral futnak, ugyanis a Beacon üzenetek vétele, illetve a Ping időrések figyelése még nem terheli meg nagyon az energiafogyasztást. Minimális beállítások mellett 128 másodpercenként (ami körülbelül 2 perc) 1 Beacon üzenettel és 1 Ping időréssel kell számolni. Ez az üzemmód a multicast átviteli lehetőség révén még olyan esetekben is hasznos, ahol az alkalmazás egyszerre több végpontot szeretne elérni, mint például okos városokban a közvilágítás.

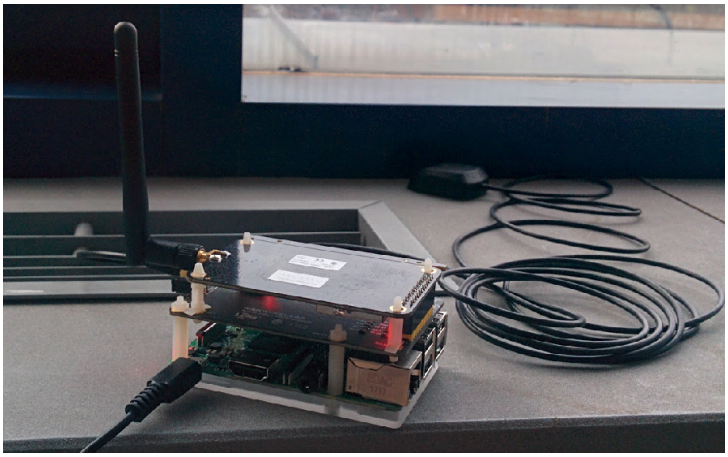
Végül a végpont választhatja a harmadik, úgynevezett *Class C* üzemmódot is. Ebben az üzemmódban a működés szintén a Class A üzemmódra épül, azonban a második vételi időrés hossza elnyúlik a következő végponttól menő adás időpontjáig, ráadásul ugyanezekkel a paraméterekkel rendelkező időrés már megnyílik közvetlenül a végpont adása után egészen az első vételi időablak pontjáig. Tulajdonképpen az adástól eltekintve a végpont folyamatos vétel üzemmódban van.

Class C üzemmódban is használhatóak a multicast üzenetek, pontosan ugyanazokkal a feltételekkel, amelyeket korábban ismertettünk.

A Class C üzemmódot csak akkor képes a végpont használni, ha folyamatos, külső tápellátása van, ugyanis a végpont fogyasztása a rádió energiatakarékossága ellenére a folyamatos működés miatt mégis jelentős. Segítségével azonban elérhető a legkisebb késleltetés az alkalmazástól jövő üzenetek esetében, hiszen a LoRaWAN hálózati szervernek nem kell az ütemezéssel az eszközre várnia, hanem közel tetszőleges időpontban küldheti az adatokat.

5.5. LORAWAN ÁTJÁRÓK

A LoRaWAN átjárók (lásd 5.8 ábra) jelentik a végpontok számára a LoRa kommunikációs partnert, egyben a LoRa rádiós szakasz végét. Feladatuk, hogy a beérkező LoRa rádiós adásokat dekódolják és azokat biztonságosan a LoRaWAN szerver irányába továbbítsák. Kiemelt szerepük van a Class B eszközök esetében, hiszen ilyenkor Beacon üzeneteket kell kibocsátaniuk nagyon pontos időzítéssel.



5.8 ábra: LoRaWAN átjáró RaspberryPi alapon és GPS vevő a Class B ütemezéshez.

(A szerző saját felvétele)

Ahogy azt korábban már írtuk, egy átjáró egyszerre minimum 8 csatornát figyel, és minden egyes csatornán képes dekódolni az egy időben érkező, de más spreading factor értékkel rendelkező adásokat. A LoRaWAN csomagok dekódolása után azonban, mivel az adatcsomag tartalma titkosított, az átjáró számára csak a fejlécek érhetőek el. Ugyancsak használhatatlan az integritásvédő kód (MIC), hiszen az átjáró nem ismeri a LoRaWAN végpont és hálózati szerver közötti, adott csomagra vonatkozó hálózati viszony kulcsot. Ennek következtében egy esetlegesen meghamisított csomag az átjárónál még továbbküldésre fog kerülni, azt majd csak a hálózati szerver dobja el.

Érdemben az átjáró ellenőrizni tudja a fizikai kerethez tartozó ellenőrző kódot, amely a természetese keletkezésű rádiós hibák ellen nyújt védelmet. Amennyiben az ellenőrző kód nem egyezik a csomag dekódolt tartalmán újraszámolt kóddal, úgy a csomag olyan mértékben sérült, hogy nem lehetett rendesen dekódolni, az átjárónak el kell azt dobnia.

Egy másik kód, amit az átjáró ellenőrizhet, az a LoRaWAN hálózati eszköz címe, ugyanis ez a mező nem titkosított. Minden LoRaWAN hálózatnak van egyedi kódja, így kiszűrhető vele, hogy a végpont az átjáróval azonos vagy eltérő hálózatban van. Amennyiben a végpont egy másik hálózathoz tartozik, az átjárónak nem kell továbbítania a dekódolt csomagot. Természetesen dönthetnek úgy is a hálózat operátorok, hogy továbbítják egymás forgalmát (roaming), ekkor viszont az átjárónak ismernie kell, hogy hogyan küldheti át a másik hálózatba a dekódolt adást. A LoRaWAN 1.1 szabvány segítségével azonban valószínűbb, hogy a roaming nem ezen a szinten fog megvalósulni, hanem eggyel feljebb, a hálózati szerver szintjén.

A legtöbb átjáró manapság a Semtech cég által gyártott SX1301 LoRa modulra épül, ez pedig két rádióval is rendelkezik. Éppen ezért az átjárónak nem kell felfüggesztenie a csatornák figyelését miközben a végpontok felé szeretne kommunikálni, a LoRaWAN végpontok vétele így zavartalan. Azt, hogy az adott végponthoz melyik frekvencián és milyen spreading factorral, illetve kódolással kell adnia, utasításban kapja a LoRaWAN szervertől.

A LoRaWAN átjáró a hálózati szerverrel tartja a kapcsolatot. A közelmúltban háromfajta protokoll terjedt el ehhez a kapcsolathoz. Az egyik, amelyet minden átjáró ismer, a „packet forwarder" nevű protokoll, amit még a Semtech cég fejlesztett ki. Ez egy viszonylag egyszerű protokoll, amely a LoRa rádión érkező adatokat kiegészíti a kommunikáció során az átjáró által érzékelt adatokkal, majd UDP csomagok segítségével elküldi azt a LoRaWAN szerver megfelelő IP címére és portjára. Ez a protokoll még több LoRaWAN hálózat esetében használható maradt, de ma már nem tartják megfelelőnek, az UDP kapcsolat ugyanis teljesen nyílt és bárki lehallgathatja az interneten, módosíthatja a tartalmat vagy éppen hamisan gyártott csomagokat küldhet.

Maga az adatforgalom ugyanakkor továbbra sincs veszélyben, hiszen annak titkosításáról és integritás védelméről a végpont és a LoRaWAN hálózati szerver a teljes szakaszon gondoskodik. A packet forwarder az adatcsomagot szövegesen *JSON (JavaScript Object Notation)* formátumban küldi (lásd 5.9 ábra), míg az adatcsomag nyers tartalma Base64 kódolással Azt feltételezik ugyanis, hogy az átjáró bekapcsolására már nagy sebességű hálózat rendelkezésre áll, ahol többet számít az adatok könnyebb értelmezhetősége, mint a tömörsége.

{„rxpk":[{„tmst":872440795,"chan":0,"rfch":0,"freq":868.099975,"stat":1,"modu":"LORA","datr":"SF9BW125","codr":"4/5","lsnr":9,"rssi":-45,"size":23,"data":"AGIIAPB+1bNw782riWdFIwGvMGl8fu4="}]}
{„txpk":{„imme":false,"tmst":1209615992,"freq":868.099975,"rfch":0,"powe":14,"modu":"LORA","datr":"SF9BW125","codr":"4/5","ipol":true,"size":33,"ncrc":true,"data":"INlvpV/AN/ZxVP6Wv8k7SXOnweD4oJYDoO6M4YD9Gdnu"}}

5.9 ábra: Példák a packet forwarder üzeneteire

A másik protokoll az UDP helyett az *MQTTS (Secure Message Queuing Telemetry Transport)* protokollt használja az üzenetek továbbítására. Maga az MQTT protokoll jól ismert a szenzorhálózatok világában, ez egy pehelysúlyú megoldás a kommunikációra. Újdonsága a hagyományos megközelítéshez képest, hogy nem végpont-végpont kapcsolatokat hoz létre, hanem akinek közlendője van, az egy adott csatornán közölheti, valamint erre a csatornára, akik érdeklődnek az üzenetek iránt, feliratkozhatnak. A LoRaWAN átjárók esetében az átjárók feliratkoznak a hálózati szerver csatornáira és itt küldik a fogadott üzeneteket, valamint itt fogadják a szervertől jövőket.

Az MQTTS protokoll lehetővé tesz hitelesítést is, így a szerverhez kapcsolódó átjárók csak akkor jutnak hozzá az üzenetekhez, illetve küldeni is csak akkor tudnak, ha hitelesítik előbb magukat. További előnye a protokollnak, hogy az MQTT kapcsolat, már egy biztonságosan kiépített kapcsolat fölött jöhet létre, erre utal az S betű az MQTTS mozaikszó végén. Így az átjárók és a szerver között küldött üzenetek nem értelmezhetőek mások által és az integritásvédelem is biztosítva van. Az üzeneteket a Google által kifejlesztett *Protocol Buffers* megoldással helyezi el az adatcsomagban, amely platform és nyelv semleges megoldást kínál. Segítségével az adatok tömörebben vannak ábrázolva, mint a korábban leírt JSON protokoll esetén.

A harmadik protokoll tulajdonképpen az előbbi két protokoll keveréke. Az MQTT protokollt használja adatátvitelre, illetve a JSON formátumot használja az adatok leírására. Az MQTT protokoll esetén itt is lehetséges a hitelesítés, illetve akár a biztonságos adatátvitel is.

5.6. LORAWAN HÁLÓZATI SZERVER

A LoRaWAN hálózati szerver feladata meglehetősen bonyolult, hiszen ez menedzseli az összes éppen csatlakozó és már csatlakozott végpontot, illetve az átjárókat. Tisztában kell lennie, hogy mely csomópont éppen hol tartózkodik, és ennek megfelelően kell küldenie a hálózat felől küldött üzeneteket. Míg az átjárók szenzorok ezreivel foglalkoznak, addig a hálózati szervernek több száz átjáróval és a hozzájuk kapcsolódott végpontokkal kell foglalkoznia. A csatornákat optimalizálni kell, figyelembe véve, hogy a végpontok ne zavarják egymást, illetve azt is, hogy ahogy a végpontok, úgy az átjárók is csak egy pici részét használhatják a csatornán töltött időnek adatküldésre.

Mindazonáltal általában egy nagy kiterjedésű LoRaWAN hálózatban is csak kevés számú szerver található. A *loriot.io* globális LoRaWAN szolgáltató az egész Földre kiterjedő szolgáltatással bír. Ez a szolgáltató összesen 4 szervert üzemeltet az európai és afrikai régióban (Frankfurt, Amszterdam, Madrid és Fokváros), 5 szervert az ázsiai és ausztráliai részen (Szingapúr, Sydney, Shenzhen, Tokió, Mumbai) és 3 szervert Amerikában (Kalifornia, New York és Sao Paulo). Ezek a szerverek képesek ellátni 70 ország forgalmát.

A hálózati szerver a végpontok forgalmát az átjárókon keresztül fogadja. Mivel az átjárók minden forgalmat további elemzés nélkül továbbítanak felé, amely az átjáróhoz érkezik, ezért előfordulhat, hogy amennyiben a végpont több átjáróhoz is megfelelően közel van, úgy ugyanazt az adatcsomagot a szerver többször is megkapja. A LoRaWAN hálózati szerver egyik feladata lesz a multiplikált csomagok összevonása és a továbbiakban egyetlen adatcsomagként kezelése. Mivel időben eltérhet a különböző átjáróktól beérkező forgalom, ezért az adatcsomagokat rövid ideig tárolni kell.

Mivel a nyugtázásnak vagy a vissza irányú adatnak alapértelmezésben egy másodpercen belül vissza kell érkeznie a végpontra, így természetesen a duplikátumokat sem érdemes tovább várnia. Ebbe az időkeretbe ugyanakkor annak is bele kell férnie, hogy a hálózati szerver ellenőrzi az adatcsomag épségét. A LoRaWAN adatcsomagban erre a célra szolgál a MAC 4 bájtos mező. Az ellenőrzéséhez a szervernél található hálózati kulcsra

van szükség. Azt azonban, hogy melyik ez a hálózati kulcs, nem is olyan egyszerű kideríteni. Ez azért van, ugyanis a LoRaWAN hálózati azonosító (Device Address) mindössze 4 bájtos és ugyan egyedinek kellene lennie egy hálózaton belül, mégsem garantálható, hogy ne legyenek ütközések a különböző alkalmazások és felhasználók között, különösen akkor, amikor ezeket a címeket akár a felhasználó is beállíthatja egy ABP csatlakozás során, illetve gondolhatunk a roaming végpontokra is.

Tulajdonképpen nagy gond akkor sincs, ha a végpont azonosítója nem egyedi a hálózatban. A LoRaWAN hálózati szerver ugyanis az eszközök azonosítója mellett tárolja minden egyes eszköz hálózati kulcsát és a hozzá tartozó alkalmazás azonosítóját. Amennyiben több azonosító is illeszkedik az éppen fogadott adatcsomag esetében, akkor a szerver az összes lehetséges hálózati kulcsot kipróbálja, ahol cím egyezés van, és amelyik kulcs esetében az integritást védő MAC mező alapján korrekt érték mutatkozik, az ahhoz tartozó alkalmazás felé továbbítja az adatcsomagot. Mivel a hálózati kulcsok 128 bit hosszúságúak, ezért a szándékosságot kizárva, elhanyagolható annak a valószínűsége, hogy a kulcsok megegyezzenek különböző alkalmazások esetében is. Az átjáróktól átvett és duplikátumoktól megtisztított adatcsomagot így végül a hálózati szerver a megfelelő alkalmazáshoz juttatja.

A LoRaWAN szerver menedzseli a hálózatból a végpontok felé irányuló adatfolyamot is. Három esetben történik ilyen kommunikáció. Egyrészt az alkalmazás adatot küldhet a végpont felé, másrészt elképzelhető, hogy a végpont nyugtát vár az utolsó elküldött adatcsomagjához, harmadrészt pedig a LoRaWAN szerve hálózati parancsokat küldhet a végpontnak. Az első esetben az alkalmazás a hálózati szerver számára küldi el az adatcsomagok és ellátja azt a megfelelő adat titkosítással.

A LoRaWAN 1.0 szabvány szerinti hálózatokban ezt sajnos nem olyan egyszerű megtennie függetlenül a hálózati szervertől, ugyanis a titkosítás során felhasználásra kerülnek olyan sorszámok is, amelyeket az alkalmazás nem tud önmaga irányítani. Ugyan megoldható olyan algoritmus, ahol a titkosítás továbbra is a felhasználónál marad és azt a LoRaWAN hálózati szerver nem ismeri, mégis, mint ahogy már utaltunk rá, ebben a verzióban inkább szorosabban összefonódik az hálózati és alkalmazás szerver szerep és a titkosítást így valójában már a hálózati szerver végzi el.

A LoRaWAN 1.1 szabványban ugyanakkor már jól elkülönül az hálózati és alkalmazás szerver szerepkör és a kulcsok és sorszámok helyes felosztásával megoldották, hogy valóban az alkalmazás szerver végezze el a titkosítást és abban a hálózati szerver ne vegyen részt. Ez fontos mozzanat volt a roaming megoldásánál is. Az alkalmazástól átvett titkosított adatcsomagot a hálózati szerver integritást védő kóddal látja el, amihez az általa ismert hálózati kulcsra van szükség.

Az adatcsomag végponthoz küldéséhez először egy átjáróhoz kell eljuttatnia azt. Amennyiben a végpont Class B vagy Class C üzemmódban működik, úgy ezt akár abban a pillanatban is megteheti, amikor az adatküldés kérés megérkezik az alkalmazástól. Ebben az esetben ugyanis a szerver feltehetően jól ismeri, hogy a végpontot mely átjárókon keresztül lehet elérni. Ráadásul ezeknél az üzemmódoknál a végpont számít is adatcsomag érkezésére. Class A esetében azonban más a helyzet, itt a szerver egészen addig nem tudja, hogy milyen átjárókat használhat, amíg egy adatcsomag meg nem jelenik az adott végpont felől. Éppen ezért a hálózati szerver ebben az esetben elraktározza az adatcsomagokat és majd csak akkor veszi elő, ha a végponttól adás érkezik.

Az adatcsomag küldéshez a megfelelő átjáró kiválasztása is a hálózati szerver feladata. Egy végpont több átjáróhoz is közel lehet. Azt, hogy pontosan melyik átjárók érhetik el a végpontot, nem is olyan egyszerű meghatározni, hiszen nem biztos, hogy csak azok, amelyek látják a végpont forgalmát. Előfordulhat ugyanis, hogy a végpont alacsony spreading factor értéket használ, így adása nem ér el távolra, míg egy távolabbi átjáró egy magasabb

spreading factorral már elérhetné azt. A hálózati szervernek biztosítani kell az optimális adatküldést és úgy kell kiválasztania a végponthoz vezető átjárót.

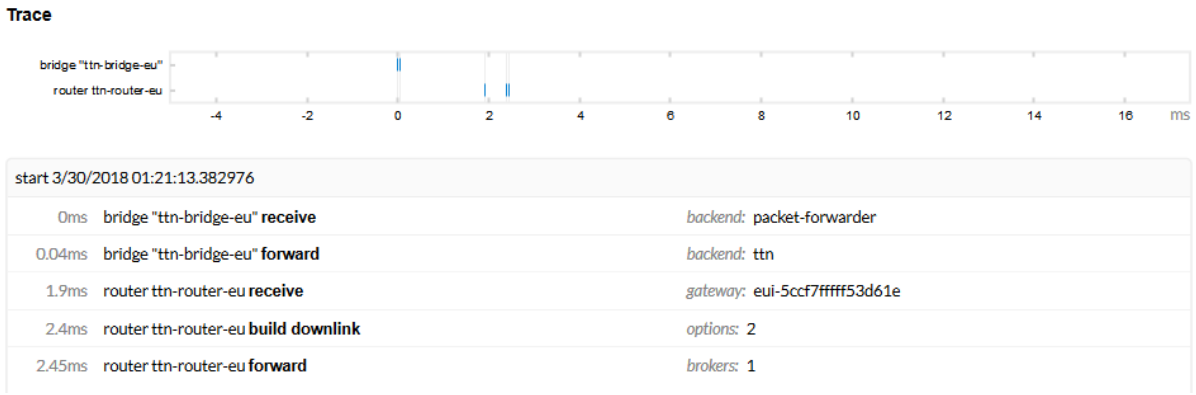
Az optimalizálásnál két faktor fog leginkább számítani, egyrészt az adás sebessége, másrészt az adott átjáró forgalma, amely egyben limitálja is, hogy mennyi adatot küldhet végpontoknak. Adott esetben lehet, hogy a hálózat inkább egy távolabbi és ezért lassabb adatátvitelt fog választani egy, a végponthoz ugyan közel lévő, de nagy forgalmú átjáróval szemben. Az időzítés a Class A üzemmód esetén kritikus, ilyenkor a kiválasztásnak és továbbításnak is bele kell férnie az 1 másodperces határidőbe, úgy, hogy ott már a duplikátumok megszüntetésével és a dekódolással is foglalkozott. A többi üzemmód esetén az adatküldés időzítése szabadabb, persze nem árt törekedni, hogy az minél hamarabb megtörténjen.

A nyugták esetében mindhárom üzemmód egyformán működik és egy nyugtázott válasz üzenetet kell küldeni 1 másodperc alatt. Amennyiben van olyan adat, amely a végpont felé menne, akkor a szerver erre az adatcsomagra fogja ráültetni a nyugtát, ellenkező esetben üres adatcsomagot küld nyugtával együtt. Az adatküldés a korábban ismertetettek szerint megy. Szintén hasonló a helyzet a hálózati parancsok (MAC parancsok) esetében is, azzal a különbséggel, hogy itt a hálózati szerver a kezdeményező fél és erről nem is értesíti az alkalmazást sem, mint ahogy a végpont esetében sem fog a parancs eljutni az ott futó alkalmazásig.

Hasonlóan viszont az adatküldés korábbi eseteihez, Class A üzemmódban lévő eszköznek csak akkor lehet parancsot küldeni, ha az éppen adatot küld az alkalmazás felé. A többi üzemmód esetén bármikor kiküldhető a parancs. Minden esetben a hálózati szerver egyetlen átjárót fog megjelölni. Amennyiben a packet forwarder protokollt használja az átjáró, úgy az átjáró fogja elkérni (*polling*) a neki szóló feladatokat a szervertől. Amennyiben a kommunikáció már a modernebb eljárásokat követi, úgy a hálózati szerver kérés nélkül elküldi az átjáró számára az elküldendő adatcsomagot. A szervernek természetesen azt is meg kell határoznia, hogy milyen paraméterekkel kell az adatcsomagot elküldenie.

A LoRaWAN hálózati szervernek szintén feladata a hálózat optimalizálása. Az egyes végpontok jelezhetik kérésüket, hogy a kapcsolatot optimalizálják az *ADR (Adaptive Data Rate)* mechanizmuson keresztül. Az optimalizálás során a szerver adatot gyűjt arról, hogy az adott végpont milyen átjárókon keresztül, milyen *jel-zaj viszonnyal (Signal-to-Noise Ratio, SNR)* érkezett. Amennyiben úgy tűnik, hogy az aktuális kommunikációs paraméterekhez képest a vétel még akkor is biztosított lenne, ha egy gyorsabb, kisebb energiaigényű átvitelt használ a végpont, akkor a hálózati szerver utasítja, hogy változtasson paramétereket és kijelöli az új spreading factor és sávszélesség értéket.

Példaként, ha a szerver úgy látja, hogy a végponttól érkező utóbbi adások egy adott átjárón keresztül 5 dBm értéket mutatnak, míg az adás 12-es spreading factor értékkel érkezik, akkor utasíthatja, hogy váltson 7-es spreading factor értékre, hiszen az így várható -7.5 dBm körüli jel-zaj viszony érték mellett még mindig képes lesz a hálózat az adatcsomagokat fogadni. Így kevesebb interferenciát okoz az adás, illetve a végpont is kevesebb energiát használ. A LoRaWAN szervernek persze a hálózat egészét kell tekintenie optimalizálásnál.



5.10 ábra: Adatcsomag megerősítés nélkül *(A szerző saját felvétele)*

5.7. LORAWAN ALKALMAZÁS SZERVER

A LoRaWAN alkalmazás szerver az a hely, ahol a felhasználó alkalmazása fut. A hálózati szerver ide továbbítja az alkalmazáshoz tartozó végpontok adatait, illetve innen érkeznek a végpontok számára küldendő adatok. Ahogy már írtuk, az LoRaWAN 1.0 specifikáció nem különíti el eléggé az hálózati és alkalmazás szerver feladatkörét, sőt a biztonsági kulcsok sem különülnek el eléggé. Ebben az esetben, a gyakorlatban a hálózati szerver és az alkalmazás szerver a hálózati operátornál integrálódik. A szolgáltató pedig hozzáférést biztosít, ahol a felhasználó a már dekódolt, tehát már nem titkosított adatokhoz hozzáfér. A LoRaWAN 1.1 specifikáció esetében azonban az adatcsomagok kezelése, és egyben a titkosítás kezelése egyértelműen az alkalmazás szerver feladata, azt a hálózati szerver már nem tudja megtenni helyette.

Bár egyelőre kevés példa van rá, hiszen a LoRaWAN 1.1 specifikáció még újnak számít, de amennyiben az alkalmazás szerver elkülönül a hálózati szervertől, úgy a kapcsolat a már említett MQTT protokoll tudja biztosítani. Hasonlóan, ahogy az adatcsomagok magához a hálózati szerverhez érkeznek, az továbbítja a titkosított adatrészt az alkalmazás szerver felé.

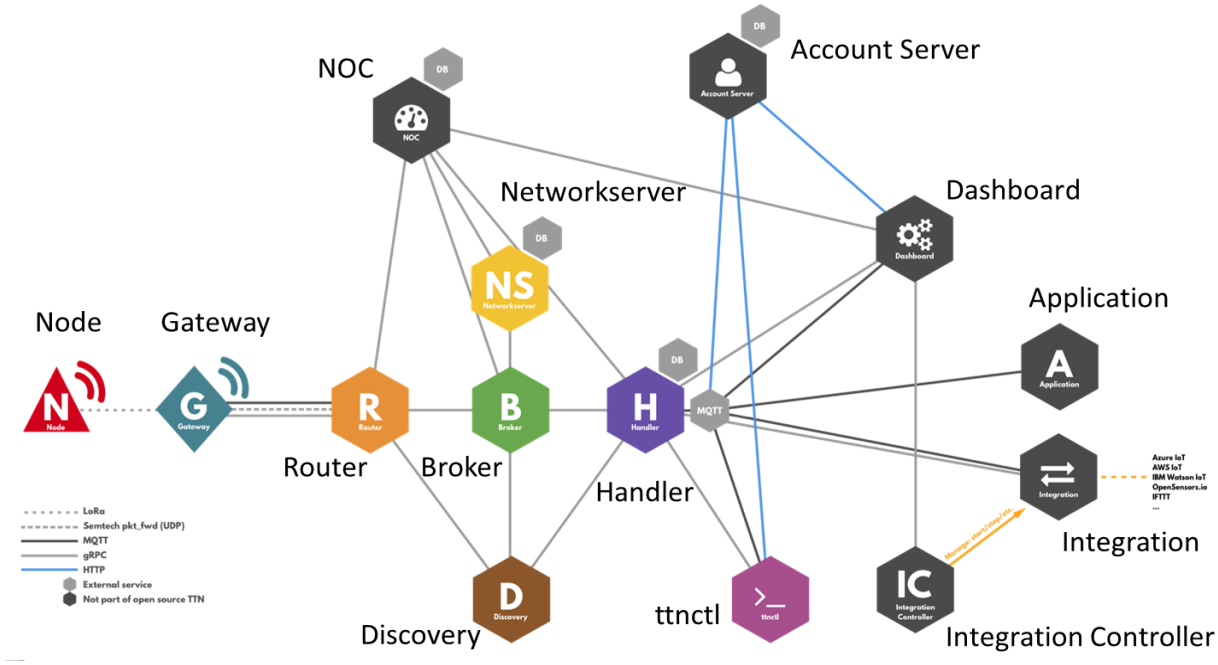
5.8. A THE THINGS NETWORK HÁLÓZATA

Az egész LoRaWAN hálózat szerveződésére példaként a *The Things Network (TTN)* felépítését mutatjuk be. A TTN hálózata egy közösségi LoRa hálózat. Mivel a frekvenciasáv nem licenc köteles, így tulajdonképpen bárki építhet saját hálózatot. Ezt valóban meg is teheti, ugyanis az interneten több forrásból is talál szoftvereket a LoRaWAN hálózat komponenseihez. Az eszközök is megvásárolhatóak magánember számára is elérhető árakon. Ahelyett azonban, hogy mindenki magának építené fel a saját LoRaWAN hálózatát egyszerűbb és ráadásul több területen is kedvezőbb, ha inkább csatlakozik egy nagyobb hálózathoz.

A TTN hálózat éppen egy ilyen közösségi hálózat, ahol a felhasználók szabadon csatlakoztathatják átjáróikat és végpontjaikat egy nagyobb hálózathoz, illetve természetesen az alkalmazás oldalon is lehetőségük van végpontjaik adatainak kinyerésére. A felhasználó és magának a hálózatnak is előnyére válik az új átjárókkal bővülés.

A felhasználó így már nem csak saját átjárója közelében, hanem az egész hálózat területén használhatja a végpontokat, míg a hálózat is nagyobb lefedést tud kínálni, biztosabb kapcsolatokkal. Annak érdekében viszont, hogy a végpontok ne árasszák el a hálózatot, egy *fair access policy* házirendet vezettek be. A házirend szerint egyetlen végpont sem adhat napi 30 másodpercnél többet. Ez persze még a legrosszabb hálózati paraméterek mellett is 100 üzenetet jelent.

A TTN hálózatának architektúrája az 5.11. ábrán látható. A már leírt hálózati elemeken túl felfedezhető, hogy a hálózati szervert és az alkalmazás szervert az implementációban jobban tagolták. Ez elősegíti a decentralizációt, amely előnyös, ha a felhasználó a saját elemeit szeretné a hálózathoz csatlakoztatni.



5.11 ábra: A The Things Network architektúrája (publikusan elérhető ábra,
https://www.thethingsnetwork.org/docs/network/architecture.html)

Az átjárók csatlakozásához a már ismertetett háromféle módszer használható. A kapcsolatok a TTN hálózatában a *Routerhez* futnak be. Minden átjárónak egyetlen Routerrel van kapcsolata. A Router funkciója, hogy menedzseli az átjárókat, figyeli folyamatosan az állapotukat. Amennyiben az adatcsomag a hálózat felől érkezik, akkor a Router az az elem, aki ütemezi, hogy melyik átjáró, mikor és hogyan fogja továbbküldeni az adatcsomagot a végpont felé.

A Router egy vagy több *Broker* elemmel áll kapcsolatban. A Broker a TTN hálózat központi eleme. A Broker találja meg a végponthoz tartozó alkalmazást és továbbítja a végponttól jövő adatcsomagokat ahhoz. Az alkalmazás felől érkező adatcsomagokat pedig a megfelelő Routerhez irányítja.

A LoRaWAN hálózat menedzselését és optimalizálását a *Network Server* komponens látja el. Az alkalmazásokkal a *Handler* tarja a kapcsolatot. Mivel a TTN is egy olyan hálózat, ahol az alkalmazás szerver és a hálózati szerver

funkciók egy kézben vannak, így a Handler egyben az a komponens, ami az adatcsomagok titkosítását és annak feloldását is végzi.

További lényeges, TTN specifikus elem az *Account Server*, amely a felhasználók, alkalmazások és átjárók hitelesítését végzi. Szintén TTN specifikus a *Discovery* szerver elem. Mivel a TTN hálózata decentralizált és nyílt, az egyes komponensek hirdetik a saját szolgáltatásaikat és ez alapján történik a hálózatba szerveződés.

A Handler, amely az alkalmazások adatait kezeli, az *Integration Controller* elem segítségével más létező webes szolgáltatásokkal kerülhet kapcsolatba. A jelenlegi kínálatban megtalálunk adatbázis kezelőt, térkép rajzolót, de például a népszerű *IFTTT (If This Then That)* szolgáltatás is integrálva van.

6. IRODALOMJEGYZÉK

Abbasi, Ameer Ahmed – Younis, Mohamed. A survey on clustering algorithms for wireless sensor networks. *Computer Communications*, Vol. 30, Issue 14, p. 2826–2841, 2007.

Allen, G.W. – Swieskowsli, P. – Welsh, M.: MoteLab: *A Wireless Sensor Network Testbed*, in Proc. of the 4th International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, USA, April 2005.

Allen, G.W. – Swieskowsli, P. – Welsh, M.: CitySense: *An Urban-Scale Wireless Sensor Network and Testbed*, in Proc. of the IEEE Conference on Technologies for Homeland Security, Waltham, MA, USA, May 2008.

Blywis, Bastian – Güneş, Mesut – Hofmann, Sebastian – Juraschek, Felix: *A Study of Adaptive Gossip Routing in Wireless Mesh Networks*, in Proc. of International Conference on Ad Hoc Networks, ADHOCNETS 2010, pp 98–113.

Burin des Roziers, C. – Chelius, G. – Ducrocq, T. – Fleury, E. – Fraboulet, A. – Gallais, A. – Mitton, N – Noel, T – Vandaele, J.: *Two demos using SensLAB: Very Large Scale Open WSN Testbed*, in Proc. of The 7th IEEE International Conference on Distributed Computing in Sensor Systems (IEEE DCOSS '11), 2011.

Chakrabarti, A – Sabharwal, A – Aazhang, B: *Using predictable observer mobility for power efficient design of sensor networks*. In Proc., 2nd Int. Workshop on Information Processing in Sensor Networks (IPSN), pages 129–145, Palo Alto, CA, USA, April 2003.

Chen, G. – Branch, J. – Pflug, M. – Zhu, L. – Szymanski, B.: *SENSE: a wireless sensor network simulator.* In Advances in pervasive computing and networking, pp. 249–267, Springer US, 2005.

Coulson, Geoff – Porter, Barry – Chatzigiannakis, Ioannis – Koninis, Christos – Fischer, Stefan – Pfisterer, Dennis – Bimschas, Daniel – Braun, Torsten – Hurni, Philipp – Anwander, Markus – Wagenknecht, Gerald – Fekete, Sandor P. – Kröller, Alexander – Baumgartner, Tobias: Flexible experimentation in wireless sensor networks, in *Communications of the ACM*, ACM, volume 55, 2012.

Gerla, Mario – Xu, Kaixin: Multimedia streaming in large-scale sensor networks with mobile swarms. *SIGMOD* Rec., 32(4):72–76, 2003.

Fekete, S. P. – Kröller, A. – Fischer, S. – Pfisterer, D.: Shawn: *The fast, highly customizable sensor network simulator.* In Proceedings of the Fourth International Conference on Networked Sensing Systems, INSS 2007, June 2007.

FIT/IoT-LAB: *IoT experimentation at a large scale* https://www.iot-lab.info/

Hata, M.: Empirical formula for propagation loss in land mobile radio services, *IEEE Transactions on Vehicular Technology,* 29(3), 317–325, 1980.

Heinzelman, W – Kulik, J – Balakrishnan, H: *Adaptive Protocols for Information Dissemination in Wireless Sensor Networks*, in Proc. 5th ACM/IEEE Mobicom, Seattle, WA, Aug. 1999. pp. 174–85.

Heinzelman, W. – Chandrakasan, A. – Balakrishnan, H.: *Energy-Efficient Communication Protocols for Wireless Microsensor Networks*, in Proceedings of the 33rd Hawaiian International Conference on Systems Science (HICSS), January 2000.

Hristov, H.D.: *Fresnel Zones in Wireless Links, Zone Plate Lenses and Antennas* (1st ed.), Artech House, Inc., Norwood, MA, USA, 2000.

Intanagonwiwat, C – Govindan, R – Estrin, D: Directed diffusion: *A scalable and robust communication paradigm for sensor networks*, in Proceedings of ACM MobiCom'00, Boston, MA, Aug. 2000, pp. 56–67.

Issariyakul, Teerawat – Ekram Hossain: *Introduction to network simulator NS2*. Springer, 2011.

Jakab László – Vida Rolland – Fehér Gábor: *Szenzorok és kommunikációs technológiáik*, Kismonográfia, NKE, 2018.

Jayaraman, Prem Prakash – Zaslavsky, Arkady – Delsing, Jerker: *Sensor data collection using heterogeneous mobile devices.* In Proc., IEEE International Conference on Pervasive Services, pages 161–164, June 2007.

Kansal, Aman – Rahimi, Mohammad – Kaiser, William J – Srivastava, Mani B – Pottie, Gregory J. – Estrin, Deborah: *Controlled mobility for sustainable wireless networks.* In Proc., IEEE Sensor and Ad Hoc Communications and Networks (SECON), Santa Clara, CA, October 2004.

Kansal, A. – Somasundara, A – Jea, D – Srivastava, M.B – Estrin, D: *Intelligent fluid infrastructure for embedded networks.* In Proc., ACM MOBISYS 2004, pages 111–124, Boston, MA, USA, June 2004.

Kim, Y. P., – Jung, E. – Park, Y. J. (2009). A radio-aware routing algorithm for reliable directed diffusion in lossy wireless sensor networks. *Sensors*, 9(10), 8047–8072.

Krishna, K. Madhava – Hexmoor, Henry – Rao, P. Subba – Chellapa, S: *A surveillance system based on multiple mobile sensors.* In Proc. of FLAIRS 2004, Special Track on AI Techniques in Multi-sensor Fusion, May 2004.

Krishnamachari, L. – Estrin, D. – Wicker, S. (2002). *The impact of data aggregation in wireless sensor networks.* In Distributed Computing Systems Workshops, 2002. Proceedings of 22nd International Conference on (pp. 575–578). IEEE.

Li, Jiakai – Serpen, Gursel: Simulating Heterogeneous and Larger-Scale Wireless Sensor Networks with TOSSIM TinyOS Emulator, *Procedia Computer Science*, Volume 12, 2012, Pages 374–379.

LoRa Alliance, LoRaWAN specification 1.1, 2017. https://lora-alliance.org/sites/default/files/2018-04/lorawantm_specification_-v1.1.pdf

Luo, Jun – Hubaux, Jean-Pierre: *Joint mobility and routing for lifetime elongation in wireless sensor networks.* In Proc., IEEE INFOCOM, pages 1735–1746. IEEE, 2005.

Montenegro, G – Kushalnagar, N – Hui, J – Culler, D: *Transmission of IPv6 Packets over IEEE 802.15.4 Networks, IETF RFC 4944*, September 2007.

Nayyar, Anand – Singh, Rajeshwar: A Comprehensive Review of Simulation Tools for Wireless Sensor Networks (WSNs), *Journal of Wireless Networking and Communications,* 5(1): 19–47, 2015.

ns-3 Network Simulator, ns-3 Tutorial Release ns-3.26, March 2017, https://www.nsnam.org/docs/release/3.26/tutorial/ns-3-tutorial.pdf.

Okumura, Y. – Ohmori, E. – Kawano, T. – Fukuda, K: Field strength and its variability in the VHF and UHF land mobile radio service, Rev. *Elec. Commun. Lab.,* 16(9/10), 825–73. 1968.

Pešović, Uroš M. – Mohorko, Jože J. – Benkič, Karl – Čučej, Žarko F.: *Single-hop vs. Multi-hop – Energy efficiency analysis in wireless sensor networks,* in Proc. 18. Telekomunikacioni forum TELFOR 2010, Srbija, Beograd, November 2010.

RadioShuttle 2017. http://www.radioshuttle.de/en/radioshuttle-2/protocol/, 2017.

Rajagopalan, R – Varshney, P: Data-aggregation Techniques in Sensor Networks: A Survey, IEEE Commun. *Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, Oct. 2006.

Rastegarnia, A – Solouk, V: *Performance evaluation of Castalia Wireless Sensor Network simulator*, in Proc. of 34th International Conference on Telecommunications and Signal Processing (TSP), Budapest, 2011, pp. 111–115.

Rault, Tifenn – Bouabdallah, Abdelmadjid – Challal, Yacine: Energy efficiency in wireless sensor networks: A top-down survey, *Computer Networks*, Volume 67, 4 July 2014, Pages 104–122.

Rensfelt, O – Hermans, F – Ferm, C – Gunningberg, P. – Larzon, L.A.: *Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes,* in Proc. of the 4th ACM International Workshop on Wireless Networks Testbeds, Experimental Evaluation, and Characterization, 2010.

Shah, Rahul C. – Roy, Sumit – Jain, Sushant – Brunette, Waylon Brunette: *Data MULEs: Modelling a three-tier architecture for sparse sensor networks.* In Proc., IEEE Workshop on Sensor Network Protocols and Applications (SNPA), pages 30–41, Anchorage, Alaska, USA, May 2003.

Shnayder, V. – Hempstead, M. – Chen, B. – Werner Allen, G. – Welsh, M.: *Simulating the power consumption of large-scale sensor network applications.* In Proceedings of the 2nd international conference on embedded networked sensor systems (SenSys ,04). ACM, New York, NY, USA, 188–200.

Sklar, B.: Rayleigh Fading Channels in Mobile Digital Communication Systems Part I: Characterization, *IEEE Communications Magazine.* 35 (7): 90–100, 1997.

SmartSantander Testbed Manual, http://www.smartsantander.eu/wiki/

Sobeih, A. – Chen, W. P. – Hou, J. C. – Kung, L. C. – Li, N. – Lim, H. – Zhang, H.: J-sim: *A simulation environment for wireless sensor networks.* In Proceedings of the 38th annual Symposium on Simulation, pp. 175–187, IEEE Computer Society, April 2005.

Tilak, S. – Abu-Ghazaleh, N. B. – Heinzelman, W. (2002). A taxonomy of wireless micro-sensor network models. *ACM SIGMOBILE Mobile Computing and Communications Review,* 6(2), 28–36.

Tong, Lang – Zhao, Qing – Adireddy, Srihari: Sensor networks with mobile agents. In Proc., *IEEE MILCOM 2003*, volume 22, pages 688–693, Boston, MA, USA, October 2003.

TOSSIM – TinyOS Wiki, Stanford University, http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM

Trajcevski, Goce – Scheuermann, Peter – Brönnimann, Herve: *Mission-critical management of mobile sensors: or, how to guide a flock of sensors.* In DMSN '04: Proceeedings of the 1st international workshop on Data management for sensor networks, pages 111– 118, New York, NY, USA, 2004. ACM.

Tseng, Yu-Chee Tseng – Wang, You-Chiun Wang – Cheng, Kai-Yang: *An integrated mobile surveillance and wireless sensor (imouse) system and its detection delay analysis.* In MSWiM '05: Proceedings of the 8th ACM International Symposium on Modelling, Analysis and Simulation of Wireless and Mobile Systems, pages 178–181, New York, NY, USA, 2005. ACM.

Varshney, M. – Xu, D. – Srivastava, M. – Bagrodia, R.: *sQualNet: A scalable simulation and emulation environment for sensor networks.* In Proceedings of the International Conference on Information Processing in Sensor Networks, New York, NY, USA, April 2007.

Vidhi S. Patel – Chandresh R. Parekh: Survey on Sensor Protocol for Information via Negotiation (SPIN) protocol, *International Journal of Engineering Research & Technology*, Vol. 3 – Issue 3, March – 2014.

Vincze, Zoltan – Vass, Dorottya – Vida, Rolland – Vidacs, Attila – Telcs, András: Adaptive sink mobility in event-driven densely deployed wireless sensor networks, in *Int. Journal Ad Hoc and Sensor Wireless Networks,* 3(2-3):255–284, 2007.

Winter, T. et al, „RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", IETF RFC 6550, March 2012.

Ye, Wei – Heidemann, John – Estrin, Deborah: *An Energy-Efficient MAC Protocol for Wireless Sensor Networks,* Proc. of Infocom 2002, pp. 1567–1576, New York, USA, June 2002.

Ye, Wei – Heidemann, John – Estrin, Deborah: *Medium Access Control With Coordinated Adaptive Sleeping for Wireless Sensor Networks*, IEEE/ACM Transactions on Networking, 12(3):493–506, June 2004.

A Nemzeti Közszolgálati Egyetem kiadványa

Kiadó:
Nemzeti Közszolgálati Egyetem;
Közigazgatási Továbbképzési Intézet
www.uni-nke.hu

Felelős kiadó:
Prof. Dr. Kis Norbert rektorhelyettes
Címe: 1083 Budapest, Üllői út 82.

Olvasószerkesztő:
Kiss Eszter

Tördelőszerkesztő:
Vörös Ferenc

ISBN (pdf) 978-963-498-119-0