KÖZTESRÉTEG SZOLGÁLTATÁSOK HATÉKONY MEGVALÓSÍTÁSA SZENZORHÁLÓZATOKBAN

DOI: 10.18136/PE.2015.584

Doktori (PhD) értekezés

Készítette:
Vakulya Gergely
Témavezető:
Dr. Simon Gyula

Pannon Egyetem
Műszaki Informatikai Kar
Informatikai Tudományok Doktori Iskola

2015

KÖZTESRÉTEG SZOLGÁLTATÁSOK HATÉKONY MEGVALÓSÍTÁSA SZENZORHÁLÓZATOKBAN

Értekezés doktori (PhD) fokozat elnyerése érdekében

Írta:
Vakulya Gergely

Készült a Pannon Egyetem Informatikai Tudományok Doktori Iskolája keretében

Témavezető: **Dr. Simon Gyula**

Elfogadásra javaslom (igen / nem)

(aláírás)

A jelölt a doktori szigorlaton ..... % -ot ért el.
Veszprém, ...
a Szigorlati Bizottság elnöke

Az értekezést bírálóként elfogadásra javaslom:

Bíráló neve: ..... (igen /nem)

(aláírás)

Bíráló neve: ..... (igen /nem)

(aláírás)

A jelölt az értekezés nyilvános vitáján ..... % - ot ért el.

Veszprém, ...
a Bíráló Bizottság elnöke

A doktori (PhD) oklevél minősítése .....

.....
Az EDT elnöke

Tartalomjegyzék

Kivonat

A szenzorhálózatok megjelenése a kis méretű, mobil működésre képes informatikai eszközök olcsó előállíthatóságával hozható összefüggésbe. A szenzorhálózatok önálló, egymással kommunikálni tudó eszközökből, szenzor csomópontokból állnak. Bár az egyes eszközök erőforrásai a leggyakrabban alkalmazott alacsony órajelű és kevés memóriával rendelkező mikrovezérlő, illetve az elemes táplálás miatt mind számítási kapacitás, mind pedig a rendelkezésre álló energia tekintetében erősen korlátozottak, az együttműködő eszközökből kialakított rendszer számos olyan feladat hatékony megoldására képes, ahol más mérési módszerek nem, vagy csak nehézkesen lennének alkalmazhatók.

A szenzorhálózatok alkalmazási területe rendkívül kiterjedt, a mezőgazdaságtól kezdve a környezetvédelmen át az egészségügyi, biztonságtechnikai, vagy akár védelmi alkalmazásokig számos különböző példával találkozhatunk. Az alkalmazások sokszínűsége ellenére számos, a felhasználó számára közvetlenül nem megjelenő szolgáltatást találhatunk, amit az alkalmazások nagy része megvalósít. Ilyenek például a kommunikációt, szinkronizációt, adattárolást, illetve lokalizációt megvalósító szolgáltatások. Ezeket közös néven *köztesréteg szolgáltatásoknak* nevezzük.

Dolgozatom 2-5. fejezeteiben különböző, kommunikációt megvalósító köztesréteg szolgáltatásokat mutatok be. A 2. fejezetben tárgyalt algoritmus egy elárasztáson alapuló forgalomirányítási módszer, ami az üzenetek eljuttatásához szükséges csomagok számának csökkentéséhez a perkoláció jelenségét használja fel. A módszer által nyújtott lefedettséget, illetve az elküldött csomagok számában mérhető javulást szimulációk segítségével mérem meg, az algoritmus helyes működését ideális, végtelen kiterjedésű, véletlenszerű elméleti hálózaton bizonyítom.

A módszert a 3. fejezetben egy körkörös, illetve többkörös hálózatokban használható, időosztásos többszörös hozzáférést (TDMA-t) használó algoritmus tárgyalása követi. Az algoritmus egyszerre biztosít időgarantált csomagtovábbítást, hibatűrést és energiahatékonyságot. A javasolt algoritmust valós hardveren implementálom, majd behatóan tesztelem.

A 4. fejezet egy olyan TDMA rendszer optimális ütemezését mutatja be, ahol egyszerre több, egymás vételét nem zavaró csomópont is adhat egyszerre. A problémára két algoritmust (OMTS és OMTS-A) mutatok be. Mind az algoritmusok által előállított TDMA ütemezés optimális voltát, mind az algoritmusok komplexitását matematikai módszerekkel bizonyítom, az utóbbit futási eredményekkel is alátámasztom.

Az 5. fejezetben egy olyan energiahatékony protokollt mutatok be, ami főleg egyirányú adatátvitelt támogat, de lehetőséget biztosít a fordított irányú kommunikációra is. Más megoldásokkal szemben a fordított irányban várakozó csomagok meglétének jelzése nem jelent többletfogyasztást. Az algoritmus kis energiafogyasztását az implementált rendszeren történő mérésekkel igazolom.

Az általam vizsgált másik köztesréteg szolgáltatás a lokalizáció, aminek célja egy objektum helyének meghatározása. A lokalizálandó objektumok lehetnek a szenzorhálózat csomópontjai, vagy egyéb tárgyak, tipikusan jelforrások is. A lokalizáció többféle fizikai jelenség alapján is történhet, így beszélhetünk például rádiófrekvenciás, illetve akusztikus lokalizációról. Dolgozatomban egy olyan akusztikus lokalizációs módszert vizsgáltam, aminek célja egy hangforrás helyének meghatározása. A sok lehetséges mérési módszer közül a beérkezési idők különbsége (TDoA) alapján működő konzisztencia-függvény alapú becsléssel kiemelkedően jó eredmények érhetők el. A 6. fejezetben a konzisztencia-függvény alapú akusztikus lokalizációs algoritmus továbbfejlesztéseit mutatom be. Először a becslőre egy adaptív módosítást javaslok, amely a régebbi mérések alapján a szenzorokhoz egy megbízhatósági értéket rendel, amit a becslésnél figyelembe vesz. Így a pozícióbecslés olyan esetekben is helyesen működik, amikor az irodalomban található módszer a konzisztensen rossz mérések nagy száma miatt hibás működést mutat. Az algoritmus működését valós mérésen alapuló példán demonstrálom. A fejezet második felében a lokalizáció gyorsítására egy véletlenített módszert mutatok be. A módszer a keresési teret egy olyan részhalmazra szűkíti le, amelyet megvizsgálva bármilyen megadott, az 1-et tetszőlegesen megközelítő valószínűséggel garantálható a globális optimum. A javasolt algoritmus működését analitikai módszerekkel és szimulációval is elemzem.

Abstract

Sensor networks first appeared when the low cost mass production of small and mobile intelligent devices became possible. Sensor networks consist of sensor nodes, autonomous devices capable of communicating with each other. Although the resources of a single device are highly limited in terms of both computational capacity and available power because of the low-end microcontroller, small memory, and the limited capacity of batteries, the network formed by the cooperating nodes can efficiently solve problems, where other measurement methods would be inefficient or impossible to use.

Sensor networks have wide range of applications, many examples from agriculture to environment protection, from healthcare to security and even military can be found in the literature. Despite of the diversity of the applications they share many similar services, which are invisible to the user. These services, called *middleware*, are responsible for e.g. communication, time synchronization, storage or localization.

In chapters 2-5 different middleware services are proposed for communication. In chapter 2 a modified flood routing algorithm is proposed which reduces the number of messages using the phenomenon of percolation. The algorithm is evaluated with simulations and the coverage and the gain in the number of transmitted messages are measured. The high coverage is mathematically proven in ideal, infinite, random theoretical networks.

In chapter 3 a TDMA (Time Division Multiple Access) algorithm is proposed, which provides guaranteed delivery time, fault tolerance and energy efficiency in ring topology networks. The proposed algorithm is implemented and tested on real hardware environment.

In chapter 4 two scheduling algorithms (OMTS and OMTS-A) are proposed for ring topology TDMA networks where multiple nodes are allowed to transmit at the same time if they don't collide. Both the optimality of the generated schedule and the computational complexity of the algorithm are mathematically proven. Computational complexity is evaluated also with test results.

In chapter 5 an asymmetric low-power communication protocol is proposed suitable for data collection applications. The protocol is designed for situations where the data flow is mainly unidirectional, but communication in the reverse direction is also supported. Unlike other common solutions the ability of the reverse communication does not require additional power consumption. The small energy consumption of the proposed method is demonstrated with measurements on the implemented system.

Acoustic localization is an interesting sensor networking service, where the goal is to determine the location of an acoustic source. There are many different approaches for solving this problem. The consistency function based estimator using Time Difference of Arrival (TDoA) measurements gives outstanding results. In chapter 6 of the thesis enhancements of the consistency function based estimator are presented. First a more

accurate localization method is proposed, where the performance of the sensors are monitored during the localization process and the calculated trustiness factors are taken into account during the calculations. This method leads to correct location estimation even in cases, when the original algorithm gives large error due to the presence of a large number of cooperatively bad measurements. The performance of the algorithm is evaluated with tests based on real measurements. In the second part of the chapter a statistical method is presented to accelerate the localization process. The algorithm reduces the search space to a subset, where the global optimum can be found with a probability arbitrary close to 1. The performance of the algorithm is analyzed with mathematical methods and evaluated with simulation.

Zusammenfassung

Die Erscheinung von Sensornetzwerken ist mit den kleinen, mobil Computer Geräten und mit deren günstigen Herstellung verbunden. Sensornetze bestehen aus autonomen Geräten, so genannten Sensorknoten, die miteinander kommunizieren. Die Ressourcen von den Geräten sind am häufigsten verwendeten Micro Controller, mit niedrigem Taktsignal und kleinem Arbeitsspeicher. Wegen dem Batteriebetrieb ist auch Rechenkapazität und auch die verfügbare Energie stark eingeschränkt. Das System, das aus den kooperativen Geräten entsteht, ist in der Lage eine Reihe von Aufgaben effektiv zu lösen, wo andere rechen Methoden nicht, oder nur gering eingesetzt werden können.

Die Anwendung von Sensornetzwerken ist außerordentlich Umfangreich. Wir können verschiedene Lösungen in der Landwirtschaft, Umweltschutz, Gesundheitspflege, Sicherheitstechnik, oder sogar Verteidigungsanwendungen finden. Trotz der Vielfalt der Anwendungen kann man eine Anzahl von Dienstleistungen finden, die nicht direkt dem Benutzer angezeigt werden, und wo die meisten Anwendungen schon implementiert sind. Dies sind die Dienste, wo Kommunikation, Synchronisation, speichern von Daten und Lokalisation ermöglicht werden. Dies wird auch als Middleware genannt.

In Kapitels 2-5 führe ich verschiede Middleware Dienste vor, die die Kommunikation realisieren. Der erste Algorithmus basiert auf einem Überschwemmung Verkehrsmanagement Protokoll, der die Zahl der Meldungen für die erforderlichen Anwendungspakete reduziert, durch den Einsatz von Versickerung. Die Abdeckung durch das Protokoll, und die Anzahl der Pakete, die eine messbare Verbesserung zeigen, stelle ich in Simulationen da. Den korrekten Betrieb des Algorithmus beweise ich durch ideal, unendlich dimensionale Zufallsnetzwerktheorie.

Im Kapitel 3 die Methode wird von einem TDMA Protokollverfahren gefolgt, das sowohl bei allseitigen, als auch bei mehrseitgien Netzwerken benutzt werden kann. Das Protokoll stellt sowohl garantierte Paketübertragungszeit, Fehlertoleranz und Energieeffizienz da. Das vorgeschlagene Protokoll wird auf einer echten Hardware implementiert, und auch getestet.

Kapitel 4 zeigt eine optimale Terminplanung für ein TDMA-System, wo mehrere Netzwerkknoten zur gleichen Zeit senden können, ohne Signalunterdrückung. Für dieses Problem führe ich zwei Algorithmen (OMTS und OMTS-A) vor. Beide TDMA Scheduling-Algorithmen durch die optimale Art und Komplexität der Algorithmen generiert beweisen mathematische Methoden, das letztgenannte unterstütze ich auch mit Ergebnisse.

Im Kapitel 5 stelle ich ein energieeffizientes Protokoll vor, das meist nur Einweg Datenübertragung unterstützt, aber auch die Möglichkeit anbietet in die umgekehrte Richtung zu kommunizieren. Im Gegensatz zu andere Lösungen in umgekehrter Richtung, die Warteanzeige von Paketen verursacht kein Mehrverbrauch. Den Energieverbrauch vom Algorithmus beweise ich mit Messungen im implementierten System.

Die von mir untersuchte Middleware Dienst ist die Lokalisation deren Ziel ist es den Ort des Objekts zu bestimmen. Die Objekte die lokalisiert werden sollten, können Sensornetzwerkknoten oder andere Gegenstände, typischerweise Signalquellen sein. Die Lokalisierung kann auf einer Vielzahl von physikalischen Phänomenen durchgeführt werden, so können wir über Lokalisierung mit Radiofrequenzen, oder über akustische Lokalisierung sprechen. In meiner Dissertation studierte ich ein akustisches Lokalisierungsverfahren, deren Ziel war es, die Position einer Schallquelle zu bestimmen. Von den vielen möglichen Messverfahren kann man das beste Ergebnis mit der Konsistenzfunktion erreichen, welche eine Schätzung abgibt basierend auf der Differenz zwischen den Ankunftszeiten (TDoA). Im Kapitel 6 zeige ich eine weiterentwickelte Konsistenzfunktion, der auf einer akustischen Lokalisationsalgoritmus basiert. Zunächst schlage ich eine adaptive Änderung vor, was den Schätzer betrifft, und auf den früheren Messungen Basierend zu den Sensoren einen Vertrauenswert angibt, die bei der Schätzung beachtet wird. So funktioniert die Position Schätzung auch richtig, wenn die in der Literatur angegebene Methode konsequent eine Fehlfunktion aufgrund der großen Anzahl von schlechten Messungen zeigt. Die Funktion des Algorithmus demonstriere ich mit der Hilfe einer realen Messung. Im 2. Teil des Kapitels zeige ich eine Methode der eine schnellere Lokalisation zeigt. Die Methode verengt das Suchraum auf eine Teilmenge, die bei einer Analyse mit jeder/aller Wahrscheinlichkeit das globale Optimum garantiert, was sich 1 nähert. Die Funktion des vorgeschlagenen Algorithmus wird durch Simulation und analytischen Verfahren analysiert.

Köszönetnyilvánítás

Köszönetemet fejezem ki témavezetőm, dr. Simon Gyula felé, aki mind a dolgozat alapját képező kutatás, mind pedig a dolgozat elkészítése közben iránymutatással és hasznos tanácsokkal látott el.

Köszönet illeti szüleimet, Böröcz Emilt és Böröczné Zakariás Máriát, akik doktori tanulmányaim alatt végig mellettem álltak és lelkileg támogattak.

Külön köszönet illeti a 3. fejezetben szereplő mérések során segítséget nyújtó kollégáimat, elsősorban Róth Gergőt és Orosz Ákost, valamint a 6. fejezet alapjául szolgáló mérések megosztásáért dr. Lédeczi Ákost.

1 Bevezetés

Az informatikai eszközök fejlődésének egyik legszembetűnőbb következménye az újabb és újabb egységek három fő paraméternek, a méretnek, az árnak, illetve az energiafogyasztásának rohamos csökkenése. A jelenség legismertebb megfogalmazása Gordon E. Moore-tól származik, ő egy negyedik fontos tényező, az integráltsági fok exponenciális növekedését jósolta meg [1]. A fentiek igen fontos velejárója az is, hogy időről időre újabb és újabb platformok születnek, melyek az eszközök újabb típusú használati módjainak kialakulását eredményezik. Ennek egyik fontos mérföldköve a személyi számítógépek, illetve az interaktív használati modell megjelenése volt.

1.1. Szenzorhálózatok

A szenzorhálózatok [2] megjelenése a kétezres évek elejére tehető, elsősorban annak köszönhetően, hogy lehetővé vált a kis méretű és alacsony fogyasztású mobil, egymással kommunikálni képes eszközök kedvező árú előállítása. Természetesen a használati modell is megváltozott, immár proaktív működésről beszélhetünk. Egy ilyen rendszerben az események nem a felhasználó direkt beavatkozásának hatására történnek; a rendszer a körülötte levő környezet bizonyos hatásait érzékeli és azokra reagál.

A szenzorhálózatok egymással kommunikáló szenzor csomópontokból állnak. Egy-egy csomópont korlátozott erőforrásokkal rendelkezik [1] [3]. Az eszközökön található feldolgozóegység jellemzően egy néhány MHz-es nyolcbites mikrovezérlő, kilobájtos tartományba eső RAM-mal, illetve flash memóriával. A kommunikáció szűk, jellemzően néhány száz kbps, vagy ennél is alacsonyabb sebességű, kis hatótávolságú rádió segítségével valósul meg. A csomópontok áramellátása leggyakrabban kis méretű szárazelem segítségével valósul meg, ami talán a legnagyobb kihívást jelenti.

Az eszközök egyszerűsége ellenére a csomópontok együttműködése során kialakult hálózat számos területen hatékonynak bizonyult, sőt olyan mérési feladatok esetén is eredményesen alkalmazható, ahol más eljárások nem, vagy nehézkesen lennének használhatók. Különösen azokban az esetekben érdemes szenzorhálózatos megoldást használni, amikor a mérési terület kiterjedt, vagy sok mérési pontból áll, illetve ha hosszú idejű, mobil, felügyelet nélküli mérésre van szükség. A szenzorhálózatok területén a hosszú, gyakran több hónapos, sok esetben akár több éves üzemidő, a robusztus, hibatűrő működés kulcsfontosságú kritériumok.

A szenzorhálózatok alkalmazási területe rendkívül szerteágazó, a mezőgazdaságtól kezdve a környezetvédelmen át az egészségügyi, biztonságtechnikai, vagy akár védelmi

[1] Az egyes méréseknél használt eszközök jellemzőit a megfelelő szakaszban ismertetem.

alkalmazásokig számos különböző példával találkozhatunk. A szenzorhálózatok egyik fontos felhasználási területe a vadállatok természetes élőhelyükön történő megfigyelése. Az ilyen jellegű projektek során a szenzor eszközöket általában az állatokra erősítik, így azok későbbi fizikai elérésére (például elemcserére) már nincs lehetőség. Az egyik leghíresebb projekt a ZebraNet [4]; a rendszer zebrák nyakára szerelt eszközök segítségével működött, melyek napelemmel voltak táplálva. Az összegyűjtött mérési adatokat az egymáshoz közel levő állatokon levő szenzorok egymással megoszthatták, tehát az egyes adatok minden eszközhöz eljuthattak. Az adatok letöltése természetesen rádión keresztül történt.

Természeti jelenségek megfigyelése is történhet szenzorhálózatok segítségével. A Glacsweb [5] projekt célja gleccserek megfigyelése volt, ami jó példa a szélsőséges körülmények közötti robusztus működésre. A jég a kommunikáció, az alacsony hőmérséklet pedig az energiaellátás kapcsán okozott nehézségeket. A [6] egy erdőtüzek megfigyelésére használt rendszert mutat be, aminek különlegessége, hogy a feldolgozás elosztottan, a hálózaton belül történik.

Nem csak természetes, hanem ember alkotta struktúrák megfigyelése is történhet szenzorhálózatok segítségével. A [7] például egy híd szenzorhálózattal történő megfigyelését írja le.

A precíziós mezőgazdaság a termőterületeknek jellemzően szenzorhálózatok segítségével történő megfigyelésén, illetve ezek alapján az egyes tevékenységek (öntözés, szüret) pontos hangolásán alapul. Tipikus példa a intelligens szőlőtermesztés [8].

A szenzorhálózatok egészségügyi alkalmazásai is széleskörűek kezdve a jelző- illetve felügyelő rendszerektől [9] a viselhető szenzorokon át [10] az implantátumokig [11].

A dolgozat egyik fő témáját is képező terület az akusztikus lokalizáció. A [12] egy védelmi alkalmazású lokalizációs rendszert mutat be, amit dolgozatom 6. fejezetében részletesen is bemutatok, illetve vele kapcsolatos javításokat eszközölök.

Egy futurisztikus elképzelés az intelligens por (smart dust) [13], ami homokszem méretű eszközök sokaságából áll, melyek észrevétlenül oldanak meg mérési, illetve beavatkozási feladatokat. A szenzorhálózati csomópontok rövidebb elnevezése is az angol mote (porszem) szóból ered.

1.2. A hardver fejlődése

Az „intelligens por” paradigma által keltett kezdeti lelkesedés ellenére az eszközök fizikai fejlődése nem volt látványos. Ugyan már léteznek kereskedelmi forgalomban kapható integrált rádió-mikrokontroller SoC (System on Chip) konfigurációk (pl. ATmega128RFA1 [14]), kis méretű MEMS (Microelectromechanical system) szenzorok, de ennek ellenére a porszem méretű szenzorok nem alakultak ki. Ennek legfőbb oka az energiatárolók kis energiasűrűsége, ebből eredően nagy mérete. A mai szenzorok energiatakarékos üzemmódban egy ceruza- vagy gombelem méretű áramforrás segítségével az alkalmazástól függően hetekig, hónapokig, vagy akár évekig is működhetnek, de a kis formafaktor (porszem) ma még nem realitás.

Az eszközök energiafogyasztása, ha nem is drámai módon, de folyamatosan csökken. A 2000-es évek elején alkalmazott tipikus mikrokontrollerek (pl. ATmega128L) fogyasz-

tása 36 mW / 40 µW (aktív/alvó üzemmód[2]), míg ma már elérhetőek ennek töredékét fogyasztó, nagyobb kapacitású eszközök is, például az MSP430 (2,6 mW / 1,1 µW)[3]. A rádiók hatékonysága is javult, például a CC1000 modell fogyasztása 50 mW / 29 mW (adás/vétel)[4] volt, míg egy mai korszerű rádió (pl. AT86RF212) fogyasztása 30 mW / 16 mW (adás/vétel)[5]. A fogyasztási adatokon kívül például a feléledési idők is mutatják a hardverek fejlődését: míg a CC1000 rádiónak 2-5 ms-ra volt szüksége, hogy alvó állapotból aktív állapotba kapcsoljon át, addig ez például az AT86RF212 rádiónak csak 900 µs ideig tart.

Mivel egy tipikus szenzorhálózati alkalmazásban a hálózati táplálás gyakran nem megoldható, a legtöbb esetben elemes vagy akkumulátoros energiaforrásokat alkalmaznak. A megújuló energiaforrások használatára számos kísérlet történt [15]. Napjainkban a legelterjedtebb megoldás a napelemek használata, melyek kis méretben is számottevő energiát tudnak termelni megfelelő környezetben. Az egyéb elektromágneses sugárzások kihasználása (például rádióhullámokból energia kinyerése) csak közeli adókkal reális (például RFID technológiák). A mozgási energiát elektromos energiává alakító berendezések (piezoelektromos átalakítók, generátorok) ugyan jó hatékonyságúak a termelt energia/méret mérték alapján, de nagy méretük csak speciális alkalmazásokban engedi meg használatukat [16]. Olyan alkalmazásokban, ahol nagy hőmérsékletkülönbségek fordulnak elő, termoelemek használata is lehetséges [17].

1.3. Operációs rendszerek

Beágyazott rendszerek számára számos operációs rendszert dolgoztak ki, pl. a uC/OS-II [18], az eCos [19], vagy a FreeRTOS [20]. Ezek általános célú beágyazott operációs rendszerek, melyek szenzorhálózatokban szintén alkalmazhatók. A speciális igények kiszolgálása érdekében azonban számos új operációs rendszer került kidolgozásra, melyek közül a legelterjedtebbek a következők:

- A TinyOS operációs rendszer központi elemei a modularitás és az esemény-alapú vezérlés. A programozói modell eseménykezelőket valamint ún. taszkokat tartalmaz, amelyek lényegében késleltetett függvényhívások. A későbbi verziók már tartalmaznak többszálú végrehajtáshoz is primitíveket. A modell jól illeszkedik a szenzorhálózatos alkalmazások esemény-orientált természetéhez, de azon feladatok megfogalmazása, melyek hagyományosan blokkoló utasításokra épülnek, a „split-phase" végrehajtási modell miatt nehézkesek [21].
- A Contiki operációs rendszer a C nyelvhez közeli programozói felületet kínál, a szolgáltatások széles halmazával. Az operációs rendszer támogatja a többszálú végrehajtást preemptív ütemezővel, tartalmazza a teljes TCP/IP vermet és támogatja az IPv6-ot is [22].

[2] 8 MHz-en
[3] MSP430F1611, 8 MHz-en
[4] 868 MHz, 1 mW kimenő teljesítmény
[5] 868 MHz, 3 mW kimenő teljesítmény

- A Mantis operációs rendszer szintén támogatja a többszálú végrehajtást és a hálózati kommunikációt, de leglényegesebb újdonsága a keresztplatfomos fejlesztés támogatása: a program tetszőleges platfomon (pl. PC vagy PDA) fejleszthető és tesztelhető, majd a kész alkalmazás portolható kisebb szenzoreszközökre [23].
- A LiteOS legfontosabb sajátsága, hogy a kis szenzorhálózati eszközökön biztosítja a Unix/Linux rendszereken megszokott szolgáltatásokat és programozói modelleket [24].

1.4. Energiatakarékos működés támogatása

Mivel a szenzorhálózatok alkalmazástechnikájának egyik legkritikusabb pontja a korlátozott energiaforrások kezelése, a hardver-támogatáson kívül számos szoftver-szolgáltatásba is szervesen beépült az energiatakarékosság támogatása. A hálózati rétegekben megjelentek az energiatakarékosságot szolgáló megoldások.

Számos energiatakarékos közeghozzáférési protokoll született. Az S-MAC megengedi, hogy a hálózati eszközök periodikusan alvó üzemmódba kapcsoljanak át, ezáltal csökkentve azok energiafogyasztását [25]. A B-MAC protokoll a szenzorhálózatokban gyakran jellemző alacsony adatforgalmat használja ki az energiahatékony működés támogatására [26]. Ennek egyik kulcseleme a több más protokoll által is hatékonyan alkalmazott alacsony energiájú figyelés (Low Power Listening, LPL) [27], amikor is a figyelő csomópont nagyon rövid időre felébredve mintavételezi a kommunikációs csatornát és ebből következtet az ébredés szükségességére. Az extrém kis fogyasztású hálózatokban gyakran alkalmaznak időosztásos közeghozzáférési technikákat is [28].

Az útvonalválasztási protokollok közül több is képes az energiafogyasztás figyelembe vételére az útvonal meghatározásakor. Pl. a GEAR protokoll az útvonal költségének számításakor figyelembe veszi az eszközök energiatartalékát is [29]. A LEACH protokoll fürtökre osztott hálózatban, a fürtön belüli időosztásos működés segítségével biztosít energiahatékony működést [30].

A mérési adatokban megjelenő redundanciák lehetővé teszik az energiatakarékosság támogatását az aggregáción keresztül. Az aggregációs algoritmusok lehetővé teszik az adatforgalom csökkentését statisztikai módszerekkel [31], Slepian-Wolf kódolással [29], vagy a legújabb irányzatok a tömörítő érzékelés (compressive sensing) segítségével [29].

Meg kell jegyezni, hogy az energiatakarékos működést számos esetben szinkronizációs mechanizmusok támogatják, melyek elengedhetetlenek a pontos időzítések megvalósításához.

1.5. Szinkronizáció

A közös időalap, az időbeli rendezhetőség követelménye számos alkalmazás eleme, például ez az időosztásos rendszerek alappillére is. Az első szenzorhálózatokra írott protokollok lefektették a jól alkalmazható szinkronizációs primitívek alapjait: a referencia

szórás (Reference Broadcast Synchronization) során egy referencia adó ad szinkronizációs üzeneteket, amelyet számos vevő egy időben vesz. Ezen vevők, kicserélve egymással adataikat, egymáshoz is tudnak szinkronizálódni [32]. A Timing-Sync Protocol for Sensor Networks (TPSN) ezzel ellentétben adó és vevő közötti szinkronizálást hajt végre, melyet a közeghozzáférési rétegbe épített precíz időbélyegző szolgáltatások támogatnak [33]. Ezen primitívekre számos sikeres protokoll épült: a FTSP a hálózatban egy referencia csomópontot választ, majd ehhez szinkronizálja a hálózat szomszédos elemeit, becsülve az offset és a drift mértékét is. A már szinkronizált csomópontok szintén referenciaként működve továbbterjesztik a szinkronizációt a hálózatban [34]. A Gradient Time Synchronization Protocol (GTSP) hasonló elvek mentén (offset és drift becslés) teljesen elosztott működést valósít meg [35]. A Glossy rendszer precíz szinkronizációt megvalósítva oly módon terjeszti a szinkronizációs üzeneteket, hogy a vett üzenetet több adó egyszerre adja tovább, kihasználva a konstruktív interferencia jelenségét [36].

1.6. Kommunikációs módszerek szenzorhálózatokban

Az adatok elterjesztésének igénye lényegében minden szenzorhálózatos alkalmazásban felmerül és összetett módon képes befolyásolni az alkalmazás egészének minőségét. A fontosabb minőségi mutatók a csomagok kézbesítési aránya, a késleltetés, a kihasználható sávszélesség, illetve az energiahatékonyság.

Az adott alkalmazás céljától, illetve a körülményektől függően többféle topológia, illetve a kommunikáció számos típusa elképzelhető. A leggyakrabban használt topológiák a következők.

- A csillag topológia [37] egyszerűsége és robusztussága miatt elterjedt. Számos alkalmazás esetében, például ipari környezetben a robusztus működés minden egyéb paraméternél fontosabb lehet, ilyenkor a csillag topológiájú hálózatok egyszerűségéből fakadó előnyök hatékonyan kihasználhatók. A módszer használatakor egyszerűvé válik a mérő csomópontok kis kitöltési tényezővel való üzemeltetése is.

- Ha a hálózat kiterjedése miatt az egy ugrásos kommunikáció már nem megvalósítható, egy gyakori továbblépési lehetőség a fa topológia. Ez a megközelítés főleg akkor lehet hasznos, ha a hálózatban kijelölhető egy olyan csomópont, ami a fa gyökerét fogja alkotni. Ez lehet egy forrás, vagy nyelő típusú csomópont (lásd lejjebb), vagy a hálózatot a külvilággal összekötő eszköz, mint például a 6LoWPAN esetében [38].

- A csillag topológia egy gerinchálózat segítségével nagyobb kiterjedésű rendszerré bővíthető. Egy ilyen hibrid hálózat többféle megközelítés előnyeit is ötvözheti, például egyszerre valósíthat meg többugrásos működést és lehet robusztus, illetve a mérő csomópontok kis kitöltési tényezővel való üzemeltetése ebben az esetben is egyszerűen megoldható. Ezt a működési elvet követi a ZigBee [39] is.

- Az általános gráffal leírható hálózati topológiát mesh-nek nevezzük [40]. Ezekben a hálózatokban az információ két csomópont között többféle útvonalon is eljuthat

és sok esetben a csomópontok is egyenrangúak.

- Igen nagy kiterjedésű hálózatban a rendszer hierarchikus kialakítása, klaszterek használata [41] egy célravezető út. Lényegében ez a megközelítés a hálózat egyfajta dekompozíciója.

Ezek alapján vegyük sorra a fontosabb működési módokat.

- A legáltalánosabb és egyben a legritkábban használt működési mód a bármely két eszköz közötti kommunikáció. Ez a teljesen elosztott működés során fellépő, csomópontok közötti információmegosztás egyik jellemző működése. Ez az általános megközelítés ad-hoc hálózatok esetén számos problémát vet fel, például a csomópontok altatása nehéz problémává válik, csak úgy, mint az útválasztás. Egyik megvalósítása a ZigBee-ben is használt AODV [42].

- Adatgyűjtő alkalmazásoknál a hálózat egy (bizonyos esetekben több) speciális, nyelő csomópontot tartalmaz, ahová a mérési adatok eljutnak. A megvalósítások leggyakrabban gradiens módszert, vagy feszítőfát alkalmaznak, például a DFRF [43] mindkettő használatára lehetőséget biztosít.

- Gyakori működési mód az üzenetszórás (broadcast), ami az adatoknak egy rögzített, vagy akár több különböző csomópontból az összes többi csomópontba való eljuttatását jelenti. Az üzenetszórásnak többféle célja is lehet, például a hálózat felébresztése [44], konfigurációs adatok, illetve parancsok küldése [45], vagy a hálózatban szereplő csomópontokon futó program frissítése [46].

Az egyes gyakorlati megvalósítások jellemzően az adatkapcsolati rétegben (azon belül is általában a közeghozzáférési alrétegben), vagy a hálózati rétegben működnek, illetve gyakoriak az olyan megvalósítások is, amik e két réteget lényegében egy egészként kezelik. Fontos még megemlíteni a két rétegen kívül egy harmadik tényezőt is. A kommunikáció természetes feltétele ugyanis az, hogy mind az adó, mind a vevő rádiója egy időben legyen bekapcsolt állapotban. Az energiatakarékos protokollok a rádiók jól szervezett be- illetve kikapcsolásával tudnak hatékonyan energiát spórolni. Vannak olyan protokollok is, amelyek ezt a kérdést nyitva hagyják; ezekben az esetekben vagy feltehetjük, hogy a rádiók folyamatosan bekapcsolt állapotban vannak, vagy azok be- és kikapcsolását a felsőbb rétegekre bízzuk. A következőkben néhány gyakorlati kommunikációs protokollt sorolok fel.

A közeghozzáférési rétegek kiindulási alapjának az ALOHA protokollt tekinthetjük. Itt bármiféle megkötés nélkül bármikor bármely csomópont adhat. Az ALOHA sem a csomagok ütközésével, sem pedig az energiafogyasztás optimalizálásával nem törődik. Ezt a protokollt viszonylag rossz teljesítőképessége miatt ritkán használják, más, komplexebb protokolloknak azonban gyakran részét képezi.

A CSMA/CA közeghozzáférési réteg az ütközések elkerülésére csatornafoglaltságfigyelést, foglalt csatorna esetén pedig véletlenszerű várakozásokat használ. Az általam vizsgált szenzorhálózatok céljára kifejlesztett operációs rendszer, a TinyOS [21] rádió

kommunikációs alrendszere a CSMA/CA felett egy egyszerű kommunikációs primitívet (ActiveMessage) biztosít, ezzel szabad kezet adva a felhasználónak. A rádió ki- illetve bekapcsolása a felhasználó feladata, maga a CSMA/CA energiatakarékosságot támogató funkciókat nem tartalmaz. A véletlenszerű várakozások megnehezítik egy, a CSMA/CA-ra épülő, rövid aktív periódusokat használó energiatakarékos protokoll működését, ahogy azt az 5.3.1. szakaszban látni fogjuk.

Egy energiahatékonyabb közeghozzáférési protokoll az S-MAC [25], ami egy szinkronizált időablakot, úgynevezett versengési ablakot használ, ahol a hálózat összes csomópontja egyszerre ébred fel egy fix időtartamra, az ezen kívüli részt pedig alacsony energiafelvételű (alvó) állapotban töltik. A versengési ablakon belül a kommunikáció CSMA/CA protokollal történik. Az S-MAC fix időablaka alacsony hálózati terhelés esetén az energiával, magas terhelés estén pedig a rendelkezésre álló sávszélességgel kapcsolatban pazarló. A módszer lehetséges javításai a D-MAC [47], illetve a T-MAC [48]; mindkét protokoll a terhelés függvényében, dinamikusan változtatja a versengési ablak hosszúságát.

Az adatáramlás számos szenzorhálózatos alkalmazásnál aszimmetrikus. Számos adatgyűjtő alkalmazás használ korlátozott energiával rendelkező (jellemzően elemmel táplált) mérő csomópontokat és hálózati táplálású központi csomópontot (csillag topológia esetén), vagy gerinchálózatot (hibrid topológia esetén). Ekkor a mérő csomópontok rádiói csak az adatküldések idejére vannak bekapcsolva. Gyakran azonban az ellenkező irányban is szükség lehet kommunikációra, például parancsok, vagy konfigurációs üzenetek küldésére. Az egyik lehetséges megoldás a problémára a Low Power Listening (LPL) [49]. Itt a mérő csomópont bizonyos időközönként bekapcsolja rádióját és egy rövid ideig hallgat, a központi csomópont pedig megpróbálja eltalálni ezt az ablakot az adásával. A hálózat szinkronizációjának hiánya miatt az adó nem tudja, hogy a vevő mikor van bekapcsolva; a problémát megnövelt hosszúságú preambulumot alkalmazó csomagokkal hidalja át. A csomag vétele akkor lesz sikeres, ha a vevő felébredése a hosszú preambulum idejére esik. A megoldás hátránya a nagy méretű preambulumok által elpazarolt sávszélesség és energia.

Egy másik megközelítés a RI-MAC (Receive Initiated MAC) [50], ahol a szenzor csomópont a rendszeres passzív hallgatás helyett rövid csomagokat küld, ezzel jelezve a vételkészségét. Így elmarad a sávszélességnek az LPL esetében tapasztalt szükségtelen pazarlása. A RI-MAC egy javításának tekinthető az A-MAC [51], ami adaptív működéssel, például a szomszédos csomópontok ütemezésének megtanulásával egészíti ki a RI-MAC-et, így jelentős energiamegtakarítást érve el.

Áttérve a hálózati rétegre, talán a legegyszerűbb forgalomirányítási algoritmus az elárasztás [52]. Ez az egyszerű algoritmus számos szempontból nem bizonyul hatékonynak, de számos összetettebb protokoll alapját képezi. Néhány példa az irányított elárasztás [43], a vezérelt elárasztás [53], a tiny ad-hoc routing [54], vagy az AODV [42], ami a ZigBee forgalomirányítási protokollja is [39].

A Collection Tree Protocol (CTP) [55] fa topológiájú hálózatokban valósít meg adatgyűjtést. Többféle MAC réteg felett képes hatékonyan működni és magas kézbesítési arányt nyújtani.

A DFRF [43] is egy tisztán hálózati rétegbeli protokoll, ami egy központi csomópont és a hálózat többi része között továbbít üzeneteket, mindkét irányban. A központi csomó-

ponttól eredő csomagok elárasztással, a hálózat csomópontjaiból eredők gradiens módszerrel, vagy feszítőfa segítségével kerülnek a gyűjtő csomópontba.

Az időosztásos többszörös hozzáférés (Time Division Multiple Access, TDMA) esetében az egyes csomópontok számára kizárólagosan használható időszeleteket biztosítunk. Így az ütközések elkerülhetők, illetve energiahatékony működésre és determinisztikus üzenetközvetítésre van lehetőség. A TDMA ütemezésének megszervezése összetett feladat, ezért bár maga a TDMA alapvetően egy közeghozzáférési elv, egy ilyen rendszer működtetéséhez a felsőbb rétegek támogatására is szükség van.

A Dozer [56] egy olyan, TDMA alapú protokoll, ami az időszeletek kiosztását dinamikusan végzi, így jelentősen csökken a TDMA rendszerekben tapasztalható komplexitás. A Dozer esetében azonban a determinisztikus csomagtovábbítás nem valósul meg.

Az [57] a szenzorhálózattokban kialakítható TDMA rendszerek kialakíthatóságát vizsgálja. A szenzorhálózatokban használható TDMA protokollokról az [58] ad átfogó képet.

1.7. Lokalizációs módszerek szenzorhálózatokban

A lokalizációkat alapvetően két nagy csoportba sorolhatjuk. Számos szenzorhálózatos alkalmazásnál elengedhetetlen az egyes szenzorok pozíciójának ismerete, a mérési adatok ugyanis a legtöbb esetben csak a forrás ismeretében hordoznak hasznosítható információt. A lokalizációnak ezt a típusát, tehát amikor a szenzorhálózat az egyes, a hálózatot alkotó csomópontok helyzetét határozza meg, önlokalizációnak nevezzük. Az önlokalizációt megvalósító rendszerek az egyes szenzorok egymáshoz képesti relatív helyzetét határozzák meg, azonban ismert pozíciójú (horgony) csomópontok használatával az abszolút koordináták kiszámítása is lehetséges. A horgony csomópontok helyzetének meghatározására többek közt GPS-t, vagy ismert helyzetű referencia pontokat használnak.

A lokalizációs módszerek másik csoportjába azok az esetek tartoznak, amikor a szenzor csomópontok helyzete ismert, és a cél egy, vagy több külső objektum lokalizálása.

A pozícióbecslési lépés a jellemzően a szenzorok egymástól, vagy az lokalizálandó objektumtól vett távolsággal, vagy bizonyos szögekkel összefüggésbe hozható mérési adatok alapján dolgozik.

A távolsággal összefüggésben levő egyik fizikai mennyiség a (jellemzően akusztikus, vagy rádiófrekvenciás) jel terjedési ideje, ami többféle módon is mérhető. A beérkezési idők (Time on Arrival, ToA) alapján működő lokalizációk esetén a lokalizálandó objektum egy jelforrás, a szenzorok pedig a jelek beérkezési idejét mérik. A ToA azonban feltételezi, hogy a jel kibocsájtási ideje ismert, ami az esetek nagy részében nem biztosítható. A ToA módszernek ezt a hibáját a beérkezési idők különbsége (Time Difference on Arrival, TDoA) alapján működő rendszerek küszöbölik ki.

A mért jel szempontjából rádiófrekvenciás, illetve akusztikus (hangjelek segítségével működő) rendszerekről beszélhetünk. A rádiófrekvenciás jel terjedési ideje annak nagy sebessége miatt nehezen mérő kielégítő pontossággal, azonban ilyen elvű rendszerre is található példa. Az [59] egy 802.11 szabványra, tehát a WiFi-re épülő alkalmazást mutat be.

Egy hatékony akusztikus lokalizációs rendszert mutat be a [12]. Ez a megoldás a lő-

fegyverek torkolati zajának beérkezési idejét méri. A pontos lokalizációhoz nagyban hozzájárul a speciális hangjel meredek felfutása, aminek köszönhetően a beérkezési idő igen pontosan mérhető. Akusztikus lokalizációs rendszer nem csak levegőben, hanem vízben terjedő hanghullámokkal is megvalósítható. A [60] egy víz alatti lokalizációs megoldást mutat be.

Egy másik, a jel által megtett távolsággal összefüggésben levő mérhető adat a jel vételi jelszintje (RSSI, Received Signal Strength Indication), azonban míg a távolság és a terjedési idő között egyszerű lineáris kapcsolat áll fenn, az RSSI esetében hasonlóan egyszerű összefüggés nem fogalmazható meg. Az RSSI elvű lokalizációs rendszerek vitathatatlan előnye azonban, hogy használatuk például a meglévő WiFi hálózat jeleinek passzív mérésével is lehetséges [59].

A lokalizálandó jelforrás irányát használják fel az iránydetekción alapuló (AoA, Angle of Arrival) rendszerek. Az AoA elv hátránya, hogy használatához drága és nagy méretű irányérzékeny eszközökre (antennákra, vagy mikrofonokra) van szükség. Szenzorhálózatos megvalósítása emiatt nem gyakori, a [61] egy ilyen típusú rendszert mutat be.

A szenzorhálózatos lokalizációs rendszerekről a [62] irodalom ad átfogó képet.

1.8. A dolgozat struktúrája

Dolgozatom törzsét öt fejezetre osztottam. A 2-5. fejezetekben különböző, a kommunikáció témakörébe illeszkedő, a 6. fejezetben pedig lokalizációt megvalósító köztesréteg szolgáltatásokat mutatok be.

A 2. fejezetben egy adaptív elárasztási algoritmust javaslok, ami az egyszerű elárasztáshoz képest jelentősen képes csökkenteni az üzenetszámot, közel teljes lefedettség megtartása mellett. A javasolt algoritmus képes mind a változó hálózati sűrűséghez, mind pedig a hibás kapcsolatokhoz adaptálódni. Az algoritmus jó tulajdonságait szimulációval és elméleti bizonyítással is alátámasztom.

A 3. fejezetben két időosztásos többszörös hozzáférést (TDMA) megvalósító algoritmust mutatok be. Mindkét algoritmust valós eszközökön implementáltam, a szakaszban az elvi működés mellett a megvalósítás részleteit is tárgyalom. A hibatűrő és robusztus működést, a csomagok átvitelének időszükségletét, illetve az alacsony energiafogyasztást mérésekkel igazolom.

A 4. fejezetben egy olyan TDMA ütemezési algoritmust közlök, ahol egyszerre több eszköz is adhat, amennyiben az adások a csomagok vételénél nem okoznak interferenciát. A rendszer modelljét gráfelméleti eszközökkel formalizálom, az optimális (legrövidebb periódusidejű) ütemezés meghatározásra egy algoritmust mutatok be, aminek helyességét a modellt felhasználta igazolom. Az algoritmus futási idejére aszimptotikus korlátot definiálok, amit tesztekkel is alátámasztok. Az algoritmus által létrehozott ütemezés fontos tulajdonságait analitikai módszerekkel határozom meg. A kézbesítési időben tapasztalható javulásra felső korlátot, az energiafogyasztásra pontos értéket adok meg.

Az 5. fejezetben egy olyan protokollt mutatok be, ami a nagyrészt egyirányú kommunikációt használó (például adatgyűjtő) alkalmazások számára nyújt lehetőséget ritka, visszafelé irányuló kommunikációra energiatakarékos módon. A protokoll a visszafelé

kommunikáció támogatásához módosított nyugtákat használ, így más módszerekkel ellentétben ehhez a művelethez nem, kizárólag magához az adatközléshez igényel többlet energiát. Az összehasonlításhoz egy másik, naiv programozott protokollt is készítettem. Mindkét protokollt valós eszközökön implementáltam, majd összehasonlító méréseket végeztem.

A dolgozatom törzsének másik felét tartalmazó 6. fejezet az akusztikus lokalizációval kapcsolatos eredményeimet írja le. Munkám alapja a [12]-ben leírt konzisztencia-függvény alapú módszer. Az áttekintés után a 6.2. alfejezetben a konzisztencia-függvény alapú lokalizáció működését foglalom össze.

A 6.3. alfejezetben a [12]-ben bemutatott konzisztencia-függvény alapú becslőre javaslok egy pontosabb módszert. A bemutatott módszer az egyes becslések során felméri a szenzorok megbízhatóságát, a későbbi becslések során ezt az információt felhasználva képes elkerülni a konzisztensen hibás mérésekből fakadó hibás működést.

A 6.4. alfejezetben egy statisztikai módszert használok a konzisztencia-alapú pozícióbecslés gyorsítására. Az algoritmus képes 1-hez tetszőlegesen közeli, megadott valószínűséggel helyes becslőt adni. Az algoritmus hatékonyságát bizonyítom, amit valós méréseken alapuló szimulációval is alátámasztok.

Munkámat a 7. fejezetben összegzem, végül eredményeimet a 8. fejezetben magyar, a 9. fejezetben angol nyelven tételesen is felsorolom.

2 Perkolációt eredményező elárasztás

2.1. Áttekintés

Ebben a fejezetben egy forgalomirányítási algoritmust javaslok, ami az elárasztáson alapul, azonban kiküszöböli annak fő hátrányát, a feleslegesen elküldött nagy számú csomagot. Emellett megtartja az eredeti algoritmus jó tulajdonságait: az elérhető nagy lefedettséget és az egyszerű működést.

Mivel a következőkben ismertetésre kerülő algoritmus az elárasztáson alapul, először ennek működését ismertetem. Amikor egy csomópont üzenetet szeretne küldeni a hálózat összes többi csomópontja számára, először a szomszédainak küld egy broadcast üzenetet. Amikor egy csomópont egy üzenetet először vesz, továbbküldi azt a szomszédainak. Az üzenet később megkapott példányai eldobásra kerülnek. Az ehhez hasonló elveken nyugvó algoritmusokat gyakran pletyka alapúnak is nevezik.

Az elárasztás főleg üzenetszórás esetén hasznos, de emellett útvonal-feltérképezésre is használható. Az elárasztáson alapuló útvonal-feltérképező algoritmusok egy kétlépcsős módszert használnak egy forrás és egy cél csomópont közötti útvonal keresésére. Először a forrás a hálózatot egy felfedezés-kérés üzenettel árasztja el, majd a cél csomópont egy választ küld vissza a forrásnak és közben a csomagban feljegyzésre kerülnek az érintett csomópontok. Így a hálózat megtanulja a forrástól a célig vezető utat és a későbbi kommunikációhoz már ezt használja. Ebben a kontextusban a felfedezett út minősége – első megközelítésben annak hossza – is egy fontos teljesítménymérő kritérium.

Az egyszerűség mellett az elárasztás másik nagy előnye a robusztusságban rejlik. Az algoritmusban rejlő implicit redundancia nagy arányú csomagvesztés, illetve a csomópontok meghibásodása esetén is védelmet ad. Az algoritmus hátránya az elküldött csomagok nagy száma, amit üzenetszórási viharnak (broadcast storm) is neveznek: a csomópontok akkor is továbbküldik a kapott üzenetet, ha ez nem lenne szükséges. Egy sűrű hálózatban a szükségtelenül nagy mennyiségű üzenet gyakori csomagütközést (ezáltal teljesítménycsökkenést) okoz, és az energiafelhasználásra nézve is pazarló.

A következőkben egy, a perkoláció jelenségén alapló, statisztikai megközelítésű megoldás kerül bemutatásra. A javasolt algoritmus a hálózati topológiához adaptálva a továbbküldési valószínűséget automatikusan állítja be úgy, hogy ehhez csak lokális információt használ. A javasolt algoritmus magas lefedettséget nyújt kis üzenetszám mellett, illetve képes dinamikusan igazodni a hálózati topológia tulajdonságaihoz. Az algoritmus az egyszerű elárasztásnál alig összetettebb, így kevés erőforrással rendelkező eszközökön is használható.

Az algoritmus működésének alapötlete az, hogy a csomópontok véletlenszerű döntést hoznak arról, hogy egy üzenetet továbbküldjenek-e, vagy sem. A véletlenszerű elárasztásnak számos változata ismert. A legegyszerűbb megoldás szerint a csomópontok az üzene-

teket az első megkapás után p valószínűséggel adják tovább és $1-p$ valószínűséggel dobják el. Egyazon üzenet további azonos példányait az algoritmus ettől a döntéstől függetlenül eldobja. Természetesen a $p = 1$ választása esetén az egyszerű elárasztáshoz jutunk. A teszteredmények azt a megérzést támasztják alá, hogy nagyobb p értékek nagyobb lefedettséget biztosítanak. Továbbá egy kellően sűrű véletlenszerű hálózatban az algoritmus egy érdekes jelenséget mutat. Ha $p \geq p_c$, ahol a p_c a topológiától függő kritikus valószínűség, akkor az üzenetek gyakorlatilag a hálózatban szereplő összes csomóponthoz elérnek. Ellenkező esetben az üzenet a hálózatnak csak egy kis részéhez jut el. Az optimális eset jól láthatóan $p = p_c$ lenne. Az algoritmus változtatásával a teljesítmény növelhető, például az üzenetek korai elhalását p értékének az ugrásszám függvényében történő változtatásával lehet csökkenteni; a forráshoz közeli csomópontok magasabb p-t használhatnak, a messzebb levők pedig kisebbet. A véletlenszerű elárasztási algoritmusokról Haas és társai [52] cikkében olvashatunk bővebben. Ezen algoritmusok közös problémája, hogy a tervparaméterek (az alkalmazott valószínűségek) a használt hálózati elrendezéstől függenek, különösen a változó sűrűségű hálózatokban mutatnak optimális alatti teljesítményt.

Bizonyos összetettebb algoritmusok ezt a problémát is hatékonyan kezelik. A pozícióalapú algoritmusok a csomópontok helyzetét használják fel a csomagok továbbküldésének optimalizálására [63]. A [64]-ben közölt algoritmus elosztott módon határoz meg lefogó halmazt a legfeljebb két ugrásra levő csomópontokra vonatkozó információt felhasználva. Ezek a módszerek változó sűrűségű hálózatokban is jól működnek, de jóval komplexebbek is.

A perkolációelmélet eredményeit a vezeték nélküli hálózatokban is alkalmazták, megalkotva ezzel a kontinuum perkoláció elméletét [65].

A [66] realisztikusabb hálózati modelleket (például nem megbízható kapcsolatokat és anizotrop sugárzási mintákat) tárgyal. A [66] egyik fontos eredménye az egyszerű kör modell érdekes tulajdonságának kimutatása: kör modell esetén nehezebben alakul ki perkoláció, mint más sugárzási mintázatok esetén, így a kör modell legrosszabb esetet leíró modellnek tekinthető. Ez a tulajdonság igazolja a modell létjogosultságát a csomópontok kapcsolatának leírása esetén.

A szenzorhálózatokat gyakran modellezik Poisson Boolean modellel. Ebben a modellben a csomópontok a síkon egy Poisson pont folyamat által meghatározott módon vannak elhelyezve λ sűrűséggel, és minden csomópont egy fix r kommunikációs hatósugárral rendelkezik (kör modell). A Poisson pont folyamat jól közelíti a szabályozott, de véletlenszerű elhelyezéseket. Valós körülmények között egy szenzor csomópont sugárzási mintázata anizotrop, amit a kör modell rosszul közelít. Azonban a kör modell egy legrosszabb esetet leíró modellnek is tekinthető, mert más kommunikációs modellek hasonló körülmények között egyszerűbben vezetnek perkolációhoz [67]. Elméleti eredményekkel igazolták, hogy ha egy Poisson Boolean hálózatban az adó csomópontok sűrűsége (vagy más megközelítésben azok teljesítménye) elér egy bizonyos λ_c küszöbértéket, akkor annak a valószínűsége, hogy a hálózatban egy végtelenül nagy komponens keletkezik, 1 lesz, vagyis a hálózat perkolál [66, 68]. Ellenkező esetben, tehát ha $\lambda < \lambda_c$, akkor 1 valószínűséggel minden, a hálózatban szereplő komponens véges méretű. Tehát a perkoláció egy hálózat fontos tulajdonsága nagy távolságú több ugrásos kommunikáció igénye esetén.

A [68] egy elosztott minimum-fokszám szabályt mutat be a hálózat egyenetlenségei-

nek kiküszöbölésére. Itt a teljes hálózat lefedése érdekében az egyes csomópontok adási teljesítményét úgy állítják be, hogy minden csomópont szomszédainak száma egy bizonyos küszöb felett legyen. A [69] egy olyan energiamegtakarítási módszert tárgyal, ahol a hálózati csomópontok úgy kapcsolódnak ki, illetve be, hogy a hálózatban megmaradjon a perkoláció. A módszer által alkalmazott egyszerű szabály az, hogy a bekapcsolt csomópontok arányát mindig $\frac{\lambda}{\lambda_c}$ felett tartja. Ez az algoritmus rosszul skálázható, mivel minden csomópontnak ismernie kell a λ paramétert. A [70] ennek a módszernek egy továbbfejlesztését mutatja be, ahol egy, a gyakorlatban is használható elosztott algoritmussal határozzák meg az alvás és az ébrenlét arányát. Ez a módszer csak a lokális sűrűséget, nevezetesen a szomszédok számát használja fel. Egy k fokú csomópontra az η_k ébrenléti arányt a következő képlettel számítja ki:

$$\eta_k = \begin{cases} \frac{\varphi}{k} & \text{ha } k > \varphi \\ 1 & \text{ha } k \leq \varphi \end{cases}, \tag{2.1}$$

ahol φ egy sűrűségfüggetlen tervparaméter. A javasolt perkolációt eredményező elárasztás egy ehhez hasonló ötletet használ. Az általam javasolt eljárás hasonlóan a [70]-hez a szenzorok sűrűségét használja fel a vezérléshez. A [70] irodalomban közölt eredmények egységnyi kommunikációs hatósugarú, véletlenszerűen ki- és bekapcsolt elemeket tartalmazó hálózatokra vonatkoznak. A következőkben ezen eredményeket használom fel tetszőleges hatósugarú, állandóan ébren levő, de az általam javasolt vezérlést alkalmazó elemeket tartalmazó hálózatokban.

2.2. A perkolációt eredményező elárasztási algoritmus

Az algoritmus alapötlete a következő. Használjunk véletlenített elárasztást és minden csomópont esetén úgy állítsuk be a továbbküldési valószínűséget, hogy az üzenet a hálózat legnagyobb részéhez elérjen. Az algoritmus a következő:

1. A forrás az üzenetet 1 valószínűséggel küldi el a szomszédainak.

2. Az n sorszámot viselő csomópont az üzenetet annak első megkapása után P_n valószínűséggel küldi tovább és $1 - P_n$ valószínűséggel eldobja azt. Az üzenetek későbbi példányai szintén eldobásra kerülnek. A P_n valószínűséget a következő képlet adja meg:

$$P_n = \begin{cases} \frac{K_{min}}{K_n} & \text{ha } K_n > K_{min} \\ 1 & \text{egyébként} \end{cases}, \tag{2.2}$$

   ahol K_n az n sorszámmal rendelkező csomópont fokszáma (azaz a szomszédainak száma), és K_{min} tervparaméter egy küszöbértéket jelöli. Két csomópont akkor tekinthető szomszédosnak, ha hallják egymást, tehát ha egy szimmetrikus kapcsolat köti őket össze.

Az algoritmus működéséhez először meg kell határozni a szomszédok számát, ami azok felderítésével történik. Minden csomópont `HELLO` üzeneteket küld, illetve fogad; ennek segítségével a fokszám egyszerűen meghatározható. A gyakorlati alkalmazásokban K_{min} értékét közelítőleg 7-re érdemes választani, ahogyan azt a szimulációk során látni fogjuk.

Az algoritmus elméleti, végtelen kiterjedésű, Poisson Boolean folyamat által generált hálózatokon való helyes működésének bizonyításához a következő, négyparaméteres modellt használom a hálózatok jellemzésére. A λ a hálózat sűrűségét, tehát az egységnyi területen levő szenzorok számát írja le. Az r a kommunikációs hatósugarat jelöli. Mindkét adat mértékegység nélküli, relatív mérőszám. A hálózat elemei szakaszos működésre képesek, tehát az időnek csak bizonyos részében aktívak; 1 időegységre vetítve η ideig vannak bekapcsolt, $1-\eta$ ideig pedig kikapcsolt állapotban. Továbbá csomópontok dönthetnek arról is, hogy a megkapott csomagot továbbadják-e, vagy sem. A P valószínűségi paraméter a továbbadás valószínűségét jelöli. A csomagok későbbi példányai a továbbküldésükről hozott első döntéstől függetlenül eldobásra kerülnek. A fenti paraméterekkel leírható hálózatra a $G(\lambda, r, \eta(.), P(.))$ jelölést használom.

A fenti modellel jellemzett hálózatokra tekintsük a [70] irodalomban közölt egyik fontos eredményt, amit a következő lemma ír le.

2.2.1. Lemma. Adott egy $G(\lambda, 1, 1, 1)$ paraméterekkel leírt hálózat, ahol $\lambda > \lambda_c$, és λ_c az a kritikus sűrűség, ahol a perkoláció először bekövetkezik. Ekkor létezik olyan λ-tól független $2 < \varphi < \infty$, hogy ha minden csomópont

$$\eta = \begin{cases} \frac{\varphi}{K} & \text{ha } K > \varphi \\ 1 & \text{ha } K \leq \varphi \end{cases}, \tag{2.3}$$

valószínűséggel adja tovább a fogadott csomagot, akkor az így nyert $G(\lambda, 1, \eta(.), 1)$ paraméterekkel leírt hálózat is perkolálni fog. K az adott csomópont fokszámát jelöli. A bizonyítást lásd a [70] irodalomban.

A 2.2.1. lemma tehát azt mondja ki, hogy egy 1 hatósugarú, folyamatosan aktív és a vett csomagokat minden esetben továbbadó csomópontokat tartalmazó perkoláló hálózatban a csomópontokhoz hozzárendelhető egy olyan K-tól függő, de λ-tól független ébrenléti arány úgy, hogy a hálózat továbbra is perkoláljon. A lemma felhasználásával már kimondható, illetve bizonyítható az általam vizsgált hálózatokra vonatkozó tétel.

2.2.1. Tétel. Ha egy Poisson Boolean folyamat által λ sűrűséggel és r kommunikációs hatósugárral előállított, folyamatosan aktív és minden üzenetet továbbadó csomópontokkal rendelkező $G(\lambda, r, 1, 1)$ végtelen hálózatban minden üzenet 1 valószínűséggel ér el végtelen sok csomóponthoz, akkor *létezik* λ-tól független $K_{min} < \infty$ tervparaméter úgy, hogy ha a minden csomópont a (2.2) egyenlet által meghatározott valószínűséget használva adja tovább a megkapott csomagot, akkor az így kapott $G(\lambda, r, 1, P(.))$ hálózatban is minden üzenet 1 valószínűséggel fog végtelen sok csomóponthoz elérni.

Bizonyítás
A bizonyítás két lépésben történik. Először a hálózaton egy térbeli transzformációt hajtunk végre, majd egy működési transzformációt. Ezáltal visszavezetjük a problémát a 2.2.1. lemmára.

Jelölje a hálózatot reprezentáló gráfot a fent definiált négyparaméteres modellben $G(\lambda, r, 1, P(.))$. A hálózat r-szeres kicsinyítésével a sűrűség r^2-szeresére változik, ami most $\lambda' = \lambda r^2$ alakban írható, míg a kommunikációs hatósugár 1 lesz. A skálázás a csomópontok aktívan töltött idejére, illetve a továbbadási valószínűségre nincs hatással, vagyis a harmadik és negyedik paraméter változatlan marad. A hálózat viselkedését ez a művelet nem változtatja meg, tehát a $G(\lambda', 1, 1, P(.))$ hálózatra kimondott állítás vonatkozik a $G(\lambda, r, 1, P(.))$ hálózatra is.

A bizonyítás második felében a működési módot transzformáljuk a következő módon. Ahelyett, hogy egy csomópont esetében a megkapott üzenetnek a (2.2) szerinti továbbküldéséről, vagy eldobásáról beszélünk, azt is mondhatjuk, hogy az üzenet megkapása előtt ugyanezen egyenlet által meghatározott valószínűséggel tartjuk bekapcsolva az adott csomópontot, így a $G(\lambda', 1, 1, P(.))$ hálózathoz jutva. Erre a hálózatra már alkalmazhatjuk a 2.2.1. lemmát, ami kimondja, hogy létezik olyan $K_{min} = \varphi < \infty$ tervparaméter, hogy G perkolál. A bizonyítás első fele miatt azonban ez az állítás igaz lesz a $G(\lambda, r, 1, P(.))$ hálózatra is. Következésképpen az elküldött üzenetek ebben a hálózatban is 1 valószínűséggel jutnak el végtelen számú csomóponthoz. ∎

A valóságban létező véges méretű hálózatok esetén a 2.2.1. tétel azt jelenti, hogy K_{min} egy alkalmas választása esetén minden üzenet a hálózatnak gyakorlatilag az összes csomópontjához elér. Az algoritmus további tulajdonságait szimuláció segítségével elemzem.

2.3. Értékelés

Ebben a szakaszban az algoritmus különböző metrikák szerinti teljesítményének értékelésére kerül sor szimulációs eredmények alapján. Először a teljesítménymérő metrikák, valamint a hálózati és kommunikációs modellek kerülnek ismertetésre, majd a szimulációs környezet, illetve a tesztbeállítások. Végül a javasolt algoritmus viselkedését demonstráló szimulációs eredmények kerülnek bemutatásra.

2.3.1. Teljesítménymérő metrikák

A legfontosabb mérőszámok egy üzenetszórásos algoritmussal kapcsolatban az üzenetet megkapó csomópontok aránya és az elküldött üzenetek száma. Ha a hálózatban szereplő csomópontok száma N, az üzenetet megkapók száma pedig N_{rec}, akkor a lefedettséget a

$$C_r = \frac{N_{rec}}{N} \tag{2.4}$$

adja meg. A jó szolgáltatásminőséghez egyértelműen 1-hez közeli C_r a kívánatos.

Egy másik fontos mérőszám az elküldött csomagok M száma. Egyszerű elárasztás esetén természetesen adódik, hogy $M = N$, ha $C_r = 1$. Egy hatékony algoritmustól alacsonyabb M értéket várunk, magas C_r mellett. Az egyszerűbb összehasonlítás érdekében az üzenetek számát normalizált formában,

$$M_{norm} = \frac{M}{N} \tag{2.5}$$

alakban használjuk.

Ha az elárasztást útvonal-feltérképezésre használjuk, egy másik fontos mérőszám a feltérképezett utak L hossza (lépésszáma).

2.3.2. Hálózati modell

A csomópontokat egy kétdimenziós területen helyeztem el, egyenletes eloszlást használva. A véges kommunikációs hatótávolság leírására a kör modellt használtam. Eszerint azok a csomópontok tudnak üzenetet váltani, melyek egy bizonyos kommunikációs hatósugáron belül helyezkednek el. A [66]-ban szereplő eredmények szerint ez a modell az idealisztikus volta ellenére is legrosszabb esetet leíró modellnek tekinthető.

A felhasznált csatornamodell nem foglalkozik külön-külön a fizikai, vagy a MAC réteg egyes részleteivel, ehelyett egy valószínűségi modellt használ. Egy, a kommunikációs hatósugáron belül levő csomópont $P_{rec} \leq 1$ valószínűséggel tudja venni a küldött csomagot. Ez a magas szintű modell nem tesz különbséget a különböző eredetű hibaforrások (csomagütközés, fading, stb.) között, ehelyett azok közös hatását egyetlen paraméterrel modellezi. Habár a modell számos hibaforrást (például az üzenetek egymásra hatását) figyelmen kívül hagy, mégis egy használható megoldást nyújt a szimulációkhoz.

2.3.3. Szimulációs környezet

A gyors és hatékony teszteléshez C nyelven valósítottam meg a szimulátor programot. A program a csomópontokat véletlenszerűen helyezi el, a tesztbeállításoknak megfelelően. A csomópontok N száma, az r kommunikációs hatósugár és a K_{min} a program bemeneti paraméterei.

Minden szimuláció elején a csomópontok az előzőektől független helyekre kerülnek, majd egy kijelölt forrás csomópont adást kezdeményez. A szimulátor a forrás csomópontot tartalmazó összefüggő komponens méretét, az üzenetet sikeresen vevő csomópontok számát és az elküldött üzenetek számát adja vissza. A szimulátor a topológiát, az aktív adatutakat, és a csomópontok állapotát is képes megjeleníteni.

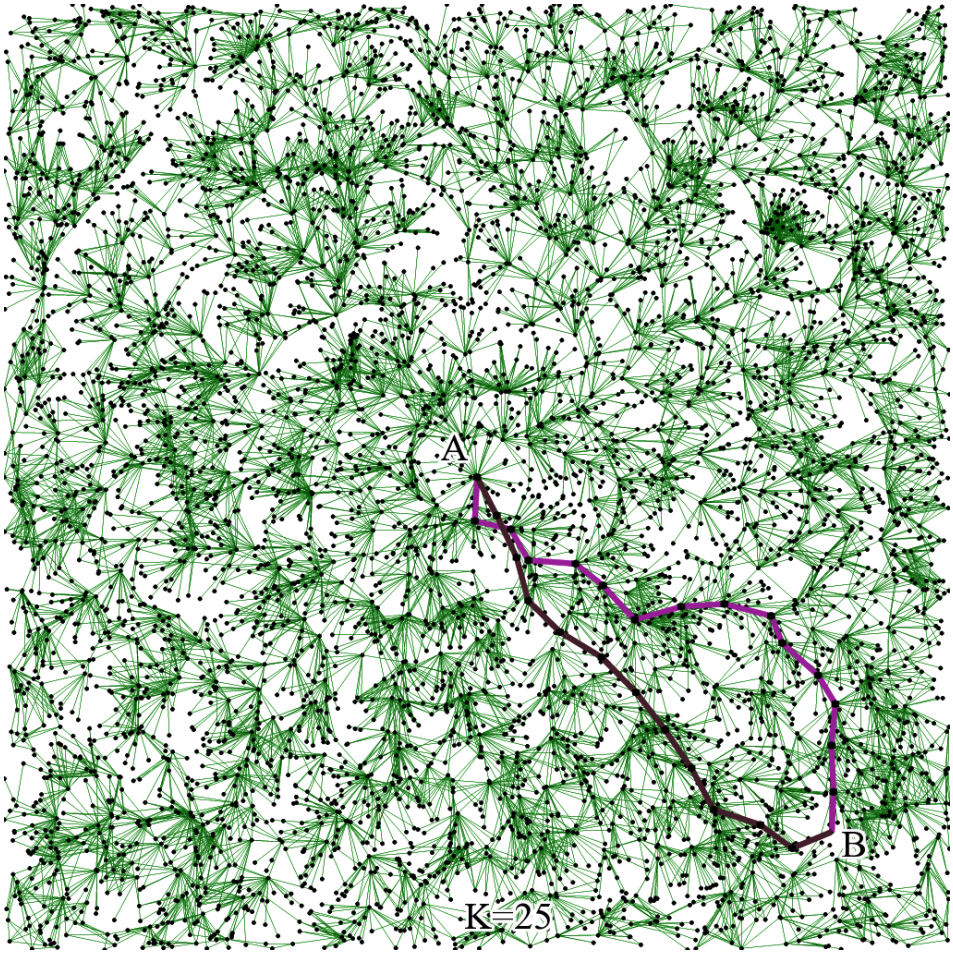
A szimulációs program forráskódja az [S1] linken érhető el.

2.3.4. Tesztbeállítások

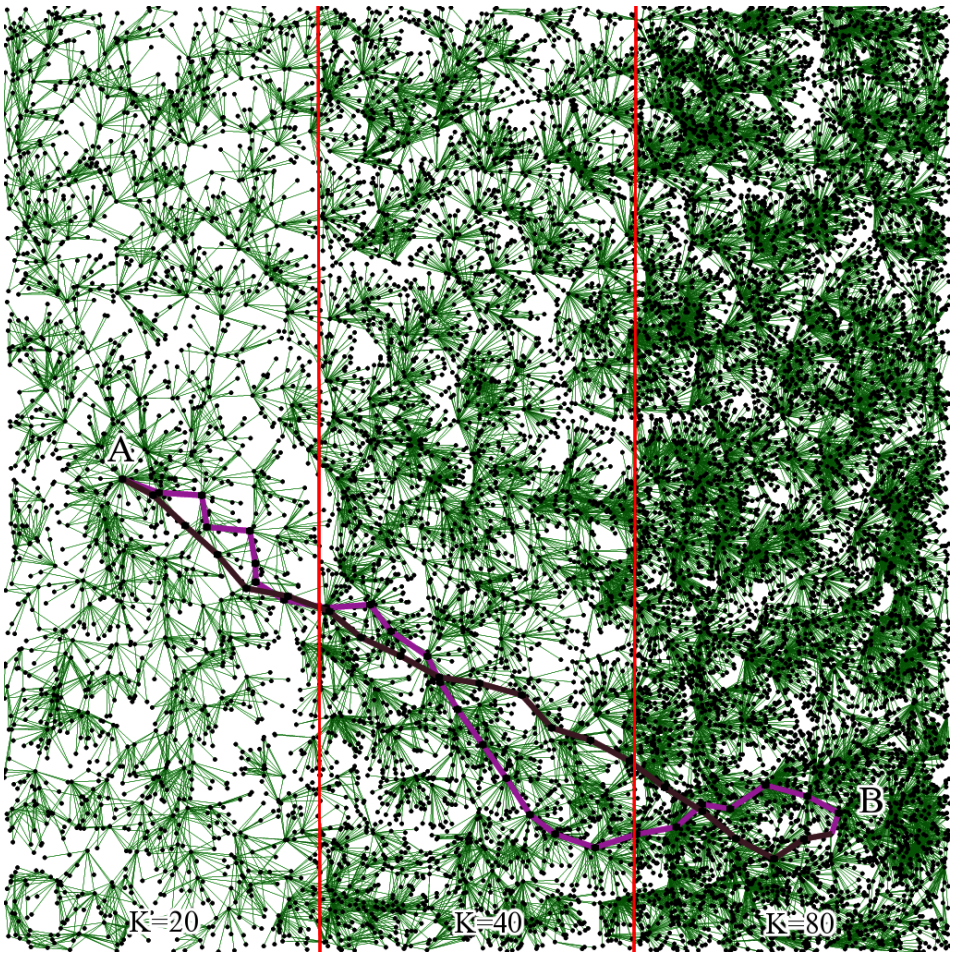
A teszt során két különböző tesztbeállítást használtam. Az első beállításban a csomópontok egyenletes eloszlást követve helyezkednek el, tehát sűrűségük állandó, a második beállításban a csomópontok által elfoglalt területet három régióra bontottam, és minden régión belül más sűrűséget használtam. Az elrendezésekre egy-egy tipikus példát láthatunk a 2.1. és a 2.2. ábrákon.

2.3.5. Teszteredmények

A C_r lefedettség és az M üzenetszám méréséhez egy négyzetes területen 3000 csomópontot helyeztem el, ahogyan az a 2.1 ábrán is látható. A különböző hálózati sűrűségek eléréséhez úgy változtattam a kommunikációs hatósugarat, hogy a szomszédok K átlagos



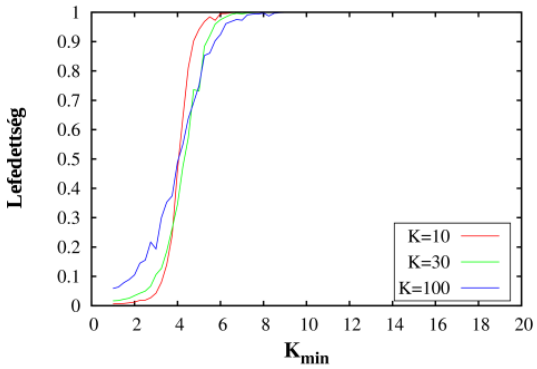
2.1. ábra. Egyenletes eloszlást használó tesztbeállítás 3000 csomóponttal. A szomszédok átlagos száma 25. A fekete, illetve a lila út az elárasztás, illetve a perkolációt eredményező elárasztás által felfedezett, A-ból B-re vezető utat jelöli.



2.2. ábra. Három különböző sűrűséget használó tesztbeállítás. Balról jobbra haladva az egyes régiókban az átlagos szomszédszám 20, 40, illetve 80. A fekete, illetve a lila út az elárasztás, illetve a perkolációt eredményező elárasztás által felfedezett, A-ból B-re vezető utat jelöli.

száma a 10, a 30, illetve a 100 értékeket vegye fel. Majd K_{min} értékét 1-től 20-ig 0.25-os lépésekben változtatva minden r kommunikációs hatósugárra, K_{min}, illetve P_{rec} értékre 100 futtatást végeztem. A grafikon minden pontja a 100 futtatás átlagát jelzi.

A 2.3. ábrán a lefedettség K_{min} függvényében, $P_{rec} = 1$ beállítás mellett látható. A különböző hálózati sűrűségű beállításokhoz tartozó grafikonok hasonlóak. Amikor K_{min} nagyon alacsony ($K_{min} < 3$), az üzenetek célba jutási aránya alacsony. K_{min} növelésével ez hirtelen megugrik, a kritikus érték $K_{min} \approx 4{,}5$. A lefedettség a 90%-os szintet $K_{min} \approx 5$-nél éri el. Ha $K_{min} > 7$, akkor gyakorlatilag az összes csomóponthoz eljut az üzenet. A 2.3. ábrán látható módon a perkolációt eredményező elárasztás többféle hálózati sűrűség esetén is jól használható, K_{min} értéke nem függ a hálózat sűrűségétől.

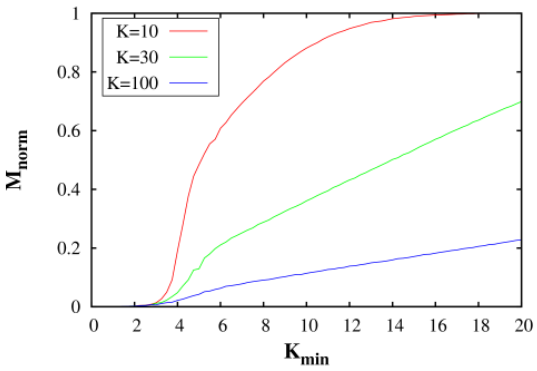


2.3. ábra. A lefedettség K_{min} függvényében különböző hálózati sűrűségekre az egyenletes eloszlású (állandó hálózati sűrűségű) tesztbeállításban (állandó sűrűség esetén) $P_{rec} = 1$ mellett.

A 2.4. ábrán a küldött üzenetek normalizált értéke (M_{norm}) látható K_{min} függvényében az előző kísérletre. Az egyszerű elárasztás esetében $M_{norm} = 1$, mert az összes csomópont továbbküldi az üzenetet. Ahogy a 2.3. ábrán látható, $K_{min} > 7$ esetén közel teljes lefedettséget kapunk. A 2.4. ábra szerint $K_{min} \approx 7$ esetén a küldött üzenetek száma erősen lecsökkent, a csomópontok sűrűségétől függően 30. . . 90%-os megtakarítás volt elérhető a tesztekben. (A magasabb megtakarítások a nagyobb sűrűségekhez tartoznak.)

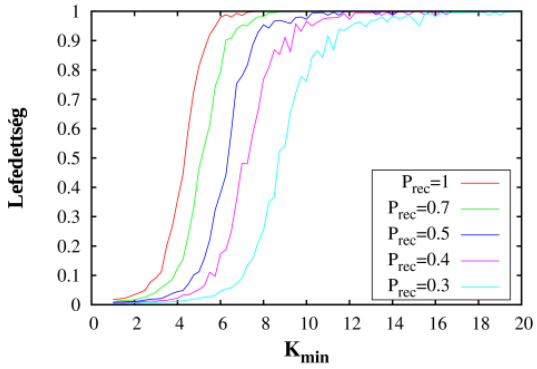
Világos, hogy ha K értéke alig haladja meg K_{min} értékét, a küldött üzenetek száma alig csökken, sűrű hálózatok esetén viszont alacsony üzenetszámmal is biztosítható a közel teljes lefedettség, ahogy az a 2.4. ábráról is leolvasható. A 2.3. és 2.4. ábrák alapján elmondható, hogy a perkolációt eredményező elárasztás sűrű hálózat esetén jelentősen képes csökkenteni az elküldött üzenetek számát a közel teljes lefedettség biztosítása mellett.

A 2.5. ábra a megbízhatatlan összeköttetések hatását mutatja be. A kísérlethez a csomópontok sűrűségét állandó ($K = 30$) értéken tartva a P_{rec} vételi valószínűséget változtattam 0,3 és 1 között. Az ábra szerint az algoritmus kevésbé megbízható összeköttetések esetén is képes nagy lefedettséget biztosítani, ehhez azonban a csatorna minőségének romlásával egyre magasabb K_{min} szükséges. A 2.6. ábrán a rossz minőségű csatornára vonatkozó kísérlet során tapasztalható üzenetszám látható. K_{min} értékét növelve az üze-

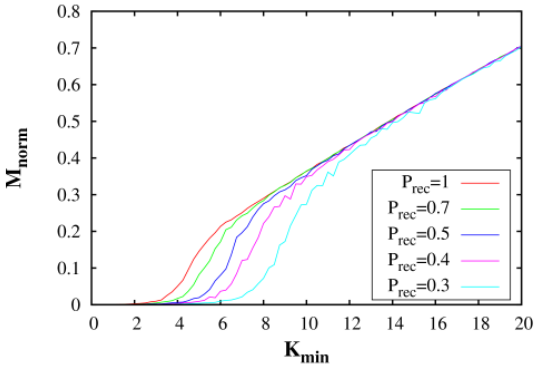


2.4. ábra. A küldött üzenetek száma K_{min} függvényében különböző hálózati sűrűségekre az egyenletes eloszlású (állandó hálózati sűrűségű) tesztbeállításban $P_{rec} = 1$ mellett.

netszám a hibátlan csatornát használó kísérlethez hasonlóan növekszik. Az egyes görbék között K_{min} közel teljes lefedettséget adó értékeinél lényegi különbség nem tapasztalható.

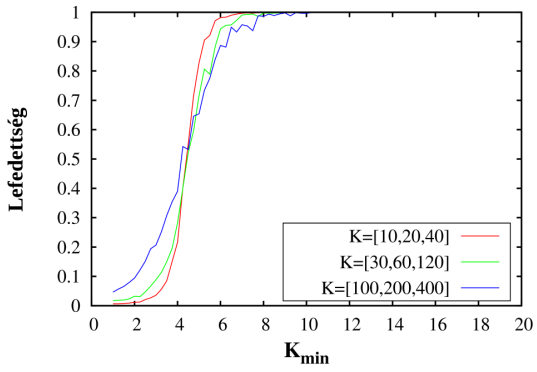


2.5. ábra. A lefedettség K_{min} függvényében különböző P_{rec} értékekre az egyenletes eloszlású (állandó hálózati sűrűségű) tesztbeállításban $K = 30$ mellett.

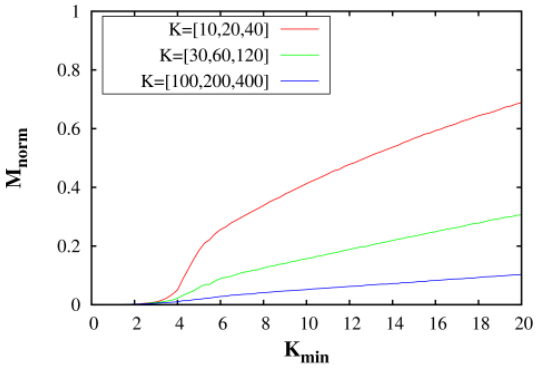


2.6. ábra. Az üzenetek száma K_{min} függvényében különböző P_{rec} értékek esetén az egyenletes eloszlású (állandó hálózati sűrűségű) tesztbeállításban $K = 30$ mellett.

A perkolációt eredményező elárasztás a hálózatban egy időben jelen levő különböző sűrűségű területek esetén is hatékonyan képes működni. Ennek tesztelésére a 2.2. ábrán látható tesztbeállítást használtam; itt a hálózat három eltérő sűrűségű területre oszlik. Ebben az esetben a továbbküldési valószínűség egy fix értékre való beállítása vagy alacsony lefedettséget, vagy feleslegesen sok üzenetet eredményezne (a ritka, illetve a sűrű tartományokban). A 2.7. ábrán a perkolációt eredményező elárasztás viselkedését láthatjuk. A grafikonokon a lefedettség látható K_{min} értékének függvényében, az egyes tartományokban $K = [10, 20, 40]$, $K = [30, 60, 120]$, illetve $K = [100, 200, 400]$ átlagos szomszédszámot alkalmazva, minden esetben $P_{rec} = 1$ beállítást használva. Az algoritmus viselkedése hasonló az első kísérletben szereplő konstans sűrűségű hálózat esetében tapasztaltakhoz. A perkoláció ugyanannál a $K_{min} \approx 5$ értéknél kezd érvényesülni, a megközelítőleg teljes lefedettséghez $K_{min} > 7$ érték szükséges. A kísérlet során mért üzenetszámok a 2.8. ábrán szereplő grafikonokon láthatók. M_{norm} értéke az első kísérlethez hasonló jelleget mutat.



2.7. ábra. A lefedettség K_{min} függvényében a változó hálózati sűrűségű tesztbeállításban $P_{rec} = 1$ mellett.



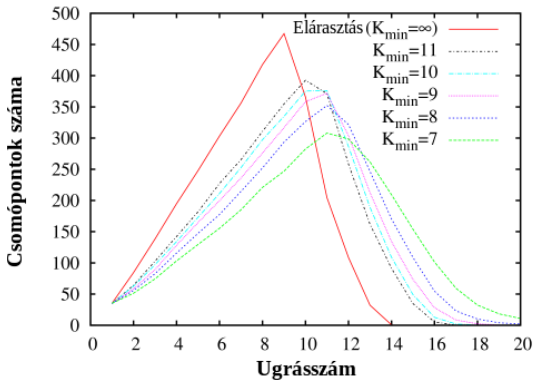
2.8. ábra. Az üzenetek száma K_{min} függvényében a változó hálózati sűrűségű tesztbeállításban $P_{rec} = 1$ mellett.

Az üzenetek által megtett utak L hossza akkor lényeges, ha az algoritmust útvonalfelfedezésre használjuk. Hogy az algoritmust ebből a szempontból is megvizsgáljam, megmértem az egyszerű elárasztás és a perkolációt eredményező elárasztás által felfedezett útvonalak hosszúságát különböző K_{min} értékekre. A kísérletben az egyenletes eloszlású tesztbeállítást használtam $K = 35$ és $P_{rec} = 1$ beállításokkal. A forrás minden estben a hálózat közepén elhelyezkedő csomópont volt, az ettől számított ugrásszámot minden csomópont esetén lemértem. A 2.9. ábrán szereplő hisztogramok 100 futtatás átlagát mutatják. A tapasztalt eloszlás kis ugrásszámokra lineáris. A magyarázat a 2.10. ábrán látható; az üzenetek ideális esetben körgyűrűszerű lépésekben terjednek. Az egymást követő körgyűrűk területe lineárisan növekszik, így az egyes körgyűrűkben levő csomópontok száma is. Ha a csomópontok sűrűsége kisebb, akkor az egyes körgyűrűkben levő csomópontok száma is kisebb, ami kisebb meredekséget eredményez a 2.9. ábra grafikonjain. A 2.9. ábra szerint a kisebb K_{min} értékek is a meredekség csökkenését eredményezik, például $K_{min} = 9$ esetén a meredekség 40%-kal kisebb, mint az egyszerű elárasztás esetén. Ez azt jelenti, hogy a felismert útvonal 40%-kal hosszabb, mint az egyszerű elárasztás által felismert.

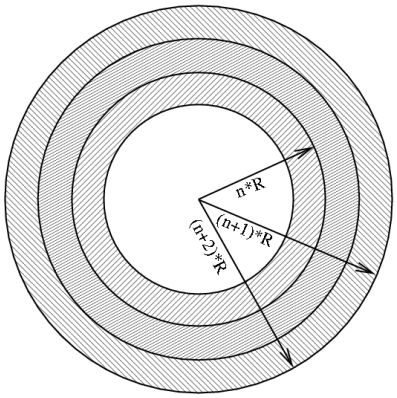
A grafikonok csökkenő szakasza a teszthálózat véges méretéből ered: az üzenetek elérik a hálózat kerületét, majd elhalnak. A hisztogram vége a leghosszabb felismert utak hosszúságát jelöli; ideális esetben a vizsgált négyzet alakú teszthálózatban, középről indított üzenetek esetén ez a hisztogram maximumhelyének $\sqrt{2}$-szörösénél található.

A nem megbízható kapcsolatok hatása a 2.11. ábrán látható. A méréshez az egyenletes eloszlású tesztbeállítást használtam $K = 35$ átlagos szomszédszámmal és $P_{rec} = 0,4 \ldots 1$ vételi valószínűséggel. A grafikon minden pontja 100 futtatás átlagát jelzi. Ahogy az intuitív módon várható, a kevésbé megbízható kapcsolatok kisebb meredekséget, tehát hosszabb felfedett útvonalakat eredményeznek. Például $P_{rec} = 0,7$ esetén az útvonalak 18%-kal lesznek hosszabbak, az igen megbízhatatlan kapcsolatokat jelentő $P_{rec} = 0,4$ a kapcsolatok 63%-kal való hosszabbodását eredményezik.

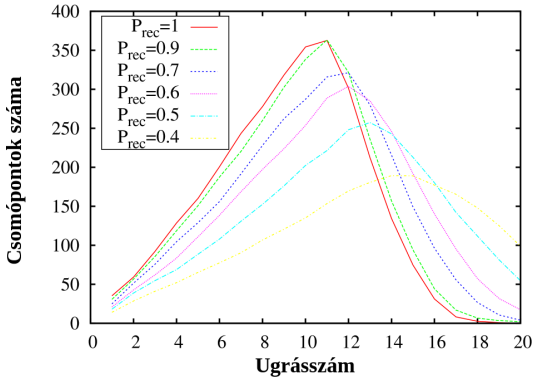
Ahogy a szimulációk alapján tapasztalható, a perkolációt eredményező elárasztás jelentős mértékben hosszabb felfedezett útvonalakat eredményez kis K_{min} értékek esetén.



2.9. ábra. Az ugrásszámok eloszlása az egyenletes eloszlású tesztbeállításban különböző K_{min} értékekre $P_{rec} = 1$ mellett.



2.10. ábra. Egy üzenet ideális terjedése sűrű hálózatban.



2.11. ábra. Az ugrásszámok eloszlása különböző P_{rec} értékek mellett az egyenletes eloszlású tesztbeállításban $K_{min} = 9$ mellett.

Természetesen a kapcsolatok alacsony minősége is hasonló hatást fejt ki. Ez a viselkedés útvonal-felderítési algoritmusok esetén nemkívánatos lehet, ezekben az esetekben az egyszerű elárasztás jobb eredményt ad.

2.4. Összefoglalás

Egy olyan, perkolációt eredményező elárasztási algoritmust mutattam be, ami kizárólag lokális információt, a szomszédok számát felhasználva képes csökkenteni az üzenetszórási vihart. Az algoritmus képes az üzenetek számának jelentős csökkentésére a magas kézbesítési arány megtartása mellett. Az algoritmus használhatóságát elméletileg is alátámasztottam: bizonyítottam, hogy kellően sűrű végtelen hálózat esetén az üzenet 1 valószínűséggel jut el a hálózat egy végtelen méretű komponenséhez. Az algoritmus teljesítményét szimuláció segítségével is vizsgáltam. Az általam használt hálózatokban a $K_{\min}$ megfelelő választása esetén a küldött üzenetek száma a hálózati sűrűségtől függően 30-90%-kal csökkent, közel 100%-os lefedettség megléte mellett. A perkolációt eredményező elárasztás változó sűrűségű hálózatok esetében is jól teljesített, tehát a hálózat lokális jellegéhez is képes adaptálódni. A tesztek során a megbízhatatlan linkek hatását is vizsgáltam; az algoritmus nagy számú nem megbízható link megléte esetén is hatékony működést mutatott.

3 Körkörös és többkörös TDMA ütemezések

3.1. Áttekintés

Ha az adattovábbítás során egyszerre szükséges valós idejű kézbesítés és magas energiahatékonyság, akkor egy időosztásos többszörös hozzáférést (Time Division Multiple Access, TDMA) használó közeghozzáférési (MAC) protokoll jelenthet kézenfekvő megoldást. TDMA használata esetén a kommunikáció ütemezett időszeletekben történik, így biztosítva az ütközésmentes kommunikációt. Emellett ez a megoldás energiahatékony is, hiszen az időszeletek előre ismert ideje elkerülhetővé teszi a sok energiát igénylő tétlen hallgatást. A fix TDMA ütemezés determinisztikus kézbesítési időt is garantál. A TDMA hátrányai közé tartozik, hogy a kihasználatlan időszeletek egyben kihasználatlan sávszélességet is jelentenek, illetve hogy a rendszer működéséhez globális időszinkronizációra van szükség, ami hálózati többletforgalommal jár [71, 72]. A TDMA két igen elterjedt, nem szenzorhálózatos alkalmazási példája a 802.11 protokollcsalád (ismertebb nevén a WiFi) [73], illetve a 2G mobil telekommunikációs szabványok jelentős része [74].

A TDMA szenzorhálózatos alkalmazások esetén is elterjedt; ezeknél a rendszereknél a korlátos késleltetés és az energiahatékonyság kulcsfontosságú tényezők [75, 76].

Számos rendszer a globális TDMA helyett a közeli csomópontokból fürtöket alakít ki, és egy fürtön a TDMA-t a fürtön belüli eszközök kommunikációjára használja, ahogy azt a [77] protokollja, illetve a LEACH [30] is teszi.

Bizonyos protokollok a TDMA elveit követik, azonban nem ügyelnek szigorúan az ütközések elkerülésére, ahogyan a Dozer esetében is megfigyelhető. Itt a protokoll arra törekszik, hogy a konkurens kommunikációt folytató eszközök kis mértékben eltérő periódusidőt használjanak, így az ütközéseket ugyan nem kerüli el, de azok valószínűségét alacsony szinten tartja [30].

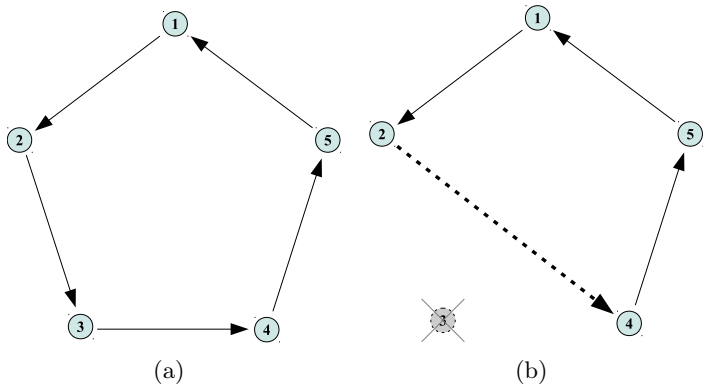
A DESYNC egy elosztott TDMA rendszer valósít meg. A megoldás alapötlete az, hogy a globális szinkronizáció helyett ennek ellenkezőjét hozza létre, vagyis a csomópontok adási idejét igyekszik a kijelölt perióduson belül minél inkább szétszórni és így elkerülni az ütközéseket [78].

A következőkben két TDMA alapú kommunikációs algoritmust mutatok be. A körkörös TDMA egy gyűrű topológiát használó algoritmus, ami hatékony energiafelhasználást és garantált kézbesítési időt nyújt. Ezen felül hibatűrő is: akkor is képes nyújtani az említett szolgáltatásokat, ha egy (sőt akárhány, nem egymást követő) csomópont, vagy kapcsolat meghibásodik. A körkörös TDMA egyszerű gyűrű topológiájával szemben a többkörös TDMA több, egymáshoz kapcsolódó, akár különböző sebességgel működő gyűrűt is tud kezelni, megtartva a körkörös TDMA előnyös tulajdonságait.

3.2. A körkörös TDMA algoritmus

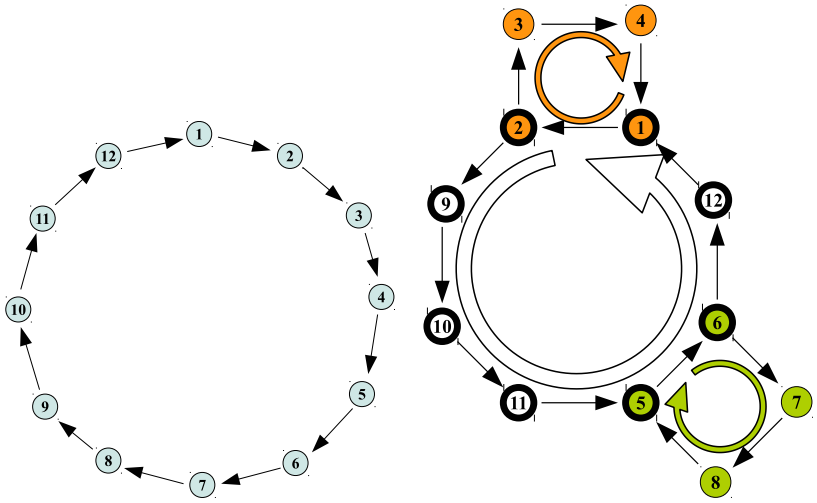
3.2.1. Az algoritmus működésének vázlata

Az algoritmus működésének alapgondolatát a 3.1. ábra mutatja be egy 5 csomópontból álló hálózaton keresztül. A 3.1(a) ábrán a hibamentes eset látható, itt mindegyik csomópont a szomszédjának adja tovább a megkapott csomagot, ami így bármely csomópontból az összes többi csomópontba 4 lépésen belül eljut. A 3.1(b) ábrán a 3-as számú csomópont működésképtelenné vált, aminek következtében az előző sémát követve a körkörös kommunikáció nem lenne lehetséges. Ekkor az algoritmus a hibás eszközt átugorva, a másodszomszédok között létező kapcsolatot felhasználva kommunikál, így az algoritmus hibatűrő működést valósít meg.



3.1. ábra. Az üzenet körbeküldése a gyűrű topológia esetén, amikor minden csomópont működőképes (a), illetve egy csomópont meghibásodása esetén (b).

A 3.2. ábrán a körkörös, illetve a többkörös topológiák összehasonlításához az egyes topológiákra egy-egy példát láthatunk. A 3.2(a) ábrán egy 12 csomópontból álló körkörös hálózat, a 3.2(b) ábrán pedig egy 8 csomópontot tartalmazó főkörből és ahhoz kapcsolva két alkörrel egy többkörös hálózat látható. Míg a körkörös hálózatban a topológia egyetlen körből áll, a többkörös esetben a főkörhöz kapcsolva alkörök is létrehozhatók. A javasolt algoritmus működését az egyes topológiák esetén a következő szekciókban ismertetem.



3.2. ábra. Egy körkörös (a) és egy többkörös (b) hálózat topológiája. A (b) ábrán a főkörön levő csomópontok vastag vonallal vannak jelölve. A zöld és a narancssárga csomópontok az alköröket jelölik.

3.2.2. A rendszer üzemmódjai és konfigurációja

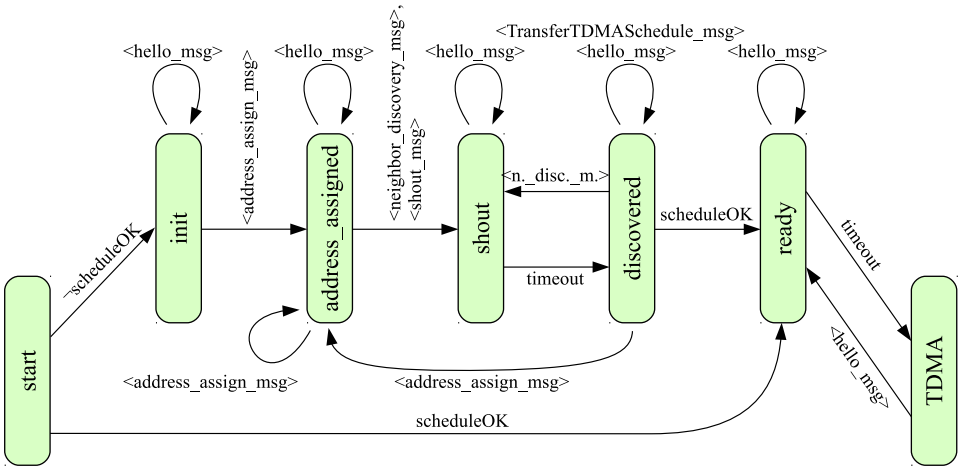
Induláskor minden szenzor csomópont megvizsgálja a nemfelejtő memóriáját. Ha itt egy valós TDMA ütemezést talál, akkor rövid várakozási idő után átlép TDMA módba. Egyéb esetben a csomópontok konfigurációs módba lépnek, aminek a végén a TDMA ütemezés letöltődik a csomópontokba.

A konfigurációs mód működését egy speciális programot futtató eszköz, a bázisállomás, illetve a hozzá kapcsolt számítógép és a rajta futó vezérlő program irányítja. A konfigurálás végeztével a bázisállomásra, illetve a számítógépre már nincsen szükség.

A felderítés elindítását a bázisállomás egy *hello_msg* üzenetet küldésével váltja ki. Az üzenet eljut a csomópontokhoz, amire azok a saját, egyedi 64-bites hardver-címükkel válaszolnak. Ezután a bázisállomás a 64-bites címekhez 16-bites rövid címeket rendel az *address_assign_msg* üzenetek segítségével. A csomópontok erre *address_assign_reply_msg* üzenetekkel válaszolnak, ami nyugtaként szolgál, így a bázisállomás szükség esetén meg tudja ismételni az esetlegesen elveszett üzeneteket. Miután minden csomópont megkapta a rövid címét, a csomópontok feltérképezik egymás szomszédságát a *neighbor_discovery_msg* üzenet hatására. A csomópontok broadcast üzeneteket küldenek és összegyűjtik a szomszédaiktól vett üzeneteket. Az érzékelt szomszédok listáját *neighbor_reply_msg* típusú üzenetek formájában küldik vissza a csomópontok a bázisállomásnak. Az üzenetek tartalmazzák, hogy melyik eszközt hányszor és milyen minőségben hallotta az adott csomópont. A szomszédlistákból a bázisállomás felépíti a hálózat kapcsolati gráfját és ez alapján generálja a TDMA ütemezést. Az ütemezés *TDMA_schedule_msg* típusú csomagok formájában töltődik le az eszközökre. A szenzorok minden vett csomagra egy-egy *TDMA_schedule_reply_msg* üzenetet küldenek vissza a bázisállomásra

nyugtaként, így az elveszett csomagok megismételhetőek.

A rendszer viselkedését részletesen a 3.3. ábrán látható állapotgép szemlélteti. Az állapotgépen a fentebb részletezett működést biztosító állapotátmeneteken kívül még néhány további is látható. Ezek a kivételes működések esetén biztosítanak helyes viselkedést. Például a *hello_msg* újra képes elküldeni az eszközök 64-bites címét, azok állapotát megtartva, illetve a *ready* állapot esetében az időtúllépés óráját újraindítva. Egy újabb cím hozzárendelésével (*address_assign_msg*) a szomszédok felderítése is elölről kezdhető.



3.3. ábra. A rendszer véges állapotú automata modellje

3.2.3. A TDMA kommunikációs primitív

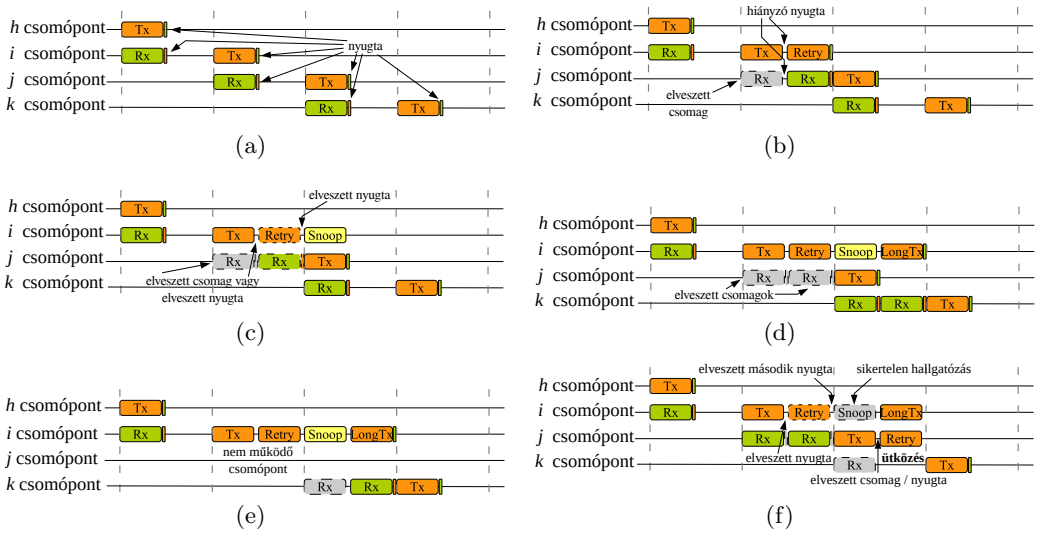
A kommunikáció működéséhez a körkörös TDMA használata esetén a kapcsolati gráfnak egy Hamilton-kört kell tartalmaznia, legyen ez a $H = (V_1, \ldots, V_n)$. A hibatűrés biztosításához ezen kívül még az is szükséges, hogy minden csomópont és a Hamilton-körnek megfelelő másodszomszédja között is legyen él. A TDMA periodikus réselt kommunikációt használ, ahol minden periódus azonos és a periódust a hálózatra nézve globális ütemezés írja le.

Az ütemezés vétel-adás-hallgatózás-áthidalás négyesekből áll, a következő módon. Legyen h, i, j és k egymást követő csomópontok H-ban. A csomópontok közötti kommunikáció a következő módon alakul:

- i egy csomagot vett h-tól.
- i egy csomagot küld j-nek.
- j, közvetlenül a csomag vétele után egy nyugtát (ACK) küld i-nek.

- Ha i megkapja a nyugtát, a kommunikációt befejezettnek tekinti és kikapcsolja rádióját. A nyugta sikertelen vétele esetén, ha az újraküldés engedélyezett (lásd később), i még egyszer megpróbálkozik a csomag elküldésével. Ha i a nyugtát semelyik esetben sem kapja meg, feltételezi, hogy j meghibásodott. Hogy erről meggyőződjön, lehallgatja j következő adását.
  - Az újraküldés csak akkor engedélyezett, ha a legutóbbi vételi időszeletben sikeres vétel történt, tehát ha a csomópont friss információval rendelkezik. Ellenkező esetben az újraküldési időszelet az áthidalásra van fenntartva.
  - Minden csomag tartalmaz egy *last_ OK* (LOK) mezőt, ami azt jelzi, hogy az előző vétel sikeres volt-e.
- A következő adási időszelet a j-hez tartozik, ekkor j ad k-nak egy csomagot.
- Abban az időszeletben, amikor j k-nak ad, i lehallgathatja azt; ezt abban az esetben teszi, ha a legutóbbi adására egyetlen nyugta sem érkezett. Ekkor három eset lehetséges:
  - Az i által lehallgatott csomagban a LOK mező igaz, tehát j sikeresen vette i legutóbbi üzenetét. Ekkor i-nek nincs további teendője, tehát elaludhat.
  - Az i által lehallgatott csomagban a LOK mező hamis, tehát j működik, de nem vette i legutóbbi üzenetét. Ebben az esetben i egy áthidaló csomagot küld k-nak j újraküldési időszeletében. Fontos megjegyezni, hogy ilyenkor j biztosan nem ismétli meg az adását, hiszen az csak egy sikeres vételt követően lenne lehetséges, így az újraküldési időszeletben nem alakulhat ki csomagütközés.
  - Az i csomópont nem hallott csomagot a lehallgatás során, ami azt jelenti, hogy j valószínűleg nem üzemel. Ekkor i, az előző szituációhoz hasonlóan egy áthidaló üzenetet küld k-nak j újraküldési időszeletében. Fontos megjegyezni, hogy ebben az esetben előfordulhat csomagütközés, de csak abban a ritka esetben, ha j mégis működik, ráadásul a csomagját is megismétli.

A kommunikáció folyamata a 3.4. ábrán látható.

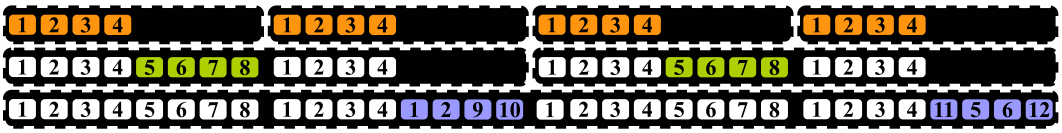


3.4. ábra. A kommunikáció folyamata. A *Tx* normál adást, az *Rx* normál vételt, a *Retry* az adás ismétlését, a *Snoop* a lehallgatást, a *LongTx* pedig az áthidalást jelzi. Az (a) esetben a kommunikáció hibamentes. A (b) esetben az i j-nek szánt csomagja elveszik, de az ismétlés sikeres. A (c) esetben i és j között az átvitel sikeres, de a nyugták elvesznek. A sikeres átvitelről i lehallgatással szerez tudomást. A (d) esetben i és j között sikertelen az átvitel, amit i k-nak szóló áthidaló üzenettel korrigál. Az (e) esetben j működésképtelen. Az (f) ábra azt a ritka esetet ábrázolja, amikor az ismétlési szeletben ütközés történik.

3.3. Az ütemezés kiterjesztése többkörös TDMA hálózatokra

Az eddig tárgyalt topológia egyetlen kört tartalmazott. A továbbiakban az egykörös hálózatot általánosítom többkörös topológiára. Többkörös hálózat esetén az eddig tárgyalt kör a főkör szerepét fogja ellátni, amihez alkörök kapcsolódhatnak. A 3.2(b) ábrán egy ily módon kialakított többkörös topológiát láthatunk. Az 1, 2, 9, 10, 11, 5, 6 és 12 csomópontok alkotják a főkört, ahol az információ a körkörös topológiával megegyező módon áramlik körbe. Az 1, 2, 3 és 4, illetve az 5, 6, 7 és 8 csomópontok egy-egy alkört alkotnak, a kommunikáció az alkörök esetében a főkörhöz hasonló módon történik. Mivel a főkör és az alkörök által kialakított struktúra egyetlen hálózatot alkot, a hálózat egy globális ütemezést használ és az összes kör összes kommunikációs eseménye számára egyetlen közös periódus áll rendelkezésre. Az egyes körök eltérő sebességgel (eltérő periódusidővel) működhetnek, ahol az arányszámok csak kettőhatványok lehetnek. Így a leglassabb kör periódusideje alatt annak minden eseménye egyszer kerül sorra, míg egy n-szeres sebességű kör eseményei n-szer. Az 1, 2, 5 és 6 csomópontok a főkörnek is részei, ezek a közös csomópontok felelősek az információ közvetítéséért a főkör, illetve a csomópontot tartalmazó alkör között. Ezek a csomópontok azért is speciálisak, mert ezek mindkét, őket tartalmazó kör megfelelő eseményeit megvalósítják.

Az ütemező algoritmus működését egy példán keresztül fogom bemutatni. Az első lépésben minden használni kívánt sebesség számára egy-egy, a sebességgel fordítottan arányos méretű, saját tárolót hozunk létre. A 3.2(b) ábrán látható példában három különböző sebesség (1×, 2× és 4×) szerepel, így egy P, egy $\frac{P}{2}$ és egy $\frac{P}{4}$ méretű tárolót használunk, ahogy az a 3.5. ábrán látható. Minden $\frac{P}{n}$ hosszúságú tárolóból n egyforma példányt hozunk létre és az egyes példányok egyenlőségét az algoritmus futása során is megőrizzük. Fontos kiemelni, hogy n értéke csak kettő-hatvány lehet.



3.5. ábra. Az egyes események elhelyezése a különböző tárolókban. Az egyes sorok a különböző sebességeknek felelnek meg.

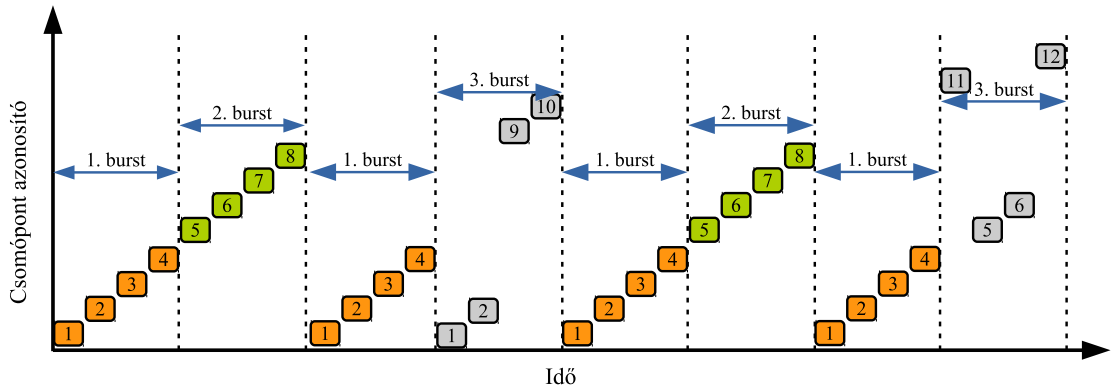
A példában a leglassabb kör periódusidejét $P = 32$ hosszúságúra választottam, így az ábrán 32 időszelet látható. Minden időszeletben pontosan egy adási esemény lehet; ezzel egy időben egy másik csomóponton egy vételi esemény történik. Ugyanennek az időszeletnek a másik felében foglal helyet az adó csomópont ismétlése, illetve egy harmadik csomópont áthidalási eseménye. A továbbiakban csak az adások idejét fogom levezetni, ezekből a másik három eseménytípus kiszámítása kézenfekvő.

Az időszeletek lefoglalása az események számára a következő módon történik. Először a leggyorsabb körben szereplő csomópontok számára foglaljuk le az időszeleteket, a legrövidebb tárolóban. A 3.2(b) ábrán szereplő példában az 1-es, 2-es, 3-as és 4-es csomópontok a négyszeres sebességű körben szerepelnek, így a legrövidebb tárolókban kerülnek

elhelyezésre a 3.5. ábrán látható módon. Mivel a most ütemezett események sebessége négyszeres, így a teljes periódus alatt mindegyik négyszer kerül sorra. Fontos megjegyezni, hogy a gyorsabb körökben lefoglalt időszeleteket a lassabb köröknek megfelelő tárolókban is foglaltnak kell jelölni (a 3.5. ábrán fehér színnel jelölve).

A következő sebességhez tartozó csomópontokat a következő sebesség üresen maradt időszeleteit felhasználva ütemezzük. A példában az 5-ös, 6-os, 7-es és 8-as csomópontok a kétszeres sebességű körhöz tartoznak, így ezek adási eseményei majd a 3.5. ábra második sorában foglalnak helyet. A lefoglalt időszeleteket természetesen az egyszeres sebességű tárolóban is foglaltnak jelezzük.

Az időszeletek lefoglalása a sebességek szerinti sorrendben folytatódik egészen addig, amíg minden esemény be nincs ütemezve. A példában végül az egyszeres sebességű körön levő csomópontokat (1, 2, 9, 10, 11, 5, 6 és 12) ütemezzük a 3.5. ábra 3. sorában üresen maradt helyekre. Az így kapott ütemezés időzítési diagramja a 3.6. ábrán látható.



3.6. ábra. A 3.2(b) ábrán látható hálózatnak, illetve a 3.5. ábrán látható tárolóknak megfelelő ütemezés időzítési diagramja.

3.3.1. Az ütemezés-végrehajtó modul

Az előre kiszámított ütemezést az egyes csomópontokon az ütemezés-végrehajtó modul működteti. A modul egyszerre ütemezi az összes sebességnek megfelelő eseményt. Erre egy egyszerű megoldást az jelentene, ha csak az adott csomóponton előforduló leghosszabb ciklust használnánk és a gyorsabb (k-szoros) sebességű körökhöz tartozó eseményeket ebben k-szor helyeznénk el, ahogy azt a 3.5. ábra sugallja. Az ábrán látható ütemezés szerint például a 2-es csomópontnak 5 adási eseményt kellene tárolnia. Ez a megoldás mind memória, mind az ütemezés letöltése közben használt sávszélesség szempontjából pazarló.

A javasolt algoritmus minden eseményt csak egyszer tárol el a memóriában. A 3.5. ábrán látható ütemezésben a 2-es csomópont csak 2 adási eseményt tárol (egy piros és egy kék színnel jelölt eseményt). Minden eseményhez a következő tulajdonságokat tároljuk:

- A periódus hosszának 2 alapú logaritmusa (SCE – SubCycle Exponent). A periódus kettő hatvány volta miatt ez a szám mindig egész.
- Az esemény ideje a perióduson belül (t_{ev}).
- Egyéb adatok, például az esemény típusa (vétel, adás, lehallgatás, áthidalás, feldolgozás), illetve a szinkronizációhoz szükséges jelzőbitek.

A körök hosszai mindig 2 pozitív hatványai, óraütésekben mérve. Az óraütés a mikrovezérlő nagy felbontású időzítőjének elemi egysége, hossza 16 µs. Bizonyos időszeletek mobil eszközök kommunikációjára vannak fenntartva [79]. A rendszerben definiált egy kitüntetett, $2^{SRV} = 2^{17}$ hosszúságú köridő, aminek az első 124925 óraütésnyi ideje (megközelítőleg 2 másodperc) normál események ütemezésére van használva, a maradék 6144 óraütés (megközelítőleg 98 ms) pedig a mobil eszközökkel való kommunikációra van fenntartva. Fontos megjegezni, hogy a 2^{SRV}-nél rövidebb körök esetén az ütemező modul a 124926 óraütésnyi időt osztja 2, 4, ... egyenlő részre, tehát ezekben az esetekben a körök hossza nem *pontos* kettő hatvány.

Az ütemező modul mindig közvetlenül az aktuálisan kiszolgált esemény után fut és választja ki a következő eseményt, tehát aminek szükséges bekövetkeztéig a legkevesebb idő van hátra. A hátra lévő idő (t_{rem}) az esemény t_{ev} perióduson belüli ideje és a t_{int} periódus kezdete eltelt idő különbségeként számolható. A számításhoz jelöljük az ébredési óra jelenlegi idejét t_w-vel. Egy adott eseményt tekintve két eset lehetséges.

Ha az esemény köridje nagyobb, vagy egyenlő, mint 2^{SRV}, akkor t_{int} egy bitenkénti és művelettel számítható ki:

$$t_{int} = t_w \wedge (2^{SCE} - 1) \tag{3.1}$$

Rövidebb köridők esetén először a 2^{SRV} hosszúságú speciális köridőn belüli t_{intsrv}-vel jelölt időt határozzuk meg:

$$t_{intsrv} = t_w \wedge (2^{SRV} - 1), \tag{3.2}$$

majd kiszámítjuk a t_{sc} köridőt:

$$t_{sc} = \frac{124926}{2^{SRV-SCE}}, \tag{3.3}$$

és végül t_{int}-et, maradékos osztást használva a következő módon:

$$t_{int} = t_{intsrv} \mod t_{sc}. \tag{3.4}$$

Mindkét esetben t_{rem} a következő módon számítható ki:

$$t_{rem} = t_{ev} - t_{int} \tag{3.5}$$

Az algoritmus azt az eseményt választja, amelyikhez a legkisebb t_{rem} érték tartozik, majd az időzítőt úgy állítja be, hogy az t_{rem} óraütés múlva jelezzen.

3.3.2. Időszinkronizáció

A rendszer egy, a [80] cikkben használthoz hasonló, speciálisan kör topológiájú hálózatokhoz illeszkedő időszinkronizációs megoldást használ, ami két különböző óra használatán alapul. Az egyes eseményekre való felébredést az *ébredési óra* szabályozza, a szinkronizációhoz viszont a *fő óra* használatos. A két órát a rendszer kijelölt időpontokban szinkronizálja egymáshoz, úgy, hogy a kommunikációs események időzítése akkor is pontos maradjon, ha a fő órát átállítjuk.

A körkörös TDMA esetén minden csomópont az őt megelőzőhöz szinkronizál, így azt mondhatjuk, hogy a kör a szinkronizáció szempontjából zárt. A többkörös TDMA esetében azonban vannak olyan csomópontok, amelyek két másik csomóponttól is kapnak adatot. Ezekben az esetekben szinkronizáció mindig csak a főkörhöz történik, mellékkörhöz nem. Így azt mondhatjuk, hogy a mellékkör fel van nyitva. Emiatt a mellékkör utolsó csomópontja és a főkörön levő következő csomópont között a szinkronizáció hibája miatt nagyobb szinkronizációból eredő hibával kell számolni.

3.3.3. Hardver architektúra

A szenzor csomópontok fő alkatrésze az ATmega128RFA1 system-on-chip (SoC) eszköz. Ez az integrált áramkör jól használható szenzorhálózatos alkalmazások céljára. A chip egyik fő egysége egy 16 MHz-es ATmega mikrovezérlő 16 kB RAM-mal és 128 kB flash memóriával. A konfigurációs adatok (például a csomópont azonosítója, illetve az ütemezés) számára 4 kB EEPROM áll rendelkezésre. A chipen található másik fontos elem egy mind adásban, mint vételben jó jellemzőkkel rendelkező rádió adóvevő. Az adóteljesítménye -17 dBm és 3,5 dBm között szabályozható, a vételi érzékenysége pedig -100 dBm. A rádió a 802.15.4 szabványnak megfelelő 250 kbps sebességen túl gyorsabb kommunikációra, akár 2 Mbps-ra is képes, ebben az esetben természetesen elvesztve a 802.15.4-gyel való kompatibilitást. A rádió igen gyors, 0.5 ms-os felébredési ideje nagyban segít az alacsony kitöltési tényezőjű TDMA használatában.

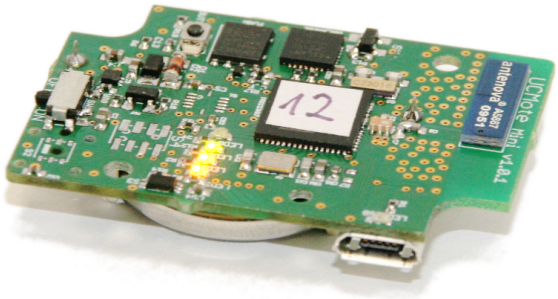
A prototípus eszközön számos szenzor is található: egy SHT21 kombinált hőmérő- és páratartalom-mérő, egy BH1740fvi ambiens fényerősségmérő, illetve egy BMA180 háromtengelyű gyorsulásmérő. Megtalálható még egy általános felhasználású nyomógomb, illetve négy LED, amik elsősorban a hibakeresés során, illetve állapotjelzés céljára hasznosak. Az eszközökön található összes szenzor alacsony energiafelvételű állapotba helyezhető. Az eszköz lehetőséget ad arra is, hogy ahhoz egyéb szenzorokat kapcsoljunk.

Egy 16 Mbites M25P16 típusú külső SPI flash memória segítségével nagyobb mennyiségű mérési adat, illetve napló elmentése is lehetséges. A szenzorokhoz hasonlóan a flash memória is kikapcsolható.

A szenzor csomópontok és a PC kommunikációja egy Silabs CP2102 soros-USB átalakító chip segítségével valósul meg. Az USB porton keresztül lehetőség van az eszközök firmware-ének frissítésére, illetve az árammal való ellátásra is.

Az eszközök elemes, vagy akkumulátoros táplálásúak; mind CR2450 típusú elem, mind pedig LIR2450 típusú akkumulátor fogadására alkalmasak. Az eszközök paneljén töltőelektronika is található; az akkumulátor USB portról tölthető.

Az eszköz fényképe a 3.7. ábrán látható.



3.7. ábra. Egy szenzor csomópont.

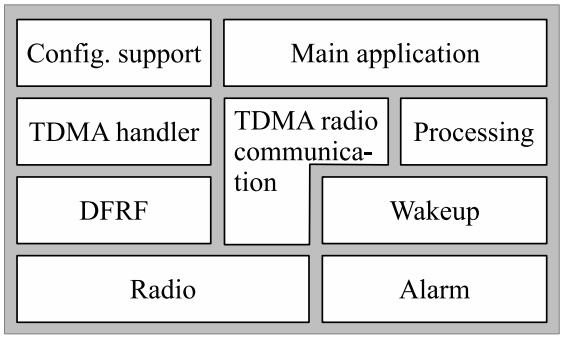
3.3.4. Szoftver architektúra

A szenzor csomópontok szoftver architektúrája a 3.8. ábrán látható. Egy nagy felbontású, 62.5 kHz-en működő óra riasztó (*Alarm*) szolgáltatását használva valósul meg az ébresztő (*Wakeup*) modul, ami az ütemezés egyes eseményeit állítja elő. Az események két csoportra bonthatók. Az egyik csoport a kommunikációval kapcsolatos eseményeket (az adást, a vételt, a lehallgatást, illetve az áthidalást) tartalmazza, a másikban az adatfeldolgozással kapcsolatos események (például mérés, döntéshozás) foglalnak helyet. A fő alkalmazás a TDMA rádiókommunikáció (*TDMA radio communication*) és a feldolgozás (*Processing*) modulok szolgáltatásaira épül.

A konfigurációs fázisban használt többugrásos forgalomirányítás a DFRF (Directed Flood Routing Framework) [43] segítségével valósul meg. Ezzel a komponenssel mind az egy pontból irányuló adatszórás (broadcast), mind pedig az egy pontba történő adatgyűjtés (convergecast) egyszerűen megvalósítható.

A *Wakeup* komponensben levő TDMA ütemező a *TDMA handler* modulon keresztül konfigurálható, ami a letöltött ütemezést tárolja. A *Configuration support* a 3.3. ábrán látható állapotgépet valósítja meg.

A csomópontokon, illetve a kezdeti konfiguráláshoz szükséges bázisállomáson futó szoftver forráskódja az [S2] linken érhető el.



3.8. ábra. A szenzor csomópontok szoftver architektúrája az alkalmazott modulokkal.

3.4. Mérési eredmények

3.4.1. Mérési elrendezések

A tesztekhez – ahol másképp nem jeleztem – két hálózatot használtam. A körkörös TDMA algoritmushoz a 3.9(a) ábrán látható 10 csomópontból álló, a többkörös TDMA algoritmushoz a 3.9(b) ábrán látható 12 csomópontból álló elrendezést alakítottam ki egy irodaépület egyik szintjén. Mindkét esetben 20 ms-os időszeleteket alkalmaztam.

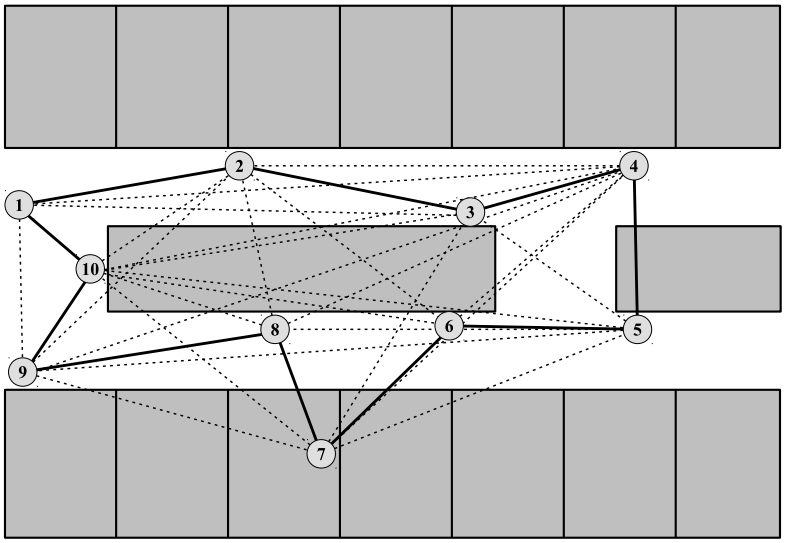
3.4.2. A helyes működés vizsgálata

A körkörös TDMA algoritmust a 3.9(a) ábrán látható 10 csomópontból álló hálózatban teszteltem; a teszt során az egyes csomópontok aktivitását vizsgáltam. A hálózatban a csomópontok növekvő sorrendben voltak elhelyezve. Az eszközök rádiójának be- illetve kikapcsolt állapotát LED-ek be- illetve kikapcsolásával indikáltam, amit egy négycsatornás oszcilloszkóppal mintavételeztem. A 3.10. ábra három szinkronizált felvétel egybeolvasztásával készült. Az ábrán a rendszer működése hibamentes estben figyelhető meg. A periódusban elsőt leszámítva mindegyik eszköz estében egy 9 ms hosszúságú vétel (az ábrán zöld színű R), majd 11 ms inaktivitás (ismétlési időszelet, illetve biztonsági ráhagyás) után egy ugyancsak 9 ms hosszúságú adás (piros színű T) következik. Az egyes vételi ablakok a körben előző csomópont adási ablakával esnek egybe. Az 1-es csomópont esete speciális, ennek vételi ablaka a kört záró 10-es adásával esik egybe.

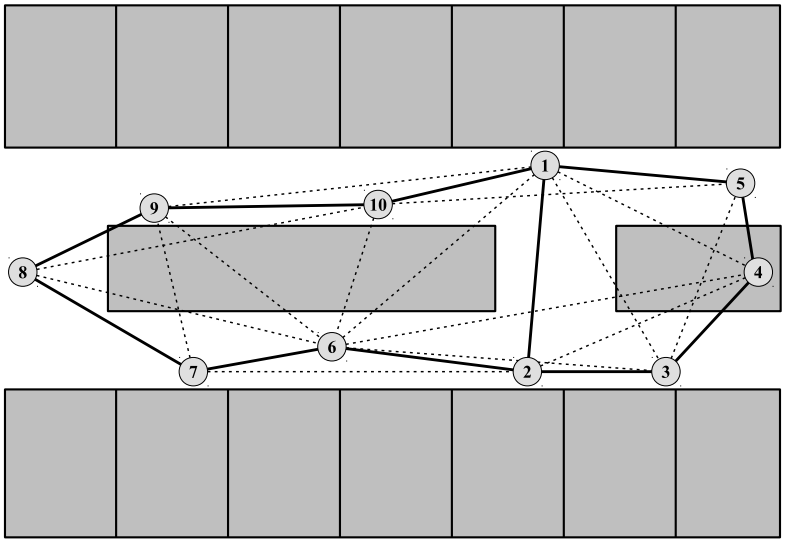
A többkörös TDMA esetében egy hasonló kísérletet végeztem. A 3.9(b) ábrán látható ábrán az 1-es, 2-es, 6-os, 7-es, 8-as, 9-es és 10-es számú csomópontok az egyszeres sebességű főkörön, az 1-es, 2-es, 3-as, 4-es és 5-ös csomópontok pedig a négyszeres sebességű mellékkörön vannak. A 3.11. ábrát több szinkronizált felvétel egybeolvasztásával készítettem. Az ábrán jól megfigyelhető, hogy az egyszeres sebességű főkörön levő csomópontok a perióduson belül egyszer, a négyszeres sebességű mellékkörön levő csomópontok pedig négyszer adnak. Mivel a mellékkör az időszinkronizáció szempontjából fel van nyitva, az alkör utolsó (5-ös) csomópontjának adását vevő 1-es csomópont hosszabb ideig van vételkész állapotban az 5-ös tényleges adásának indulása előtt, hiszen itt nagyobb szinkronizációs hibára kell felkészülni.

3.4.3. Kézbesítési idő

A tesztek során egy kijelölt forrás csomópontnál egy aszinkron eseményt hoztam létre és az üzenet vételét egy másik kijelölt cél csomópontban figyeltem meg; a kézbesítési idő az eseménynek a forrás csomóponton való létrejötte és annak a célban történő vétele közt eltelt idő. Ahogy azt látni fogjuk, a kézbesítési időt a forrás és a cél csomópont megválasztása nagyban befolyásolja, de minden forrás-cél pár esetén megadható a kézbesítési idő valószínűségi eloszlása.

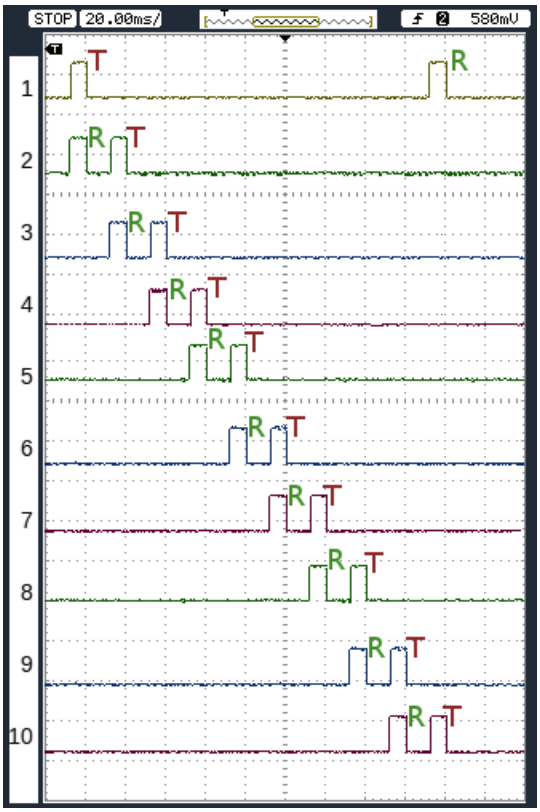


(a)

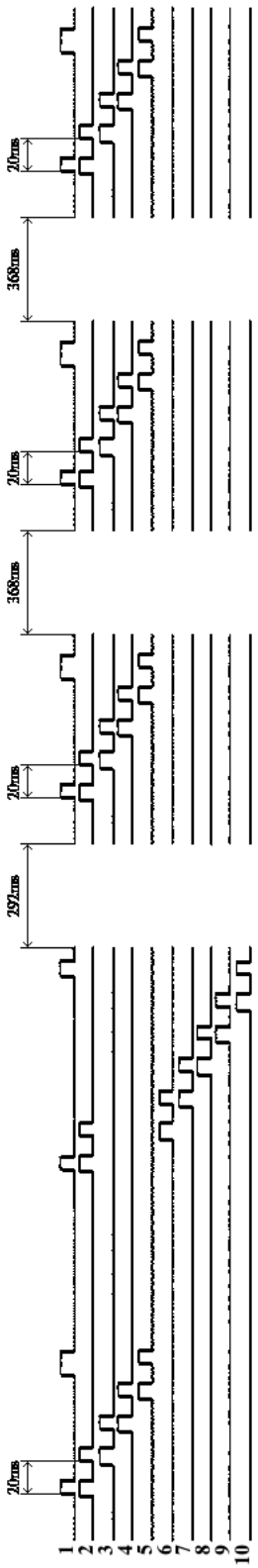


(b)

3.9. ábra. A teszteléshez használt terület alaprajza a körkörös (a), illetve a többkörös (b) TDMA algoritmusokhoz. A szenzor csomópontokat számokkal ellátott körök, a lehetséges összeköttetéseket folytonos és szaggatott vonalak, a Hamilton kört vastag vonal jelzi.



3.10. ábra. A 3.9(a) ábrán látható 10 csomópontból álló rendszer TDMA ütemezésének oszcilloszkópos felvétele hibamentes esetben.

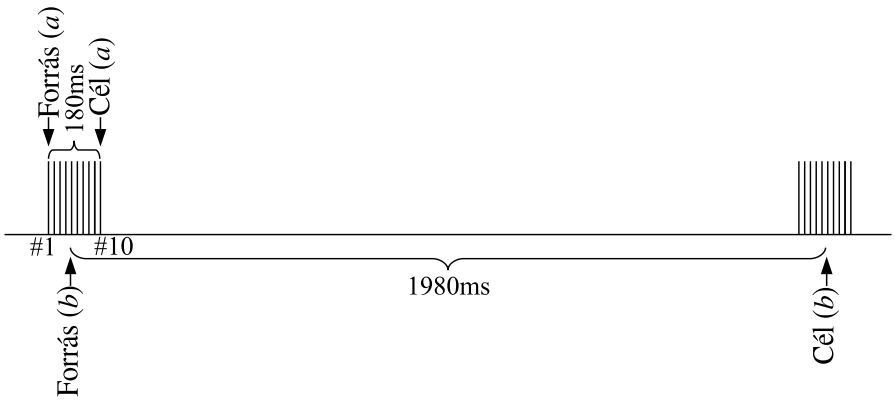


3.11. ábra. A 3.9(b) ábrán látható 12 csomópontból álló rendszer TDMA ütemezésének oszcilloszkópos felvétele hibamentes esetben. Az esemény nélküli szakaszokat a jobb láthatóság érdekében elhagytam.

Körkörös TDMA

A körkörös TDMA esetében a 3.9(a) ábrán látható 10 csomópontos hálózatot használtam a kézbesítési idő mérésére. A 20 ms-os időszelet alkalmazása miatt a kommunikáció 200 ms-os burst-ökben történik. Egy burst-ön belül az információ az 1-es csomóponttól a 10-esig áramlik, majd burst végén a 10-es csomópont az 1-esnek ad. A burst-ök 2,1 másodpercenként ismétlődnek.

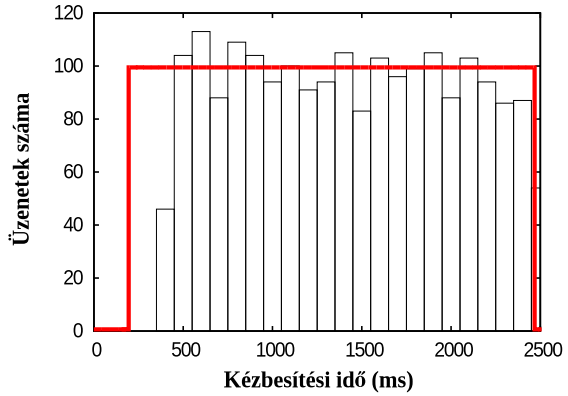
A teszt során két forrás-cél párt használtam, a 3.12. ábrán ezek időzítése látható. Az a esetben a burst első csomópontja (az 1-es) a forrás és az utolsó (a 10-es) a cél. Ebben az esetben a kézbesítési idő két részből áll. Először az esemény után a forrásnak meg kell várnia az adási időszeletét, majd a burst alatt az üzenet eljut a forrástól a célig. Az első összetevő 0 és 2100 ms közötti egyenletes eloszlással rendelkezik, a második komponens konstans 180 ms időt vesz igénybe. A kézbesítési idő elméleti eloszlása a 3.13(a) ábrán piros színnel szerepel.



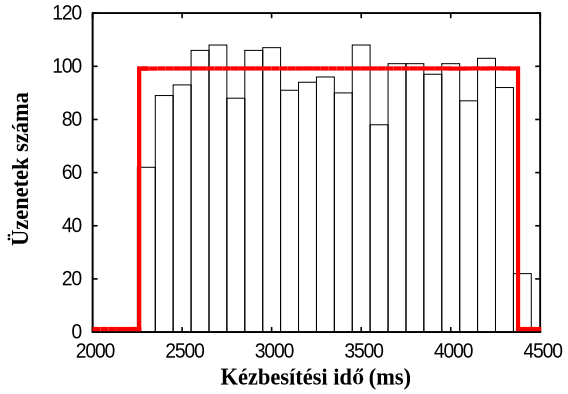
3.12. ábra. Két egymást követő burst időzítése.

A b esetben a kommunikáció iránya az a-hoz képest fordított; itt a 6-os csomópont küld adatot az 5-ösnek. Hasonlóan az előző esethez, az esemény után itt is meg kell várni a forrás csomópont adási időszeletét; ez az időtartam 0 és 2100 ms közötti egyenletes eloszlással rendelkezik. Majd a burst-nek a 6-os csomóponttól kezdődő része következik, ami alatt az üzenet eljut a forrástól a burst-ben szereplő utolsó csomópontig. A burst után egy inaktív, kommunikáció nélküli időszak következik, majd egy újabb burst kezdődik, ami az üzenetet a célba juttatja, ahogy az a 3.12. ábrán látható. A két részidő összege 2280 és 4360 ms közötti egyenletes eloszlással rendelkezik, ahogy az a 3.13(b) ábrán piros színnel szerepel.

A forrás csomópont aszinkron eseményének létrehozására és a cél csomóponton az üzenet vételének jelzésére egy mérőeszközt készítettem. Az adatgyűjtő szoftver a triggereket véletlenszerű időközönként állította elő és az esemény létrejöttétől az üzenetnek a cél csomóponton való megjelenéséig eltelt időt mérte. Mindkét beállítást 2000 eseménnyel teszteltem; az egyes beállítások esetében mért idők hisztogramjai a 3.13. ábrán láthatók. A hisztogramokat 100 ms méretű rekeszekkel készítettem. A mért értékek jól egybevágnak a piros színnel jelölt elméleti eloszlásokkal.



(a)



(b)

3.13. ábra. A mért kézbesítési idő eloszlása a körkörös TDMA esetében az első (a), illetve a második (b) kísérlet során. Az elméleti eloszlást piros szín jelöli.

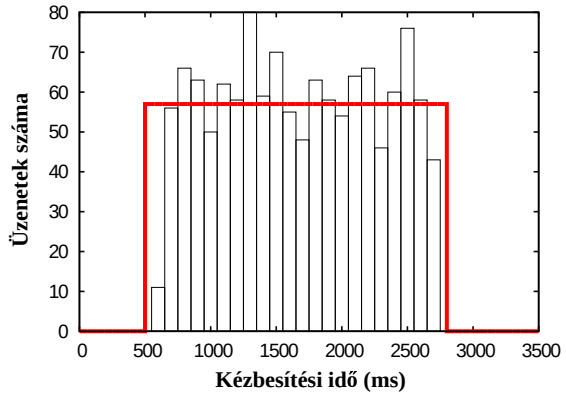
Többkörös TDMA

A többkörös TDMA esetében a forrás és a cél kiválasztására sokkal több lehetőségünk van. A két csomópont sorrendjén kívül fontos tényező még az is, hogy melyikük melyik körön helyezkedik el. A következőkben két tipikus, szignifikánsan eltérő esetet mutatok be.

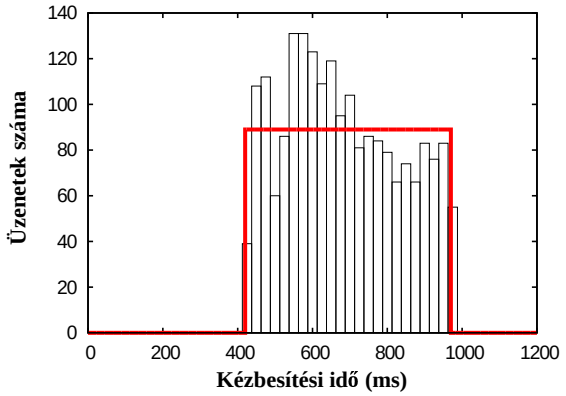
Az *a* esetben a forrás a lassabb főkörön levő 6-os csomópont, a cél pedig a gyorsabb alkörön levő 3-as csomópont (3.9(b) ábra). Hasonlóan az előző kísérletekhez, a forrás csomópontnak itt is meg kell várni az adási időszeletét; ez az idő 0 és 2100 ms közötti egyenletes eloszlással rendelkezik. Ezt követően az információt a csomópontok egymásnak átadva eljuttatják a cél csomóponthoz, ami a 3.11. ábrán látható ütemezés szerint 450 ms-ig tart. Így a kézbesítési idő 450 ms és 2550 ms közötti egyenletes eloszlással rendelkezik (lásd a 3.14(a) ábrán piros színnel).

A *b* esetben mind a forrás (5-ös csomópont), mint a cél (2-es csomópont) a négyszeres sebességű körben van. A késleltetés itt is két részből áll. Először a forrás csomópont adási időszeletét kell megvárni, ami a négyszeres sebességű kör miatt csak a teljes periódusidő negyede, tehát 525 ms. Majd az üzenet további 420 ms alatt a célba ér. Így a teljes kézbesítési idő 420 ms és 945 ms közötti egyenletes eloszlással rendelkezik (lásd a 3.14(b) ábrán piros színnel).

A kézbesítési idő mérésénél a körkörös TDMA esetében leírtaknak megfelelően jártam el. Az *a* esetben 1250 kísérletet végeztem, a mért kézbesítési időkből készült hisztogram a 3.14(a) ábrán látható. A *b* esetben 2000 kísérletet végeztem, a kísérletek hisztogramja a 3.13(b) ábrán látható. A mérési eredmények a többkörös TDMA esetében is jó egyezést mutatnak az elméleti eloszlásokkal.



(a)



(b)

3.14. ábra. A mért kézbesítési idő eloszlása a többkörös TDMA esetében az első (a), illetve a második (b) kísérlet során. Az elméleti eloszlást piros szín jelöli.

3.4.4. Hibatűrés

A hibatűrést egy 4 eszközt tartalmazó hálózatban vizsgáltam. A kevés eszköz lehetővé tette, hogy egyszerre figyeljem meg az összes csomópont működését, de már elegendő a hibajavító mechanizmus működésének bemutatásához. A hibákat csomópontok kikapcsolásával, illetve a nyugta küldésének mesterséges letiltásával szimuláltam. A rádiók ki- illetve bekapcsolt állapotát az előző kísérletekhez hasonlóan négycsatornás oszcilloszkóppal rögzítettem. Az 1., 2., 3., illetve 4. eszközhöz rendre a sárga, zöld, kék, illetve piros görbe tartozik. A 3.15(a) ábrán a referenciának tekinthető hibamentes eset látható, ebben az esetben a 3.10. ábrával analóg működést tapasztalhatunk.

A 3.15(b) ábrán a 2. csomópontot kikapcsoltam. Az 1. csomópont a (sikertelen) küldés (Tx) után megpróbálja még egyszer elküldeni az üzenetet (Retry). A két elmaradó nyugta után az 1. csomópont lehallgatja a 2. és 3. csomópontok közötti kommunikációt. Ez idő alatt a 3. csomópont is vételkész állapotban van, de nem vesz üzenetet. A kimaradó adás után az 1. csomópont egy áthidaló üzenetet (LongTx) küld a 3. csomópontnak. A 3. és a 4. csomópontok ezt követően a szokásos ütemezés szerint folytatják működésüket.

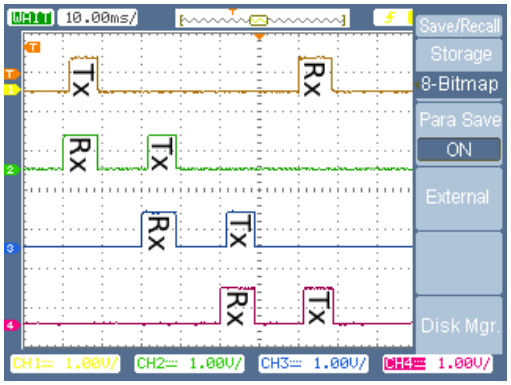
A 3.15(c) ábrán a rendszernek egy elveszett nyugta esetén tapasztalható viselkedése látható. Az 1. csomópont egy üzenetet küld a 2. csomópontnak, amit az vesz, viszont a csomag nyugtája elveszik. Az 1. csomópont ennek megfelelően megismétli az üzenetet. Mivel az első üzenet rendben megérkezett, a 2. csomópont kikapcsolja a rádióját, így a második üzenetet már nem veszi, tehát nyugtát sem küld. A 2. csomópont a saját adási időszeletében egy üzenetet küld a 3. csomópontnak, amit az sikeresen vesz. Ugyanebben az időszeletben az üzenetet az 1. csomópont lehallgatja és az LOK mezőt igazra állítva találja. Ebből arra következtet, hogy a 2. csomópont megkapta az üzenetet, így nincs szükség áthidaló üzenetre.

3.4.5. Időszinkronizáció

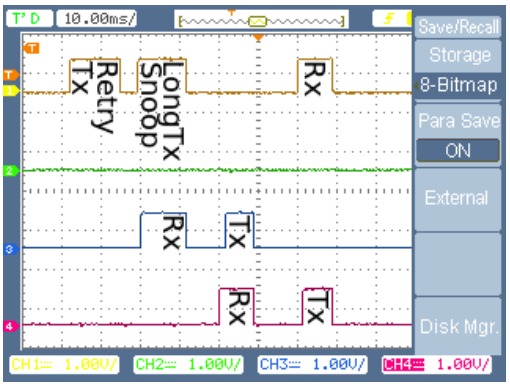
Az oszcilloszkóppal az időszinkronizációs hibák is mérhetők voltak. A körkörös TDMA ütemezés esetén a maximális időszinkronizációs hiba 150 µs, a kommunikáló párok közötti maximális hiba pedig 16 µs volt. A többkörös TDMA esetén a körkörös TDMA-hoz hasonló értékek adódtak. A maximális globális hiba ebben az estben is 150 µs, a kommunikáló párok között mért maximális hiba a főkörön 16 µs volt. A mellékkörök, azok nyílt volta miatt magasabb, 72 µs-os maximális páronkénti hibát mutattak. Fontos megjegyezni, hogy a szinkronizációhoz használt óra felbontása 16 µs.

3.4.6. Energiahatékonyság

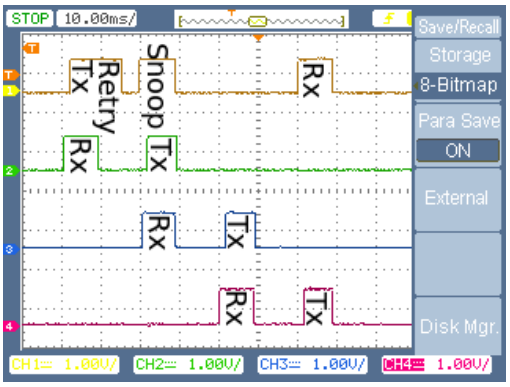
Az energiahatékonyság mérésénél két tényezőt vettem figyelembe: a rádió és a mikrovezérlő fogyasztását. Ezeket közvetett módon, csak az aktivitási időket mérve becsültem. Az előző mérésekhez hasonlóan a rádió bekapcsolt állapotát egy LED-del jeleztem. A TinyOS egy speciális lehetőségét felhasználva a processzor aktív állapotát ugyancsak egy LED-en jelenítettem meg. Méréseim alapján mind a körkörös, mind a többkörös TDMA esetében a korábbiakban ismertetett időzítési beállításokat használva a rádióhasználat kitöltési tényezője 0,8%, az átlagos processzorhasználat 0,2% volt.



(a)



(b)



(c)

3.15. ábra. A rádiók ki- illetve bekapcsolt állapota egy 4 csomópontot tartalmazó hálózatban. (a) Hibamentes működés. (b) A 2-es csomópont nem működik. (c) Elveszett nyugta.

3.5. Összefoglalás

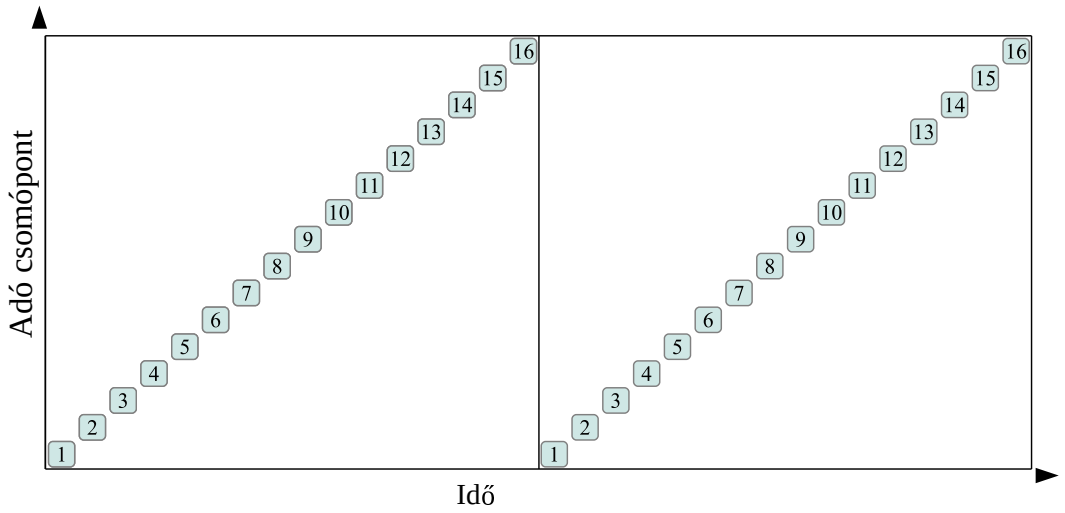
Az előzőkben egy körkörös hálózatokban használható garantált kézbesítési időt nyújtó, hibatűrő és energiahatékony TDMA algoritmust, illetve az algoritmust megvalósító köztesréteg szolgáltatást mutattam be. A rendszer működését többkörös topológiára is kiterjesztettem, a rendszer jó tulajdonságainak (a garantált kézbesítési időnek, a hibatűrésnek és az energiahatékonyságnak) megtartásával. A rendszert valós hardverkörnyezetben implementáltam, illetve behatóan teszteltem. A tesztek során a mért kézbesítési idők jó egyezést mutattak az elméleti értékekkel. A hibatűrő mechanizmus révén a rendszer többféle hiba fellépése esetén is működőképes maradt. A tesztek során a rádiók kitöltési tényezője 0,8%, a processzorhasználat pedig átlagosan 0,2% volt, ami igen alacsony energiafogyasztást [1] jelent.

[1] A szakirodalom szerint jellemzően az 1%-os, vagy az alatti kitöltési tényező tekinthető igen alacsonynak.

4 Optimális multi-TDMA ütemezések

4.1. Áttekintés

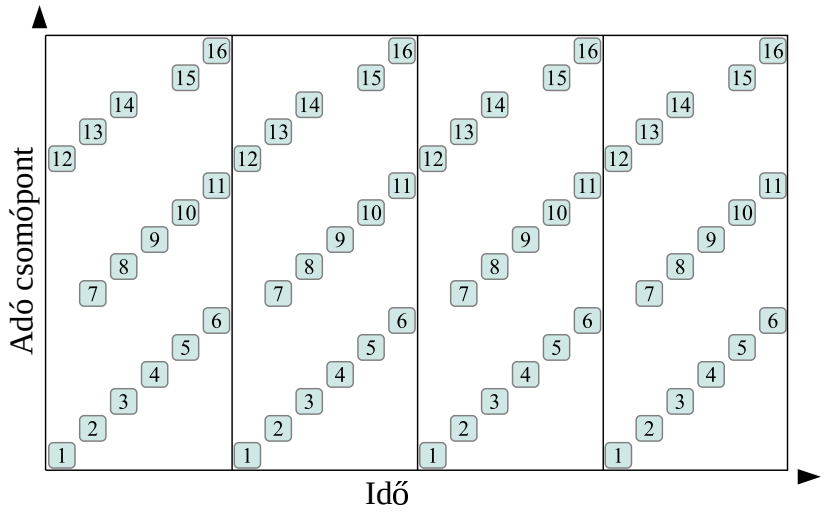
A TDMA lényege az, hogy az adó csomópontok számára dedikált időszeleteket biztosítunk, így elkerülve az ütközést. A lineáris körkörös TDMA egy előre definiált periodikus ütemezést használ. A periódusok időszeletekre vannak osztva, és minden adó-vevő párhoz egy-egy időszelet van rendelve. A csomópontok egy kör mentén helyezkednek el. A periódusok első időszeletében az első csomópont ad egy csomagot a szomszédjának, a későbbi időszeletekben pedig minden alkalommal az a csomópont fog egy csomagot küldeni a szomszédjának, amelyik az előző időszeletben mint vevő szerepelt, egészen az utolsó időszeletig, ahol az utolsó csomópont ad az elsőnek. A periódus a rendszer működése során folyamatosan ismétlődik, ahogy az a 4.1. ábrán látható.



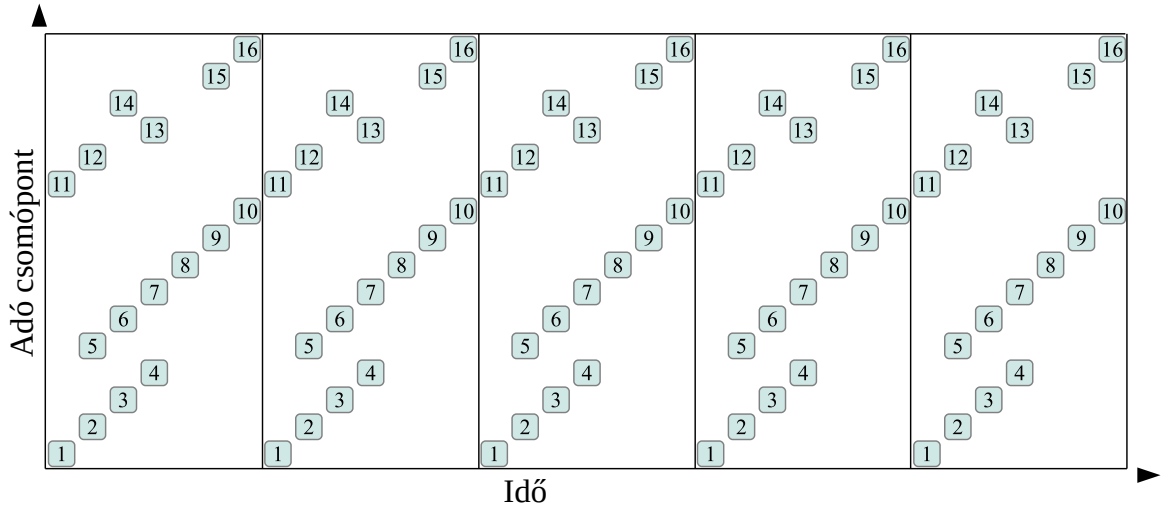
4.1. ábra. A lineáris TDMA ütemezés időzítési diagrammja 16 csomópontra. Minden csomópont a számára kijelölt időszeletben adhat, ilyenkor az időszeletet kizárólagosan használja. Az ábrán két periódus látható, melyek 16-16 időszeletet tartalmaznak.

Multi-TDMA esetén egy időben több adás is megengedett, amennyiben azok nem zavarják egymás vételét. A 4.2. és a 4.3. ábrákon két lehetséges multi-TDMA ütemezés látható 16 csomópontra. A 4.2. ábrán a periódus hossza 6, amit háromszor kell megismételni ahhoz, hogy egy üzenet minden csomóponthoz eljuthasson. A 4.3. ábrán a periódus hossza 7, itt 4 periódus szükséges ahhoz, hogy egy üzenet minden csomóponthoz eljusson.

A multi-TDMA hálózatok használata főleg akkor előnyös, ha alacsony késleltetésre van szükség. Tegyük fel, hogy egy csomópont egy eseményt érzékel és erről egy üzene-



4.2. ábra. Egy multi-TDMA ütemezés időzítési diagramja 16 csomópontra. A független adások ugyanabban az időszeletben foglalnak helyet. Egy periódus 3 időszelet hosszúságú. A példában 3 periódus szükséges ahhoz, hogy egy üzenet eljusson egy csomóponttól az összes többi csomópontig.

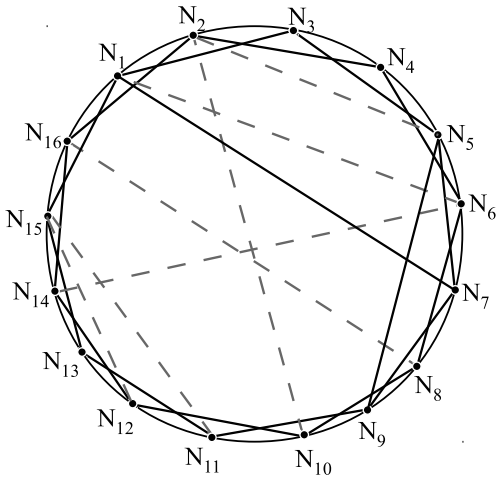


4.3. ábra. Egy multi-TDMA ütemezés időzítési diagramja 16 csomópontra. A független adások ugyanabban az időszeletben foglalnak helyet. Egy periódus 3 időszelet hosszúságú. A példában 3 periódus szükséges ahhoz, hogy egy üzenet eljusson egy csomóponttól az összes többi csomópontig.

tet küld szét a hálózatban. Tekintsük a 4.1. ábrán látható lineáris TDMA ütemezést. Itt legrosszabb esetben 31 időszeletre van szükség ahhoz, hogy az eseményről minden csomópont értesüljön. Először meg kell várni, amíg az adott csomópont időszelete következik, ez legfeljebb 16 időszeletnyi várakozást jelent. Ezt követően a csomópontok egymás után adnak, így elterjesztve az üzenetet a hálózatban, ami újabb 15 időszeletbe kerül. A 4.2. ábrán látható multi-TDMA ütemezést használva a legrosszabb estre számított késleltetés csak 23 időszelet (legfeljebb 6 időszelet amíg egy adott csomópont adhat, majd $3 \cdot 6 - 1$ időszelet amíg az információ a teljes hálózatban elterjed). Jegyezzük meg, hogy a multi-TDMA ütemezés nem mindig ad rövidebb késleltetési időt; a 4.3. ábrán látható ütemezés esetében a késleltetés a legrosszabb esetet tekintve 34 $(7 + 4 \cdot 7 - 1)$ időszelet.

4.2. Az ütemezési algoritmus

A hálózatot egy $C = (N, L)$ kapcsolati gráf írja le, ahol N az egyes szenzor csomópontokat jelképező csúcsok, L pedig a köztük lehetséges kapcsolatokat leíró élek halmaza. Az N halmaz elemszámát n jelöli.



4.4. ábra. Egy lehetséges kapcsolati gráf (folytonos és szaggatott vonalak) és egy kommunikációs gráf (folytonos vonalak).

Az $lq : L \to (0, 1]$ *kapcsolatminőség-függvény* az egyes kapcsolatok minőségét írja le egy előre definiált metrika szerint. Csak azok a linkek használhatók tényleges kommunikációra, ahol a kapcsolatminőség átlép egy bizonyos z küszöbértéket:

$$L_C = \{p \in L | lq(p) > z\}, \tag{4.1}$$

a C_C *kommunikációs gráf* pedig a következő:

$$C_C = (N, L_C) \tag{4.2}$$

A multi-TDMA ütemezés kialakításakor a gyengébb kapcsolatokat is figyelembe kell venni, mert a gyengébb jelszintű interferencia is okozhat csomagütközést, ha egy vevő több adót is hall egy időben.

A 4.4. ábrán egy 16 csomópontot tartalmazó hálózat látható. A gráfon a folyamatos vonallal jelölt élek a jó ($lq > z$), a szaggatott vonallal jelölt élek a gyengébb ($0 < lq \leq z$) kapcsolatokat jelentik.

A tárgyalt kommunikációs modellben minden csomópont a rákövetkezőjének ad, tehát a C_C kommunikációs gráf egy Hamilton-kört kell, hogy tartalmazzon. Az általánosság megszorítása nélkül feltehetjük, hogy $N = \{N_1, \ldots, N_n\}$ és a Hamilton-kör élei az $(N_1, N_2), (N_2, N_3), \ldots, (N_{n-1}, N_n), (N_n, N_1)$ élek, ahogy a 4.4. ábra 16 csúcsú gráfján is látható.

4.2.1. Definíciók

4.2.1. Definíció. Minden N_i csomópont a *vevőjének* ad, amit $r(N_i)$-vel jelölünk.

$$r(N_i) = N_{(i \bmod n)+1}, 1 \leq i \leq n \tag{4.3}$$

4.2.2. Definíció. A $G_I = (N, I)$ *adás-interferencia gráf* jelöli az adási interferenciákat. Két csomópont között akkor vezet él az adás-interferencia gráfban, ha ezek egy időszeletben történő adása nem megengedett, vagyis:

$$(p, q) \in I \text{ ha } (p, r(q)) \in L \text{ vagy } (q, r(p)) \in L \text{ vagy } q = r(p) \text{ vagy } p = r(q) \tag{4.4}$$

4.2.3. Definíció. Csúcsok egy T halmaza akkor *udvarias*, ha a halmazban levő összes csomópont ütközés nélkül adhat ugyanabban az időszeletben. POL jelöli az összes udvarias csomóponthalmazt tartalmazó halmazt, tehát:

$$POL = \{T \subseteq N \mid \forall p, q \in T, p \neq q : (p, q) \notin I\} \tag{4.5}$$

4.2.4. Definíció. Egy *ütemezés* csomóponthalmazok egy $S = (S_1, \ldots, S_l)$ sorozata, ahol $S_i \subseteq N$, l az ütemezés hossza és S_i az i-edik időszelet adó csomópontjait tartalmazza. Minden csomópont pontosan egyszer ad a vevőjének egy periódus során, tehát

$$1 \leq i < j \leq l : S_i \cap S_j = \emptyset, \forall i, j \tag{4.6}$$

és

$$\bigcup_{1 \leq i \leq l} S_i = N. \tag{4.7}$$

A 4.5 ábrán a 4.3 ábrán látható idődiagramhoz tartozó ütemezés látható, ahol $S_1 = \{N_1, N_{11}\}$, $S_2 = \{N_2, N_5, N_{12}\}$, $S_3 = \{N_3, N_6, N_{14}\}$, $S_4 = \{N_4, N_7, N_{13}\}$, $S_5 = \{N_8\}$, $S_6 = \{N_9, N_{15}\}$ és $S_7 = \{N_{10}, N_{16}\}$.

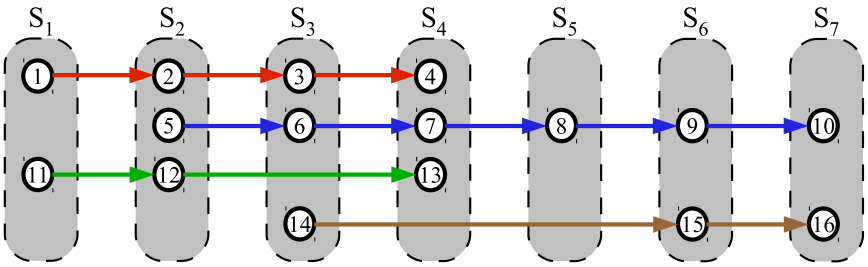
4.2.5. Definíció. Egy adott S ütemezéshez egy $J(S) = (N, Y)$ *ütemezés-kapcsolati gráf* definiálható, ahol

$$(p, q) \in Y \text{ if } \exists i, j : i < j, p \in S_i, q \in S_j, q = r(p). \tag{4.8}$$

A (p, q) irányított él jelentése az, hogy a q csomópont továbbadhatja a p csomóponttól vett üzenetet valamelyik következő, de még az aktuális periódushoz tartozó időszeletben. A 4.5 ábrán a 4.3 ábrán látható időzítési diagramhoz tartozó ütemezés-kapcsolati gráf látható. Itt a 2-es és a 3-as csomópont között vezet irányított él, mert a 3-as csomópont tovább tudja adni a 2-estől a 2. időszeletben vett üzenetet a 4-esnek, még ugyanennek a periódusnak a 3. időszeletében. Hasonlóan a 4-es is tovább tudja ezt adni az 5-ösnek, még ugyanebben a periódusban, ennek megfelelően a 3-astól a 4-esig is vezet irányított él. De az 5-ös már csak a következő periódusban tudja továbbadni a 4-estől kapott üzenetet, így a 4-es csomópontból nem mutat él az 5-ösbe,

4.2.6. Definíció. Egy S *ütemezés szélessége* a $J(S)$ ütemezés-kapcsolati gráfban található független utak számával egyenlő.

A 4.5 ábrán szereplő ütemezés szélessége 4. Az ütemezés szélességének van egy igen fontos gyakorlati jelentése is: Egy k szélességű ütemezést k-szor kell megismételni ahhoz, hogy egy csomag a hálózat minden csomópontjához eljusson.



4.5. ábra. A 4.3 ábrán látható példa ütemezés-kapcsolati gráfja. Magát az ütemezést az $S_1, S_2, \ldots, S_7$ halmazok jelölik. Az ütemezés szélessége 4, ahogy a 4 egyszínű út alapján látható.

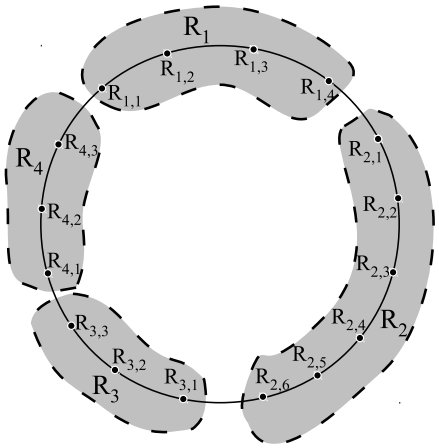
4.2.7. Definíció. A Hamilton-kör diszjunkt utakra való felosztása után az egyes utak csúcshalmazát *szegmensnek* nevezzük.

Az ütemezés-kapcsolati gráfon szereplő független utak egyben (független) szegmensek is. A 4.5 ábrán a piros, kék, zöld és barna utak egy-egy független szegmenshez tartoznak. Ezek alapján a következő definíció természetesen adódik.

4.2.8. Definíció. Bármely k szélességű ütemezéshez egy $R = (R_1, \ldots, R_k)$ *szegmentáció* definiálható a Hamilton-körön, ahol $R_i = (R_{i,1}, \ldots, R_{i,l_i})$ egy szegmens, továbbá $(R_{1,1}, \ldots, R_{1,l_1}, \ldots, R_{k,1}, \ldots, R_{k,l_k}) = (N_1, \ldots N_n)$, ahol az i-edik szegmens hosszát l_i, az R_i halmazokban található csúcsokat pedig $R_{i,1}, \ldots, R_{i,l_i}$ jelöli.

Egy k szélességű ütemezés tehát k független szegmenst tartalmaz, amelyek uniója N.

A 4.6 ábrán a 4.5 ábrához tartozó szegmentáció látható.



4.6. ábra. A Hamilton-kör egy lehetséges szegmentálása.

4.2.9. Definíció. Egy adott R szegmentáció esetén a $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ *állapot* csomópontok egy olyan R_{j,i_j}, $0 \leq i_j \leq l_j$ halmaza, ahol az R_{j,i_j} az a csomópont, amelyik a saját R_j szegmensében utoljára adott. Ha egy adott R_j szegmensben még nem történt adás, akkor ott az $R_{j,0} = \otimes$ jelölést használjuk. Arra az állapotra, ahol még semelyik szegmensben sem történt adás, a $\langle \otimes, \otimes, \ldots, \otimes \rangle = (\otimes)_k$ jelölést használjuk. Egy P állapot i-edik elemét, vagyis azt az elemet, amelyik az R_i szegmenshez tartozik, $P[i]$-vel jelöljük.

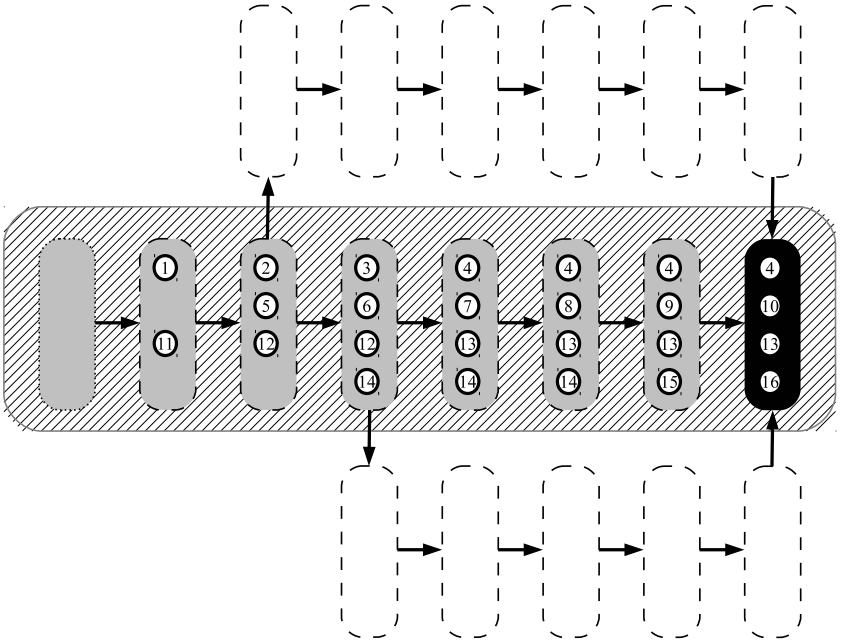
A k szélességű ütemezés megalkotásához az $A_R = (\mathcal{G}, D)$ irányított gráfot használjuk. Az A_R élei jelölik a lehetséges ütemezések állapotait. Az élek az állapotok lehetséges sorrendjét definiálják.

Egy $(P, Q) \in D$ átmenet esetén az aktuálisan adó csomópontok halmazát a

$$\bigcup_{i=1}^{k} (\{Q[i]\} \setminus \{P[i]\}) \tag{4.9}$$

kifejezés adja meg.

A 4.7. ábra kiemelt része a 4.5. ábrán látható ütemezéshez tartozó állapotokat és a köztük levő átmeneteket adja meg. Az állapotgráf a $(\otimes)_4 = \langle \otimes, \otimes, \otimes, \otimes \rangle$ állapotból indul. Az S_1-hez tartozó időszelet után az 1-es és a 11-es azok a csomópontok, amelyek a saját szegmensükben utoljára adtak (a másik két szegmensben ekkor még nem történt adás), tehát ez az $\langle N_1, \otimes, N_{11}, \otimes \rangle$ állapot. Az S_2-höz tartozó időszelet után a 2-es, 5-ös és 12-es csomópontok voltak a saját szegmensükben utoljára adók, ami az $\langle N_2, N_5, N_{12}, \otimes \rangle$ állapotot eredményezi. Az S_3-hoz tartozó időszelet után a 3-as, 6-os, 12-es és 14-es lesznek a szegmensükben utoljára adó csomópontok, tehát ez az állapot az $\langle N_3, N_6, N_{12}, N_{14} \rangle$. Érdemes megjegyezni, hogy a 12-es csomópont szegmenséből nem történt adás, tehát a 12-es csomópont maradt az állapotban. A 4.5. ábrán ezt jelöli az S_3 állapoton keresztül haladó zöld színű út.



4.7. ábra. Az állapotgráf egy részgráfja. A kiemelt út egy optimális ütemezést ír le.

Az A_R formális definíciója a következő.

4.2.10. Definíció. Egy $R = (R_1, \ldots, R_k)$ szegmentáció esetén az $A_R = (\mathcal{G}, D)$ gráfot az R szegmentáció *állapotgráfjának* nevezzük. Az A_R gráf $\mathcal{G}$ csúcshalmaza azokat az állapotokat jelöli, amelyek valamely, az R szegmentációval kompatibilis ütemezés során előfordulhatnak. Az állapotgráf élei a következő definícióval adottak.

$(P, Q) \in D$ ha:

TR1 $\forall i : Q[i] \neq \otimes \wedge P[i] = \otimes \Rightarrow Q[i] = R_{i,1}$, tehát minden R_i szegmensben az első csomópont ($R_{i,1}$) fog először adni.

TR2 $\forall i : P[i] \neq \otimes \Rightarrow Q[i] \neq \otimes$. Ez a szabály azt a triviálisnak tűnő feltételt fogalmazza meg, mely szerint ha egy szegmensben történt már adás, akkor ez az állítás később is igaz marad, tehát található olyan csúcs, amelyik az adott szegmensből utoljára adott.

TR3 $\forall i : P[i] \neq \otimes \Rightarrow V \vee W$, ahol $V = (Q[i] = P[i]), W = (P[i] = R_{i,j} \wedge Q[i] = R_{i,j+1} \wedge j < l_i)$, tehát ha valamely szegmensben ad egy csomópont, akkor a csomópont vevője vagy a következő szegmensben fog adni (W feltétel), vagy valamelyik későbbiben (V feltétel). Utóbbi esetben az adott szegmensbeli utoljára adó csomópont változatlan marad.

TR4 $\left(\bigcup_{i=1}^{k} (\{Q[i]\} \setminus \{P[i]\}) \right) \in POL$, tehát csak egy udvarias adóhalmaz adhat ugyanabban az időszeletben.

A TR1 – TR4 szabályokat *levezetési szabályoknak* nevezzük.

4.2.2. Ütemező algoritmusok

Ebben a szakaszban a Fixed-Segmentation-Scheduler (FSS) és az Optimal-Multi-TDMA-Scheduler (OMTS) algoritmusok, illetve az OMTS gyorsított változata, az OMTS-A kerülnek bemutatásra.

Fixed-Segmentation-Scheduler (FSS)
input
  $C = (N, L)$: kapcsolati gráf
  $C_C = (N, L_C)$: kommunikációs gráf
  $\mathcal{R} = \{R_1, \ldots, R_k\}$: szegmensek halmaza
variables
  $\widehat{\mathcal{G}}$: az A levezetett állapotainak halmaza
  $\mathcal{O}$: az A nyílt csúcsainak halmaza
  P: az aktuálisan levezetés alatt álló csúcs
  T: adó szegmensek egy lehetséges kombinációja
  Q: egy lehetséges csomópont, amibe mutat irányított él T-ből
  parent: csomópontokhoz csomópontokat rendelő leképezés
begin
  $\widehat{\mathcal{G}} = \{(\otimes)_k\}$
  $\mathcal{O} = ((\otimes)_k)$
  while $E_R \notin \mathcal{O}$
    $P \leftarrow dequeue(\mathcal{O})$
    foreach T **in** $\mathcal{P}(\mathcal{R}) \setminus \emptyset$
      $Q = \{R_{i,1} : R_i \cap P = \emptyset, R_i \in T\}$
        $\cup\{r(R_{i,j}) : R_i \cap P = \{R_{i,j}\}, R_i \in T\}$
      if (P, Q) kielégíti a levezetési szabályokat **and** $Q \notin \widehat{\mathcal{G}}$:
        $\widehat{\mathcal{G}} \leftarrow \widehat{\mathcal{G}} \cup Q$
        $enqueue(\mathcal{O}, Q)$
        $parent(Q) \leftarrow P$
  $Q = E_R$
  $S = ()$
  while $Q \neq (\otimes)_k$
    $P = parent(Q)$
    $enqueue\left(S, \bigcup_{i=1}^{k} (\{Q[i]\} \setminus \{P[i]\})\right)$
    $Q = P$
  return $reverse(S)$
end

Jelölésekkel kapcsolatos megjegyzések:

- $dequeue(L)$ kiveszi az L lista első elemét, $enqueue(L, X)$ az X elemet hozzáfűzi az L lista végéhez, lásd a 16., 21. és 27. sorokat.
- $reverse(L)$ visszaadja az L lista megfordítását, lásd a 29. sort.
- $parent(Q)$ tárolja azt az állapotot, amiből Q-t levezettük, lásd a 22. és 26. sorokat.

Az algoritmus bemenete a C kapcsolati gráfból, a C_C kommunikációs gráfból és az $R = (R_1, \ldots, R_k)$ szegmentálásból áll. Az algoritmus futása során az $\widehat{A}_R = (\widehat{\mathcal{G}}, \widehat{D})$ gráfot építi fel, ahol $\widehat{\mathcal{G}} \subseteq \mathcal{G}$, $\widehat{D} \subseteq D$, és a $\widehat{D}$ élhalmazt a $parent()$ leképezés adja meg, lásd a 17–22 sorokat.

Az iterációk során $\mathcal{G}$ a szélességi keresés által eddig érintett állapotokat tartalmazza (röviden érintett csúcsok), $\mathcal{O}$ pedig azokat a (már $\mathcal{G}$-ben is szereplő) csúcsokat, amik lehetséges gyerekeit még nem vizsgálta meg a szélességi keresés (röviden nyílt csúcsok).

Kezdetben a nyílt csúcsokat tartalmazó $\mathcal{O}$ lista csak az üres halmazt tartalmazza (14. sor); az algoritmus minden iterációban a legrégebben a halmazban levő állapotból generálja az adó csomópontok összes lehetséges kombinációját (18. sor), amivel az éppen vizsgált állapot a levezetési szabályoknak megfelelően folytatható (19–21. sorok). Az algoritmus a kimenetként szolgáltatott ütemezést az $\widehat{A}_R$-ben található állapotok sorozatából állítja össze az E_R végállapottól az $(\otimes)_k$ kezdőállapotig visszafelé haladva (23–28. sorok).

Fontos megjegyezni, hogy az FSS algoritmus csak egy rögzített szegmentálásra vonatkozóan generálja az optimális ütemezést. Az összes szegmentálásra vonatkozó optimum az OMTS algoritmus segítségével generálható, ami az összes lehetséges szegmentálásra futtatja az FSS-t.

Az OMTS algoritmus pszeudokódja a következőkben látható. A csillag nélküli sorok az OMTS algoritmust adják meg. Az OMTS-t a csillagozott sorokkal kiegészítve kapjuk az OMTS-A algoritmust, ami az OMTS korai vágásokkal gyorsított variánsának tekinthető.

Optimal-Multi-TDMA-Scheduler (OMTS és OMTS-A*)

1 **input**
2 $C = (N, L)$: kapcsolati gráf
3 $C_C = (N, L_C)$: kommunikációs gráf
4 $k_{\max}$: az ütemezés maximális szélessége
5 **variables**
6 $minSchLen$: a legrövidebb ismert ütemezés hossza
4 k: a jelenleg vizsgált szélesség
5 $startNodes$: a vizsgálandó szegmentációk szegmenseinek kezdő csomópontjaiból képezett k-asok halmaza
6 $maxSegLength$: egy szegmentáció leghosszabb szegmensének mérete

```
7    B: egy szegmentáció kezdő csomópontjaiból képezett k-as
8    R: egy szegmentáció
9    S_R: R egy optimális ütemezése
10   S: egy optimális ütemezés
11 begin
12   minSchedLen = ∞
13   for k = k_max downto 1
14     startNodes = ()
15     foreach B in {B : B ⊆ N ∧ |D| = k}
16       startNodes = enqueue(startNodes, B)
17*      maxSegLength(B) =
18*        max((j − i + n) mod n), i ≠ j, N_i, N_j ∈ B
19*    sort startNodes by maxSegLengths increasing order
20     foreach B in startNodes
21*      if maxSegLength(B) < minSchedLen
22         R = (R_1, ..., R_k) : R_{i,1} ∈ B, 1 ≤ i ≤ k
23         S_R = FSS(C, C_C, R)
24         if length(S_R) < minSchedLen
25           S = S_R
26           minSchedLen = length(S_R)
27*      else
28*        break
29   return S
30 end
```

Az algoritmus bemenete a C kapcsolati gráf, a C_C kommunikációs gráf és a vizsgálni kívánt $k_{\max}$ maximális ütemezési szélesség. A *len* változó tárolja az aktuálisan legrövidebbnek ismert ütemezés hosszúságát; ezt az algoritmus kezdetben ∞-nek állítja be (12. sor). A gyorsítás nélküli algoritmus $k_{\max}$-tól 1-ig iterál végig az egyes k értékeken (13. sor); minden esetben az összes lehetséges, a szegmentációk kezdő csúcsaiból álló k-as generálásra kerül (15. sor); ezeket a kezdő k-asokat a *startNodes* lista tárolja (16. sor). A *maxSegLength* leképezés minden kezdő k-ashoz a hozzá tartozó szegmentációban előforduló legnagyobb szegmens hosszát adja meg (17–18. sorok). Az algoritmus a *startNodes* listán iterál végig (20. sor), és a 22. sorban generálja a kezdő csomópontokból a szegmentációkat (22. sor). Minden szegmentációra lefuttatja az FSS algoritmust (23. sor), ami megadja az aktuális szegmentációra vonatkozó legrövidebb S_R ütemezést.

Az OMTS algoritmus az összes lehetséges szegmentációt megvizsgálja, az OMTS-A ezzel szemben kihagyja azokat az eseteket, amik semmiképpen nem eredményezhetnének rövidebb ütemezést, mint az eddigi legjobb; ezekre az esetekre nem futtatja az FSS algoritmust. Ehhez a szegmentáció legnagyobb szegmensének hosszát vizsgálja, ez ugyanis egy alsó korlátja a lehetséges ütemezések hosszának. Az OMTS-A tehát csak azokat a szegmentációkat vizsgálja, ahol a $maxSegLength < maxSchedLen$ feltétel teljesül (lásd a 21., 27. és 28 sorokat). A korai vágások hatékonyságának növelése érdekében az algoritmus a szegmentációkat az iterációk megkezdése előtt a *maxSegLength* értéke szerint rendezi (lásd 17–19 sorok).

4.2.3. Az algoritmusok tulajdonságai

4.2.1. Lemma. Egy adott $(R_1, \ldots, R_k) = ((R_{1,1}, \ldots, R_{1,l_1}), \ldots, (R_{k,1}, \ldots, R_{k,l_k}))$ szegmentációra az $E_R = \langle R_{1,l_1}, R_{2,l_2}, \ldots, R_{k,l_k} \rangle$ végállapot az összes lehetséges ütemezés végállapotát, tehát az adott szegmentációra vonatkozó állapotgráfban található összes lehetséges irányított út végpontját adja meg.

Bizonyítás
Minden ütemezésre igaz, hogy egy perióduson belül minden csomópont pontosan egyszer ad, tehát minden szegmens minden csomópontja szerepelni fog legalább egy állapotban. Ha egy szegmens egy csomópontja szerepel egy P állapotban és (P, Q) az állapotgráf egy irányított éle, akkor a Q állapotban is szerepelni fog egy csomópont ugyanebből a szegmensből. Az egy ütemezéshez tartozó állapotokban az egyes szegmensekhez tartozó csomópontok növekvő sorrendben szerepelnek. Így az ütemezések utolsó állapotában minden szegmensből az utolsó csomópont fog szerepelni. ∎

4.2.2. Lemma. Az $R = ((R_{1,1}, \ldots, R_{1,l_1}), \ldots, (R_{k,1}, \ldots, R_{k,l_k}))$ szegmentációhoz tartozó legrövidebb ütemezés az A_R gráfon a $(\otimes)_k$ kezdőállapottól az E_R végállapotig futó legrövidebb út mentén található állapotok sorozatához tartozik.

Bizonyítás
Egy adott szegmentációra az állapotgráf élei a lehetséges ütemezések időszeleteinek felelnek meg. Így minden ütemezés hossza az állapotgráfon neki megfelelő, az $(\otimes)_k$ kezdőállapottól az E_R végállapotig futó út hosszának felel meg. Tehát az $(\otimes)_k$-tól az E_R-ig tartó legrövidebb út a legrövidebb ütemezésnek felel meg. ∎

A következő lemma legalább egy lehetséges ütemezés létezését garantálja.

4.2.3. Lemma. Minden $P \neq E_R$-re létezik olyan $Q = \langle R_{1,j_1}, R_{2,j_2}, \ldots, R_{k,j_k} \rangle$ állapot, ahol (P, Q) kielégíti a levezetési szabályokat.

Bizonyítás
Mivel $P \neq E_R$, létezik olyan i, hogy $R_{i,j_i} \neq R_{i,l_i}$. Legyen $Q[i] = R_{i,j_i+1}$ és $Q[k] = R_{k,j_k}, k \neq i$. Ebben az esetben (P, Q) kielégíti a levezetési szabályokat, hiszen csak egy csomópont (R_{i,j_i+1}) ad. ∎

4.2.11. Definíció. Egy adott $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ állapot és $E_R = \langle R_{1,l_1}, R_{2,l_2}, \ldots, R_{k,l_k} \rangle$ végállapot esetén a P és E_R közötti távolság $D(P) = \sum_{k=1}^{k} l_k - i_k$. Fontos megjegyezni, hogy $D(P) = 0$ akkor és csak akkor, ha $P = E_R$.

4.2.4. Lemma. Bármely két $P = \langle R_{1,i_1}, R_{2,i_2}, \ldots, R_{k,i_k} \rangle$ és $Q = \langle R_{1,j_1}, R_{2,j_2}, \ldots, R_{k,j_k} \rangle$ állapotra, ha (P, Q) kielégíti a levezetési szabályokat, akkor $\Delta_D < 0$, ahol $\Delta_D = D(Q) - D(P)$.

Bizonyítás
$\Delta_D = D(Q) - D(P) = \sum_{m=1}^{k} (l_m - j_m) - \sum_{m=1}^{k} (l_m - i_m) = \sum_{m=1}^{k} ((l_m - j_m) - (l_m - i_m)) = \sum_{m=1}^{k} (i_m - j_m)$

A TR3 szerint minden m-re vagy $j_m - i_m = 0$ (V eset), vagy $j_m - i_m = 1$ (W eset) teljesül. Mivel az állapot legalább egyik eleme megváltozik (W eset), $\Delta_D < 0$. ∎

4.2.5. Lemma. Az FSS algoritmus megáll, tehát véges lépés után $E_R \in \mathcal{O}$ és $E_R \in \widehat{\mathcal{G}}$.

Bizonyítás
Vezessük be a $D_{min}(\mathcal{O}) = min\,(D(O[i]))$ és $i_{min} = \arg\min_i (D(O[i])), 1 \leq i \leq |\mathcal{O}|$ jelöléseket az egyes iterációkra.

Mivel az algoritmus az $\mathcal{O}$ sor elemeit egymás után vizsgálja meg (15–23. sorok), minden iterációban igaz az, hogy az algoritmus a jelenlegi $\mathcal{O}[i_{min}]$ állapotot i_{min} iteráció múlva fogja megvizsgálni. Így a 4.2.3 és a 4.2.4 lemmák szerint D_{min} értéke i_{min} iteráción belül mindenképpen csökkenni fog.

A fentiek ismétlése $D_{min}(\mathcal{O})$ folyamatos csökkenését eredményezi. Mivel $D_{min}(\mathcal{O})$ egész, így értéke véges lépésen belül el fogja érni a 0 értéket, ahol szükségszerűen $E_R \in \mathcal{O}$ (és $E_R \in \widehat{\mathcal{G}}$, lásd a 21–22. sorokat). Az iterációk a 15. sorban állnak meg, ezt követően az algoritmus a 23–30. sorokban fejezi be működését. ∎

4.2.1. Tétel. Az állapotoknak az FSS algoritmus által generált listája egy adott szegmentálásra vonatkoztatva egy legrövidebb ütemezést ad meg.

Bizonyítás
Legyen $\widehat{A}_R = (\widehat{\mathcal{G}}, D)$ az FSS algoritmus által generált ütemezés-kapcsolati gráf. (A D élhalmaz a *parent* leképezés által implicit módon adott.)

Az algoritmus először az $(\otimes)_k$ kezdőállapotot vizsgálja meg (13. sor), így $\widehat{A}_R$ tartalmazza a kezdőállapotot. A 4.2.5 lemma szerint $\widehat{A}_R$ az E_R végállapotot is tartalmazza.

Az algoritmus keresőmódszere (szélességi keresés) garantálja a legrövidebb utat a kezdőállapottól a végállapotig. Mivel az algoritmus betartja a levezetési szabályokat (19. sor), $\widehat{A}_R \subseteq A_R$, ennek megfelelően az $\widehat{A}_R$ az A_R egy komponense lesz. A 4.2.2 lemma szerint az így nyert út egy, az adott szegmentációra lehetséges legrövidebb ütemezéshez tartozik. ∎

4.2.2. Tétel. Az OMTS és az OMTS-A algoritmusok egy optimális (vagyis legrövidebb) ütemezést állítanak elő a megadott (N, L, L_C) csomópont halmaz, link halmaz, kommunikációs link halmaz hármasra, illetve maximális k szélességre.

Bizonyítás
Az OMTS algoritmus a kommunikációs gráfra lehetséges összes szegmentációt végigpróbálja. A 4.2.1 tétel szerint az FSS algoritmus bármely szegmentációra a legrövidebb ütemezést állítja elő. Tehát az OMTS valóban globális optimumot ad.

Az OMTS-A is minden lehetséges szegmentációt leszámlál, azonban nem vizsgálja meg azokat az eseteket, amikor biztosan nem születhetne jobb megoldás, mint a jelenlegi legjobb. Így az OMTS-A szintén globális optimumot ad. ∎

4.2.3. Tétel. Ha a kommunikációs gráfban szereplő csomópontok száma n, az OMTS algoritmus számítási komplexitása az n polinomfüggvénye egy rögzített k_{max} maximális megengedett szélesség esetén.

Bizonyítás
Az FSS algoritmus a 15-22. sorokban vizsgálja meg az egyes állapotokat. A *dequeue* művelet konstans idő alatt végrehajtható. A $\mathcal{P}(\mathcal{R}) \setminus \emptyset$ halmaz mérete rögzített k esetén konstans, így 17-22. sorokban helyet foglaló ciklus iterációinak száma szintén konstans. A 18–22. sorokban levő ciklusmagban szereplő halmazműveletek, a levezetési szabályok vizsgálata, valamint a sor és leképezés frissítések legfeljebb $O(n)$ komplexitással megvalósíthatók. Így tehát egy-egy állapot vizsgálata $O(n)$ komplexitású.

Legyen az FSS algoritmus által megvizsgált állapotok száma (ami megegyezik a 15–22. sorokban levő ciklus iterációinak számával) I_{FSS}. I_{FSS} nyilvánvalóan nem lehet nagyobb az összes állapotok számánál. Tekintsünk egy k szélességű szegmentációt $l_1, l_2, \ldots, l_k$ hosszúságú szegmensekkel. Ekkor egy állapot i-edik eleme $R_{i,1}$, $R_{i,2}$, …, R_{i,l_i} vagy $\otimes$ lehet, ami összesen $l_i + 1$ lehetőség. Tehát egy R szegmentáció esetén lehetséges állapotok száma ($I_{\mathrm{FSS},R}$)

$$I_{\mathrm{FSS},R} = (l_1 + 1) \cdot (l_2 + 1) \cdots (l_k + 1). \tag{4.10}$$

Felhasználva a számtani és a mértani közepek közötti összefüggést:

$$I_{\mathrm{FSS},R} \leq \left(\frac{l_1 + l_2 + \cdots + l_k + k}{k} \right)^k = \left(\frac{n + k}{k} \right)^k, \tag{4.11}$$

ami aszimptotikusan $O(n^k)$ rendű. Figyelembe véve egy állapot vizsgálatának $O(n)$ komplexitását az FSS algoritmus komplexitása

$$C_{\mathrm{FSS}}(n) = O(n^k \cdot n) = O(n^{k+1}). \tag{4.12}$$

Az OMTS algoritmusban lefutó iterációk I_{OMTS} száma, amikor az FSS algoritmus meghívásra kerül, egyenlő az m szélességű szegmentációk összegével, ahol $m = 1, 2, \ldots, k$:

$$I_{\mathrm{OMTS}}(n) = \binom{n}{k} + \binom{n}{k-1} + \cdots + \binom{n}{1} < \; < n^k + n^{k-1} + \cdots + n^1 < k \cdot n^k \tag{4.13}$$

Így:

$$I_{\mathrm{OMTS}}(n) = O(n^k), \tag{4.14}$$

és az OMTS algoritmus komplexitása az I_{OMTS} és C_{FSS} a következő képlet szerint számítható ki:

$$C_{\mathrm{OMTS}}(n) = O(n^k \cdot n^{k+1}) = O(n^{2k+1}) \tag{4.15}$$

Tehát az OMTS algoritmus rögzített k esetén polinomiális komplexitású, $O(n^{2k+1})$ rendű. ∎

4.3. Analitikai eredmények

Az ütemezési módszereket a következőkben két metrika, az üzenetek célba érési ideje, illetve az energiaszükséglet szerint fogom összehasonlítani.

4.3.1. Az üzenet célba érési ideje

A célba érési idő (T_D) az az idő, amíg egy csomag egy csomóponttól az összes többi csomópontba eljut.

Lineáris TDMA ütemezés esetén a célba érkezési idő egy n csomópontból álló hálózat esetén a következő módon számítható ki. A hálózatban aszinkron események (például mérések) történnek, amikre a hálózat üzenetek (például riasztások) küldésével reagál. Az üzenetküldést kiváltó esemény bekövetkezte után a csomópontnak először a saját adási időszeletét kell megvárnia. A legkedvezőtlenebb eset ilyenkor az, ha az adott csomópont épp most adott; ekkor egy teljes periódust (n időszeletet) kell várnia a következő adási lehetőségig. Ezt követően $n-1$ időszelet szükséges ahhoz, hogy az üzenet az összes többi csomópontig is eljusson. Összesen tehát legrosszabb esetben

$$T_{D,linear} = 2n - 1 \tag{4.16}$$

időszelet szükséges ahhoz, hogy egy üzenet egy csomópontból az összes többi csomópontba eljusson.

Egy k szélességű multi-TDMA ütemezés esetén alsó korlát adható a célba érési időre. Legyen p a periódus hossza. Legrosszabb esetben egy csomópontnak p időszeletet kell várnia a legközelebbi adási lehetőségig és $k \cdot p - 1$ időszelet szükséges ahhoz, hogy az üzenet az összes többi csomópontig eljusson, így

$$T_{D,multi} = p + k \cdot p - 1 = (k+1) \cdot p - 1. \tag{4.17}$$

A periódus hossza legalább a szegmentáció legnagyobb szegmensének mérete, így p-re a következő alsó korlát adható:

$$p \geq \frac{n}{k} \tag{4.18}$$

Tehát

$$T_{D,multi} \geq \frac{k+1}{k} n - 1 > n - 1 = \frac{T_{D,linear} - 1}{2}. \tag{4.19}$$

Érdemes megjegyezni, hogy a multi-TDMA ütemezéssel a lineáris TDMA ütemezés periódusa legkedvezőbb esetben is körülbelül 1/2 részére csökkenthető. Ez a korlát akkor közelíthető meg, ha

- a szegmentáció egyenlő méretű szegmenseket tartalmaz,
- minden időszeletben minden szegmensből tud adni egy csomópont,
- és k nagy.

4.3.2. Energiaszükséglet

Az energiaszükséglet két módon is tárgyalható. Azt az energiát, ami egy csomag továbbításához szükséges egy csomóponttól az összes többi csomópontig, lineáris TDMA esetén $E_{packet,linear}$-ral, multi TDMA esetén $E_{packet,multi}$-val jelöljük. Az átlagos energiafogyasztás lineáris TDMA esetén $E_{avg,linear}$, multi TDMA esetén pedig $E_{avg,multi}$.

A modellünkben E_{tx} energia szükséges egy csomag adásához és E_{rx} energia egy csomag vételéhez, az egyéb paraméterektől (például a kapcsolat minősége) függetlenül.

Mind lineáris, mind multi TDMA esetén minden csomópont minden mérési adatot egy perióduson belül pontosan egyszer továbbít. Minden időszeletben E_{tx} energia szükséges a csomag adásához, illetve E_{rx} energia annak vételéhez. Így mind lineáris, mint multi TDMA esetén az egy üzenet elterjesztéséhez szükséges energia az alábbi képlettel számítható ki:

$$E_{packet,linear} = E_{packet,multi} = (n-1) \cdot (E_{tx} + E_{rx}). \tag{4.20}$$

Lineáris TDMA esetében minden időszeletben pontosan egy csomópont ad, illetve egy vesz üzenetet, így az átlagos energiaszükséglet:

$$E_{avg,linear} = E_{tx} + E_{rx} \tag{4.21}$$

Multi TDMA esetében minden csomópont egyszer ad, illetve vesz üzenetet egy perióduson belül, így a multi TDMA ugyanannyi energiát igényel egy p hosszúságú periódushoz, mint a lineáris TDMA egy hosszabb, n hosszúságú periódushoz. Így a multi TDMA átlagos energiaigénye a következő képlet szerint alakul:

$$E_{avg,multi} = \frac{n}{p} \cdot (E_{tx} + E_{rx}) = \frac{n}{p} \cdot E_{avg,linear}. \tag{4.22}$$

Tehát a gyorsabb célba érkezési idő ára a magasabb energiaszükséglet; mivel $p \geq n/k$,

$$E_{avg,multi} \leq k \cdot E_{avg,linear}. \tag{4.23}$$

Az átlagos energiaszükséglet akkor a legnagyobb, amikor a szegmensek egyenlő hosszúságúak és minden időszeletben minden szegmensből ad valamelyik csomópont; ebben az esetben a multi TDMA ütemezés k-szoros energiaigényt eredményez a lineárishoz képest.

4.4. Teszt eredmények

A komplexitás valós környezetben történő méréséhez mind az OMTS, mint az OMTS-A algoritmusokat Perl nyelven implementáltam. Az implementáció, illetve a tesztgráfok generálásához és a futási idők statisztikáinak elkészítéséhez szükséges segédprogramok az [S3] linken érhetők el.

Összehasonlításként egy brute-force algoritmust használó program is készült, ami nem használja a javasolt, állapot alapú megoldást. A teszteléshez véletlenszerűen generált, kontrollált csúcsszámmal (n) és élszámmal (c) rendelkező gráfokat használtam, amiket a következő módon állítottam elő. Egy négyzet alakú területen n csomópontot helyeztem

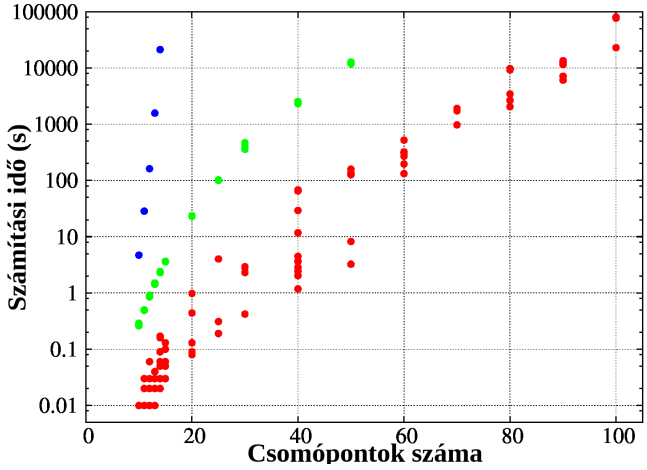
el és n él hozzáadásával egy Hamilton-kört alakítottam ki. A maradék $c - n$ élet iteratív módon adtam hozzá a gráfhoz, mindig a két legközelebbi, közös éllel még nem rendelkező csomópont között. A tesztekhez c értékét az összes lehetséges kapcsolat 15%-ára választottam, tehát $c = 0{,}15 \cdot \frac{n \cdot (n-1)}{2}$ volt a tesztek során. n értékét 10 és 100 között változtattam. Minden n-re 10 különböző véletlenszerű topológiát generáltam és minden esetben megmértem az algoritmusok futási idejét, 24 órás időkorlátot alkalmazva. A tesztekhez egy Lenovo Thinkpad T430 típusú számítógépet használtam, 2,5 GHz-es Intel Core i5-3210M processzorral és 8 GB memóriával. A tesztekhez $k = 3$ szélességet használtam. A 4.1. táblázatban az egyes beállítások esetén mért futási idők statisztikái (átlag és szórás) láthatók. Az egyes futások eredménye a 4.8. ábrán is látható. A futási idők nagy különbsége miatt az ábra függőleges tengelyén logaritmikus léptéket használtam.

4.1. táblázat. Az OMTS, OMTS-A, illetve a brute-force (bf) algoritmusok futási ideje.

n	t_{OMTS}(s)		$t_{\text{OMTS-A}}$(s)		t_{bf}(s)
	Átlag	Szórás	Átlag	Szórás	
10	0.3	0.01	0.005	0.005	4.7
11	0.5	0.004	0.014	0.007	28.5
12	0.9	0.02	0.033	0.01	161.4
13	1.5	0.02	0.023	0.01	1578
14	2.3	0.03	0.07	0.05	21 182
15	3.6	0.03	0.08	0.03	–
20	23	0.2	0.22	0.3	–
25	101	0.8	0.6	1.1	–
30	383	33	2.6	0.7	–
40	2380	71	19	25	–
50	12 434	280	107	52	–
60	–	–	228	122	–
70	–	–	1736	261	–
80	–	–	7404	3091	–
90	–	–	11 029	2462	–
100	–	–	64 066	23 836	–

A brute-force algoritmus már $n = 15$ estén elérte a 24 órás időkorlátot, az OMTS algoritmus legfeljebb $n = 50$-es ütemezési feladatokat tudott megoldani, az OMTS-A pedig még $n = 100$ esetén is használható volt. Az egyes algoritmusok közti sebességkülönbség a 4.8. ábráról is leolvasható. A brute-force algoritmus futási idejének növekedése jóval gyorsabb, mint a javasolt algoritmusoké. Az OMTS és az OMTS-A viselkedése hasonló, de a korai vágások alkalmazása miatt az OMTS-A két nagyságrenddel gyorsabbnak bizonyul. Figyeljük meg az OMTS-A által mutatott nagy varianciát, ami abból ered, hogy a korai vágások hatása nagy mértékben függ a konkrét topológiától.

A tesztek során alkalmazott $k = 3$ szélességre az OMTS és az OMTS-A elméleti komplexitása $O(n^7)$, ami a mérési eredmények $n \geq 20$ eseteivel is jól egybevág.



4.8. ábra. Az egyes algoritmusok futási idejei különböző csúcsszámok esetén. Az OMTS zöld, az OMTS-A piros, a brute-force algoritmus pedig kék színnel van jelölve.

4.5. Összefoglalás

A fentiekben bemutattam az OMTS algoritmust, ami kör topológiájú hálózatokban képes optimális Multi-TDMA ütemezést előállítani. Matematikailag bizonyítottam, hogy az algoritmus egy optimális (legrövidebb periódusidejű) ütemezést állít elő, ami egy tetszőleges csomópontból eredő csomag teljes hálózatban való elterjesztésének leggyorsabb módját eredményezi. Bizonyítottam továbbá azt is, hogy a Multi-TDMA ütemezés legfeljebb kétszeres gyorsulást tud eredményezni a lineáris ütemezéshez képest az energiafogyasztás kis mértékű növekedése árán.

Az OMTS-nek egy egy gyorsított, OMTS-A nevű változatát is bemutattam, ami heurisztikus úton csökkenti a keresési teret, de így is optimális megoldást garantál. Az alkalmazott heurisztika kiszűri a keresési tér azon részeit, amik már garantáltan nem adnának az aktuálisan ismert legjobbnál jobb megoldást.

A bemutatott algoritmusok matematikai vizsgálatán kívül beható tesztek segítségével is összehasonlítottam a bemutatott algoritmusok, illetve egy egyszerű brute-force módszer teljesítményét.

Míg a brute-force algoritmus csak kis (legfeljebb 15 csomópontból álló) hálózatok esetén volt használható, a bemutatott OMTS-A algoritmus 100 csomópontos hálózatra is kivárható időn belül lefutott.

5 Kétirányú aszimmetrikus protokoll extra alacsony kitöltési tényezőjű eszközök működtetéséhez

5.1. Áttekintés

Szenzorhálózatok tipikus felhasználási formái az olyan adatgyűjtő hálózatok, ahol a szenzor csomópontok méréseket végeznek, a mérési adatokat pedig egy központi csomópontnak [81], vagy egy gerinchálózat megléte esetén az adott csomóponthoz asszociált szülő eszköznek küldik el [82]. Az adatok a központi csomópontról, vagy a gerinchálózatról a felhasználóhoz kerülnek, ahol a további feldolgozás megtörténik. Ezekben a hálózatokban az adatáramlás egyirányú és a mérő csomópontoktól a központi csomópont, vagy a gerinchálózat felé mutat. Az általam vizsgált rendszermodell nem csak az adatáramlás iránya tekintetében mutat aszimmetriát, hanem a rendelkezésre álló energia szempontjából is. Míg a mérő csomópontok tápellátása jellemzően elemről történik, a központi csomópont, illetve a gerinchálózat hálózati tápellátású. Így a cél a mérő csomópont minimális energiafogyasztása, akár a központi csomópont, vagy a gerinchálózat energiafogyasztásának növelésével is. Egy ilyen rendszerben a jellemzően elemmel táplált mérő csomópontok úgy tudnak maximális működési időt elérni, hogy rádióikat kizárólag az adatküldés idejére kapcsolják be, a központi csomópont, illetve a gerinchálózat elemei pedig folyamatosan ébren vannak.

Bizonyos esetekben azonban szükséges lehet a fordított irányú kommunikáció is, jellemzően konfigurációs üzenetek, vagy parancsok küldésének céljából. Ennek megvalósítása jellemzően egy, a méréseket tartalmazó adatcsomag elküldését követő rövid vételi ablak beiktatásával lehetséges. A vételi időablak természetesen többlet energiát igényel, abban az esetben is, amikor valódi kommunikáció nem történik. Ennek a megközelítésnek egy lehetséges megvalósítását *kérdés-válasz protokoll* néven tárgyalom a későbbiekben.

Léteznek olyan protokollok is, amelyek eleve kétirányú kommunikációt biztosítanak, alacsony energiafogyasztás mellett. Ilyen az 1. fejezetben ismertetett S-MAC [25], a D-MAC [47], illetve a T-MAC [48]. A [83] két energiahatékony protokollt mutat be. Mindkettő periodikus kis kitöltési tényezőjű működést valósít meg mind adó, mind vevőoldalon. A TICER protokollban a kommunikációt az adó kezdeményezi oly módon, hogy addig küld RTS csomagokat, amíg a vevő fel nem ébred; ekkor az egy CTS csomaggal válaszol és megkezdődik a tényleges adatcsere. A RICER a vevő jelzi éber állapotát, speciális csomagokkal. Mindkét protokoll paraméterei opcionális időszinkronizációval is javíthatók.

A [84] az általános kommunikációs és szinkronizációs séma karakterizációját tárgyalja. A probléma modelljét matematikai módszerekkel és szimulációval vizsgálja, ezek segítsé-

gével megállapítja a vizsgált módszerek optimális paramétereit.

A vizsgálat tárgyát képező alkalmazások jellemzője, hogy a gerinchálózat elemei, illetve csillag topológia esetén az adatgyűjtő csomópont hálózati áramról, vagy nagy kapacitású, rendszeresen (például napelemről) feltöltött akkumulátorról működik, így ott az alacsony energiafogyasztás nem játszik szerepet. A szenzor csomópontok számára rendelkezésre álló energia ezzel szemben igen korlátozott, hiszen ezek az eszközök elemes tápellátásúak.

Az ilyen típusú rendszerek alkalmazási területe például lassan változó paraméterek (például kültérben mérhető hőmérséklet, vagy páratartalom) mérése, illetve ritkán előforduló jelenségek (például talajban mérhető rezgések) jelzése. Emiatt a kommunikáció lassú ütemben történik, ami óránként, vagy akár naponta néhány üzenetet is jelenthet.

A fordított irányú adatok, tehát a parancsok, illetve konfigurációs adatok kézbesítési idejével kapcsolatban az alkalmazás nem támaszt szigorú elvárásokat, azonban a rendszernek vissza kell tudnia jelezni a parancsok továbbításának sikerességét.

Az általam javasolt protokoll a ráültetés (piggybacking) technikát használja, ami többféle szabványos protokollnak is része; a TCP protokoll például a nyugtákat nem külön csomagként küldi, hanem más adatcsomagokba ágyazza [85], a [86] pedig egy olyan megoldást közöl, ami a 802.15.4 protokoll nyugta kereteiben levő kihasználatlan biteket használja RSSI információ továbbítására.

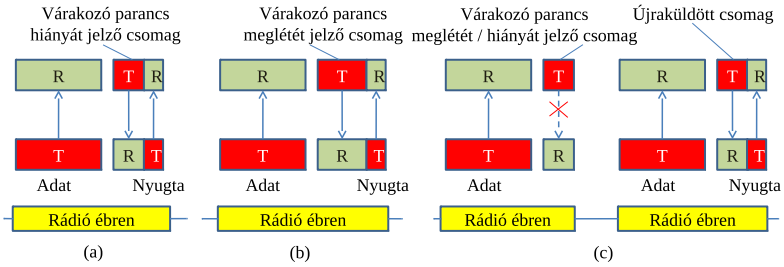
5.2. A javasolt módszer

A következőkben két különböző protokollt fogok bemutatni: egy egyszerű, hagyományos üzeneteket használó kérés-válasz protokollt és egy ráültetéses protokollt, ami módosított nyugtákat használ. Mindkét protokoll használata esetén a szenzor csomópontok az idő nagy részében alacsony energiafelvételű, alvó állapotban vannak és a nekik címzett információt mindaddig egy relé csomópont tárolja, amíg a megcímzett eszköz fel nem ébred. A relé csillag topológia estén a bázisállomás, fa topológia esetén a szenzor csomópont (mint levél) szülője lehet.

5.2.1. A kérés-válasz protokoll

A protokoll működése a következő. A szenzorok periodikusan kéréseket küldenek a hozzájuk tartozó relé felé. A periódusidőt úgy kell megválasztani, hogy az rövidebb legyen, mint a parancsok elvárt kézbesítési ideje. Amikor egy szenzor mérést küld a relé felé, a mérést tartalmazó csomag egyúttal kérésként is értelmeződik. A kérésre a relé egy üzenettel válaszol. Ha a relé tárolt üzenetet a csomópont számára, akkor a válasz a teljes (tárolt) üzenet, vagy információ a relén tárolt üzenetekről, ami egy hosszabb adatcserét készíthet elő. A javasolt protokoll a válaszüzenet struktúráját nem írja elő; az az adott alkalmazástól függ.

Az 5.1. ábrán a kérés-válasz protokoll kommunikációs diagramja látható. A szenzor egy kérést, vagy egy mérést tartalmazó csomagot küld a relének. Ha relén nincs eltárolt csomag a szenzor számára, akkor egy, a várakozó parancs hiányát jelző üzenettel válaszol. Ebben az esetben a szenzor egy nyugtát küld vissza a relének, majd alvó állapotba kapcsol, ahogy az az 5.1(a) ábrán látható.



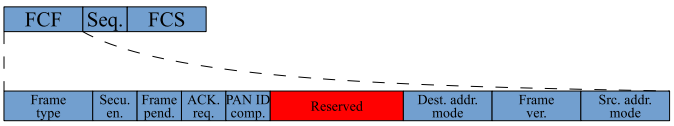
5.1. ábra. A kérés-válasz protokoll kommunikációs diagramja. A felső sorban a relé, a középsőben a szenzor csomópont tevékenységei láthatók. Az alsó sor a szenzor csomópont alvó / aktív állapotát jelzi. (a) A relé nem tárol üzenetet a szenzor számára. (b) A relén egy parancs várakozik, ami a szenzorhoz kerül. (c) Egy elveszett válaszüzenet, majd ismételt kommunikáció.

Ha a relé tárolt parancsot a szenzor számára, akkor ezt a válaszüzenetben juttathatja el a szenzornak. A szenzor erre egy nyugtával válaszol, ahogy az az 5.1(b) ábrán látható. A relé csak a nyugta megérkezésekor törli az eltárolt parancsot, ellenkező esetben később újra megpróbálja elküldeni azt.

Az 5.1(c) ábrán egy olyan eset látható, amikor kommunikációs hiba lépett fel és a szenzor nem kapja meg a válaszüzenetet. Ekkor egy véletlenszerű alvási periódus után újra megpróbálkozik a relével történő kommunikációval. A véletlenszerű késleltetés az esetleges ütközések valószínűségét csökkenti.

5.2.2. A nyugtára ültetéses protokoll

Egy nyugta elküldése a legtöbb közeghozzáférési (MAC) protokollba (például a 802.15.4-be) épített mechanizmusoknak köszönhetően lényegesen kevesebb energiát vesz igénybe, mint ami egy normál üzenethez szükséges. A nyugtára ültetéses protokoll alapötlete az, hogy a relé a várakozó csomaggal kapcsolatos meta-információt egy nyugtába ágyazva juttatja el a szenzor csomópontnak. A 802.15.4 szabvány által definiált nyugta kihasználatlan biteket is tartalmaz, amik e célra alkalmazhatóak. Ahogy az az 5.2. ábrán is látható, a nyugta FCF (Frame Control Field) mezője 3 kihasználatlan (fenntartott) bitet tartalmaz, amit felhasználva a nyugta (igen limitált mennyiségű) metainformációt is szállíthat. A nyugtára ültetéses protokollban egy bit jelöli a relén várakozó csomag meglétét, illetve hiányát. A fennmaradó két bit alkalmazásspecifikus módon használható fel, a javasolt protokoll ezek jelentését nem definiálja.

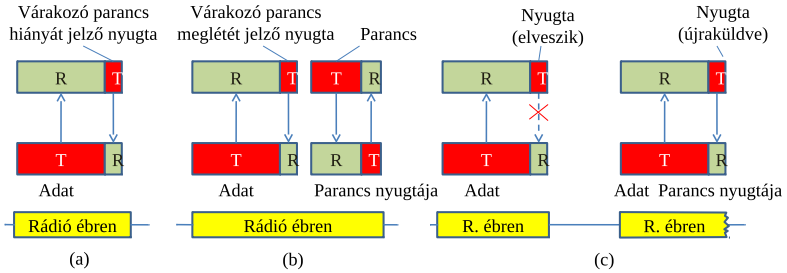


5.2. ábra. A 802.15.4 szabvány által definiált nyugta keret felépítése, illetve az FCF (Frame Control Field) mezői.

A protokoll vázlata az 5.3. ábrán látható. Az 5.3(a) ábrán az az eset látható, amikor a relén nem várakozik csomag, így a módosított nyugta megfelelő jelzőbitje hamis értékű. A nyugta vétele után a szenzor alvó állapotba válthat.

Az 5.3(b) ábrán látható esetben egy parancs várakozik a relén, így a módosított nyugta jelzőbitje igaz értéket tartalmaz, aminek hatására a szenzor ébren marad. Ezt követően a relé elküldi a várakozó parancsot a szenzornak, amire az egy nyugtával válaszol. A nyugta vétele után a relé törli az elküldött parancsot a listájából. A tranzakció után a szenzor alvó állapotba kapcsol.

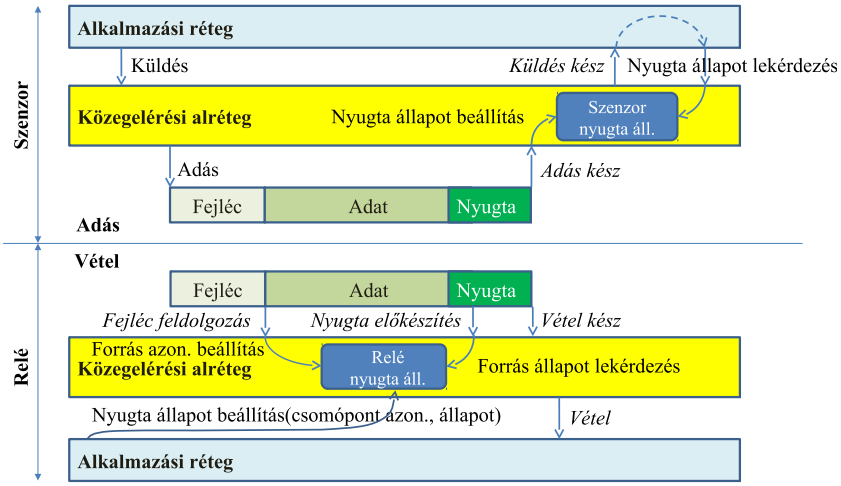
Az 5.3(c) ábrán az az eset látható, amikor a relé által elküldött nyugta elveszik. Ilyenkor a szenzor alvó állapotba vált, majd egy véletlenszerű várakozás után újra megpróbálja elküldeni üzenetét a relének.



5.3. ábra. A nyugtára ültetéses protokoll kommunikációs diagramja abban az esetben, ha a relén nincs várakozó csomag (a), egy várakozó csomag átvitele esetén (b), illetve elveszett nyugta, majd a kommunikáció ismétlése esetén (c).

A protokoll implementációjának nehézsége abban áll, hogy a vett csomag fejlécében található, a csomag forrására vonatkozó információ alapján a nyugtába ágyazandó adatot a csomag vétele közben kell előkészíteni, hogy az a nyugta összeállításakor már rendelkezésre álljon, a teljes csomag megérkezése után ugyanis erre a műveletre már nem lenne idő. A megvalósításhoz a beérkező csomag fejlécének minél előbb történő elérésére, illetve a nyugta összeállításának módosítására volt szükség. Az 5.4. ábrán a nyugtára ültetéses protokoll belső működése látható. Az ábra felső részén a szenzor csomópont, az alsón a relé egyes tevékenységei láthatók. A tranzakció előtt a relé a *Relé nyugta állapot* moduljának *Nyugta állapot beállítás* hívásával állítja be az egyes szenzoroknak küldendő jelzőbitet. (Az egyes szenzoroknak szánt jelzőbitek természetesen egymástól függetlenül kerülnek beállításra és tárolásra.) Később, amikor valamelyik szenzor egy üzenetet küld a relének, az alkalmazásrétegbeli *Küldés* parancs kerül meghívásra, majd ezt követően a vezérlést a közeghozzáférési (MAC) réteg veszi át, ami a tényleges *Adást* indítja. A csomag fejlécének vétele a *Fejléc feldolgozás* függvény meghívását váltja ki, ahol hozzáférhetővé válik a csomag forrásának azonosítója. Ezzel az argumentummal kerül meghívásra a *Relé nyugta állapot* modul *Forrásazonosító beállítás* függvénye, ami kikeresi a táblázatából az adott azonosítóhoz korábban eltárolt státuszinformációt, amit egy belső változóban eltárol.

A nyugta összeállítása rögtön az üzenet vétele után történik. Eközben a relé meghívja a *Relé nyugta állapot* modul *Forrás állapot lekérdezés* függvényét, ami visszaadja a



5.4. ábra. A nyugtára ültetéses protokoll belső működése

nyugtába beleírandó, előzőleg már előkészített jelzőbitet. A nyugta elküldését a relé közeghozzáférési rétegébe érkező *Vétel kész* hívás jelzi; az eseményről a közeghozzáférési réteg az alkalmazási réteget egy *Vétel* hívással értesíti. A szenzoron a nyugta megkapása a *Szenzor nyugta állapot* modulban levő *Nyugta állapot beállítás* függvény meghívását váltja ki és a modul eltárolja a nyugta által hordozott státusz információt. Ezt követően a közeghozzáférési réteg az alkalmazási réteget a *Küldés kész* hívással értesíti a sikeres (vagy sikertelen) küldésről. A státusz információ ez után a *Szenzor nyugta állapot* modul *Nyugta állapot lekérdezés* függvényével olvasható ki. Fontos megjegyezni, hogy a nyugta összeállítása időkritikus, emiatt a *Forrás állapot lekérdezés* függvénynek igen gyorsnak kell lennie; az általam bemutatott rendszerben ez a függvény az előzőleg már előkészített állapot információval tér vissza; az információ előkészítése a *Forrás azonosító beállítás* függvényben történik, ami az üzenet feldolgozásának egy sokkal kevésbé időkritikus pontján (a fejléc vétele után) fut le.

5.2.3. Hardver és szoftver architektúra

A tesztekhez UCmote Proton B szenzorokat használtam. Az eszközök fő egysége egy 16 MHz-es ATmega128RFA1 mikrovezérlő. A program és a futás közbeni adatok tárolására 16 kB RAM és 128 kB flash memória áll rendelkezésre. A mikrokontroller egy 2,4 GHz-es rádió adóvevőt is tartalmaz, de a protokoll tesztelésére a szenzoron levő másik, AT86RF212 típusú, 868, illetve 915 MHz-en működő rádiót használtam.

A szenzoron LED-ek is találhatóak, ezeket az események jelzésére használtam. A szenzor flash memóriáját JTAG letöltővel programoztam fel. A szenzor és a számítógép közti kommunikáció USB porton, a szenzoron levő soros-USB átalakítón keresztül történt.

A szenzoron futó szoftver a TinyOS [21] komponens alapú, valós idejű, szenzorhálózatos alkalmazások számára kifejlesztett operációs rendszer, illetve az ahhoz tartozó keretrendszer felhasználásával készült. Az alkalmazási-rétegbeli programozáson kívül a rádió alrendszer bizonyos részeiben is eszközöltem módosításokat. A nyugtához, a csomagfej-

léchez és a CSMA protokoll csatornafoglaltság-vizsgálattal és visszalépéssel kapcsolatos paramétereihez való hozzáférés az rfxlink rétegek módosításával történt.

A tesztekhez használt demóalkalmazás az [S4] linken érhető el.

5.3. Mérési eredmények

Ebben a szakaszban a szenzorok rádióinak ébren töltött idejét és az eszközök energiafogyasztását vizsgálom különböző paraméterek esetén, valamint összehasonlítom a bemutatott protokollok teljesítményét.

5.3.1. A rádiók ébren töltött ideje

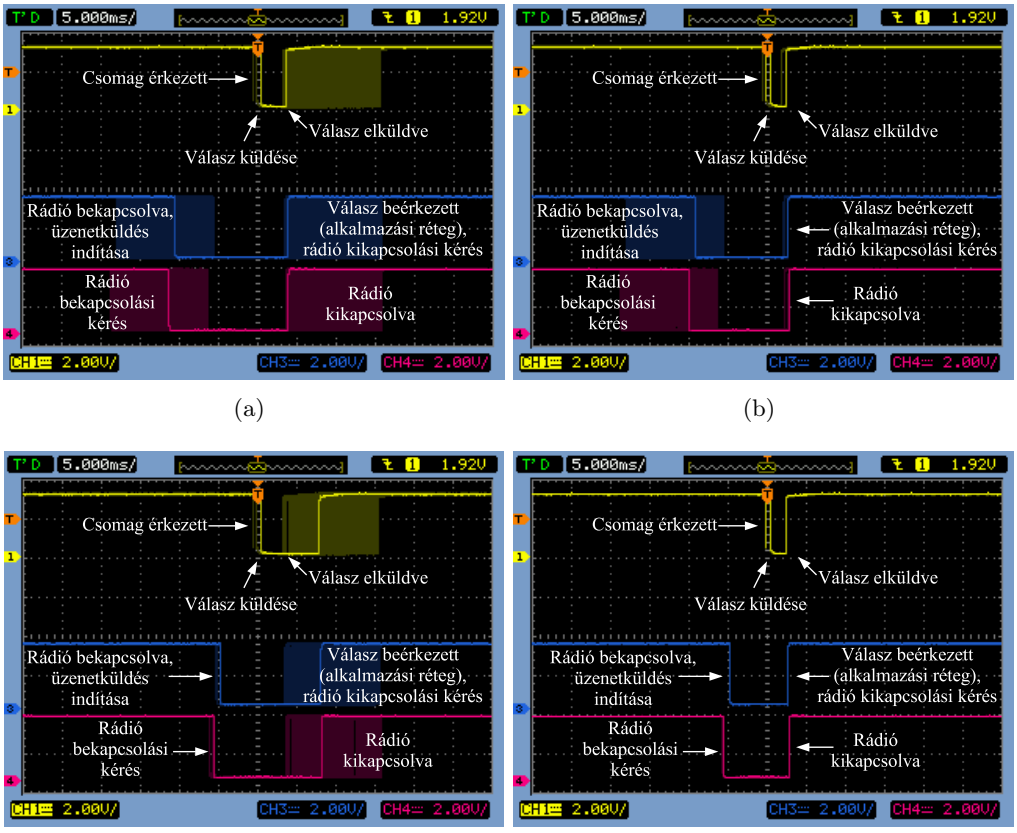
A bemutatott protokollok teljesítményét nagyban befolyásolják a rádió kommunikációs alrendszer, elsősorban a közeghozzáférési (MAC) réteg beállításai. A következőkben különböző konfigurációkat fogok bemutatni és elemezni. Az időzítési ábrákat egy négy csatornás oszcilloszkóppal rögzítettem, perzisztens módban; ebben a módban a virtuális elektronsugár összes régebbi nyoma is látható a képernyőn, halványabb színnel. A diagramok felső része a reléhez, az alsó a szenzorhoz tartozik.

A kérés-válasz protokoll

Az 5.5(a) ábrán a kérdés-válasz protokoll időzítése látható. A lila vonal a szenzor rádiójának ki- illetve bekapcsolt állapotát jelzi (a jel inverz). A *Rádió bekapcsolási kérés* által jelzett pillanatban kéri a szenzor a rádió bekapcsolását. A bekapcsolás egy kis időt vesz igénybe, a bekapcsolt állapotot *Rádió bekapcsolva, üzenetküldés indítása* felirat jelzi. Ugyanekkor, tehát rögtön a bekapcsolást követően történik az adás küldésének kérése is. A sárga vonalon jelölt *Csomag érkezett* jelzés azt az időpillanatot jelzi, amikor a relé megkapja a csomagot. A diagramot ehhez az időponthoz szinkronizáltam. A CSMA/CA protokollal és annak alapértelmezett beállításával a csomag alkalmazási rétegből való elküldése és annak vétele közti idő a véletlenszerű várakozások miatt széles tartományban változik. A bázisállomás a válasz üzenetet a *Válasz küldése* szöveggel megjelölt időpontban küldi, és a csomag küldése a *Válasz elküldve* időpontban ér véget. A szenzor az üzenetet a *Válasz beérkezett* időpontban kapja meg, majd rögtön kéri a rádió kikapcsolását. A kikapcsolás a *Rádió kikapcsolva* időpontban fejeződik be.

A kérés-válasz protokoll, illetve alapértelmezett CSMA/CA beállítások esetén a rádió a kliens oldalon 12–36 ms ideig van bekapcsolva. A késleltetések többsége a CSMA/CA protokoll véletlenszerű várakozásaiból fakad. A *Küldés* parancs alkalmazási rétegbeli kiadása után a közegelérési réteg véletlenszerű várakozásokat használ. Ezek minimális ideje $t_{\text{backoff,min}}$, maximális ideje $t_{\text{backoff,max}}$, tehát ha a csomag hosszát t_{message}-vel jelöljük, akkor a *Rádió bekapcsolva és csomagküldés indítása* pillanattól *Csomag érkezett* pillanatig eltelt idő $t_{\text{backoff,min}} + t_{\text{message}}$ és $t_{\text{backoff,max}} + t_{\text{message}}$ között van, ha az egyéb késleltetésektől eltekintünk.

Az 5.5(b) ábrán a relé oldalon kikapcsolásra kerültek a CSMA/CA protokoll véletlenszerű késleltetései. Ebben az esetben a szenzor ébren töltött ideje szignifikánsan csök-



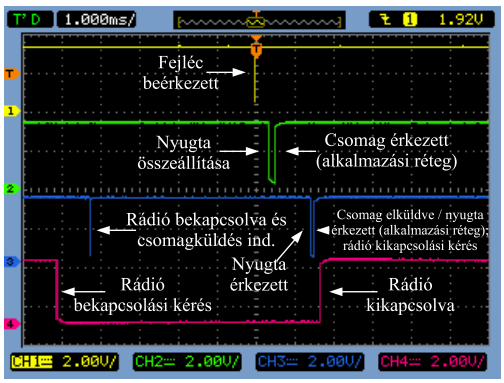
5.5. ábra. A kérdés-válasz protokoll időzítésének oszcilloszkópos felvételei. (a) A CSMA/CA várakozásai alapbeállításon. (b) A bázisállomáson kikapcsolt várakozások. (c) A mérő csomóponton kikapcsolt várakozások. (d) Mindkét oldalon kikapcsolt várakozások.

kent, hasonlóan ahhoz az esethez, amikor ugyanezeket a szenzor csomóponton kerültek kikapcsolásra (5.5(c) ábra). Mindkét esetben eltűnt valamelyik (szenzor, vagy relé) oldal küldési késleltetésének véletlenszerű mivolta. Mindkét konfiguráció esetén a szenzor oldalon 10–25 ms ébren töltött idő volt mérhető.

Az 5.5(d) ábrán a CSMA/CA várakozásait mindkét oldalon kikapcsoltam. Ennek hatására az időzítésben tapasztalható véletlenszerűség gyakorlatilag teljesen eltűnt. Ebben az esetben a szenzor oldalon 9 ms ébren töltött idő volt tapasztalható.

A nyugtára ültetéses protokoll

A nyugtára ültetéses protokoll esetében felvett időzítési mintázat az 5.6. ábrán látható. A lila vonal a szenzor rádiójának állapotát jelzi (inverz jel). A jel lefutó éle a rádió bekapcsolási kérésének időpillanatét jelzi (*Rádió bekapcsolási kérés*). Amint a rádió üzemkész, a protokoll egy üzenetküldést kezdeményez (*Rádió bekapcsolva és csomagküldés indítása*). A szenzor rádió alrendszere a fejléc beérkezése után egy jelzést bocsájt ki (*Fejléc beérkezett* szöveggel jelölve). A nyugta összeállítása a *Nyugta összeállítása* címkével ellátott időpontban kezdődik, megközelítőleg 300 µs-mal a fejléc beérkezésének jelzése után; ennyi idő áll rendelkezésre a relé számára a fejléc feldolgozására, a forrás címének kinyerésére és a megfelelő szenzor számára eltárolt állapotinformáció kikeresésére. A relé alkalmazási rétege a *Csomag érkezett (alkalmazási réteg)* pillanatban kapja meg a csomagot. A szenzor a nyugtát az *Nyugta érkezett* pillanatban kapja meg, az alkalmazási réteg pedig a *Csomag elküldve / nyugta érkezett (alkalmazási réteg); rádió kikapcsolási kérés* időpillanatban értesül a sikeres (vagy sikerelen) küldésről. Ekkor rögtön kéri a rádió kikapcsolását, ami a *Rádió kikapcsolva* időpontban fejeződik be. Ennek a protokollnak a használatával egy tranzakcióhoz a szenzor oldalon 6,8 ms aktív (bekapcsolt) idő szükséges. Ez 2,2 ms-mal kevesebb, mint ami az ebből a szempontból legjobb paraméterekkel használt kérdés-válasz protokoll esetében mérhető (mindkét oldalon kikapcsolt véletlen várakozások, 5.5(d) ábra), illetve 5–29 ms-mal kevesebb, mint az alapértelmezett CSMA/CA beállításokkal használt kérés-válasz protokoll esetén mérhető érték (5.5(a) ábra).

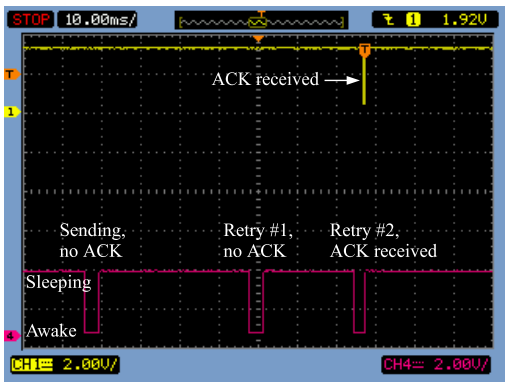


5.6. ábra. A nyugtára ültetéses protokoll időzítéseinek oszcilloszkópos felvétele.

5.3.2. A protokoll hibatűrése

A protokoll hibatűrését a relé a rádiójának ki- majd bekapcsolásával szimuláltam, és egy olyan eseménysorozatot rögzítettem az oszcilloszkóppal, amikor a szenzor első és második (ismételt) csomagjának küldése közben a relé rádiója még kikapcsolt, a harmadik (másodszor ismételt) csomag esetében viszont már bekapcsolt állapotban van. A tesztet csak a nyugtára ültetéses protokoll esetén végeztem el.

Az 5.7. ábrán a hiba esetén történő ismétlés működése látható. A szenzor rádiójának ki- illetve bekapcsolt állapotát a piros vonal jelzi (inverz jel). A sárga csúcs a nyugta megérkezését jelzi. Az első adás után a szenzor nem vette a nyugtát, és alvó állapotba kapcsolt. Véletlenszerű, a példában 38 ms-os várakozás után újra megpróbálja elküldeni a csomagot a relének, de a tranzakció megint sikertelen (nem érkezik nyugta). Egy újabb alvással töltött, ebben az esetben 24 ms hosszúságú időtartam után egy harmadik próbálkozás is látható, most már sikeresen vett nyugtával.



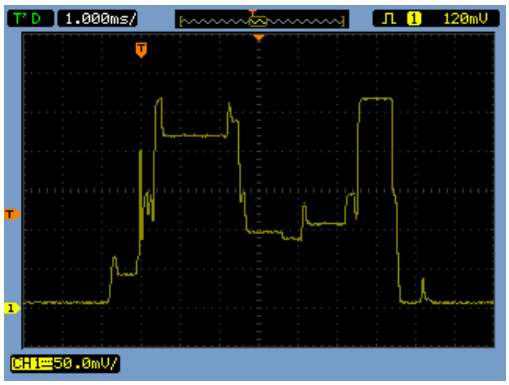
5.7. ábra. A hiba esetén működésbe lépő ismétlés időzítéseinek oszcilloszkópos felvétele. A csomag összesen háromszor került adásra az első két esetben elmaradó nyugta miatt. Az adások közötti várakozás mindig véletlenszerű, a várakozás ideje alatt a rádió kikapcsolt állapotban van.

5.3.3. Energiafogyasztás

A szenzor áramfelvételének méréséhez a tápfeszültséggel sorosan kapcsolt 10 Ω-os ellenállást használtam; az ellenálláson eső, időben változó feszültséget oszcilloszkópon figyeltem meg. A zaj csökkentésének érdekében 256 szinkronizált minta átlagát jelenítettem meg az oszcilloszkópon. A kérdés-válasz protokoll esetében a mérés a CSMA/CA véletlenszerű késleltetéseit kiiktató variánssal történt. Az egy kommunikációs periódus által igényelt energia az áramfelvételi görbe alatti terület grafikus úton történő kiszámításával történt. A szenzort labortápegységről, 3 Voltos tápfeszültségről működtettem. (Az ellenálláson eső feszültséget elhanyagoltam.)

A kérés-válasz protokoll

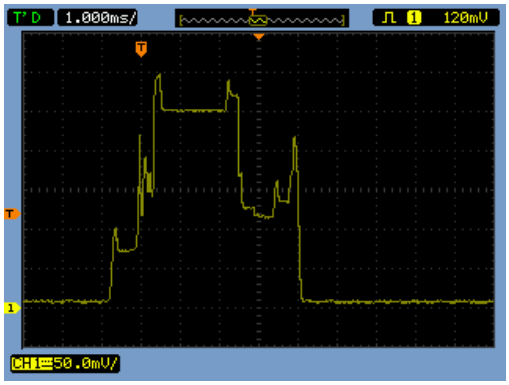
A kérdés-válasz protokoll esetében mért áramfelvételi görbe az 5.8. ábrán látható. Egy tranzakcióhoz 0,39 mJ energia szükséges.



5.8. ábra. A kérdés válasz protokoll egy tranzakciójának áramfelvétele az idő függvényében (256 szinkronizált mérés átlaga).

A nyugtára ültetéses protokoll

A nyugtára ültetéses protokoll áramfelvételi görbéje az 5.9. ábrán látható. egy tranzakció 0,27 mJ energiát igényel. Méréseim szerint tehát a nyugtára ültetéses protokoll energiaszükséglete 30%-kal kevesebb, mint a kérés-válasz protokollé.



5.9. ábra. A ráültetéses protokoll egy tranzakciójának áramfelvétele az idő függvényében (256 szinkronizált mérés átlaga).

5.4. Összefoglalás

Ebben a szakaszban egy olyan, alacsony energiaigényű kommunikációs protokollt mutattam be, ahol az adatgyűjtő szenzorok kis kitöltési tényezővel működhetnek, ami alacsony

energiafogyasztást és ezáltal hosszú élettartamot biztosít a hálózat számára. Mindkét protokoll támogatja a fordított irányú, tehát a relé csomópontoktól a szenzorok felé irányuló adatforgalmat is. Az egyik javasolt megoldás egy hagyományos üzeneteket használó kérés-válasz protokoll, amivel elsősorban a naiv megközelítést modelleztem. A másik megoldás egy nyugtára ültetéses protokoll, ami módosított nyugtákat használ. Mindkét megoldást TinyOS keretrendszer alatt implementáltam, a TinyOS 802.15.4 szabványt megvalósító rádió kommunikációs alrendszerében módosításokat eszközölve. Mindkét protokollt valós mérésekkel, többféle szempont szerint teszteltem. A nyugtára ültetéses protokoll 30%-kal kisebb energiafogyasztást mutatott a legkisebb energiafelhasználású kérdés-válasz protokollhoz képest. Az alapbeállításokat használó kérdés-válasz protokollhoz képest tapasztalható javulás még erőteljesebb.

6 Hatékony lokalizációs szolgáltatások

6.1. Bevezetés

A lokalizáció célja egy objektum pozíciójának meghatározása; az objektumok szenzorhálózat egyik csomópontjai, vagy egyéb tárgyak, jellemzően rádiófrekvenciás, vagy akusztikus jelforrások lehetnek. Az akusztikus lokalizáció a lokalizációk egy széles körben használt változata, amire az idők során sokféle konkrét megoldás született. Az egyik sikeresen alkalmazott megoldás a nyalábolás, ami több egyidejű forrás helyének zajos környezetben való becslésére képes [87]. Szenzorhálózatok alkalmazása esetén, ahol az egyes szenzorok egymástól távol helyezkednek el, a vezetéknélküli kommunikáció nem teszi lehetővé a nyers mintavételezett jelek elküldését a szenzoroktól a feldolgozó egységig. A kommunikációs teher enyhíthető, ha az egyes szenzorok a jel helyett annak erősségét, észlelési idejét, vagy még gyakrabban ezek különbségét (TDoA – Time Difference on Arrival) számítják ki és küldik el, maga a pozícióbecslés pedig egy későbbi lépésben történik. A lokalizációs lépés egyik általánosan használt módszere a maximum likelihood becslő, például a [88]-ban tárgyalt jelenergia mérések felhasználásával, vagy a [89] és a [90] szerinti TDoA becslőkkel. Távolságbecslésre is számos megoldás létezik, például a jelterjedési idő direkt mérése [91], illetve korreláció-alapú megközelítések [92].

A TDoA adatok alapján a pozíció elméletileg egyszerűen kiszámítható, például egy túlhatározott egyenletrendszer megoldásával [90]. A becsült pozíciót azonban erősen befolyásolják a nagy mérési hibák, amik visszaverődésekkel terhelt környezetben igen gyakoriak. Ugyanezzel a hibával a maximum likelihood becslő is rendelkezik, így az ilyen jellegű megoldások használhatósága valós körülmények között, nagy mérési hibák előfordulása esetén korlátozott.

Egy, az előzőektől lényegesen eltérő megközelítést alkalmaz a [12], ami egy olyan megoldást mutat be, ami a TDoA méréseket automatikusan szűri a pozícióbecslési folyamat alatt, így a kiugró értékeket a becsléshez nem használja fel, ezáltal nagy számú hibás mérés esetén is nagy pontosságú becslést ad. A becslési folyamat a konzisztencia függvényen alapul, ami azt adja meg, hogy hány szenzor támogatja a forrás pozíciójára vonatkozó hipotézist. A pozícióbecslőt a konzisztenciafüggvény maximumhelye adja meg.

A 6.2. szakaszban a [12] irodalomban közölt konzisztencia alapú lokalizációs módszert tekintem át. A 6.3. szakaszban a pozícióbecslő adaptív módosítására teszek javaslatot. A 6.4. szakaszban a konzisztenciafüggvény hatékony kiértékelésére kidolgozott módszeremet mutatom be.

6.2. Konzisztencia alapú lokalizáció

A következőkben a [12] irodalomban közölt konzisztencia alapú lokalizációs módszert tekintem át. A lokalizációhoz használt szenzorhálózatban N vezeték nélküli szenzor csomópont ($S_1, \ldots, S_N$) szerepel, ahol minden S_i szenzor térbeli pozíciója az (x_i, y_i, z_i) koordinátákkal adott. Minden csomópont egy mikrofont és egy nagy felbontású órát is tartalmaz, melyek egymáshoz vannak szinkronizálva. A szenzorok bizonyos, jellemzően meredek felfutású hangjelek beérkezési idejét mérik. Jelölje a hangforrás pozícióját (x_s, y_s, z_s), és a hang kibocsátási idejét t_0. Ekkor az S_i szenzoron a beérkezési idő a következőként alakul:

$$t_i = \tau_i + e_i, \tag{6.1}$$

ahol τ_i az ideális detekciós idő és e_i a mérési hiba. A hang levegőben mért terjedési sebességét állandó v értékűnek tekintjük. Az ideális detekciós idő, a hangjel kibocsátási ideje, a szenzor és a forrás távolsága, valamint a hang terjedési sebessége között a következő összefüggés írható fel:

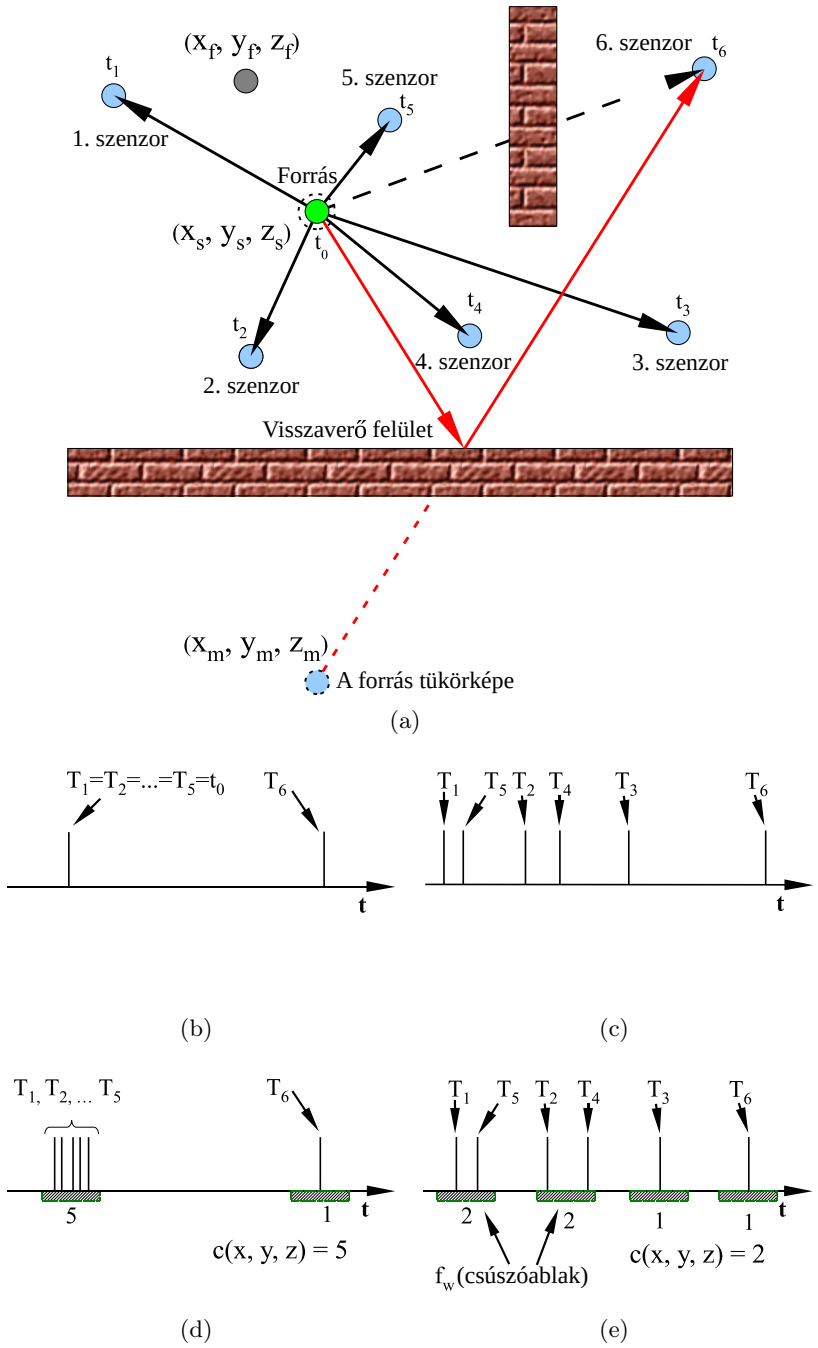
$$\sqrt{(x_i - x_s)^2 + (y_i - y_s)^2 + (z_i - z_s)^2} = v(\tau_i - t_0) \tag{6.2}$$

Definiáljuk minden S_i szenzorra a T_i-t a következő módon:

$$T_i(x, y, z) = t_i - \frac{\sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}}{v} \tag{6.3}$$

A (6.1), (6.2) és (6.3) egyenletek alapján világos, hogy $T_i(x_s, y_s, z_s) = t_0 + e_i$. Ha nincs mérési hiba, vagyis ha minden i-re $e_i = 0$, akkor minden i-re $T_i(x_s, y_s, z_s) = t_0$. Ha a mérések helyesek, de e_i kis mértékű zajt tartalmaz, akkor a T_i értékek nem esnek pontosan egybe, hanem e_i-nek megfelelően t_0 körül helyezkednek el. Ha valamelyik e_i nagy mérési hibát tartalmaz, akkor a hozzá tartozó T_i t_0-tól távolra esik. Másrészt, ha $(x, y, z) \neq (x_s, y_s, z_s)$, akkor a $T_i(x, y, z)$ egyes i-khez tartozó értékei jellemzően eltérőek lesznek, ahogy a 6.1. ábrán látható.

A 6.1(a) ábrán egy mérési elrendezés látható 5 szabad rálátással rendelkező szenzorral ($S_1, \ldots, S_5$) és egy hatodik, szabad rálátással nem rendelkezővel (S_6), ami egy visszavert jelet érzékel. A 6.1(b) ábrán a T_i értékek eloszlása látható zajmentes esetben, $(x, y, z) = (x_s, y_s, z_s)$ esetén. A szabad rálátással rendelkező szenzorok ugyanazt a T_i értéket becsülték, T_6 esetén azonban ez az érték a hosszabb út miatt nagyobb. Ebben az esetben 5 szenzor támogatja azt a hipotézist, hogy a forrás pozíciója (x_s, y_s, z_s) és a hang kibocsátási ideje t_0. A 6.1(c) ábrán egy olyan eset látható, ahol a hipotézis hamis, vagyis $(x, y, z) = (x_f, y_f, z_f) \neq (x_y, y_s, z_s)$. Ebben az esetben a T_i értékek az időben szétszórva helyezkednek el és semelyik két szenzor nem támogatja azt a hipotézist, hogy a forrás pozíciója (x_f, y_f, z_f). A 6.1(d) ábra a 6.1(e) ábra zajjal terhelt megfelelője, ahol a mérésben jelentkező zaj miatt a T_i értékek nem esnek pontosan egybe, de egymáshoz igen közel helyezkednek el. Ebben az esetben egy ablakot használhatunk a hipotézist támogató szenzorok maximális számának – ebben az esetben 5 – meghatározására. A 6.1(e)



6.1. ábra. A konzisztenciafüggvény számítása. (a) Mérési elrendezés 5 direkt rálátással rendelkező (LOS) és 1 direkt rálátással nem rendelkező (NLOS) szenzorral. (b) A T_i értékek eloszlása zajmentes esetben a forrás tényleges pozíciójában. (c) A T_i értékek eloszlása zajmentes esetben egy, a forrás pozíciójától eltérő helyen. A (d) és (e) a (b) és (c) esetek zajjal terhelt megfelelői.

ábra a 6.1(c) ábra zajjal terhelt megfelelője, itt ugyanekkora ablakot használva a támogató szenzorok maximális száma 2. Ebben a példában az (x_s, y_s, z_s) nyilvánvalóan jobb hipotézis, mint az (x_f, y_f, z_f).

A kis hibák tolerálására egy $w \in \mathbb{R}^+$ ablakot és egy $f_w : \mathbb{R} \Rightarrow \mathbb{R}^+$ ablakfüggvényt definiálunk a következő módon:

$$f_w(\tau) = \begin{cases} 1 & \text{ha } |\tau| \leq \frac{w}{2} \\ 0 & \text{egyébként} \end{cases} \tag{6.4}$$

A w ablakot úgy állítjuk be, hogy lefedje a konzisztens, de kis mértékű hibát tartalmazó méréseket, de az inkonzisztens mérések, illetve a kiugró értékek már ne férhessenek bele. Ha w túl kicsi, akkor az esetleg helyes, de zajos mérések nem tudják az egyébként helyes hipotézist támogatni (például ha a 6.1(d) ábrán az ablakméretet jelentősen csökkentenénk, akkor ablak nem tudná mind az 5 támogató mérést lefedni). Másrészt egy túl nagy ablak inkonzisztens méréseket is lefedne (például a 6.1(e) ábrán egy túlságosan nagy ablak még több inkonzisztens mérést tudna lefedni). Ha a maximálisan megengedett hiba $e_{\max}$, akkor az ablaknak körülbelül $w = 2e_{\max}$ méretűnek kellene lennie. Az ablakméret hatását a 6.4.4. szakaszban fogom részletesen vizsgálni.

A konzisztenciafüggvény tehát azt jelzi, hogy hány szenzor támogatja azt a hipotézist, hogy a forrás pozíciója az (x, y, z) pont. A pontos definíció a következő:

$$c(x, y, z) = \max_{t \in \mathbb{R}} \sum_{i=1 \ldots N} f_w(T_i(x, y, z) - t) \tag{6.5}$$

A becsült pozíciót a konzisztenciafüggvény maximumhelye adja, tehát:

$$(\hat{x}_s, \hat{y}_s, \hat{z}_s) = \arg\max_{(x,y,z)} c(x, y, z) \tag{6.6}$$

6.3. A pozícióbecslő adaptív javítása a szenzorok megbízhatósága alapján

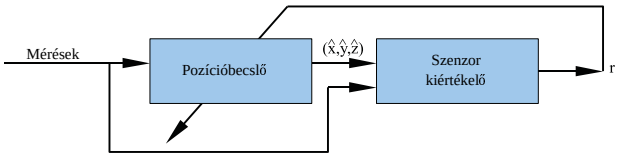
6.3.1. Áttekintés

A konzisztenciafüggvény alapú becslő nagyszámú független kiugró hibát tartalmazó mérés esetén is robusztus megoldást ad. Bizonyos esetekben azonban a szenzorok szisztematikus hibával is rendelkezhetnek, konzisztensen rossz eredményt adva. Ilyen jellegű hiba például visszaverődésekkel terhelt mérési területeken jelentkezhet, közvetlen rálátással nem rendelkező szenzorok esetén. Ezek a konzisztensen hibás adatok a konzisztenciafüggvényben hamis maximumot eredményezhetnek, hibás becslést eredményezve.

A következőkben a konzisztencia alapú lokalizációs séma adaptív javítását fogom bemutatni. A módszer az egyes szenzorok hibáját figyeli, a pozícióbecslés pedig egy módosított, a szenzorok megbízhatóságát figyelembe vevő konzisztencia függvénnyel történik. Minden pozícióbecslés után értékelésre kerülnek az egyes szenzorok. Azok a szenzorok, amelyek az aktuálisan becsült pozícióval konzisztens mérést adnak, megbízhatóbbnak tekinthetők, mint azok, amelyek ezzel inkonzisztensek.

6.3.2. A megbízhatóság alapú adaptív lokalizációs algoritmus

A megbízhatóság alapú adaptív lokalizációs algoritmus vázlata a 6.2. ábrán látható. A méréseket először a *pozícióbecslő* értékeli ki, egy $(\hat{x}, \hat{y}, \hat{z})$ becslést adva eredményül. A becsült pozíció és a mérések a *szenzor kiértékelő* blokkba kerülnek, ami a bemenő adatok alapján eldönti, hogy mely szenzorok támogatták a becsült pozíciót, melyek adtak ezzel inkonzisztens mérést, illetve melyektől nem érkezett adat. Ezek alapján kerülnek módosításra az egyes szenzorokhoz tartozó r_i $(0 \leq r_i \leq 1, 1 \leq i \leq n)$ megbízhatósági faktorok, amik a pozícióbecslő működését hangolják.

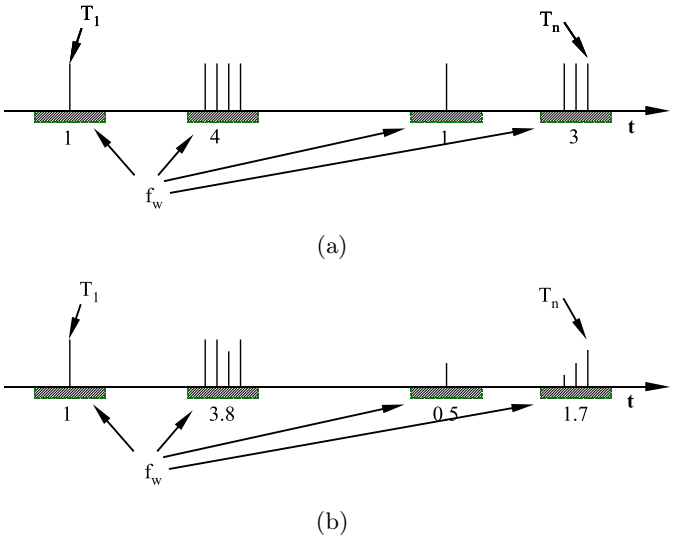


6.2. ábra. Adaptív lokalizáció az előző becslések alapján kiszámított megbízhatósági faktor segítségével.

A módosított, megbízhatósági faktorokat is figyelembe vevő konzisztenciafüggvény a következő:

$$c(x, y, z) = \max_{t \in \mathbb{R}} \sum_{i=1...n} r_i f_w(T_i(x, y, z) - t) \tag{6.7}$$

A megbízhatósági faktor hatása a 6.3. ábrán látható. A 6.3(a) ábrán a konzisztenciafüggvény minden szenzort egyforma súllyal vesz figyelembe. Az ablak csúsztatásával található legnagyobb csoport 4 elemű, tehát a konzisztenciafüggvény értéke 4. A 6.3(b) ábrán



6.3. ábra. A megbízhatósági faktor hatása. A T_i értékek eloszlása egy bizonyos pontra az eredeti algoritmussal (a), illetve a megbízhatósági faktort használó adaptív algoritmussal (b).

a megbízhatósági faktort használó adaptív algoritmus által használt konzisztenciafüggvény kiszámításának grafikus ábrázolása látható. A módosított konzisztenciafüggvény az egyes szenzorokat eltérő, a megbízhatósági faktornak megfelelő súllyal veszi figyelembe. Az itt megtalált legnagyobb összsúlyú csoport 3,8 értékű, tehát a konzisztenciafüggvény értéke 3,8.

Az adaptív algoritmus esetén a pozícióbecslő a következő:

$$(\hat{x}_s, \hat{y}_s, \hat{z}_s) = \arg\max_{(x,y,z)} c(x, y, z) \tag{6.8}$$

a kibocsátási idő alsó korlátja pedig a következő képletek segítségével számítható ki:

$$\hat{t}_{0,L} = \min(\arg\max_{t} \sum_{i=1\ldots n} f_w(T_i(\hat{x}_s, \hat{y}_s, \hat{z}_s) - t)) \tag{6.9}$$

A megbízhatósági faktorok kezdeti értéke $r_i = 1$ $(1 \leq i \leq n)$. A megbízhatósági faktort az algoritmus minden pozícióbecslés után a következő módon korrigálja:

$$\left.\begin{array}{ll} r_i := \min(r_i + \alpha, 1) & \text{ha } |T_i(\hat{x}_s, \hat{y}_s, \hat{z}_s) - \hat{t}_{0,L}| \in [0, w] \\ r_i := \max(r_i - \alpha, 0) & \text{ha } |T_i(\hat{x}_s, \hat{y}_s, \hat{z}_s) - \hat{t}_{0,L}| \notin [0, w] \\ r_i \text{ változatlan marad} & \text{ha az } i \text{ csomópont nem küldött adatot} \end{array}\right\}, \tag{6.10}$$

ahol $1 \leq i \leq n$ és α az adaptációs paraméter. A pozícióbecslés, illetve az azt követő kiértékelő és adaptív korrekciót végző lépés a közel egy időben beérkező (jellemzően egy lövéshez tartozó) mérésekből álló méréscsoportra fut le.

A megbízhatóság alapú adaptív lokalizációs algoritmus (TBALA) részletes működése a következő:

TRUSTINESS-BASED-ADAPTIVE-LOCALIZATION-ALGORITHM (TBALA)

input
  n: szenzorok száma
  $(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)$: szenzorok pozíciója
  $t_1, \ldots, t_n$: beérkezési idők
  α: adaptációs paraméter
  $x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}$: a keresési tér határai
  (res_x, res_y, res_z): a keresés felbontása
változók
  $location_{\{}x, y, z\}$: az aktuálisan vizsgált pont koordinátái
  c: a konzisztenciafüggvény értéke az aktuális pontra
  i: futóváltozó
  $r_1, \ldots r_n$: a megbízhatósági faktorok
  currentbest: a jelenleg ismert legmagasabb konzisztencia érték
  loc_est: a jelenleg ismert legmagasabb konzisztenciával rendelkező pont
begin
for $i \leftarrow 1$ **to** n
  load(r_i)
for $location_x \leftarrow x_{\min}$ **to** $x_{\max}$ **step** res_x do
  for $location_y \leftarrow y_{\min}$ **to** $y_{\max}$ **step** res_y do
    for $location_z \leftarrow z_{\min}$ **to** $z_{\max}$ **step** res_x do
      $c \leftarrow c(location_{\{x,y,z\}})$
        r és α paraméterek és a (6.7) felhasználásával
      if $c(location_{\{x,y,z\}}) > currentbest$ **then**
        $currentbest \leftarrow c(location_{\{x,y,z\}})$
        $loc\_est \leftarrow location$
for $i \leftarrow 1$ **to** n
az r_i megbízhatósági faktor kiszámítása a (6.10) segítségével
store(r_i)
return(*loc_est*)
end

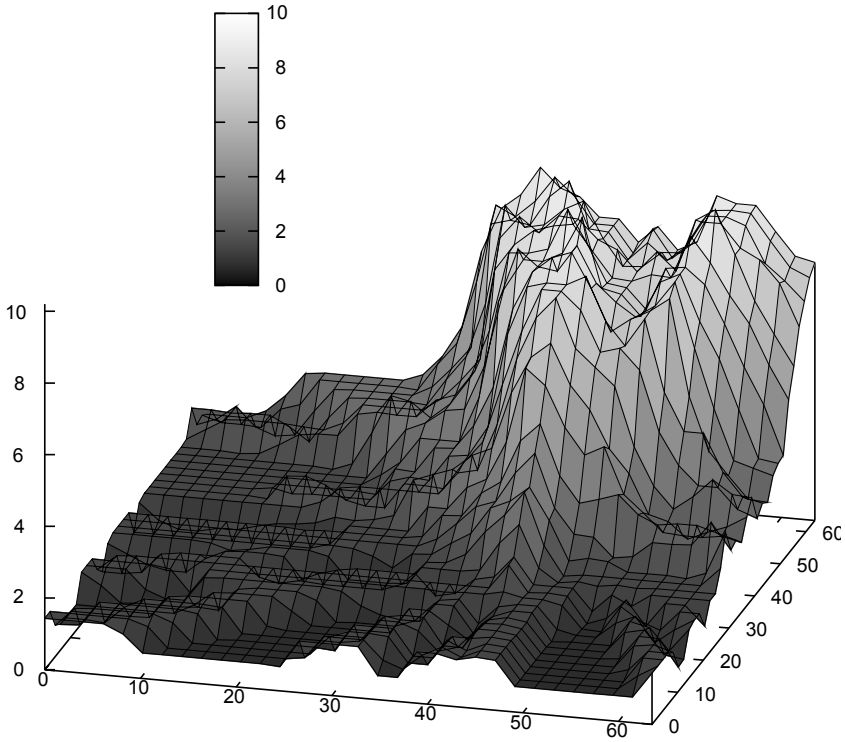
6.3.3. Értékelés

A konzisztenciafüggvény értelmezési tartománya az $\mathbb{R}^3$. A könnyebb megjeleníthetőség érdekében a függvényt két dimenzióba transzformáltam a következő módon:

$$c_{2D}(x, y) = \max_{z \in \mathbb{R}}(c(x, y, z)) \tag{6.11}$$

A 6.4. ábrán egy példa látható a c_{2D} függvényre. A nagy számú szabad rálátással nem rendelkező szenzor és a visszaverődésekkel terhelt környezet miatt a konzisztenciafüggvény globális maximuma mellett egy erős lokális maximum is jelen van. A globális

maximumot támogató szenzorok kis részének átmeneti meghibásodása, vagy hibás mérése esetén a globális maximum már a másik csúcs lesz, így a lokalizációs algoritmus a konzisztensen hibás mérésekből fakadó csúcs következtében a pozíciót rossz helyre becsülheti.



6.4. ábra. A $c_{2D}(x, y)$ transzformált konzisztenciafüggvény háromdimenziós megjelenítése

A bemutatott adaptív lokalizációs algoritmus hatékony működését demonstrálandó, valós méréseken alapuló kísérleteket végeztem. A tesztekhez előre felvett mérési adatokat használtam. A kísérletekben használt hangjelek fegyverlövésekből származtak, ahol a szenzorok a torkolati zaj beérkezési idejét mérték [12].

Mérési elrendezés

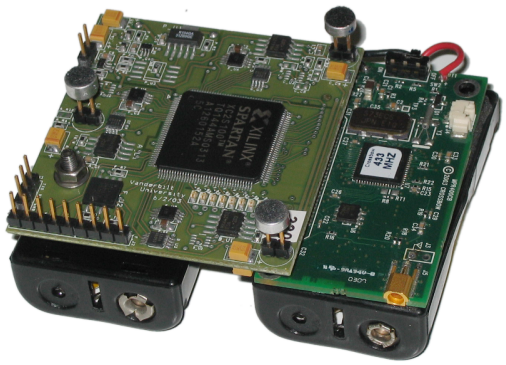
A méréshez használt 56 szenzor városi környezetben, a szabad átlátást gátló, illetve visszaverődéseket okozó épületek között, rögzített pozíciókban kerül elhelyezésre. A szenzorok nagyrészt a földön foglaltak helyet, kisebb részük ablakokba, illetve háztetőkre került. A szenzorok, illetve a lövész helyzetének meghatározása kézzel történt.

A szenzor csomópontok Mica2 [3] eszközök voltak. A szenzorok egy 7,2 MHz-es órajellel működő ATmega128L mikrovezérlőt tartalmaznak. A mikrovezérlőben 4 kB RAM és 128 kB flash memória található. A szenzorok közötti kommunikációt egy CC1000 típusú rádió adóvevő biztosítja, ami a 433 MHz-es ISM sávon működik, maximálisan 38,5 kbps átviteli sebességgel, illetve 100 méteres hatótávolsággal. A szenzorok szoftvere a TinyOS

[21] beágyazott operációs rendszerre épül. Az időszinkronizáció az FTSP [34] segítségével valósult meg, ami néhány mikroszekundumon belüli szinkronizációt eredményezett.

Mivel a szenzor mikrovezérlőjén nem valósítható meg a mérés szempontjából elengedhetetlen nagy sebességű jelfeldolgozás, erre a célra egy speciális kiegészítő kártya készült. A kártya három független csatornával rendelkezik, melyekből a mérések során csak egy került felhasználásra. Mindhárom csatornához egy-egy saját mikrofon tartozik, amelyek jele egy-egy változtatható erősítésű erősítőn át egy-egy 1 MSPS mintavételezési sebességű A/D átalakítóra kerül. A jelfeldolgozási algoritmusok egy Xilinx XC2S100 FPGA áramkörön futnak. A 6.5. ábrán a szenzor, illetve a kiegészítő kártya fényképe látható.

A mérési adatok kiértékeléséhez használt TBALA algoritmus C nyelvű implementációja, illetve a felhasznált mérési adatok az [S5] link alatt érhetők el.



6.5. ábra. A méréshez használt Mica2 típusú mote a hozzá kapcsolódó FPGA alapú jelfeldolgozó kártyával.

Mérési eredmények

A 6.6. ábrán a szenzorok helye, illetve a valós és a becsült forráspozíció látható. A kísérlet során 5 lövést használtam fel, melyek ugyanabból, az ábrán kereszttel jelölt forrás pozícióból (67,46; 25,13; 0,10) származtak. A szenzorokat nagy körökkel jelöltem (a szürkével jelzettek egyetlen mérést sem küldtek a kísérlet során, a feketék legalább egy pozícióbecslésben részt vettek). A számokkal jelölt kis körök az egyes lövések becsült pozícióját jelölik. Az eredeti algoritmus (6.6(a) ábra) az első 4 lövést helyesen becsülte, de az utolsó esetben nagy hibát figyelhetünk meg. A jelenség oka nagy valószínűséggel az, hogy bizonyos, takarásban levő szenzorok egy visszaverő felületről érkező jelet érzékelnek, így konzisztensen rossz méréseket adnak. Elegendő számú együttműködő szenzor esetén a hibás pozíció konzisztenciája magasabb lehet. Az adaptív algoritmus (6.6(b) ábra) megtanulta az egyes szenzorok viselkedését, tehát hogy jellemzően melyek a becslést támogató és melyek a nem támogató példányok. Így az algoritmus az együttműködően rossz mérést adó szenzorokat kisebb súllyal vette figyelembe, ami mind az 5 esetben helyes pozíciót eredményezett. A kísérlet során az adaptivitási paraméter $\alpha = 0,2$ volt. A paraméter értéke azzal a megfontolással került kiválasztásra, hogy egyetlen hibás mérés még ne rontsa

le jelentősen az adott szenzor becsült megbízhatóságát, azonban sorozatosan hibás viselkedés esetén a megbízhatósági faktor gyorsan (néhány mérés alatt) beálljon. A paraméter érékére a 0,2 kísérletileg megfelelőnek bizonyult.

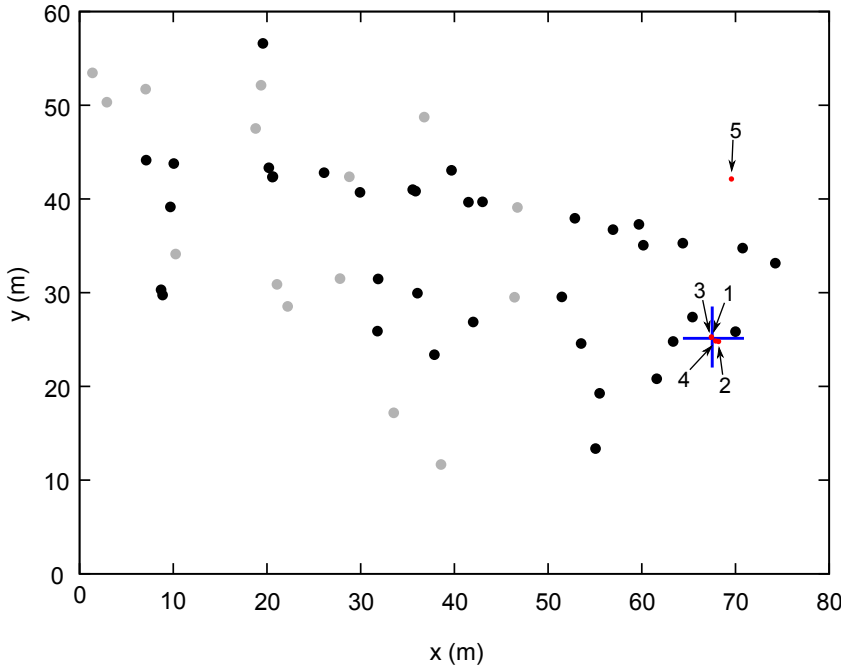
Az adaptív algoritmussal az 5. lövésre kiszámított konzisztenciafüggvény a 6.7. ábrán látható. A világosabb árnyalatok kisebb, a sötétebbek nagyobb értékeket jeleznek. A zöld és piros körök a jó, illetve rossz méréseket adó szenzorokat jelölik. A becsült pozíció a ténylegeset igen jól közelíti. A részletes numerikus eredmények a 6.1. táblázatban szerepelnek.

6.1. táblázat. A hangforrás becsült pozíciója az eredeti, illetve az adaptív lokalizációs algoritmus használatával

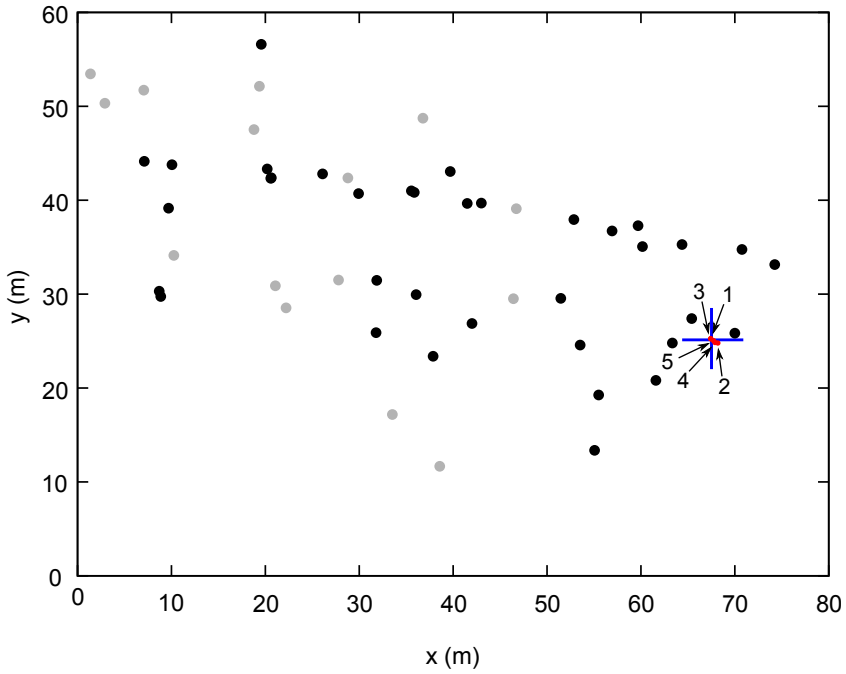
Lövés	Eredeti algoritmus		Adaptív algoritmus	
sorszáma	Pozíció (m)	Hiba (m)	Pozíció (m)	Hiba (m)
1	(67,4; 25,3; −0,5)	0,1	(67,4; 25,3; −0,5)	0,1
2	(68,2; 24,8; −0,5)	0,8	(68,2; 24,8; −0,5)	0,8
3	(67,4; 25,2; −0,5)	0,1	(67,4; 25,2; −0,5)	0,1
4	(67,9; 24,8; −0,5)	0,5	(67,9; 24,8; −0,5)	0,5
5	(69,6; 42,1; 0,9)	17,1	(67,7; 25,0; −0,5)	0,3

6.3.4. Összefoglalás

Ebben a szakaszban egy, a szenzorok megbízhatóságát folyamatosan értékelő és figyelembe vevő, adaptív konzisztencia-függvényt használó lokalizációs algoritmust mutattam be, ami konzisztensen hibás mérések esetén is helyes pozícióbecslőt ad. A módszer különösen visszaverődésekkel terhelt mérési helyszínek esetén alkalmazható sikerrel. A bemutatott algoritmust valós mérésekből származó adatokon teszteltem és összehasonlítottam az eredeti konzisztencia alapú algoritmussal. A bemutatott algoritmus akkor is helyes eredményt adott, amikor az eredeti algoritmus a visszaverődések miatt hibás becslést végzett.

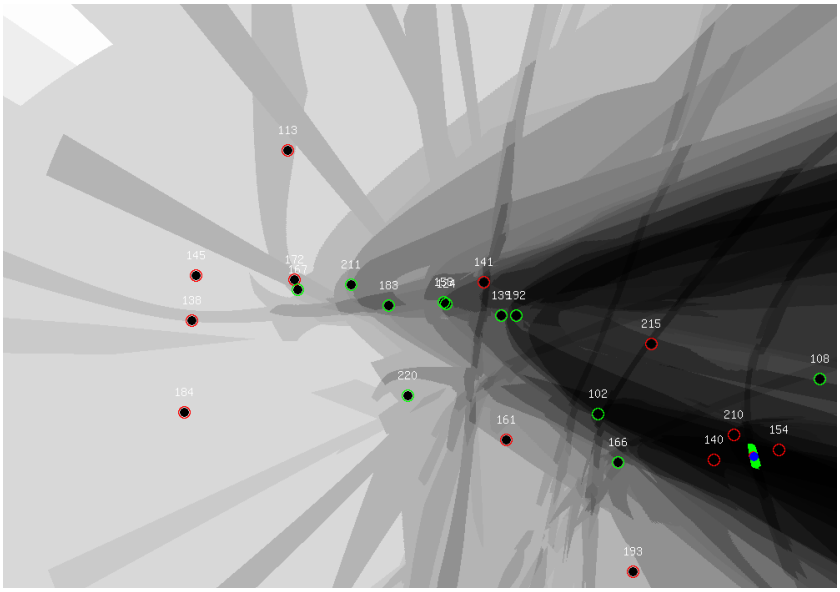


(a)



(b)

6.6. ábra. A mérés során használt terület. A forrást kereszt, a szenzorokat nagy méretű körök, a forrás becsült pozícióit az 5 lövésnek megfelelően számozott kis méretű körök jelzik. A feketével jelölt szenzorok legalább egy kísérlet során küldtek adatot, a szürkével jelöltektől egy mérés sem érkezett. (a) Az eredeti konzisztencia alapú lokalizációs algoritmus csak az első 4 esetben adott helyes eredményt. (b) Az adaptív algoritmus mind az 5 lövés során helyes pozíciót becsült.



6.7. ábra. Az módosított, a megbízhatósági faktorokat is figyelembe vevő konzisztenciafüggvény két dimenzióba transzformálva az 5. lövés esetén.

6.4. A konzisztencia alapú becslő hatékony számítása

6.4.1. Áttekintés

Az előző szakaszban ismertetett konzisztencia-függvény alapú megoldás a függvény maximumának megkeresésére globális keresést használ. Egy nagy méretű háromdimenziós keresési tér estén az algoritmus elfogadhatatlanul sokáig is futhat. Az algoritmus gyorsítására a [12] egy tartományfelezésen alapuló módszert közöl. A következőkben a lokalizáció gyorsítására egy statisztikai alapú algoritmust fogok ismertetni, ami beállítható valószínűséggel képes a konzisztencia-függvény maximumának megtalálására. A javasolt módszer igen nagy kitűzött találati valószínűség mellett is gyorsabb működést mutat az irodalomban közöltnél.

Először az algoritmus alapjául szolgáló RANSAC módszer alapelvét fogom ismertetni, majd bemutatom a módszernek a lokalizáció gyorsítására használt adaptációját. Maga a pozícióbecslés egy lineáris egyenletrendszer megoldásával történik, azonban az így kapott pozíció csak az esetek kis részében pontos. A becslés a lineáris egyenletrendszer nagy mennyiségű szenzorhalmazra való megoldásával, illetve a lehetséges megoldások értékelésével és a legjobb becslés kiválasztásával történik. A szenzorok, illetve a becslés hibájának részletes analízise után ismertetem a szenzorcsoportok elégséges számának meghatározását, illetve a gyorsított lokalizációs algoritmust. A szakaszt végén elvégzem a becslő hibaanalízisét és a mérési hibát összevetem annak elméleti korlátjával.

6.4.2. Az algoritmus vezérelve

A következőkben egy, a [12] megoldásánál hatékonyabb, RANSAC-alapú gyorsítást mutatok be, ami a konzisztencia-függvény maximumát 1-hez tetszőlegesen közeli valószínűséggel képes megtalálni. A RANSAC [93] alapötlete a következő:

1. Egy véletlenszerű (kezdő) halmaz kiválasztása a rendelkezésre álló mérésekből,
2. a kezdő halmazból egy kezdeti becslő kiszámítása,
3. a kimaradó méréseknek a kezdeti becslőhöz való illeszkedésének vizsgálata, illeszkedés esetén a halmazhoz adása,
4. a becslés ismétlése a bővített részhalmazzal,
5. az 1-4. lépések sokszoros ismétlése után a RANSAC a legjobb becslőt választja ki.

A RANSAC előnye, hogy a végső becslés abban az esetben is pontos lehet, ha a kezdeti becslések hibásak (például a nagy mennyiségű hibás mérés miatt), feltéve hogy elegendő jó minőségű mérés áll rendelkezésre, illetve elegendő iterációt végeztünk.

A javasolt megoldás a RANSAC-hoz hasonló elven alapul, fő lépései a következők (a számozással a fenti RANSAC lépéseket követve):

1. 5 véletlenszerű mérés kiválasztása,
2. a kiválasztott ötös alapján a pozícióbecslő meghatározása egy gyors algebrai módszer segítségével,

3-4. a konzisztenciafüggvény kiszámítása a becsült pozícióban az összes rendelkezésre álló mérést felhasználva,

5. a konzisztenciafüggvény vizsgálata alapján az iterációk számának felülvizsgálata, esetleges újabb iterációk beiktatása. Az iterációk számát az algoritmus úgy határozza meg, hogy garantálni tudja a globális maximum megtalálásának előírt valószínűségét.

A 2. lépés becslője (TDoA probléma megoldása) lineáris egyenletrendszer segítségével történik. A [90] eredményei alapján a probléma megoldásához háromdimenziós térben, ismert terjedési sebesség esetén 5 szenzorra (legyenek $S_1, \ldots, S_5$) van szükség. A hangjelek t_i beérkezési idejeire az általánosság megszorítása nélkül feltehetjük, hogy $t_i \leq t_j$, ha $1 \leq i < j \leq 5$. A beérkezési idők között négy független különbség írható fel:

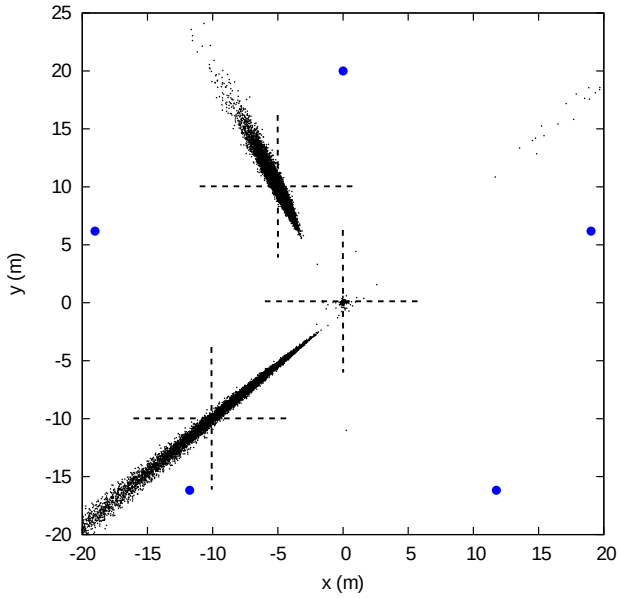
$$\begin{aligned} \Delta t_{12} &= t_2 - t_1 \\ \Delta t_{13} &= t_3 - t_1 \\ \Delta t_{14} &= t_4 - t_1 \\ \Delta t_{15} &= t_5 - t_1 \end{aligned} \tag{6.12}$$

Ha az S_1 szenzor forrástól vett távolságát d-vel jelöljük, akkor a következő egyenletrendszer teljesül:

$$\begin{bmatrix} 2(x_1-x_2) & 2(y_1-y_2) & 2(z_1-z_2) & -2v\Delta t_{12} \\ 2(x_1-x_3) & 2(y_1-y_3) & 2(z_1-z_3) & -2v\Delta t_{13} \\ 2(x_1-x_4) & 2(y_1-y_4) & 2(z_1-z_4) & -2v\Delta t_{14} \\ 2(x_1-x_5) & 2(y_1-y_5) & 2(z_1-z_5) & -2v\Delta t_{15} \end{bmatrix} \cdot \begin{bmatrix} x_s \\ y_s \\ z_s \\ d \end{bmatrix} = \begin{bmatrix} x_1^2+y_1^2+z_1^2-x_2^2-y_2^2-z_2^2 \\ x_1^2+y_1^2+z_1^2-x_3^2-y_3^2-z_3^2 \\ x_1^2+y_1^2+z_1^2-x_4^2-y_4^2-z_4^2 \\ x_1^2+y_1^2+z_1^2-x_5^2-y_5^2-z_5^2 \end{bmatrix} \tag{6.13}$$

Ennek megfelelően a kezdeti pozícióbecslő egy ötös alapján a (6.12) lineáris egyenletrendszer megoldásával számítható ki. Azonban az így kiszámított pozíció két okból kifolyólag is hibás lehet. Egyrészt a kiválasztott ötösben lehetnek olyan mérések, amik nagy, nem kontrollálható mérési hibával rendelkeznek. Másrészt a megoldás az (x_i, y_i, z_i) szenzorpozíciókban, illetve t_i beérkezési időkben előforduló kis hibákra is érzékeny. (Fontos megjegyezni, hogy a szenzorpozíciók hibáját közvetlen módon az időkülönbségekben mérhető hibákra lehet transzformálni, illetve fordítva.) Bizonyos szenzor elhelyezkedések érzékenyebbek a hibákra, mint mások. Ezt a GDOP (geometrical dilution of precision – a pontosság felhígulása) néven ismert jelenséget a GPS irodalma tárgyalja részletesen [94]. Sajnos a kiválasztott ötösök esetén a GDOP nem befolyásolható, így az esetleges rossz GDOP a becslésben nagy hibát eredményezhet, még kis mérési hiba esetén is. A jelenség illusztrálására egy 5 szenzorból álló teszt elrendezést használtam. A szenzorokat egy

szabályos ötszög csúcsaiban helyeztem el, és három különböző forráspozíciót használtam, ahogy az a 6.8. ábrán látható. Mindegyik esetre 10000 pozícióbecslést végeztem a (6.13) egyenletrendszer használatával. A becsült pozíciókat az ábrán pontok jelzik. A szenzorok $\sigma = 0{,}3$ m szórású normál eloszlású zajjal rendelkeztek. Míg a középpontban elhelyezett forrás pozícióját a módszer alacsony hibával becsülte, a másik két pozíció esetében nagy, akár 10 métert is elérő becslési hiba is jelentkezett.

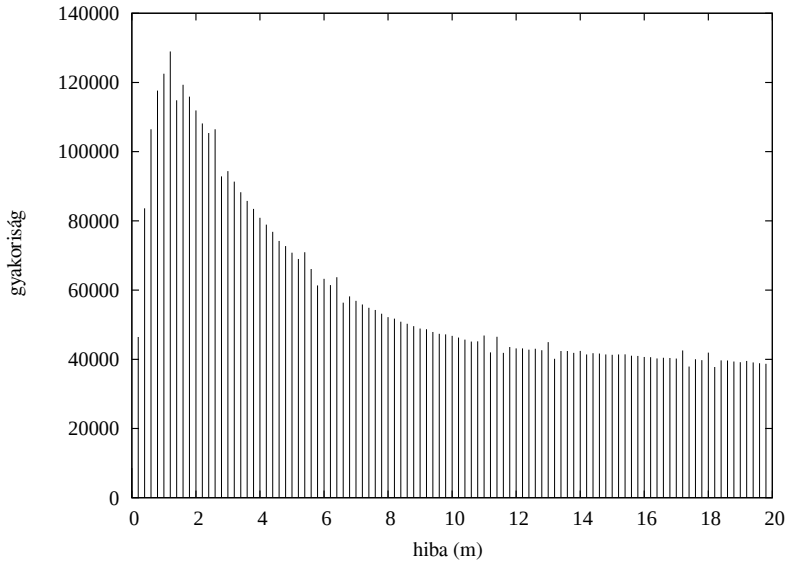


6.8. ábra. A kezdő pozícióbecslők eloszlása (kis pontok) a (6.13) egyenletrendszer használatával. A három forrás (keresztekkel jelölve) pozíciója (0; 0), (-10; -10) és (-5, 10). A kék körrel jelölt szenzorok $\sigma = 0{,}3$ m szórású normál eloszlású hibával rendelkeznek.

A lineáris egyenletrendszeren alapuló megoldást különböző szenzor- illetve forráspozíciókra valós mérésekkel is teszteltem. A teszt során $2 \cdot 10^7$ esetben számítottam ki a becsült forráspozíciót, amit összevetettem a hozzá tartozó valóssal. A kísérlet során használt valós méréseken alapuló, a 6.10. ábrának megfelelő e_i szenzorhibát használtam. Az egyenletrendszer alapján meghatározott becslő hibájának eloszlása a 6.9. ábrán látható. A becsléseknek csak igen kis része esik 1 m alá.

6.4.3. Véletlenítéssel gyorsított lokalizációs algoritmus

Látható, hogy a (6.13) egyenletrendszer által előállított becslő még akkor is igen kevéssé megbízható, ha a mérések helyesek. Nagy számú ötöst kiértékelve azonban az előállított becslések között helyesek is találhatók, tehát elegendően nagy számú ötöst megvizsgálva biztosítható legalább egy helyes pozícióbecslő. Ebben a szakaszban arra fogok becslést adni, hogy korrekt méréseket feltételezve hány iteráció szükséges egy helyes becslés előállításához, illetve ezt felhasználva arra is, hogy adott számú helyes, illetve hibás mérést adó szenzor esetén hány próbálkozásra, tehát a konzisztenciafüggvénynek hány véletlenszerűen választott ötösből képezett pozícióbecslőn való kiértékelésére van szükség legalább egy



6.9. ábra. A kezdő pozícióbecslők hibájának eloszlása a (6.13) egyenletrendszer használatával, valós mérések alapján. A teszt során csak a legfeljebb 0,5 m hibával rendelkező méréseket használtam fel.

helyes pozícióbecslés előállításához. Az optimális pozícióbecslés előállítására egy algoritmust javaslok, aminek bemenő paraméterei a kívánt találati valószínűség, a szenzorok száma, illetve a helyes mérést adó szenzorok becsült száma. Az algoritmus képes a helyesen működő szenzorok számára vonatkozó becslés futás közbeni korrekciójára.

Jelöljük a véletlenszerűen kiválasztott ötösök számát K-val. A következőkben egy olyan módszert fogok bemutatni, amivel egy bizonyos garantált találati valószínűség mellett K értéke meghatározható.

Jelöljük a k-adik ötöst Q_k-val és a Q_k alapján a (6.13) egyenletrendszer segítségével becsült pozíciót L_k-val ($1 \leq k \leq K$). Egy szenzort *megbízhatónak* nevezünk, ha a w ablakméret felénél kisebb mérési hibát mutat, Q_k-t pedig akkor nevezzük *megbízhatónak*, ha a benne szereplő 5 szenzor mindegyike megbízható. L_k-t akkor nevezzük *helyesnek*, ha a becslés hibába egy bizonyos δ küszöb alatt van. Definiáljuk annak valószínűségét, hogy a (6.13) egyenletrendszer megbízható mérések alapján helyes becslőt eredményez:

$$P^* = P(L_k \text{ helyes} | Q_k \text{ megbízható}) \tag{6.14}$$

Jelölje P_{hit} annak valószínűségét, hogy a K becsült pozíció közül legalább egy helyes. Célunk egy olyan, elegendően magas K választása, ami magas P_{hit}-et eredményez. A következőkben a P_{hit} és K között fennálló összefüggést fogom levezetni. Ha $P^+ = P(Q_k \text{ megbízható} | L_k \text{ helyes})$, akkor

$$P(L_k \text{ helyes}) P^+ = P(Q_k \text{ megbízható}) P^* \tag{6.15}$$

Ha a megbízható szenzorok száma n, akkor

$$P(Q_k \text{ megbízható}) = \frac{\binom{n}{5}}{\binom{N}{5}}. \tag{6.16}$$

A (6.15) és (6.16), valamint $P^+ \leq 1$ alapján a következő egyenlőtlenség teljesül:

$$P(L_k \text{ helyes}) \geq \frac{\binom{n}{5}}{\binom{N}{5}} P^*. \tag{6.17}$$

Tehát annak valószínűsége, hogy K ötösből legalább egy esetben helyes pozícióbecslést kapunk, a következő:

$$P_{\text{hit}} = 1 - (1 - P(L_k \text{ helyes}))^K \geq 1 - \left(1 - \frac{\binom{n}{5}}{\binom{N}{5}} P^*\right)^K. \tag{6.18}$$

Ha P_{hit} tervparaméter, akkor K-ra a következő felső korlát számítható:

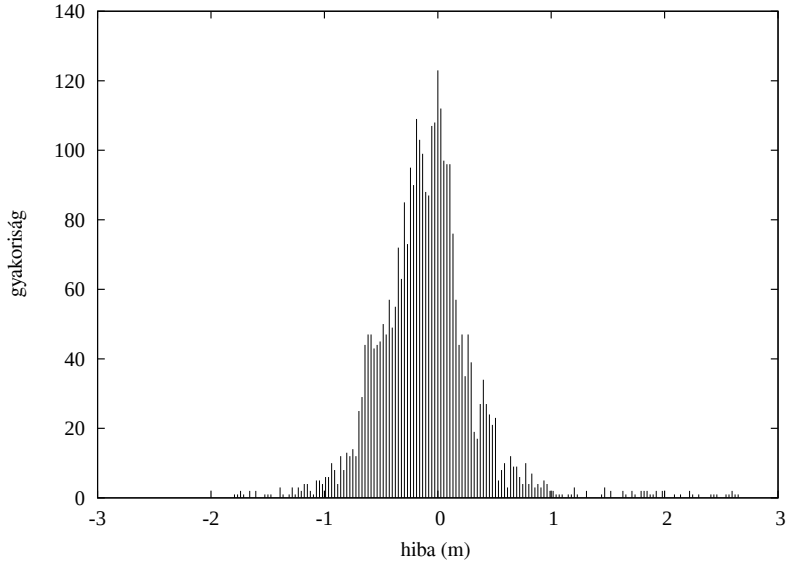
$$K = \frac{log(1 - P_{\text{hit}})}{log\left(1 - \frac{\binom{n}{5}}{\binom{N}{5}} P^*\right)} \tag{6.19}$$

Ahogy a 6.8. ábrán is látható, P^* erősen függ a szenzorok, illetve a forrás elhelyezkedésétől (az ábrán látható példában a középső forráspozícióban P^* jelentősen nagyobb, mint a másik két estben), illetve a mérési hibák Γ eloszlásától. P^* becslésére a következő, szimuláció-alapú megközelítést alkalmaztam.

$counter \leftarrow 0$
repeat E times
  Válasszunk az ismert pozíciójú szenzorok közül véletlenszerűen
    egy Q ötöst, illetve a keresési tér bármely pontján egy L forrást.
  Számítsuk ki a τ_i pontos mérési adatokat a Q ötösre, illetve
    az L forrásra, majd adjunk a mérésekhez e_i, Γ eloszlással
    rendelkező zajt, így a t_i $(1 \leq i \leq 5)$ zajos mérésekhez jutva.
  Számítsuk ki az L_k pozícióbecslőt a (6.13) egyenletrendszert használva.
  if $|L_k - L| < \delta$ **then**
    $counter \leftarrow counter + 1$
$P^* = counter/E$

A szenzorok mérési hibájának Γ eloszlása specifikáció (ha elérhető), vagy teszt mérések alapján határozható meg. Γ meghatározását előzőleg gyűjtött mérési adatok alapján végeztem. A hiba eloszlása a 6.10. ábrán látható. A 3 méternél nagyobb hibákat a vizsgálat során nem vettem figyelembe.

Hogy ellenőrizzem a P_{hit} és K között fennálló összefüggést, a következő kísérletet végeztem. Egy bizonyos eseményhez tartozó mérésekből K véletlenszerű ötöst választottam



6.10. ábra. A szenzorok mérési hibájának eloszlása.

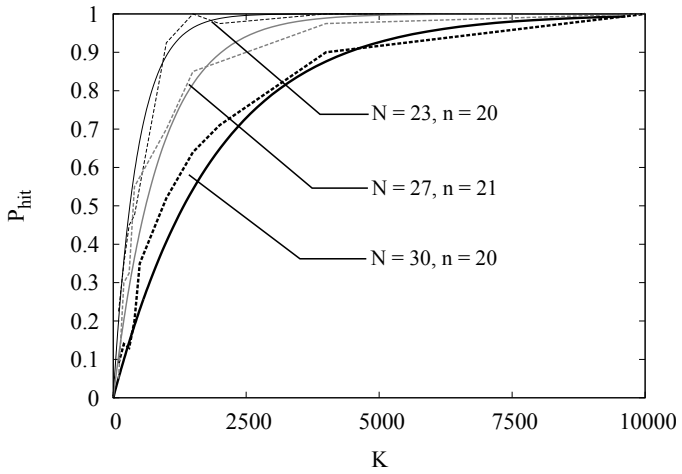
ki, majd minden esetben kiszámítottam az ötösből számított pozíció hibáját. Azokban az esetekben, amikor a hiba a $\delta = 0{,}5$ m küszöb alatt volt, a $counter_K$ számlálót növeltem. K értékét 100 és 10000 között változtattam, minden esetben 1000 kísérletet végezve. A kísérlet során kapott P_{hit} (tehát a $\frac{counter_K}{K}$) értékek a 6.11. ábrán láthatók, K függvényében ábrázolva, három különböző esetben. Az ábrán az egyes görbékhez tartozó, (6.19) egyenlet alapján számított $P_{\text{hit}}(K)$ függvények is szerepelnek. A számításhoz $P^* = 0{,}0048$ értéket használtam, a fentebb bevezetett szimuláció alapú módszer alapján, a 6.10. ábrán látható hibaeloszlást $[-0{,}5\text{ m}; 0{,}5\text{ m}]$ közé szűkítve, illetve a kísérleteknél is használt $\delta = 0{,}5$ m küszöbértéket használva. A három görbepárhoz tartozó N (a mérésben szereplő szenzorok száma), illetve n (a megbízható szenzorok konzisztencia-függvény alapján, a posteriori kiszámolt száma) 23 és 20, 27 és 21, illetve 30 és 20 voltak. Az ábrán világosan látható az elméleti, illetve a gyakorlati görbék jó egyezése.

Az imént bemutatott módszerre épülő gyorsított lokalizáció folyamatábrája a 6.12. ábrán látható.

- Az *1. lépésben* a bemenő paraméterek beállítása történik. N a rendelkezésre álló mérések száma, $P_{\text{hit}}(K)$ egy 1-hez közeli tervparaméter, n (a megbízható szenzorok száma) pedig csupán egy becslés. Az n paraméter kezdeti meghatározása például előzetes mérések alapján történhet, az algoritmus ezt a számot adaptív módon helyesbíti.

- A *2. lépésben* K kezdeti értékének kiszámítása történik.

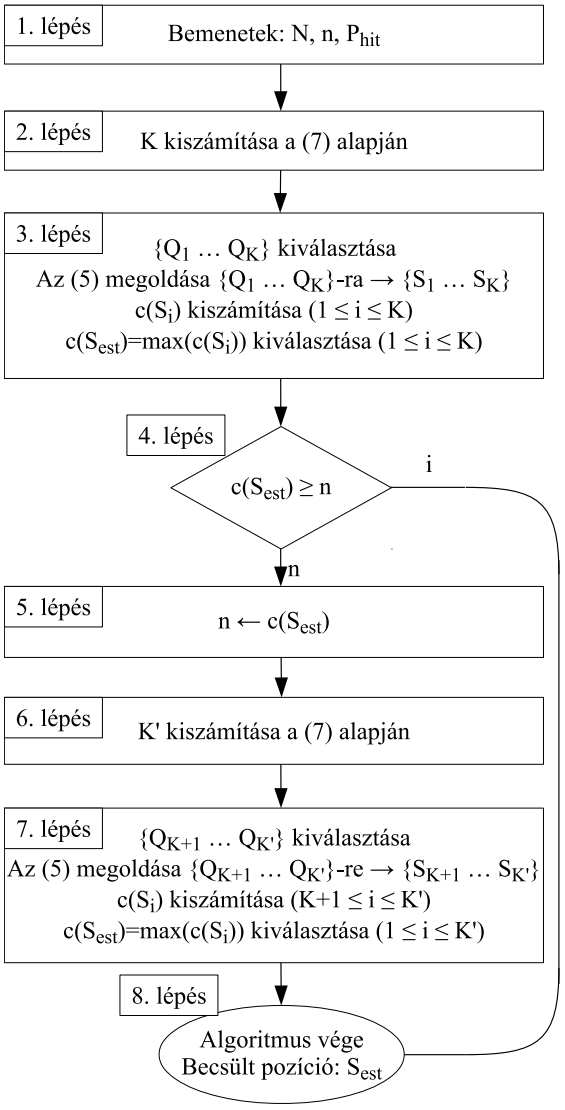
- A *3. lépésben* K pozícióbecslő kiszámítása, majd mindegyik pontban a konzisztenciafüggvény kiértékelése történik. A legjobb becslés L_{est}, $c(L_{\text{est}})$ konzisztencia értékkel.



6.11. ábra. A P_{hit} elméleti (folytonos görbék), illetve mért (szaggatott görbék) értéke három különböző mérési beállításra (vékony, normál, illetve vastag vonalak), K értékének függvényében.

- Ha a megbízható szenzorok számára tett n becslés nem volt túl magas, tehát ténylegesen volt legalább n megbízható szenzor, akkor a folyamat a *4. lépésben* véget ér.
- Ha a legnagyobb megtalált konzisztencia érték kisebb, mint n, akkor az algoritmus az *5. lépésben* n értékét $c(L_{\text{est}})$-re csökkenti.
- Majd a *6. lépésben* egy új (magasabb) K' érték kerül kiszámításra
- A pozícióbecslő a *8. lépésben* az újabb $K' - K$ ötösre is kiszámításra kerül, majd az algoritmus az új pontok konzisztenciáját is kiértékeli. Az algoritmus a K' pozíció közül a legmagasabb konzisztenciafüggvénnyel rendelkezőt választja mint pozícióbecslő.

  Az véletlenítéssel gyorsított lokalizációs algoritmus C nyelven történt implementációja, illetve a futtatáshoz szükséges bemenő adatok az [S6] linken érhetők el.



6.12. ábra. A véletlenítéssel gyorsított lokalizációs algoritmus folyamatábrája

6.2. táblázat. Az 5 lövés esetén mért számítási idő az eredeti, illetve a véletlenített algoritmussal

Lövés	Eredeti algoritmus		Véletlenített algoritmus				Gyorsulás
sorszáma	T_{comp}	N_{cf}	T_{comp}	T_{init}	T_{cf}	N_{cf}	
1	408 s	$4,2 \cdot 10^7$	95 ms	25 ms	70 ms	$11 \cdot 10^3$	4316
2	547 s	$4,2 \cdot 10^7$	139 ms	29 ms	110 ms	$14 \cdot 10^3$	3948
3	342 s	$4,2 \cdot 10^7$	41 ms	14 ms	27 ms	$5.2 \cdot 10^3$	8442
4	408 s	$4,2 \cdot 10^7$	98 ms	28 ms	70 ms	$12 \cdot 10^3$	4152
5	428 s	$4,2 \cdot 10^7$	389 ms	103 ms	286 ms	$48 \cdot 10^3$	1102

6.4.4. Hibaanalízis

A pozícióbecslő hatékonyságát a becsült pozíciók átlagos négyzetes hibájának a Cramér-Rao korláttal (CRLB) való összevetésével vizsgáltam. A számításokat a [95] és a [96] alapján végeztem.

$$CRLB(x, y, z) = trace(\mathbf{F}^{-1}), \tag{6.20}$$

ahol $\mathbf{F}$ a Fisher-féle információs mátrix:

$$\mathbf{F} = \frac{1}{v^2}\mathbf{G}^T\mathbf{Q}^{-1}\mathbf{G}, \tag{6.21}$$

ahol

$$\mathbf{Q} = \sigma^2[\mathbf{I}_{(N-1)\times(N-1)} + \mathbf{1}_{N-1}\mathbf{1}_{N-1}^T] \tag{6.22}$$

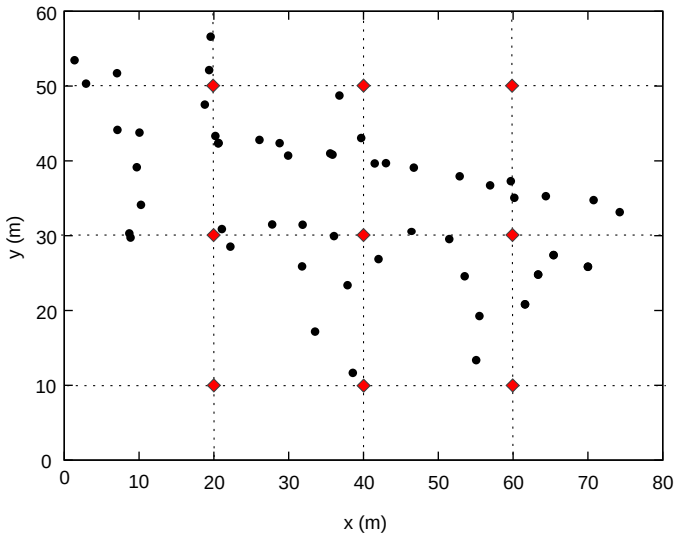
és

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_2^T - \mathbf{g}_1^T \\ \vdots \\ \mathbf{g}_N^T - \mathbf{g}_1^T \end{bmatrix} \tag{6.23}$$

valamint

$$\mathbf{g}_i^T = \frac{[x, y, z] - [x_i, y_i, z_i]}{||[x, y, z] - [x_i, y_i, z_i]||_2}, 2 \leq i \leq N. \tag{6.24}$$

Az összehasonlításhoz 9 tesztpontot vettem fel a keresési térben, ezeket a 6.13. ábrán 45 fokos szögben álló négyzetekkel jelöltem. Az egyes pontok különböző lefedettségű területeken helyezkednek el. A tesztpontokban a CRLB-t a (6.20)-(6.24) egyenleteknek megfelelően számítottam ki, az átlagos négyzetes hibát pedig az egyes pontokban 100-100 mérés átlagolásával nyertem. A felhasznált szimulációs adatok előállításához e_i helyén 0 várható értékű $\sigma = 0{,}3$ m szórású normál eloszlású zajt használtam, az ablak szélességét pedig $w = 0{,}4$ m-re választottam. A kapott eredményeket a 6.3. táblázat foglalja össze.



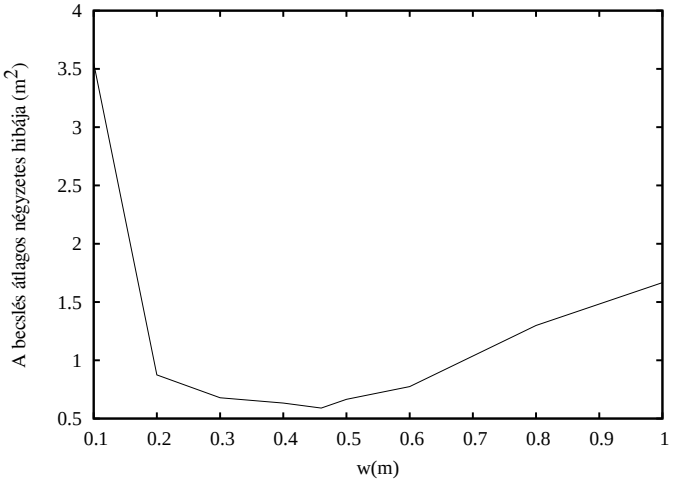
6.13. ábra. A szenzorok elhelyezkedése a kísérleti helyszínen (fekete körök), illetve a hibaanalízis során használt 9 forráspozíció (piros négyzetek).

6.3. táblázat. A pozícióbecslő átlagos négyzetes hibája (MSE) és a hozzá tartozó Cramér-Rao korlát (CRLB) a keresési tér különböző pontjain.

Pozíció	MSE	CRLB
(20; 10; 0)	0,96	0,51
(20; 30; 0)	0,25	0,11
(20; 50; 0)	0,18	0,08
(40; 10; 0)	1,11	0,43
(40; 30; 0)	0,29	0,13
(40; 50; 0)	0,17	0,08
(60; 10; 0)	1,77	0,68
(60; 30; 0)	0,47	0,21
(60; 50; 0)	0,48	0,21

Habár a becslés nyilvánvalóan nem optimális, a kapott eredmények igen jól megközelítik a Cramér-Rao korlátot.

A becslés hibája a w ablakmérettől is függ. Bár logikusan a $w = 2e_{\max}$ egy jó választás lehet, érdemes w hatását pontosabban is megvizsgálni. A vizsgálathoz az átlagos négyzetes hiba és a Cramér-Rao korlát összevetéséhez is használt tesztpontokat használtam. Az ablakméretet 0,1 és 1 m között változtattam, a szenzorok ebben az esetben is $\sigma = 0{,}3$ m nagyságú, normál eloszlású zajjal rendelkeztek. Minden pontra, illetve minden w értékre 100 kísérletet végeztem, a kapott négyzetes hibákat pedig minden w-re átlagoltam. Az egyes ablakméretek esetén tapasztalt átlagos négyzetes hibák a 6.14. ábrán láthatóak.



6.14. ábra. Az átlagos négyzetes hiba különböző w ablakméretek estén, $\sigma = 0{,}3$ m szenzorhibával.

Az optimális ablakméret 0,4 és 0,5 m között van, ami jó egyezést mutat az előzetes becsléssel. Az ábráról az is leolvasható, hogy a becslő w pontos értékére nem túlságosan érzékeny; a grafikon az optimális érték körül meglehetősen lapos. A hiba w túlságosan alacsony értékeinél gyors emelkedést mutat, ami azt mutatja, hogy az ablakméretet nem célszerű a zajszint alá választani. A becslő a magasabb w értékek irányában kevésbé érzékeny.

6.4.5. Összefoglalás

Ebben a szakaszban egy RANSAC-alapú gyorsítást mutattam be a konzisztenciafüggvény alapú pozícióbecslőhöz. A módszer a keresési térben egy olyan ponthalmazt választ ki, amely pontjain a konzisztenciafüggvényt kiértékelve az egész keresési térre vett globális optimum nagy valószínűséggel megtalálható. Ez, az algoritmus paraméterezésével állítható valószínűség az 1-et tetszőlegesen megközelítheti; nagyobb valószínűségekhez nagyobb számítási idő tartozik. Valós méréseken tesztelve, nagy megbízhatóságú ($P_{\text{hit}} = 99,99\%$) beállításban a bemutatott gyorsítási eljárás 1000-szeres és 8000-szeres közötti gyorsulást mutatott a kimerítő kereséssel szemben. Az algoritmus az eddig létező egyéb gyorsítási eljárásoknál is hatékonyabb számítást eredményez.

7 Összefoglalás

Dolgozatom 2-5. fejezeteiben különböző kommunikációs megoldásokat, illetve az azokat megvalósító köztesréteg szolgáltatásokat mutattam be. A 2. fejezetben egy elárasztáson alapuló forgalomirányítási algoritmust mutattam be, ami a perkoláció jelenségét használja fel arra, hogy nagy lefedettséget érjen el az egyszerű elárasztáshoz képest jóval kevesebb üzenettel. A módszer működését végtelen nagy hálózatokra bizonyítottam, a véges hálózatokon való viselkedést szimulációk segítségével vizsgáltam, illetve meghatároztam a $K_{\min}$ tervparaméter célszerűen használható értéktartományát. A módszer használatával különböző sűrűségű hálózatok esetén az egyszerű elárasztáshoz képest 30-90%-kal kevesebb csomagra volt szükség a közel 100%-os lefedettség eléréséhez. Az algoritmust különböző csomagvesztési hibával rendelkező hálózatokra is teszteltem, a közel teljes lefedettség ezekben az esetekben is elérhető volt.

A 3. fejezetben egy körkörös topológiákban használható garantált kézbesítési idővel rendelkező, hibatűrő, alacsony energiafogyasztású TDMA algoritmust, illetve az algoritmust megvalósító köztesréteg szolgáltatást mutattam be, amit többkörös hálózatokra is kiterjesztettem. Az algoritmust valós eszközökön teszteltem; a tesztek során mind az időgarantált kézbesítést, mind a hibatűrő működést, mind pedig az alacsony energiafogyasztást részletes tesztek segítségével vizsgáltam.

A 4. fejezetben olyan, körkörös topológiákban használható TDMA ütemezéseket vizsgáltam, ahol egyszerre több szenzor csomópont is adhat, amennyiben egymás vételét nem zavarják. Az ütemezéseket gráfelméleti módszerekkel modelleztem, az optimális (legrövidebb) ütemezés meghatározására egy polinom idejű algoritmust (OMTS) adtam, aminek egy heurisztikákkal gyorsított változatát (OMTS-A) is kidolgoztam. Bizonyítottam, hogy mind az OMTS, mind pedig az OMTS-A által előállított ütemezés optimális. Az implementált algoritmusok futási idejét tesztek segítségével mértem.

Az 5. fejezetben egy olyan kommunikációs protokollt, illetve annak implementációját mutattam be, amely adatgyűjtő alkalmazásokban kis kitöltési tényezőjű kommunikációt valósít meg, illetve a szenzoroktól a nyelő felé mutató irányon kívül a fordított irányban is lehetőséget ad adatok továbbítására. A bemutatott módszer a fordított irányban várakozó csomagok jelzésére a rendszerben egyébként is jelen levő nyugta egy kihasználatlan bitjét használja, így gyakorlatilag nem igényel többlet energiát. A javasolt protokollt valós hardveren implementáltam, az energiafogyasztást egy referencia protokollhoz hasonlítva vizsgáltam.

A dolgozat 6. fejezetében az irodalomban megtalálható konzisztencia-függvény alapú akusztikus lokalizációs módszerhez javasoltam pontosságot, illetve sebességet javító eljárásokat. A 6.3. alfejezetben egy olyan módszert mutattam be, ami a szenzorok megbízhatóságát folyamatosan mérve, illetve ezt a faktort figyelembe vége növeli a becslő pontosságát. A módszer előnyei főleg visszaverődésekkel terhelt környezetben, szabad rá-

látással nem rendelkező szenzorok esetén mutatkoznak meg. Az implementált algoritmust valós mérési adatokon alapuló szimulációval teszteltem.

A 6.4. alfejezetben az akusztikus lokalizációs módszer gyorsítására egy véletlenítésen alapuló algoritmust javasoltam, ami megfelelő paraméterezéssel az 1-hez tetszőlegesen közeli valószínűséggel képes a globális keresés által megtalált legjobb pozícióbecslő előállítására. A becslő hibáját összevetettem az elméleti korláttal, amit az igen jól megközelít. Az algoritmus alacsony hibáját, illetve futási idejét valós méréseken alapuló szimulációkkal is alátámasztottam.

8 Új tudományos eredmények

1. Perkoláció alapú elárasztási algoritmust javasoltam (2 fejezet), ami kizárólag lokális információ felhasználásával képes csökkenteni az üzenetszórási vihart. Az algoritmus képes az üzenetek számának jelentős csökkentésére a magas kézbesítési arány megtartása mellett. Az algoritmus használhatóságát elméleti eredményekkel bizonyítottam és gyakorlati tesztekkel támasztottam alá.

   1.1. A javasolt algoritmus (2.2. alfejezet) szerint az első lépésben a kezdeményező csomópont egy csomagot küld a szomszédainak, a további lépések során a többi csomópont a vett csomagot annak első vétele után egy, a szomszédainak számától és egy $K_{\min}$ tervparamétertől függő valószínűséggel (lásd (2.2) egyenlet) ismétli meg. A csomagok újabb példányai eldobásra kerülnek.

   1.2. Az algoritmus használhatóságát elméletileg is alátámasztottam (2.2. alfejezet). Bebizonyítottam, hogy egy Poisson Boolean folyamat által λ sűrűséggel és r kommunikációs hatósugárral előállított perkoláló végtelen hálózatban létezik λ-tól független $K_{\min} < \infty$ tervparaméter úgy, hogy ha a minden csomópont az 1.1 pontban definiált algoritmus szerint működik, akkor minden üzenet 1 valószínűséggel fog végtelen sok csomóponthoz elérni.

   1.3. Az algoritmus helyes működését és performanciatulajdonságait kiterjedt szimulációs vizsgálatokkal támasztottam alá (2.3. alfejezet). Több eltérő globális, illetve lokális sűrűséggel rendelkező hálózatban megmértem a lefedettséget, illetve az elküldött csomagok számát az algoritmus $K_{\min}$ tervparaméterének, illetve a csomagvesztési aránynak függvényében. Meghatároztam $K_{\min}$ gyakorlatban használható értéktartományát. A vizsgált hálózatokban $K_{\min}$ megfelelő választása mellett az egyszerű elárasztáshoz képest 30-90%-os üzenetszám csökkenést tapasztaltam közel 100%-os lefedettség mellett.

   A tézis eredményei az alábbi publikációkban jelentek meg: [P3], [P9], [P10].

   A perkoláció alapú elárasztással nagy sűrűségű hálózatokban hatékonyan, az egyszerű elárasztáshoz képest jelentősen csökkentett üzenetszámmal terjeszthető el olyan információ, ami az összes csomópont, vagy azok legnagyobb része számára fontos, például az egész hálózatra vonatkozó globális beállítások, illetve programok. Egy másik lehetséges alkalmazási terület az útvonalak feltérképezése.

2. Körkörös és többkörös hálózatokban használható TDMA algoritmusokat és ezeket megvalósító köztesréteg szolgáltatásokat javasoltam, amelyek garantált kézbesítési időt, hibatűrést és energiahatékony működést valósítanak meg (3. fejezet). Eljárást dolgoztam ki a körkörös TDMA hálózatokat egyszerre több párhuzamos adóval működtető optimális ütemezés meghatározására (4. fejezet).

2.1. Körkörös hálózatokban használható garantált kézbesítési idejű, hibatűrő és energiahatékony TDMA alapú kommunikációs algoritmust és az azt támogató köztesréteg szolgáltatást javasoltam (3.2. alfejezet). Az előre megtervezett, rögzített TDMA ütemezés garantálja a szigorú valós idejű működést, ezáltal garantálja a kézbesítési időt is. A hibatűrő működést az algoritmusba épített önellenőrző mechanizmus, valamint a javításra fenntartott időszeletekben végrehajtott önjavító kommunikáció biztosítja. Az algoritmus a szomszédok között szigorú időszinkronizációt használva képes nagyon kis kitöltési tényezővel üzemelő, ezáltal energiahatékony hálózatok megvalósítására.

2.2. A körkörös topológiában használható algoritmust kiterjesztettem többkörös hálózatokra is (3.3. alfejezet). A kifejlesztett algoritmus többkörös hálózatban lehetőséget ad az egyes alkörök eltérő sebességgel való működtetésére. A kiterjesztett algoritmus továbbra is garantált kézbesítési idejű, hibatűrő és energiahatékony működést tesz lehetővé.

2.3. Megvizsgáltam a körkörös topológiájú TDMA-t alkalmazó hálózatokban a kézbesítési idő csökkentésének lehetőségét abban az esetben, amikor egyszerre több, egymás vételét nem zavaró csomópont is adhat egyszerre (4. fejezet). A párhuzamos ütemezéssel elérhető sebességnövekedésre elméleti korlátot adtam (4.3. alfejezet). Polinom idejű algoritmust adtam (OMTS), amely képes az optimális (legrövidebb kézbesítési időt adó) ütemezés előállítására. Az OMTS algoritmust heurisztikus módszerekkel gyorsítottam (OMTS-A), így a gyakorlati esetek többségében az algoritmus futási idejét – az optimum megtartása mellett – jelentősen sikerült csökkenteni.

A tézis eredményei az alábbi publikációkban jelentek meg: [P2], [P5], [P6], [P7].

A javasolt módszer például elosztott megközelítésű biztonságtechnikai alkalmazásokban használható fel. Itt a megszokott (sérülékeny) központi egység helyett minden eseményről minden csomópont értesül és a döntéseket a beavatkozó eszközök autonóm módon hozzák. A javasolt módszer jól illeszkedik a biztonsági rendszerek természetes követelményeihez, az időgarantált kézbesítés, valamint a hibatűrés igényéhez. Az alacsony energiafogyasztás hatékonyan támogatja az elemről történő hosszú idejű működést.

3. Eljárást dolgoztam ki a konzisztencia-függvény alapú pozícióbecslő szolgáltatás szisztematikusan hibás mérések jelenléte esetén tapasztalható bizonytalanságának javítására. Véletlenítésen alapuló módszert adtam a konzisztencia-függvény alapú pozícióbecslő gyorsítására. Matematikai módszerrel bizonyítottam, hogy a módszer pontossága jól közelíti a hiba elméleti minimumát (6. fejezet).

3.1. Módszert adtam a konzisztencia alapú lokalizációs módszer továbbfejlesztésére olyan esetekben, amikor a hálózat szisztematikusan jó, illetve szisztematikusan hibás méréseket adó szenzorokat is tartalmaz (6.3. szakasz). A javasolt eljárásban a szenzorok megbízhatóságát folyamatosan becsüljük, a szenzorok

méréseit pedig adaptív módon a megbízhatóság szerint vesszük figyelembe a fúzió során.

3.2. Módszert adtam a konzisztencia-függvény alapú lokalizáció kiértékelő mechanizmusának gyorsítására (6.4.3. szakasz). A módszer a keresési tér egy olyan részhalmazát állítja elő, amelyben a keresés gyorsabb, mint a teljes térben, viszont a globális optimum megtalálásának valószínűsége az algoritmus megfelelő paraméterezésével 1-hez tetszőlegesen közeli értékre állítható be.

3.3. Meghatároztam a javasolt módszer pontosságának elméleti (Cramér-Rao) korlátját és valós méréseken alapuló szimulációk segítségével megvizsgáltam az algoritmus pontosságát (6.4.4. szakasz). A vizsgálatok tanúsága szerint a javasolt algoritmus hibája gyakorlati szempontból közel van az elméleti határhoz.

A tézis eredményei az alábbi publikációkban jelentek meg: [P1], [P8].

A módszer a beérkezési idők különbsége (TDoA) mérési elven működő lokalizációs szolgáltatásokhoz nyújt gyors és nagy pontosságú pozícióbecslőt. A módszer előnyös tulajdonságai különösen visszaverődésekkel terhelt környezetben, illetve nagy számú takarásban levő szenzor esetén jelentkeznek, emiatt a javasolt eljárás valós alkalmazásokban jól használható.

9 Major results and summary of accomplishments

1. I proposed a percolation based flooding algorithm, which can reduce the broadcast storm using only local information. The algorithm is able to highly reduce the number of transmitted messages while producing high delivery ratio. I proved the usability of the algorithm mathematically and confirmed it with practical tests.

   1.1. According to the proposed algorithm first the source node broadcasts the message to its neighbors, then in the consecutive steps the other nodes relay the received message after its first reception with a probability depending from the number of neighbors and the design parameter K_{min}. The later copies of the same message are ignored.

   1.2. I theoretically confirmed the usability of the algorithm. I proved, that in any infinite percolating network generated by a Poisson Boolean process with density λ and r communication radius there exists a design parameter $K_{min} < \infty$ independent from λ, such that if all nodes apply the algorithm described in (1.1), then the message reaches all nodes with probability 1.

   1.3. I validated the performance of the algorithm with extensive simulations. I measured the delivery ratio and the number of transmitted messages as a function of the K_{min} design parameter and the packet loss ratio in networks with different local and global densities. I determined the practical interval for K_{min}. I observed 30-90% less transmitted messages in the examined networks compared to the original flood routing, with near 100% coverage.

   The results of the thesis were published in [P3], [P9], and [P10].

   The percolation based flooding algorithm is an effective solution for data distribution, where all nodes, or a large number of them is interested in the disseminated information, e.g. for network-global configuration data, or node firmwares. Another possible application area is the route discovery. This method requires highly reduced number of transmitted messages, compared to the original flood routing.

2. I proposed TDMA algorithms and their middleware implementations for ring and multiring topology networks, providing guaranteed delivery time, fault tolerance and high energy efficiency. I created a procedure to determine the optimal schedule for ring topology TDMA networks operating with parallel transmissions.

   2.1. I proposed a TDMA-based communication algorithm and its middleware implementation for ring topology networks, providing guaranteed delivery time,

fault tolerance and high energy efficiency. The precalculated and fixed TDMA schedule guarantees the hard real time operation, and therefore the guaranteed delivery time. The fault tolerant operation is provided by self-checking and self-healing mechanisms. The low duty cycle energy efficient operation is supported by tight pairwise synchronization.

2.2. I extended the ring topology algorithm to multiring networks. The extended topology makes it possible to operate different subcycles with different speeds. The extended algorithm also provides guaranteed delivery time, fault tolerance and high energy efficiency.

2.3. I investigated the possibility of decreasing the delivery time in ring topology TDMA networks, when multiple parallel transmissions are allowed if they do not cause message collisions. I derived a theoretical upper bound to the achievable acceleration. I proposed a polynomial-time algorithm (OMTS) to generate the optimal schedule (which gives the shortest delivery time). I accelerated the OMTS algorithm with heuristics (OMTS-A), achieving significant gain on speed, retaining the optimum.

The results of the thesis were published in [P2], [P5], [P6], and [P7].

The proposed method can efficiently be used in distributed security systems. In such a system the decisions are made autonomously by the actuators, as opposed to the traditional centralized approach. The proposed system fits well to the natural requirements of a security system, i.e. to the guaranteed delivery time and to the fault tolerant operation. The high energy efficiency supports the battery supplied devices.

3. I proposed a method to enhance the performance of the consistency function based localization, which provides good quality estimates even when the original algorithm had big error due to the presence of consistently bad measurements. I proposed a randomized algorithm for the fast evaluation of the consistency function based localization estimator. I proved that the error of the estimation is close to the theoretical lower bound.

3.1. I proposed a method to enhance the consistency function based localization in cases when the network contains both systematically good and systematically bad sensors. The trustiness of the sensors is estimated during the localization process and the measurement of each sensor is taken into account according to its trustiness factor.

3.2. I proposed a method to accelerate the evaluation mechanism of the consistency function based localization. The method reduces the search space to a set, where the search is faster and with the appropriate parametrization of the algorithm the global optimum can be found with a probability arbitrarily close to 1.

3.3. I computed the error of the estimator with simulations based on real measurements and compared it to the theoretical (Cramér-Rao) lower bound. The

investigation showed that the error of the algorithm is close to the theoretical lower bound.

The results of the thesis were published in [P1] and [P8].

The proposed method gives a fast and accurate position estimator for Time Difference on Arrival (TDoA) based localization systems. The advantages of the proposed algorithms can be noticeable especially in highly reverberant areas and with high number of non line-of-sight sensors, therefore it can effectively be used in real applications.

Az értekezés témájában született publikációk jegyzéke

[P1] G. Vakulya, G. Simon, „Fast Adaptive Acoustic Localization for Sensor Networks", *IEEE Transactions on Instrumentation and Measurement*, Vol. 60, No. 5, May 2011, pp.1820 - 1829, May 2011.

[P2] G. Vakulya, Z. Tuza, G. Simon, „Optimal multi-TDMA schedulingin ring topology networks", *Mathematical Problems in Engineering*, Vol. 2015, Article ID 837074, 14 pages, 2015.

[P3] G. Vakulya, G. Simon, „Percolation Driven Flooding for Energy Efficient Routing in Dense Sensor Networks", *Journal of Telecommunications and Information Technology (JTIT)*, Vol. 3, No. 2009/2, pp. 5-12, 2009.

[P4] G. Vakulya, G. Simon, „Low-Power Communication Protocol for Low Duty Cycle Data Acquisition Applications", *in Proceedings of IEEE 2013 International Workshop on Measurements and Networking*, Naples, Italy, Oct. 7-8, 2013, pp. 58-62

[P5] G. Vakulya, G. Simon, „Extended Round-Robin TDMA Scheduling Scheme for Wireless Sensor Networks", *in Proceedings of 2013 IEEE International Instrumentation and Measurement Technology Conference*, Minneapolis, USA, May 6-9, 2013, pp.253-258

[P6] G. Vakulya, G. Simon, „Energy-Efficient and Reliable Round-Robin TDMA for Wireless Sensor Networks", *in Proceedings of 2012 IEEE International Instrumentation and Measurement Technology Conference*, Graz, Austria, May 13-16, 2012, pp.1179-1183

[P7] Gergely Vakulya, Gyula Simon, „Design of a Sensor Network Based Security System", *in Proceedings of IEEE International Symposium on Sintellingent Signal Processing*, Floriana, Malta, September 19-21, 2011, pp. 15-19.

[P8] Gergely Vakulya, Gyula Simon, „Adaptive Acoustic Localization Algorithm for Sensor Networks", *in Proceedings of 2010 IEEE International Instrumentation & Measurement Technology Conference*, Austin, TX, May 3-6, 2010, pp. 407-411.

[P9] G. Vakulya, G. Simon, „Energy Efficient Percolation-Driven Flood Routing for Large-Scale Sensor Networks", *in Proceedings of the International Multiconference on Computer Science and Information Technology (Workshop on Wireless and Unstructured Networking)*, Wisla, Poland, 20-22 October 2008, pp. 877-883.

[P10] G. Vakulya, G. Simon, „An Efficient Flood Routing Algorithm for Dense Randomly Deployed Sensor Networks", *in Proceedings of the Regional Conference on Embedded and Ambient Systems*, Budapest, Hungary, 22-24 November 2007, pp. 1-6

Az értekezés témájában született szoftverek jegyzéke

[S1] A 2. fejezetben használt szimulációs programok.
`http://dcs.uni-pannon.hu/~vakulya/research_software/percolation.zip`

[S2] A 3. fejezetben használt szenzor csomópontok szoftvere.
`http://dcs.uni-pannon.hu/~vakulya/research_software/tdma_mote.zip`

[S3] A 4. fejezetben használt algoritmus implementációja TinyOS környezetben.
`http://dcs.uni-pannon.hu/~vakulya/research_software/multi_tdma_generator.zip`

[S4] Az 5. fejezetben használt protokoll implementációja TinyOS környezetben.
`http://dcs.uni-pannon.hu/~vakulya/research_software/piggyback.zip`

[S5] A 6.3. alfejezetben használt feldolgozó szoftver forráskódja.
`http://dcs.uni-pannon.hu/~vakulya/research_software/adaptive_loc.zip`

[S6] A 6.4. alfejezetben használt feldolgozó szoftver forráskódja.
`http://dcs.uni-pannon.hu/~vakulya/research_software/random_loc.zip`

Irodalomjegyzék

[1] G. E. Moore, „Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, april 19, 1965, pp.114." *Solid-State Circuits Society Newsletter, IEEE*, vol. 11, no. 5, pp. 33–35, Sept 2006.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, „Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[3] J. L. Hill and D. E. Culler, „Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12–24, Nov. 2002.

[4] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, „Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet," *SIGARCH Computer Architecture News*, vol. 30, no. 5, pp. 96–107, 2002.

[5] K. Martinez, R. Ong, and J. Hart, „Glacsweb: a sensor network for hostile environments," in *Proceedings of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, Oct 2004, pp. 81–87.

[6] L. Yu, N. Wang, and X. Meng, „Real-time forest fire detection with wireless sensor networks," in *Proceedings of International Conference on Wireless Communications, Networking and Mobile Computing*, vol. 2, Sept 2005, pp. 1214–1217.

[7] J. P. Lynch, Y. Wang, K. J. Loh, J.-H. Yi, and C.-B. Yun, „Performance monitoring of the geumdang bridge using a dense network of high-resolution wireless sensors," *Smart Materials and Structures*, vol. 15, no. 6, p. 1561, 2006.

[8] J. Burrell, T. Brooke, and R. Beckwith, „Vineyard computing: sensor networks in agricultural production," *Pervasive Computing*, vol. 3, no. 1, pp. 38–45, Jan 2004.

[9] D. Malan, T. Fulford-Jones, M. Welsh, and S. Moulton, „Codeblue: An ad hoc sensor network infrastructure for emergency medical care," in *Proceedings of International workshop on wearable and implantable body sensor networks*, vol. 5, 2004, pp. 1–4.

[10] A. Pantelopoulos and N. G. Bourbakis, „A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 1, pp. 1–12, 2010.

[11] S. Ullah, H. Higgin, M. A. Siddiqui, and K. S. Kwak, „A study of implanted and wearable body sensor networks," in *Agent and Multi-Agent Systems: Technologies and Applications*, 2008, pp. 464–473.

[12] A. Lédeczi, A. Nádas, P. Völgyesi, G. Balogh, B. Kusy, J. Sallai, G. Pap, S. Dóra, K. Molnár, M. Maróti, and G. Simon, „Countersniper system for urban warfare," *ACM Transactions on Sensor Networks*, vol. 1, no. 2, pp. 153–177, Nov. 2005.

[13] B. Warneke, M. Last, B. Liebowitz, and K. Pister, „Smart dust: communicating with a cubic-millimeter computer," *Computer*, vol. 34, no. 1, pp. 44–51, Jan 2001.

[14] A. Corporation, „ATmega128RFA1 datasheet," 2014.

[15] J. M. Gilbert and F. Balouchi, „Comparison of energy harvesting systems for wireless sensor networks," *International journal of automation and computing*, vol. 5, no. 4, pp. 334–347, 2008.

[16] J. Kymissis, C. Kendall, J. Paradiso, and N. Gershenfeld, „Parasitic power harvesting in shoes," in *Second International Symposium on Wearable Computers, 1998. Digest of Papers.* IEEE, 1998, pp. 132–139.

[17] T. Starner, „Human-powered wearable computing," *IBM systems Journal*, vol. 35, no. 3.4, pp. 618–629, 1996.

[18] J. J. Labrosse, *Microc/OS-II.* R & D Books, 1998.

[19] A. J. Massa, *Embedded software development with eCos.* Prentice Hall Professional, 2003.

[20] R. Goyette, „An analysis and description of the inner workings of the FreeRTOS kernel," *Carleton University*, vol. 5, 2007.

[21] P. Levis and D. Gay, *TinyOS Programming.* Cambridge University Press, 2009.

[22] A. Dunkels, B. Gronvall, and T. Voigt, „Contiki – a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks.* IEEE, 2004, pp. 455–462.

[23] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, „MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms," *Mobile Networks and Applications*, vol. 10, no. 4, pp. 563–579, 2005.

[24] Q. Cao, T. Abdelzaher, J. Stankovic, and T. He, „The LiteOS operating system: Towards unix-like abstractions for wireless sensor networks," in *International Conference on Information Processing in Sensor Networks, IPSN'08.* IEEE, 2008, pp. 233–244.

[25] W. Ye, J. Heidemann, and D. Estrin, „An energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 3, 2002, pp. 1567–1576.

[26] J. Polastre, J. Hill, and D. Culler, „Versatile low power media access for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 95–107.

[27] M. Sha, G. Hackmann, and C. Lu, „Energy-efficient low power listening for wireless sensor networks in noisy environments," in *Proceedings of the 12th international conference on Information processing in sensor networks*. ACM, 2013, pp. 277–288.

[28] L. Lin, K.-J. Wong, A. Kumar, Z. Lu, S.-L. Tan, and S. J. Phee, „Evaluation of a tdma-based energy efficient mac protocol for multiple capsule networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, p. 54, 2011.

[29] Y. Yu, R. Govindan, and D. Estrin, „Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks," Technical report ucla/csd-tr-01-0023, UCLA Computer Science Department, Tech. Rep., 2001.

[30] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, „Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Jan 2000, p. 10 pp. vol.2.

[31] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, „TAG: A tiny aggregation service for ad-hoc sensor networks," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 131–146, 2002.

[32] J. Elson, L. Girod, and D. Estrin, „Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, 2002.

[33] S. Ganeriwal, R. Kumar, and M. B. Srivastava, „Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*. ACM, 2003, pp. 138–149.

[34] M. Maróti, B. Kusy, G. Simon, and A. Lédeczi, „The flooding time synchronization protocol," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004, pp. 39–49.

[35] P. Sommer and R. Wattenhofer, „Gradient clock synchronization in wireless sensor networks," in *Proceedings of the 2009 International Conference on Information Processing in Sensor Networks*. IEEE Computer Society, 2009, pp. 37–48.

[36] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh, „Efficient network flooding and time synchronization with Glossy," in *10th International Conference on Information Processing in Sensor Networks (IPSN'11)*. IEEE, 2011, pp. 73–84.

[37] C. K. Singh, A. Kumar, and P. Ameer, „Performance evaluation of an IEEE 802.15. 4 sensor network with a star topology," *Wireless Networks*, vol. 14, no. 4, pp. 543–568, 2008.

[38] Y. Mazzer and B. Tourancheau, „Comparisons of 6LoWPAN implementations on wireless sensor networks," in *Proceedings of Third International Conference on Sensor Technologies and Applications (SENSORCOMM)*, June 2009, pp. 689–692.

[39] S. Safaric and K. Malaric, „ZigBee wireless standard," in *Proceedings of 48th International Symposium on Multimedia Signal Processing and Communications (ELMAR)*, 2006, pp. 259–262.

[40] S. Waharte, R. Boutaba, Y. Iraqi, and B. Ishibashi, „Routing protocols in wireless mesh networks: challenges and design considerations," *Multimedia Tools and Applications*, vol. 29, no. 3, pp. 285–303, 2006.

[41] A. Manjeshwar and D. P. Agrawal, „TEEN: A routing protocol for enhanced efficiency in wireless sensor networks," in *Proceedings of the 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing*, vol. 3, 2001, pp. 30 189a–30 189a.

[42] C. E. Perkins and E. M. Royer, „Ad-hoc on-demand distance vector routing," in *Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999, pp. 90–100.

[43] M. Maróti, „Directed flood-routing framework for wireless sensor networks," in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, 2004, pp. 99–114.

[44] B. S. Chlebus and D. R. Kowalski, „A better wake-up in radio networks," in *Proceedings of the Twenty-third Annual ACM Symposium on Principles of Distributed Computing (PODC)*, 2004, pp. 266–274.

[45] G. Vakulya and G. Simon, „Low-power communication protocol for low duty cycle data acquisition applications," in *Proceedings of International Workshop on Measurements and Networking Proceedings (M&N)*, 2013, pp. 58–62.

[46] S. S. Kulkarni and L. Wang, „MNP: Multihop network reprogramming service for sensor networks," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, 2005, pp. 7–16.

[47] G. Lu, B. Krishnamachari, and C. Raghavendra, „An adaptive energy-efficient and low-latency mac for data gathering in wireless sensor networks," in *Proceedings of 18th International Parallel and Distributed Processing Symposium*, April 2004, pp. 224–.

[48] T. Van Dam and K. Langendoen, „An adaptive energy-efficient MAC protocol for wireless sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 171–180.

[49] D. Moss, J. Hui, and K. Klues, „Low power listening," *TinyOS Core Working Group, TEP*, vol. 105, 2007.

[50] Y. Sun, O. Gurewitz, and D. B. Johnson, „RI-MAC: a receiver-initiated asynchronous duty cycle MAC protocol for dynamic traffic loads in wireless sensor networks," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*. ACM, 2008, pp. 1–14.

[51] P. Dutta, S. Dawson-Haggerty, Y. Chen, C.-J. M. Liang, and A. Terzis, „Design and evaluation of a versatile and efficient receiver-initiated link layer for low-power wireless," in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, 2010, pp. 1–14.

[52] Z. Haas, J. Y. Halpern, and L. Li, „Gossip-based ad hoc routing," *IEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 479–491, June 2006.

[53] L. H. M. K. Costa, M. D. De Amorim, and S. Fdida, „Reducing latency and overhead of route repair with controlled flooding," *Wireless Networks*, vol. 10, no. 4, pp. 347–358, Jul. 2004.

[54] P. Gburzynski and W. Olesinski, „On a practical approach to low-cost ad hoc wireless networking," *Journal of Telecommunications and Information Technology*, no. 1, pp. 29–42, 2008.

[55] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, „Collection tree protocol," in *Proceedings of the 7th Conference on Embedded Networked Sensor Systems*, 2009, pp. 1–14.

[56] N. Burri, P. Von Rickenbach, and R. Wattenhofer, „Dozer: ultra-low power data gathering in sensor networks," in *Proceedings of 6th International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 450–459.

[57] V. Cionca, T. Newe, and V. Dadarlat, „TDMA protocol requirements for wireless sensor networks," in *Proceedings of Second International Conference on Sensor Technologies and Applications, 2008. SENSORCOMM '08.*, Aug 2008, pp. 30–35.

[58] P. K. Pal and P. Chatterjee, „A survey on TDMA-based MAC protocols for wireless sensor network," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 6, pp. 219–230, 2014.

[59] S. B. Wibowo, M. Klepal, and D. Pesch, „Time of flight ranging using off-the-self IEEE 802.11 wifi tags," in *Proceedings of the International Conference on Positioning and Context-Awareness (PoCA'09)*, 2009.

[60] H.-P. Tan, R. Diamant, W. K. Seah, and M. Waldmeyer, „A survey of techniques and challenges in underwater localization," *Ocean Engineering*, vol. 38, no. 14–15, pp. 1663 – 1676, 2011.

[61] C. Park, H. Cho, D. Park, S. Cho, J. Park, and Y. Lee, „AoA localization system design and implementation based on zigbee for applying greenhouse," in *2010 5th IEEE International Conference on Embedded and Multimedia Computing (EMC)*, 2010.

[62] G. Mao, B. Fidan, and B. D. Anderson, „Wireless sensor network localization techniques," *Computer networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[63] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih, „Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network," *IEEE Transactions on Computers*, vol. 52, no. 5, pp. 545–557, May 2003.

[64] J.Yang, B. Kim, M.-T. Sun, and T.-H. Lai, „Location-aided broadcast in wireless ad hoc networks," *Journal of Information Science and Engineering*, vol. 23, pp. 869–884, 2007.

[65] I. Glauche, W. Krause, R. Sollacher, and M. Greiner, „Continuum percolation of wireless ad hoc communication networks," *Physica A: Statistical Mechanics and its Applications*, vol. 325, pp. 577–600, 2003.

[66] G. Grimmett, *Percolation*. Springer-Verlag, 1989.

[67] V. D. Park and M. S. Corson, „A highly adaptive distributed routing algorithm for mobile wireless networks," in *Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 1997, pp. 1405–1413.

[68] M. Franceschetti, L. Booth, M. Cook, R. Meester, and J. Bruck, „Percolation of multi-hop wireless networks," EECS Department, University of California, Berkeley, Tech. Rep., 2003.

[69] J. Jonasson, „Optimization of shape in continuum percolation," *Annuals of probability*, vol. 29, pp. 624–635, 2001.

[70] O. Dousse, P. Mannersalo, and P. Thiran, „Latency of wireless sensor networks with uncoordinated power saving mechanisms," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2004, pp. 109–120.

[71] S. C. Ergen and P. Varaiya, „TDMA scheduling algorithms for wireless sensor networks," *Wireless Networks*, vol. 16, pp. 985–997, 2010.

[72] K. Kredo, II and P. Mohapatra, „Medium access control in wireless sensor networks," *Computer Networks*, vol. 51, no. 4, pp. 961–994, 2007.

[73] M. Gast, *802.11 wireless networks: the definitive guide*. O'Reilly Media, Inc., 2005.

[74] M. Mouly, M.-B. Pautet, and T. Foreword By-Haug, *The GSM system for mobile communications*. Telecom Publishing, 1992.

[75] L. Shi and A. O. Fapojuwo, „TDMA scheduling with optimized energy efficiency and minimum delay in clustered wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 7, pp. 927–940, 2010.

[76] M. Nesa Sudha and A. S. Babu, „TDMA based energy conservation in wireless sensor networks," *IJCA Proceedings on International Conference on VLSI, Communications and Instrumentation (ICVCI)*, no. 9, pp. 1–6, 2011.

[77] K. Arisha, M. Youssef, and M. Younis, „Energy-aware tdma-based mac for sensor networks," in *System-level power optimization for wireless multimedia communication*. Springer, 2002, pp. 21–40.

[78] J. Degesys, I. Rose, A. Patel, and R. Nagpal, „DESYNC: Self-organizing desynchronization and tdma on wireless sensor networks," in *6th International Symposium on Information Processing in Sensor Networks (IPSN), 2007.*, April 2007, pp. 11–20.

[79] G. Zachár and G. Simon, „Providing energy efficient mobility in TDMA-controlled wireless sensor networks," in *Proceedings of IEEE International Instrumentation and Measurement Technology Conference*, 2012, pp. 1174–1178.

[80] G. Simon, „Efficient time-synchronization in ring-topology wireless sensor networks," in *Proceedings of International Instrumentation and Measurement Technology Conference*, 2012, pp. 958–962.

[81] S. Choi, S.-J. Song, K. Sohn, H. Kim, J. Kim, J. Yoo, and H.-J. Yoo, „A low-power star-topology body area network controller for periodic data monitoring around and inside the human body," in *Proceedings of 10th International Symposium on Wearable Computers*, 2006, pp. 139–140.

[82] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic, „An advanced wireless sensor network for health monitoring," in *Proceedings of Transdisciplinary Conference on Distributed Diagnosis and Home Healthcare (D2H2)*, 2006, pp. 2–4.

[83] E.-Y. Lin, J. M. Rabaey, and A. Wolisz, „Power-efficient rendez-vous schemes for dense wireless sensor networks," in *Proceedings of 2004 IEEE International Conference on Communications*, vol. 7, 2004, pp. 3769–3776.

[84] D. Macii, A. Ageev, and L. Abeni, „An energy saving criterion for wireless sensor networks with time synchronization requirements," in *Proceedings of International Symposium on Industrial Embedded Systems (SIES)*, 2010, pp. 166–173.

[85] A. Tanenbaum, *Computer Networks*, 4th ed. Prentice Hall Professional Technical Reference, 2002.

[86] W.-B. Pottner, S. Schildt, D. Meyer, and L. Wolf, „Piggy-backing link quality measurements to IEEE 802.15.4 acknowledgements," in *Proceedings of 8th International Conference on Mobile Adhoc and Sensor Systems (MASS)*, 2011, pp. 807–812.

[87] J. C. Chen, R. E. Hudson, and K. Yao, „Maximum-likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1843–1854, 2002.

[88] X. Sheng and Y. hen Hu Fellow, „Maximum likelihood wireless sensor network source localization using acoustic signal energy measurements," *IEEE Transactions on Signal Processing*, vol. 53, 2003.

[89] T. Ajdler, I. Kozintsev, R. Lienhart, and M. Vetterli, „Acoustic Source Localization in Distributed Sensor Networks," in *Proceesings of Asilomar Conference on Signals, Systems and Computers*, vol. 2, 2004, pp. 1328–1332.

[90] A. Mahajan and M. Walworth, „3-D position sensing using the differences in the time-of-flights from a wave source to various receivers," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 91–94, 2002.

[91] J. Sallai, G. Balogh, M. Maroti, A. Ledeczi, and B. Kusy, „Acoustic ranging in resource-constrained sensor networks," in *Proceesings of ICWN '04*, June 2004, p. 467.

[92] J. N. Ash and R. L. Moses, „Acoustic time delay estimation and sensor network self-localization: Experimental results," *Journal of the Acoustical Society of America*, vol. 118, no. 2, pp. 841–850, 2005.

[93] M. A. Fischler and R. C. Bolles, „Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[94] R. B. Langley, „Dilution of precision," *GPS World*, vol. 10, no. 5, pp. 52–59, May 1999.

[95] Y. Chan and K. Ho, „A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, Aug. 1994.

[96] J. T. Isaacs, D. J. Klein, and J. P. Hespanha, „Optimal sensor placement for time difference of arrival localization," in *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009, pp. 7878–7884.