EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

Önszervező, elosztott algoritmusok Smart Dust eszközökhöz

Készítette:

Szabó Attila

Nappali tagozatos programtervező matematikus

Témavezető:

Dr. habil. Lőrincz András

Tudományos főmunkatárs, ELTE IK Információs Rendszerek Tanszék

Budapest, 2007.

EÖTVÖS LORÁND TUDOMÁNYEGYETEM

INFORMATIKAI KAR

Diplomamunka-téma bejelentő

Név: **Szabó Attila**
Tagozat: **nappali** Szak: **programtervező matematikus**
Témavezető neve: **Dr. Lőrincz András**
munkahelyének neve és címe: **ELTE IK Információs Rendszerek Tanszék**
beosztása és iskolai végzettsége: **Tudományos főmunkatárs**

A dolgozat címe: **Önszervező, elosztott algoritmusok Smart Dust eszközökhöz**

A dolgozat témája:

Napjainkban az internet bővülésével lehetőség van több ezer olcsó, személyi számítógépet összekapcsoló gridek használatára, míg a kis méretű, akár testen hordható céleszközök fejlesztése szintén sokat haladt az elmúlt években. A számítások elosztásánál azonban figyelembe kell venni, hogy a névlegesen rendelkezésre álló erőforrások közül több, előre nem meghatározható gép nem elérhető, ami késleltetést jelent az alkalmazások futtatásánál. Alapvető feladatok merülnek fel a különböző kapacitással rendelkező, hálózatba kötött eszközök minél jobb kihasználásával, a feladatok automatikus kiosztásával kapcsolatban. Ehhez nyújthat segítséget egy megbízhatósági skálán alapuló protokoll, ami önkéntes vállalásra alapozva osztja rekurzívan a feladatot a hálózati entitások között (PC, Smart Dust), szelekciós/evolúciós elvek segítségével „bizalmi" alapon működik, és a bizalmi hálózatokban biztonságos és gyors, csővezeték-szerű rendszerviselkedést tesz lehetővé.

Elosztott és gyorsan változó rendszerekben folyamatos feladat az ügynökök által csak részben észlelt hálózatok topológiájának feltérképezése, a topológia optimalizálása központi ügynök nélkül. A topológia ismeretében számítási feladatok dekompozíciójára és a részfeladatok felosztására a fenti elvek alapján felépülő SPP (Survivable Pipeline Protocol) és predikciós algoritmusokat fogom használni. Helymeghatározásban és elosztott feladatok megszervezésében, illetve azok futás közbeni újraszervezésében fogom vizsgálni a mesterséges intelligencia algoritmusait.

A témavezetést vállalom:

.....
(a témavezető aláírása)

Kérem a diplomamunka témájának jóváhagyását.
Budapest, 2006. …….

.....
(a hallgató aláírása)

A diplomamunka-témát az Informatikai Kar jóváhagyta.

Tartalomjegyzék

Köszönetnyilvánítás

Köszönettel tartozom témavezetőmnek, Lőrincz Andrásnak a kapott útmutatásért és ösztönzésért. Ezúton is köszönet jár Palotai Zsoltnak végetlen türelméért, illetve a MEMS-programozással és SPP-vel kapcsolatban nyújtott folyamatos segítségéért.

Köszönöm az ELTE NIPG azon tagjainak hozzájárulását, akik – akár közvetve – részt vettek a munkában: közülük kiemelném Szirtes Gábort, aki építő kritikáival, és segítőkészségével jelentős mértékben támogatta a fejlesztést.

Használt rövidítések

NIPG: Neural Information Processing Group
SKMK: Segítő Módszertani Kommunikáció Központ

SPP: Survivable Pipeline Protocol
PO: Pipeline Operation

RF-MEMS: Radio Frequency-Microelectromechanical System

1. Bevezető

Az Internet, illetve általában az informatika fejlődésével egyre intelligensebb, komplex problémákat megoldó algoritmusok tervezésére nyílik lehetőség, hiszen egyrészt adott a mai eszközökkel feldolgozhatatlan mennyiségű elektronikus úton hozzáférhető információ, másrészt a rendelkezésre álló számítógép-kapacitás még messze kihasználatlan. Mind a több ezer személyi számítógépet összekapcsoló

grid-ek, mind az akár testen hordható céleszközök kapcsán előtérbe kerülnek a számítások szervezésével összefüggő problémák. Ezekben, és az ezekhez hasonló hálózatokban nagy jelentősséggel bír annak vizsgálata, hogy a különböző erőforrások folyamatosan változó rendelkezésre állása mellett hogyan garantálható állandó működés és egyenletes számítási kapacitás. A tömeggyártott mikroprocesszorok elterjedésével a mindennapi elektronikai eszközök jelentős része programozottá vált. A jövőben ezen programozott eszközök szabványosításával és hálózatokba szervezésével új távlatok nyílnak például a gyógyászat, a közlekedés, vagy a biztonságtechnika terén.

A dolgozat első hat fejezetében az ELTE NIPG kutatócsoportja által fejlesztett Survivable Pipeline Protocol-t, és annak működését mutatom be. Mivel a megoldás során ötvözni kellett a hálózatokkal kapcsolatos ismereteket a szenzoros beágyazott rendszerek speciális lehetőségeivel, itt összefoglalom a hálózatok topológiájával, a beágyazott rendszerekkel, és a Crossbow Cricket Mote szolgáltatásaival, és programozásával kapcsolatos lényegesebb ismereteket. Ez a rész tartalmazza az SPP korábban elkészített implementációjának és a vizsgált példa alkalmazás programjának áttekintő bemutatását.

A dolgozat második részében az SPP működésének vizsgálatát, illetve funkcionalitásának bővítését dokumentáltam. Javítottam a meglévő algoritmus robusztusságán, így kiegyensúlyozottabbá vált a rendszer viselkedése. Sikerült megoldást adni több taszk egymás mellett működésére egy erőforrás-megosztási eljárással. Az elvégzett kísérletek eredményei a protokoll működőképességét és használhatóságát támasztják alá.

Az SPP újszerűsége abban rejlik, hogy megteremti a számítógépes hálózat feladatszervezés szemszögéből kényelmes absztrakcióját. A számításszervezés automatizálása megengedhető mértékű overhead mellett biztosítja a felügyelet nélküli futtatás lehetőségét. A protokoll definiálja az adott feladat megoldásához szükséges adatcsatorna létrehozásának rekurzív algoritmusát, az ehhez szükséges szerepeket, és kommunikációs csatornákat. Megadja a javításhoz és fenntartáshoz szükséges algoritmusokat, a feladat megfogalmazásának módját és a feladat megadásának különböző részeit. Figyelembe veszi a számítógépes hálózatok jellemzőit, emellett előzetesen becsli a nyilvántartott rendelkezésre álló erőforrások megbízhatóságát, és a feladat gazdája számára fölső korlátot ad az adatcsatorna áteresztési-, illetve újraindítási idejére.

A példa alkalmazás fejlesztése a magyar Bliss alapítvánnyal együttműködésben folyik, célja kerekes székek helyzetének meghatározására alkalmas rendszer készítése. A rendszert alkotó MEMS szenzoros hálózat elemei rádión adják át a koordináták számításához szükséges adatokat, illetve a saját, számolt helyzetüket. Emiatt a számítási feladatok megoldása mellett a hálózatnak szüksége van egy időosztásos rádió-hozzáférésre, mely által definiált sorrend minden, a hálózathoz tartozó eszközön rendelkezésre áll a csatlakozástól a

leválásig. Cél, hogy a pozíció meghatározása központi egység nélkül, folyamatosan változó topológiájú hálózatokon megbízhatóan működjön. A rendszer segítségével meghatározható a betegek helyzete, elkerülhetőek balesetek, illetve lehetőséget biztosít arra, hogy a betegek segítséget kérjenek. Ebben a környezetben merültek föl a különböző, egymástól függetlenül kialakult rendszerek együttműködésével, illetve egymás mellett működésével kapcsolatos lehetőségek és lehetséges problémák.

Az elkövetkezendőkben az SPP implementálható lenne személyi számítógépek hálózatán végzett számítások szervezésére alkalmas formában. Ez utat mutathat az ipari alkalmazások felé, szabványt adva nemcsak elosztott számítási-, hanem akár gyártási folyamatok vezérlésére.

2. Skálamentes kisvilágok

A természetes rendszerekben fellelhető szabályszerűségek a természettudományok inspirálói. Míg néhány ilyen rendszerben magas szintű szabályszerűség figyelhető meg (például a kristályrácsokban), addig a legtöbb komplex rendszer távolról sem mutat ilyen mértékű szervezettséget, inkább véletlenszerű és megjósolhatatlan felépítéssel, illetve viselkedéssel bír.

A számítógépes hálózatok kutatása a biológiai-, társadalmi-, ember alkotta rendszerek kutatásának eredményeire építkezik, és a kezdetektől azzal együtt fejlődik. A nagy rendszerek viselkedését szimuláló jelentősebb modellek - többek

között - az Erdős-Rényi véletlen modell, a Watts-Stogratz magas klaszterezettségű modell, és a Barabási-Albert preferenciális kapcsolódásos modell. Ezeken a modelleken keresztül sikerült nagy kiterjedésű hálózatok fontosabb tulajdonságait leírni, úgymint skálamentesség, klaszterezettség, és kisvilág-tulajdonság.

Ma már – a fentebb említett kutatók nyomán- tudjuk, hogy az internet, mint több más, ember alkotta hálózat kisvilág tulajdonságú, azaz ezekben a hálózatokban tetszőleges két csúcs közti legrövidebb út hossza kicsi (a legrövidebb utak hossza aszimptotikusan tart a csúcsok számának logaritmusához). Ezeknek a gráfoknak hierarchikus a szerkezete, és (a véletlen gráfokhoz képest) magas a klaszterezettsége: azaz ha két csúcsnak van legalább egy közös szomszédja, akkor nagy valószínűséggel a csúcsok közt is van él.

Másik jellemző tulajdonsága ezen a hálózatoknak a skálamentesség: olyan gráfot alkotnak, melyben a csúcsoknak nincs jellemző fokszáma (a fokszám-eloszlás hatványfüggvény típusú). Ezekben a rendszerekben a kevés nagy fokszámú csúcs az erősen klaszterezett komponenseket köti egymással össze.

Az SPP protokollt a fenti tulajdonságú hálózatokra tervezték, hiszen nem csak megalapozott az internethez hasonló hálózati szerkezetet feltételezni, hanem akár a feladat konkrét alkalmazása lehet az internetre csatlakozó számítógépek erőforrásainak elosztása.

3. Survivable Pipeline Protocol

Az SPP célkitűzései

Az SPP célja egy, a hardverrel megvalósított adatcsatorna-rendszerekhez hasonló, valamely erőforrás-hálózat fölé definiált ütemezett adatcsatorna készítése. Az SPP integrálja és elfedi az elérhető erőforrásokat és egységes felületet ad a számítási feladat végrehajtásához. Lehetővé teszi, hogy egy dinamikusan változó topológiájú hálózati réteget – az SPP szintjén – állandó, illetve lassan változó tulajdonságú rendszerként kezeljünk.

Az SPP ötlete a hardveres adatcsatornák koncepciójának hálózatra történő kiterjesztése. A számítási feladatokat (más néven taszkokat) úgynevezett PO-kként (Pipeline Operation) kell megfogalmazni, melyek szerkezete lehetővé teszi az automatikus feladatszervezést.

A protokoll az álladóságot az adatcsatorna minőségének fenntartásával éri el. A megbízható és folyamatos működés alapja a résztvevő erőforrások cseréjének lehetősége, és a megbízhatósági skálán alapuló feladat-kiosztási algoritmus. Ha a számítás végzése közben például az egyik processzor kiesik, akkor a protokoll megpróbálja pótolni úgy, hogy felkéri a többi nyilvántartott processzor valamelyikét a számításban való részvételre. Hasonló az eljárás akkor is, ha az adatcsatornából több, akár egy egész részfeladatot ellátó processzorcsoport esik ki, vagy a teljes adatcsatorna válik működésképtelenné. A folyamatos működés garantálása érdekében a protokoll nyilvántartja azokat az erőforrásokat, melyek már részt vettek valamely feladat megoldásában, és minden erőforráshoz megbízhatósági értéket rendel (például a válaszidő alapján). A feladatok kiosztásánál az eddig megbízhatóbban teljesítő jelentkezők vannak előnyben, így a várhatóan rosszabbul teljesítők ritkábban vesznek részt az adatcsatorna működtetésében.

Az adatcsatorna és elemeinek működése központi felügyelet nélkül zajlik, emiatt az ellenőrzésért a részfeladatok megszervezésére kijelölt erőforrások felelősek. Ezek megszervezik a közvetlen utánuk következő részfeladatok elvégzését az adatcsatorna létrehozásakor, és szükség esetén újraszervezik, illetve futás közben felügyelik. A szervezés egy, a részfeladatért felelős erőforrás kereséséből, és ezen erőforrás felügyeletéből áll, azaz a részfeladatok delegálása és felügyelete rekurzívan történik.

Az adatcsatorna minőségi mutatói az áteresztési- és az újraindítási ideje, melyeket a protokoll folyamatosan becsül és előre jelez a számítási feladat elvégzése közben. A feladat specifikációja tartalmazza az ezekkel kapcsolatos maximális időkorlátokat. Egy számítási (rész)feladat felelőse köteles szavatolni a saját- illetve az utána következő részfeladat megadott időn belüli elvégzését. Ha például egy processzor elvállalja egy művelet számítását, akkor becslést is ad a művelet kiszámítási idejére, melyet tartania kell. Ezek összességéből lehet becsülni a teljes rendszer tulajdonságát.

A protokollnak skálamentes kisvilág tulajdonságú hálózatokon kell jól teljesítenie, tervezésekor a hálózati topológiára más megkötés nem történt.

Az SPP felépítése és működése [1]

Egy SPP feladat megszervezéséhez a következő adatok szükségesek: egy adatfolyam-gráf, melynek csúcsai PO-k; a megengedett legnagyobb újraindítási- és áteresztési idők; végül megbízhatósági paraméterek (például a különböző erőforrások minimálisan szükséges mennyisége). A konkrét hálózat erőforrásainak megbízhatóságát a weblog értékeik adják meg. A taszk érzékelési-, számítási-, és kommunikációs feladatokból állhat.

Az SPP által definiált általános szerepkörök a következők: irányító (manager), disztribútor (distributor), és processzor (processor). A processzor egység felelős a vállalt résztaszk megadott időn belüli elvégzéséért adat érkezése esetén, és megfelelő mennyiségű disztribútor fenntartásáért. A disztribútorok figyelik az érkező adatokat és továbbítják a megfelelő processzor felé: több részből álló adat egyes részei nem feltétlenül érkeznek azonos disztribútorhoz, de az összes rész szigorúan egy processzorhoz kell kerüljön. Az disztribútorok felelősek egy, a résztaszkot kiszolgáló manager fenntartásáért, mely szolgáltatja az adattovábbításhoz szükséges információkat. A manager feladata, hogy a több erőforrással rendelkező processzorok több adatot kapjanak, és keres a következő résztaszk számára manager-t. A szerepek nem zárják ki egymást: manager lehet egyben disztribútor, és disztribútor lehet egyben processzor is.

A különböző szerepek közti adatforgalomhoz az SPP hat kommunikációs csatornát definiál:

SPP Pipe: Egyirányú, multicast csatorna, amin a taszkok szervezésének hirdetése, és a processzor-igények közlése történik.

Data Pipe: Egyirányú, unicast csatorna az adatok továbbításához.

Distributor Pipe: Egyirányú, multicast csatorna, amin a manager-ek küldik az adatok továbbításához szükséges információkat a disztribútoroknak.

Task Pipe: Egyirányú, multicast csatorna, amin a manager tud üzenni az összes processzornak (például a taszk megszűnéséről), illetve a disztribútorok jelzik meglétüket a processzorok felé.

Manager Pipe: Kétirányú, unicast csatorna a manager-ek közti kommunikáció számára: ezen történik a résztaszk delegálása, vállalása, a következő taszk ellenőrzése, és a megelőző taszk értesítése.

Task Manager Pipe: Kétirányú, unicast csatorna, amin a manager-rel kommunikálhatnak a processzorok, illetve a rákövetkező taszk megszervezésére vállalkozók.

A feladat megszervezésének és a működés fenntartásának általános lépései a következők:

Kezdő feltétel:

a) adott egy SPP csoport (melynek tagjai részt vesznek majd a rendszer kiépítésében)
b) létezik egy SPP Pipe ezen csoporton belül.

Feladat megszervezése:

a) az SPP Pipe-on valaki jelzi, hogy szervezésre váró taszk van
b) legalább egyvalaki megkapja az üzenetet a csoportból
c) legalább egy tag elvállalja az egyik részfeladatot és a szervezés folytatását
d) ez a tag engedélyt kér a taszk gazdájától
e) engedélyezés esetén létrehozza a résztaszk csoportját és ő lesz a csoport manager-e, első disztribútora, illetve első processzora

Ellenőrző tevékenységek:

a) manager meglétének lekérdezése (résztaszkon belül)
b) az disztribútorok számának figyelése
c) a processzor (saját) működésének ellenőrzése
d) annak ellenőrzése, hogy a következő résztaszk manager-e működik-e

A részfeladat elvégzésének biztosításához kapcsolódó műveletek:

a) annak ellenőrzése, hogy az elvállalt résztaszk megbízhatóan működik
b) amennyiben a működés nem kielégítő, (például a meghatározotthoz képest túl kevés erőforrás van) a manager a nyilvántartott weblog értékek alapján új partnereket választ, akiknek az SPP Pipe-on processzor-igénylést küld
c) vár valameddig
d) ha nincs elegendő válasz, akkor mindenkinek elküldi a processzor-igénylést az SPP Pipe-on
e) vár valameddig

f) ha a válaszok még mindig nem felelnek meg a támasztott elvárásoknak, akkor értesíti a megelőző résztaszk manager-ét, hogy a vállalt résztaszk meghiúsult

A rákövetkező részfeladat elvégzésének biztosítása:

a) annak ellenőrzése, hogy a következő résztaszk megbízhatóan szervezett
b) amennyiben nem, (például a következő manager nem válaszol időben) a manager a weblog értékek alapján új partnereket választ, akiknek az SPP Pipe-on jelzi, hogy szervezésre váró taszk van
c) vár valameddig
d) ha a válaszok nem elégítik ki az igényeket, mindenkinek megismétli az előző üzenetet az SPP Pipe-on
e) vár valameddig
f) ha a válaszok még mindig nem felelnek meg a támasztott elvárásoknak, akkor értesíti a megelőző résztaszk manager-ét, hogy a vállalt résztaszk meghiúsult

Feladat befejezése:

a) ha az adatcsatorna megszakad, vagy a taszk befejeződik, akkor a Manager- és Task Pipe-okon keresztül ezt a manager jelzi, ezzel felszabadítva az erőforrásokat.

4. RF-MEMS eszközök

Beágyazott rendszerek

A nem általános célú számítógépek alkalmazása főleg az ipari termelésben terjedt el, ezen belül is olyan területeken, ahol kisméretű, speciális perifériával ellátott architektúrákra van szükség. A MEMS-ek (Microelectromechanical System) általános jellemzői a korlátozott erőforrások, szenzorok, és mikro-vezérlők. Ezek az általában kisméretű eszközök viszonylag alacsony szintű programnyelveken programozhatók. Az elsődleges tervezési szempontok a célnak megfelelő – általában mennél kisebb – méret, illetve a feladatnak megfelelő, magas szintű rendelkezésre állás és megbízhatóság

Az első tömeggyártásra szánt, beágyazott rendszer az 1960-as évek elején készült el. A Boeing a „Minuteman" rakétairányító berendezés részeként rendszeresítette, és 1962-től állt hadrendben az USA nukleáris fegyverarzenáljában. Ezután még hat évnek kellett eltelnie, mire megjelent az első békés célra gyártott rendszer: a Volkswagen cég 1968-ban, elsőként szerelt programozott üzemanyag injektort autóiba. Manapság MEMS eszközök vezérlik a

magas szinten automatizált járművek egyes részeit, így például repülőgépek, vagy akár gépjárművek működtetésében vesznek részt. Használnak MEMS-eket vér-, illetve légnyomás mérésére (ellenőrzésére), földrengések előrejelzésére, különböző helymeghatározó és biztonsági rendszerekben.

Bár a technológiát széles körben alkalmazzák és fejlesztik, a programok méretének optimalizálása, a sok különböző célhardver használata és e rendszerek egymástól elszigetelt működése eddig a szabványosítás ellen hatott. A változást ezen eszközök hálózatba szervezése hozhatja, mivel az együttműködés természetszerűleg igényli a szabványos interfészeket. Sok, hálózatba szánt MEMS eszköz kommunikációját rádióval oldják meg, ami lehetővé teszi a szabad mozgást, illetve nyílt, Smart Dust hálózatok építését. Az ilyen hálózatok – mint például a dolgozatban tárgyalt pozíciómonitorozó rendszer – egyrészt rugalmas és megbízható, másrészt olcsó megoldást jelentenek alkalmazási körükben.

Magyarországon a BME Méréstechnika és Információs Rendszerek Tanszékén foglalkoznak RF-MEMS-ek építésével és fejlesztésével. Az „mitmót” névre hallgató MEMS-ek terepre kihelyezett hálózatát hőmérséklet, páratartalom és széndioxid-koncentráció mérésére használják.

Crossbow/Berkeley Mote

Az alkalmazáshoz használt Mica architektúrájú mote-okat a Crossbow vállalat a Berkeley egyetemmel közösen fejleszti. Ezek a programozható lapok rádión keresztül kommunikálnak, és különböző (fény-, gyorsulás-, ultrahang-) érzékelőkkel lehetnek felszerelve. A mote-okból kis energiafelhasználású, megbízható vezeték nélküli szenzor hálózatok építhetők, melyekhez a gyártó a következő felhasználói felületeket adja:

- Adatmegjelenítés és analízis
- Hálózati topológia megjelenítése
- Hibajelzés, riasztás
- Konfigurációs- és vezérlőfelület
- Hálózat-analízis

Néhány referencia

A Crossbow által fejlesztett rendszereket többek között az alábbi területeken használják:

A BP cég olajszállító hajóin a forgó alkatrészek (motorok, szivattyúk) állapotát ellenőrzi mote-okból álló vezeték nélküli szenzoros hálózat, ami segítségével lehetővé vált, hogy a berendezések rendszeres, teljes körű szervizelése helyett a megfelelő alkatrészeket a szükséges időben cserélhessék le.

Földbe ásott szenzorok hálózatát használják a mezőgazdaságban, öntözés vezérlésére. A programozott vezérlés az érzékelők segítségével takarékosabb vízhasználatot eredményez.

Érdekességként: hangérzékelőkkel felszerelt Mica mote-ok hálózatát az USA hadseregében kísérleti jelleggel használják orvlövészek helyzetének meghatározására.

Technikai jellemzők

A Crossbow Mica mote-ok típusait és főbb jellemzőit az 1. táblázat tartalmazza.

Mote típusa	**MICA**	**MICA 2**
Chip	ATmega103L	ATmega128L
Típus	4 MHz, 8 bit	8Mhz, 8 bit
Program Memória	128KB	128KB
RAM	4KB	4KB
Külső tároló	512KB	512KB
Áramellátás	2xAA	2xAA
Kapacitás	2850mAh	2850mAh
Rádió Frekvencia	868/916MHz	868/916MHz, 433, vagy 315 MHz
Moduláció típusa	Amplitude Shift key	Frequency Shift

1. táblázat

A pozíciómonitorozó rendszerhez használt Cricket mote egy Mica2 programozható lap alapú, kísérleti célokra szánt eszköz, mely ultrahang adó-vevő cilinder párral van felszerelve. Ennek segítségével – a körülményektől függően – akár centiméteres pontosságú távolságmérésre képes. Ezt az RF-MEMS-et navigációs eszközök, helymeghatározó rendszerek számára tervezték. A távolság méréséhez két különböző szerepre van szükség: az egyik a „jelzőfényt” adó, a másik az ultrahang jelet fogadó mote. Egy úgynevezett „jelzőfény” egyszerre kibocsátott rádió-, és ultrahang hullámokból áll: a fogadó mote a rádió és az ultrahang eltérő sebessége alapján, az érkezési idők segítségével, különböző algoritmusok alapján határozza meg a mérésben résztvevő mote-ok légvonalbeli távolságát.

5. SPP alkalmazása

Motiváció és informális feladatleírás

A feladat központi vezérlés nélküli pozíciómonitorozó rendszer készítése síkban (esetleg térben) mozgó Crossbow Cricket mote-ok ad-hoc hálózatán. A központi vezérlés hiánya azt jelenti, hogy a koordináták meghatározása rögzített koordináta-rendszer nélkül történik, azaz a viszonyítási pontok meghatározása is a beágyazott rendszer feladata, míg a befogadó rendszer – PC-n futó grafikus program - mindössze az adatok feldolgozását és megjelenítését végzi.

Az adatcsatorna PO-i a koordináta-rendszer bázisai (mely az origóból az „x" tengelyt, és az „y" tengely meghatározó mote-okból áll), illetve a többi – bázis-relatív helyzet-meghatározást végző - elemének közös PO-ja. A bázisok feladata jelzőfények meghatározott sorrendben történő sugárzása. Ezen felül a bázisok a saját koordinátáikat is számítják (vagy, az origó esetében kijelölik). A negyedik PO-hoz tartozó mote-ok feladata távolságmérések segítségével koordinátáik kiszámítása, és ezek továbbítása. Az adatokat a rendszer indítás után külső hatás nélkül generálja, dolgozza fel, majd továbbítja. A számítási idő - élő rádió kapcsolat esetén - egy mote-ra vetítve minimális, így a rendszer áteresztési ideje (az összes mote pozíciójának egymás után történő meghatározása) kevés (öttizenöt) mote esetén nem jelentős. Tapasztalatok alapján maximálisan harminc-harmincöt mote működhet egy koordináta-rendszeren belül, úgy hogy a számolt koordináták megfelelő időközönként frissüljenek. Mivel az adatok itt nem

argumentumok, hanem a rendszer generálja azokat (jelzőfények, mérési eredmények), az újraindítási időnek nincs szerepe. (A rádió kölcsönös kizárásos használata miatt megegyezik az áteresztési idővel.)

Egy körben sorra következik az origó, az „x” irányt jelölő, majd az „y” irányt jelölő mote, végül a negyedik PO egyik tagja. Ez addig ismétlődi, amíg az utolsó PO-ból minden mote sorra kerül, utána a ciklus újra kezdődik.

A rendszer használatára az SKMK-ban kerül majd sor, ahol kerekes székek helyzetének meghatározására fogják használni. Ennek segítségével az ápolók nyomon követhetik a páciensek tartózkodási helyét, illetve szükség esetén riaszthat is a rendszer. Egy kerekes székre kettő-négy mote kerül többek közt azért, hogy a pozíció mellett a szék állásának iránya is követhető legyen, illetve mindig rendelkezésre álljon a feladatban rögzített minimális számú mote.

Az fenti példaalkalmazás olcsó, kis erőforrás igényű hálózaton fut, amely rendelkezik skálamentes kisvilág tulajdonsággal. A rádión történő kommunikáció miatt a hálózat gráfja erősen klaszterezett komponensekből áll.

A feladathoz készített adatcsatorna működése viszonylag könnyen áttekinthető, és valós környezetben is használható, tesztelhető. Mivel az eszközök fizikai mérete nem jelentős, ezért helytakarékosan építhető belőlük hálózat, alacsony energiafogyasztás mellett. (Összehasonlítva például egy PC-kből álló grid-del.) A rendszer által használt erőforrások a mote-ok számítási kapacitása és a hálózathoz (itt: rádióhoz) való hozzáférés.

A megbízhatósági skála a következőn alapul: minden mote meghatározott (tárolt, illetve folyamatosan frissített) sorrend alapján következik, ezért ha valamely mote a sorrendnek megfelelő időben jelzi adat fogadását és várakozási időn belül teljesíti a feladatát, akkor nő, ha valamelyik a fentiek közül nem teljesül, akkor csökken a weblog értéke.

Egyszerűsítések

Az előző szakaszban meghatározott feladat megoldásához az SPP-t bemutató fejezetben részletesen leírtak az alábbi módon kerültek alkalmazásra.

Mivel az adatok nem kívülről érkeznek, az adatcsatorna eleje és vége „rövidre zárt”, azaz az újraindítást kiváltó esemény a helyzet-meghatározó PO-ból érkező, számított koordinátákat tartalmazó üzenet.

A koordináta-rendszert kifeszítő három PO-ban egy-egy mote vesz részt, így az SPP által definiált szerepek összetettek: itt a PO-t vállaló mote egyszerre lesz irányító, disztribútor, illetve processzor. A negyedik PO–ba a résztvevő mote-ok számától függő engedély birtokában lehet belépni. Az engedélyezés a weblog

érték alapján történik. Itt minden processzor egyben disztribútor is, azaz minden mote maga felel azért, hogy a saját számításhoz szükséges adatokhoz a megfelelő időben hozzájusson.

Az SPP Pipe mellett összesen négy üzenetküldési csatornára van szükség PO-nként, amik a szomszédos PO-val való kapcsolattartást szolgálják. Mindegyik egyirányú, multicast csatorna. Páronként egy-egy bejövő és kimenő csatorna biztosítja a kommunikációt a következő és a megelőző PO-val. Az ellenőrzési feladatok PO-n belül a weblog értékek karbantartásából állnak, míg a rákövetkező következő taszk működésének ellenőrzése az adatok nyugtázásának fogadását jelenti.

Itt minden hálózati entitás azonos architektúrájú, ezért a teljesítménybeli különbségek legfeljebb a mote-ok egymástól vett távolságából, esetleg a rádiót árnyékoló tereptárgyak helyzetéből adódnak. Emiatt nem kell foglalkozni különböző erőforrások nyilvántartásával.

A pozíciómonitorozó rendszer szervezésének lépései, a kezdeti feltételek fennállása esetén:

1) A feladat hirdetése az SPP Pipe-on
2) Valamelyik mote elvállalja az origó PO-jának részfeladatát.
3) Valamelyik mote elvállalja az „x” tengelyt meghatározó PO részfeladatát.
4) Valamelyik mote elvállalja az „y” tengelyt meghatározó PO részfeladatát.
5) Legalább egy mote vállalja a helyzet-meghatározó PO feladatát.

PO-hoz csatlakozás menete:

1) Kérelem küldése a PO manageréhez.
2) Válaszok fogadása.
3) Elégséges engedély esetén csatlakozás.
4) Nem elegendő engedély esetén késleltetés után újra próbálkozás.

Javítás menete a PO-k szintjén:

1) Ha várakozási időn belül nem jön visszajelzés a következő részfeladat manager-étől, akkor manager weblog értékének csökkentése, új manager keresése.

Javítás menete a PO-n belül:

2) Ha várakozási időn belül nem érkezik visszajelzés, vagy adat a soron következő mote-tól, akkor a mote weblog értéke csökken, és a sorrendben következő végezheti a számítást.
3) Ha így valamely mote-nál küszöb alá esik weblog érték, akkor az adott mote-ot a többek törlik a rákövetkezési sorból, és kódváltással kizárják a PO-ból.

6. A NIPG által készített rendszer

TinyOS

Az alkalmazás implementálása a Mica platformon futó TinyOS operációs rendszer alatt, NesC programnyelven készült. A TinyOS operációs rendszert a kaliforniai Berkeley egyetemen fejlesztették ki vezeték nélküli beágyazott rendszerek hálózatai számára. A rendszer forráskódja nyílt, ma a fejlesztés az ipar, és más egyetemek bevonásával folyik. A programcsomag többek közt tartalmazza a szenzorok driver programjait, kommunikációs protokollokat, elosztott számításokhoz szükséges eszközöket.

A beágyazott rendszerek speciális követelményei miatt a TinyOS egy mindössze néhány kilobyte-os operációs rendszer, ami kezeli a MEMS eszközök megszakításait, illetve ütemezi a programszálakat. Az operációs rendszer mindig egyetlen programot futtat, amely a kiválasztott, rendszer által definiált, vagy fejlesztő által készített komponensekből áll. A végrehajtásnak két szála van: taszkok és hardver-eseménykezelők. A taszkok futása ütemezett: ha egy taszk sorra kerül, akkor befejezésig fut anélkül, hogy másik taszk átvenné a vezérlést. A hardver eseménykezelők megfelelő megszakítás hatására indulnak el, és befejezésig futnak, de megszakíthatják taszkok vagy más eseménykezelők futását. Az eseménykezelőkben hívott parancsokat (command) és eseményeket (event) „async” kulcsszóval kell deklarálni, hogy az atomi végrehajtást a TinyOS szavatolni tudja.

NesC programnyelv, beépített modulok

A NesC a TinyOS operációs rendszerhez készített C szintaxisú programozási nyelv, mely támogatja a kód-újrafelhasználást, illetve optimális méretű programok készítését.

A NesC alkalmazások jól definiált, kétirányú interfészekkel kapcsolódó komponensekből állnak, a típusok meghatározása fordítási időben történik. A nyelv konkurens programozási modellje taszkokra (task) és eseménykezelőkre (event-handler) épül.

Egy futtatható alkalmazás egy vagy több egymáshoz kapcsolt komponensből áll, mely (melyek) megvalósítanak és használnak különböző interfészeket. Ezek a kétirányú interfészek a komponensek kizárólagos kapcsolódási pontjai. A kétirányúságot az interfész által definiált parancsok (a komponens által nyújtott szolgáltatások) és események (a komponens használója által megvalósítandó függvények) biztosítják. Egy komponens parancsainak használatához az adott komponens eseményeinek implementálása szükséges. Egyetlen komponens több interfészt is megvalósíthat és használhat, akár egyazon interfész több példányát is.

A nesC komponensek két típusa van: modul és konfiguráció. A modulok tartalmazzák az interfészek megvalósítását, a konfigurációk pedig komponenseket kötnek össze azok interfészeinek összekapcsolásával (wiring). Egy nesC alkalmazást egy legfelső szintű konfiguráció ír le, amely összeköti a felhasznált komponenseket.

Az SPP különböző részeinek implementálásához a következő TinyOS programmodulok (interfészek és megvalósításaik) kerültek felhasználásra:

Timer interfész

Egyszerű időzítőt definiáló interfész két paranccsal és egy eseménnyel. A parancsok segítségével indítható el, illetve állítható le, míg az időzítő lejártakor eseményt vált ki. Az időzítő eseményét kezelő függvények jellemzően valamilyen hibakezelési eljárást valósítanak meg. A különböző időzítők a várakozási idők mérését végzik az alkalmazásban, például az elküldött üzenet után a küldő az adat nyugtázására szánt idővel indít egy timert.

SendMsg és ReceiveMsg interfészek

Az SPP által definiált kommunikációs csatornák alapjául szolgáló paraméteres interfészek. A paraméter a küldendő, (vagy fogadni kívánt) üzenet típusát adja meg. A kommunikációs csatornák ezeknek az interfészeknek a kiterjesztései:

megkapják, és a csatorna azonosítója alapján szűrik az adott típusú üzeneteket, hogy a hozzájuk regisztrált mote, vagy mote-ok csak a nekik szólóakat kapják meg.

UltrasoundControl interfész

Az ultrahang adó-vevő működését vezérlő eljárásokat meghatározó interfész. Beépített megvalósítása tartalmazza a távolságméréshez szükséges algoritmusokat, ezen belül a jelzőfényt adó (beacon) és az azt vevő (listener) szerepekhez szükséges parancsokat. A koordináta-rendszert meghatározó három PO processzorai a beacon, a negyedik PO-hoz tartozó processzorok a listener szerephez tartozó szolgáltatásokat használják.

PositionEstimator interfész

Az ultrahangos mérések eredményeit feldolgozó, háromszögeléssel koordinátákat számító algoritmuscsomagot tartalmazza.

SPP implementálása, elkészített programmodulok

A TinyOS beépítetett programmoduljainak kiterjesztése mellett szükség volt teljesen új modulok készítésére az SPP funkcionalitásának megvalósításához. Ezek az alkalmazás magas szinten elhelyezkedő komponensei, amik felhasználják és összefogják a beépített, illetve „kiterjesztett” modulokat. Ezek valósítják meg az erőforrások által ellátott szerepeket.

SPPProcessor

A processzor absztrakt interfésze, tartalmazza az általános műveleteket (indítás, adatküldés, jelenlét lekérdezése, stb.)

BasePOM modul

Az origó, az „x” és az „y” irány PO-inak közös tulajdonságait összefoglaló modul: megadott időközönkénti jelzőfény küldését definiálja. Az absztrakt PO interfészt implementálja a BaseSPPM modul számára.

BaseSPPM modul

A konkrét bázisokhoz tartozó SPPProcessor interfészt implementálja. Ez a bázisok PO-inak legmagasabb szintű implementációja. A modul összefogja a kommunikációs csatornákat, a PO processzorainak állapotait, tárolja a szükséges adatokat, megvalósítja a részfeladatot megoldó, üzenetfogadó, üzenetküldő, hibakezelési, és méréseket feldolgozó algoritmusokat.

DMPosEstPOM modul

Felhasználja a jelzőfényeket feldolgozó DMPosEst interfészt, és implementálja a kiszámolt koordináták továbbküldését. Az absztrakt PO interfész DMPosEstSPPM modul számára megfelelő, specifikus implementációja.

DMPosEstSPPM modul

A negyedik PO számára megfelelő SPPProcessor interfészt implementálja. Felépítése és funkciója hasonló a BaseSPPM moduléhoz. Ebben a modulban szükség van a PO létszámának, a PO-n belüli sorrendnek, és a PO-ban résztvevők weblog értékeinek nyilvántartására, és viszonylag gyakori frissítésére.

WebLog interfész

A weblog értékek nyilvántartásához szükséges parancsokat definiálja. A DMPosEstSPPM modul használja.

RFSPPSystem konfiguráció

Az teljes alkalmazás legfelső szintű komponense. Többek között elküldi a feladatokat az SPP Pipe-on, és indítja a PO-khoz történő csatlakozást. Az RFSPPSystemM programmodul futásának leállítása a feladat számításának vége.

7. Startteszt

Probléma ismertetése, algoritmus robusztusságának javítása

Az SPP projekt átvételekor az első feladat az implementált rendszer tulajdonságainak, ezen belül az adatcsatorna megszervezési idejének tesztje volt. Bár az átvett programcsomag átesett funkcionális tesztelésen, hiányzott még a különböző körülmények közötti, sok futtatást tartalmazó teszt, melyből következtetni lehet a működés kiegyensúlyozottságra.

A szervezési idő mérésekor fény derült arra, hogy bizonyos esetekben nem tud az összes, rádió hatókörén belül lévő mote csatlakozni az adatcsatornához (ez először 10 mote esetén fordult elő). A probléma a felállási idők szórásából tűnt ki: az egyes kísérletek ideje maximálva volt, és a mote-ok számának növekedésével a kísérletek között egyre több, jóval az átlag idő feletti, maximális idejű eredménytelen futás fordult elő. A vizsgálat során kiderült, hogy a nyilvántartott sorrend frissítésével van probléma.

Azért, hogy a rendszer konzisztenciája fennmaradjon, a mote-ok a számított adatokhoz a saját nyilvántartásuk egy részét is mellékelik. Így – több más ellenőrző adat mellett – a koordináták továbbításakor a helyzet-meghatározó PO tagjai elküldik a nyilvántartott sorrend prefixét is. Mivel a rádió használata költséges művelet, ezért ellenőrző adatokat korlátozott mennyiségben tartalmazhat a kommunikáció (nem küldhető el a teljes sorrend). A hiba feltárásakor kiderült, hogy önmagában nem elég a nyilvántartott sorrendet a kapott ellenőrző prefixre cserélni a többiektől való eltérés esetén, mert előállhat olyan helyzet, ahol a mote a saját nyilvántartott sorrendjében rendre a prefix hossza utáni első indexre kerül. Ilyenkor előfordulhat olyan helyzet, hogy a kérdéses mote valójában csak a saját nyilvántartása szerint a PO tagja, míg a többi mote rákövetkezési sorában viszont nem szerepel.

A fent ismertetett probléma miatt a protokoll üzenetkezelési algoritmusát kiegészíttettem. A megoldás lényege az, hogy a csatlakozó mote korlátot szab sorra kerülésének idejére, a PO-hoz tartozó mote-ok száma alapján. Az időkorlát túllépése pontosan azt jelenti, hogy a többiek nem tartják nyilván. A korlát elérésének vizsgálatára az ellenőrző adatokat tartalmazó üzenetek fogadásakor

kerül sor. A megfelelő feltételek teljesülése esetén hibakezelés lép életbe, és a mote újraindítja a csatlakozási eljárást.

Ezzel a kiegészítéssel a robusztussági jellemzők annyival javultak, hogy a további kísérletek során az indulási időkkel kapcsolatos szórási anomália többé nem fordult elő. A lentebb bemutatott, rendszer-felállási időre vonatkozó mérések a javítás után történtek.

A kísérletek körülményei

A kísérletek során a mote-okat egy megfelelő méretű kör kerülete mentén egyenletesen helyeztem el. Így zártam ki azt az esetet, amikor a koordináta-rendszert meghatározó három mote egy egyenesre esik. A gyakorlatban, megfelelő elhelyezés mellett ennek az esetnek az előfordulása dinamikus rendszer esetén is minimálisra csökkenthető.

A teljes rendszer felállásának feltételei a következők voltak: egyrészt, a feladatban szereplő négy Pipeline Operation mindegyikét elvállalta valamely mote, másrészt a kísérletben részt vevő összes mote csatlakozott valamelyik PO-hoz, és legalább egyszer elküldte az általa számolt koordinátákat. A futtatások során egyetlen adatcsatorna felépítésére került sor, a felállás ideje a fentebb meghatározott feltételek első teljesülésének indítástól számított ideje.

A kísérleteket a TinyOS TinyViz nevű szimulátorában végeztem az egyszerűbb vezérlés, kényelmesebb paraméterezés és energiatakarékosság miatt. A rádió elérésének kölcsönös kizárásos volta miatt el lehet tekinteni a valódi párhuzamosságkor esetleg előforduló egyidejű kommunikációtól. A valós rendszerben az időben átfedve küldött csomagok az interferencia miatt elvesznek, így a sikeresen átvitelre kerülő (fogadott és nyugtázott) üzenetek - az üzenetküldés absztrakciós szintjén – időben egymástól teljesen elkülönülő elemekből álló sorozatot alkotnak abban az esetben, ha minden mote közvetlenül el tudja érni az összes többit. Ez megfelel az egyetlen processzoron futó szimulátor ütemezési lehetőségeinek. A tesztek során egyetlen adatcsatorna kialakításának idejét mértem, és a hálózati topológia megválasztásánál szempont volt, hogy a mote-ok kölcsönösen egymás rádió-hatókörén belülre essenek.

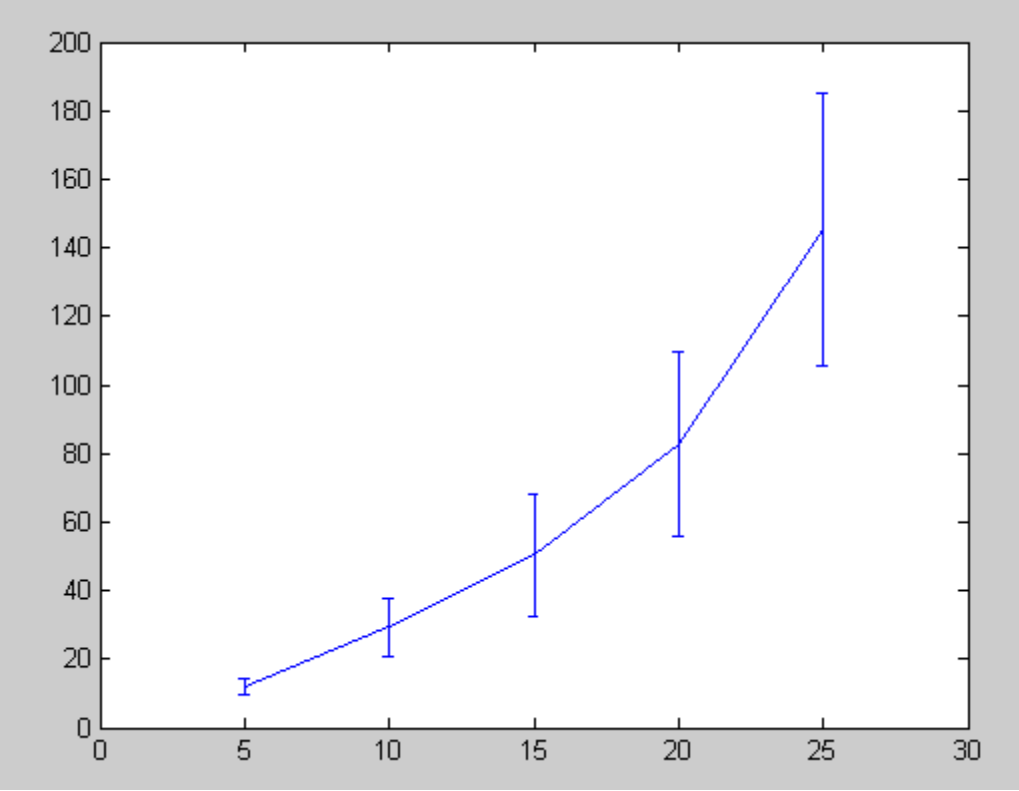
Mérési eredmények

A méréseket rendre öt, tíz, tizenöt, húsz, huszonöt mote-ból álló rendszerrel végeztem. Különböző méretű rendszerekkel tizenöt-tizenöt tesztet futtattam. A mérési eredményeket a 2. táblázat tartalmazza.

	1.	**2.**	**3.**	**4.**	**5.**	**6.**	**7.**	**8.**	**9.**	**10.**	**11.**	**12.**	**13.**	**14.**	**15.**
5	16	9	13	13	11	14	11	16	9	12	12	13	9	13	11
10	31	23	19	33	31	26	33	41	20	23	26	30	20	27	23
15	56	35	34	45	36	47	49	47	47	47	43	87	44	42	97
20	101	86	46	87	67	68	59	132	50	125	107	79	111	61	62
25	174	117	145	139	131	135	139	223	104	115	237	112	146	159	105

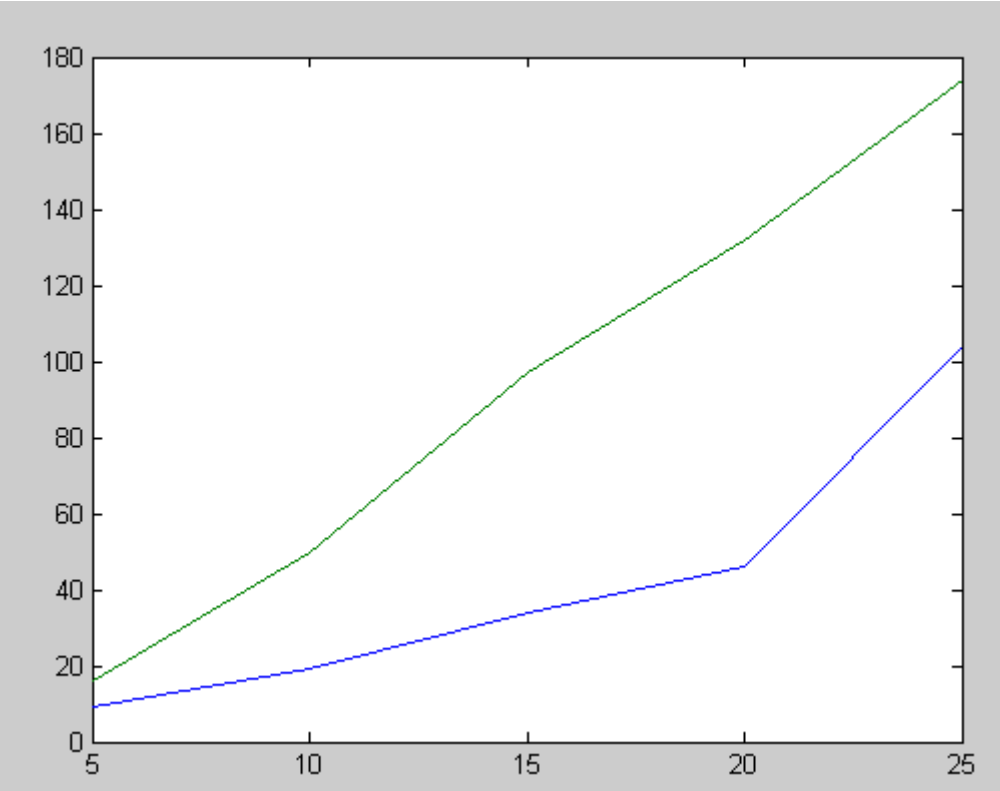
2. táblázat: rendszer felállási ideje a különböző kísérletek során

A mérések átlaga és szórása az 1. ábrán látható. Az ábráról nagyjából négyzetes növekedés olvasható le.



1. ábra

A legjobb, illetve legrosszabb eredmények összehasonlító grafikonja a 2. ábrán található.



2. ábra

8. Taszkok találkozása

A probléma megfogalmazása

Mivel a rádióhoz, mint erőforráshoz kölcsönös kizárásos módon lehet hozzáférni, illetve mivel az eddig leírtakban nem került specifikálásra erőforrás-megosztási probléma, a következővel kell szembe nézni: csak a taszkon, mint zárt rendszeren belül meghatározott a rádiózási sorrend, így taszkok közti ütemezés definiálatlan. Dinamikus ad-hoc hálózatokról lévén szó, a taszkok tetszőleges számban és mértékben fedhetik át egymást. Jelen esetben holtpont-helyzet nem állhat fenn, de fenn kell tartani a minőségi követelményeknek megfelelő működést.

Ebben a fejezetben a pozíciómonitorozó rendszerrel kapcsolatban felmerülő konkrét problémával foglalkozom. A rendelkezésre álló, rádió alapú hálózat sajátosságait fogom figyelembe venni, azonban az ismertetett problémák, szerepek ezektől elvonatkoztatva más környezetben is érvényesek lehetnek.

A taszkok találkozása a PO-k szintjén értendő. Egy taszk megjelenése egy mote rádiójának hatáskörében csak az érintett mote és PO-ja rádió használatának felfüggesztését jelenti, míg az adatcsatorna többi taszkja fut tovább a rendszer telítődéséig.

Az SPP feladat eredményét kiszámító adatcsatornán belül az egyes résztaszkok kommunikációs sorrendje meghatározott. Egy másik adatcsatorna valamely elemének megjelenésekor a kommunikáció a zavarás alatt lévő résztaszknál mind bejövő, mind kimenő irányban le kell, hogy álljon. Ezután az érintett, különböző adatcsatornákhoz tartozó taszkoknak valamilyen egymás közti sorrendet kell meghatározniuk (pontosabban az igények alapján el kell osztani a rádió-hozzáférést). A kialakított váltakozó működés lehetőleg minimálisan hátráltassa az adatok fogadását és küldését, mindkét oldalon.

A fentiek alapján a szükséges tevékenységek:

- konkurens helyzet felismerése,
- rádiózás szüneteltetésének, illetve újraindításának vezérlése PO-k között, a saját adatcsatornán belül,
- a megegyezéshez szükséges adatok átviteléhez dedikált csatorna allokálása,

- az adatok segítségével, valamely protokoll alapján a rádió-hozzáférés elosztása,
- a konkurens helyzet megszűnésének felismerése,
- a működés visszaállítása a találkozást megelőző állapotba.

A konkurens helyzet felismerésekor figyelembe kell venni, hogy mely beérkező üzenetekből lehet további üzenetforgalomra következtetni. Például leállást jelző üzenet vétele esetén nincs szükség a helyzet kezelésére.

A konkurens helyzetet felismerő mote-nak értesítenie kell a az esetlegesen a PO-jához tartozó többi mote-ot, a megelőző-, illetve a következő PO-kat. Erre azért van szükség, hogy az adatcsatorna esetleges szüneteltetését az érintett PO-k irányítói ne hiba miatti leállásként értékeljék, és ne kezdjenek a résztaszk újraszervezésébe.

Másik adatcsatornával történő kapcsolattartáshoz a harmadik fejezetben ismertetett protokoll nem definiál csatornát. Az adatcsatornában részt vevő összes entitás képes kell legyen az SPP által definiált csatornákon kívüli adatforgalom észlelésére, szükség esetén a korlátozott mértékben történő olvasására. Emellett adott helyzetben képesnek kell lennie arra, hogy a másik érintett mote-tal felvegye a kapcsolatot.

A mote-oknak tárolnia kell egy előre definiált eljárást a rádió-hozzáférés megosztásához, amely alapján az egymást zavaró mote-ok a megfelelő adatok ismeretében megvalósíthatják a kölcsönös kizárást.

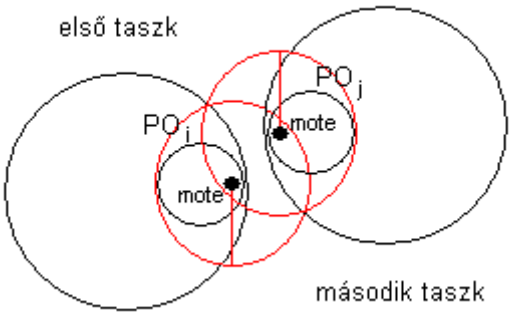
Az egyeztetett adatok alapján rendszeresen ellenőrizni kell, hogy a fenn áll-e a zavarás, és a konkurens helyzet megszűnésekor értesíteni kell a megfelelő PO-kat a helyzet változásáról.

A megoldás ismertetése két taszk esetére

Feladatom az volt, hogy az egy taszk megszervezését, és működtetését implementáló alkalmazást bővítsem olyan módon, hogy egymás közelében működő két taszk ne zavarja egymás kommunikációját. Ez erőforrás-megosztási probléma, ahol az erőforrás a taszkok rádiózási hatásköreinek metszete, amihez a hozzáférés kizárólagos (egyszerre egy mote küldhet üzenetet). Mivel ez az egyetlen olyan erőforrás, aminek „várakozási sora” lehet, ezért holtpont-helyzet [8]

nem alakulhat ki. Megjegyzendő, hogy több taszk esetére az alábbi megoldás általánosítható, a kiéheztetés [8] problémájának figyelembe vételével.

Taszkok találkozása alatt a következőkben azt értjük, hogy valamely taszk legalább egy mote-jának rádió vevőkörében üzenetet küldő, másik taszkhoz tartozó mote van. Azt az esetet, amikor a vételi sugárban van ugyan másik taszkhoz tartozó mote, de az nem küld rádióüzenetet, jelen szempontból nem vizsgálom.



3. ábra, taszkok találkozása: a piros körök a mote-ok rádiójának hatókörét jelzik.

Elméletben elképzelhető, hogy az érintett mote-ok különböző erősségű rádiójeleket adnak, így meghiúsulhat a kapcsolatfeltétel. Mivel azonos architektúrájú mote-ok vesznek részt a számításokban, elérhető, hogy a különböző adatcsatornákhoz tartozó taszkok kommunikációjában a gyengébb jelet sugárzó mote a rádióját a megfelelő erősségűre hangolja. Ez szükség esetén megoldható úgy, hogy az első üzenet a lehető legerősebb jelerősséggel kerül sugárzásra, és a résztvevő felek megegyeznek a használandó erősségben. A továbbiakban felteszem, hogy minden mote azonos erősségű jelet sugároz.

A fentiek alapján a mote-ok szintjén két szerep különböztethető meg: legyen *A* és *B* más-más taszkhoz tartozó mote. A vázolt módszerhez szükség van egy időzítőre, melynek lejárta pontosan azt jelzi, hogy másik taszkból nincs senki a közelben.

1) *A* észleli *B* üzenetét, ekkor:

   a) Az adatküldés szüneteltetésére szólítja fel az előtte lévő PO-t, várakozásra az utána következőt.
   b) Üzenetet küld *B*-nek, melyben közli az adatait.
   c) Elindítja az időzítőt a válasz megérkezéséhez szükséges idővel.

2) Ha *B* megkapja *A* üzenetét, akkor eltárolja belőle a taszk és a mote azonosítót, és a másik mote számítási idejére vonatkozó felső korlátot. Ezután választ küld *A*-nak a saját adatival.

3) Ha *A* megkapja a választ, akkor újraindítja az időzítőt a kapott számítási korláttal és várakozik.

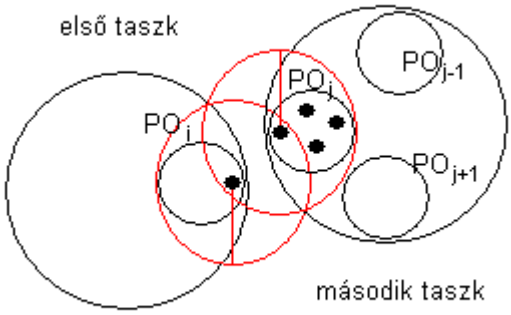
4)
   a) Ha *A* hallja *B* adatot tartalmazó üzenetét, és nem hall további üzenetet, akkor leállítja az időzítőt és engedélyezi az adatküldést az előtte lévő PO-nak.
   b) Ha *A* hallja a *B* által küldött adat nyugtázását, akkor újraindítja az időzítőt a számítási korláttal, és ezt ismétli addig, amíg annak lejárta előtt hallja a rendre elküldött adatok nyugtázását.

5) Ha *B* adatot kap, és nyilvántartása szerint *A* várakozik rá, akkor az adatküldés szüneteltetésére szólítja fel az előtte lévő PO-t, várakozásra az utána következőt és elindítja az időzítőjét *A* számítási idejének felső korlátjával. Ha ezt az üzenetet *A* hallja, akkor leállítja az időzítőt és engedélyezi az adatküldést az előtte lévő PO-nak, ezen felül eltárolja, hogy *B* rá várakozik. (*A* és *B* szerepe ekkor megcserélődik.)

6) Ha *A* időzítője lejár, akkor engedélyezi az adatküldést az előtte lévő PO-nak.

Megjegyzés: közös számítás végzése (például egymás koordinátáinak saját bázisba történő átszámítása) új PO meghatározását igényli, a fent ismertetett módszer csupán a hibamentes működés fenntartására törekszik.

Példa: két taszk találkozásának egyszerű esete



4. ábra, taszkok közti konkurens helyzet egy-egy érintett PO esetén

1) Az első taszk i. PO-jának egyik tagja hallja a második taszk j. PO-jába tartozó mote üzenetét, de a j-1. és j+1. PO-ba tartozó mote-okat nem

2) Kapcsolatot kezdeményez a hallott üzenetben szereplő azonosító segítségével, és értesíti a szomszédos PO-kat.

3) A zavaró mote fogadja a kapcsolatot kezdeményező üzenetet, és választ küld a megfelelő adatokkal.

4) A kezdeményező megkapja a választ, továbbítja a várakozási időt a szomszédos PO-knak, és a másik taszkra várakozik.

5) A kezdeményező hallja a második taszk mote-jának számolt adatot tartalmazó üzenetét, és a várakozási időn belül nem érkezik hozzá más üzenet. Ekkor értesíti a szomszédos PO-kat a várakozás beszüntetéséről és folytatja a munkát.

Az implementáció ismertetése

PO2POMsg kiegészítése

A PO-k közötti üzenetküldésekre használt üzenettípust kiegészítettem az adatcsatorna azonosítójának, a másik taszkhoz tartozó mote azonosítójának, illetve a számítási idő felső korlátjának fenntartott mezőkkel. Bővítettem az üzenettípus altípusainak listáját. Az új altípusok segítségével értesíthetőek a megfelelő PO-k a várakozásról és az adatküldés szüneteltetéséről.

A taszkok közti kapcsolat felépítése utáni kommunikáció során ezt az üzenettípust használják a mote-ok.

PingMsg üzenettípus

Ha valamelyik mote egy konkurens helyzetet észlel, egy ilyen típusú üzenettel indítja a taszkok közötti kommunikációt. Az üzenetben elküldi a fontosabb adatait (saját azonosítóját, az adatcsatornája azonosítóját, az általa vállalt számítási feladat elvégzési idejének felső korlátját). Ha erre az üzentre válasz érkezik, akkor mindkét oldalon létrejön az ütemezés a kicserélt adatok alapján. Ennek az üzenettípusnak a további kommunikációban nincs szerepe.

Ha szükség van a rádió sugárzási erejének hangolására, akkor a kommunikációt indító üzenet maximális jelerősséggel kerül küldésre, és az üzenetváltás tartalmazza a mote-ok által használt jelerősségeket is. Ennek segítségével sor kerül a rádió hangolására.

DMPosEstSPPM és BaseSPPM modulok kiegészítése

Az processzorok implementációját több helyen kiegészítettem. Szükség volt új, az összes többitől független állapothalmaz definiálására, mely segítségével egyrészt leírhatóak a taszkok találkozásának egyes lépései, másrészt a szétváláskor meghatározzák, hogy mely állapotba kell a rendszert visszaállítani.

A CtrlMsgListener komponens segítségével ezekben a modulokban valósítottam meg a konkurens helyzet detektálását, itt implementáltam az előző alfejezetben ismertetett megoldáshoz szükséges új üzenetküldéseket, mind a másik taszkkal történő kapcsolattartáshoz, mind a PO-k értesítéséhez. Ezután az üzenetek fogadását bővítettem az új üzenetek kezelésével.

Az új állapotok segítségével finomítottam a hibakezelést, így a rendszer szükség esetén megfelelő ideig várakozik a másik taszkra.

A konkurens helyzet megszűnésének ellenőrzését a CtrlTimer komponens segítségével valósítottam meg.

CtrlMsgListener komponens

A processzorok által használt kommunikációs csatornák (pipe-ok) csak a csatornához regisztrált mote-ok üzeneteit továbbítják. Az teljes, saját adatcsatornán kívüli üzenetforgalmat a CtrlMsgListener komponens felügyeli és szűri.

A CtrlMsgListener interfész lehetőséget ad a megfelelő taszk-azonosító beállítására, ami alapján az üzenetek szűrése történik.

A CtrlMsgListenerM modul akkor váltja ki a hozzá tartozó „receive" eseményt, amikor másik taszkhoz tartozó, PO-k közötti üzenetet kap: az esemény első kiváltása a konkurens helyzet felismerése, ezután a taszkok közti kommunikációban vesz részt.

CtrlTimer komponens szerepe

A CtrlTimer komponens által kiváltott esemény jelenti a konkurens helyzet megszűnését. Az eseménykezelő felelős a taszkok találkozását megelőző állapot visszaállításáért.

9. Teszt

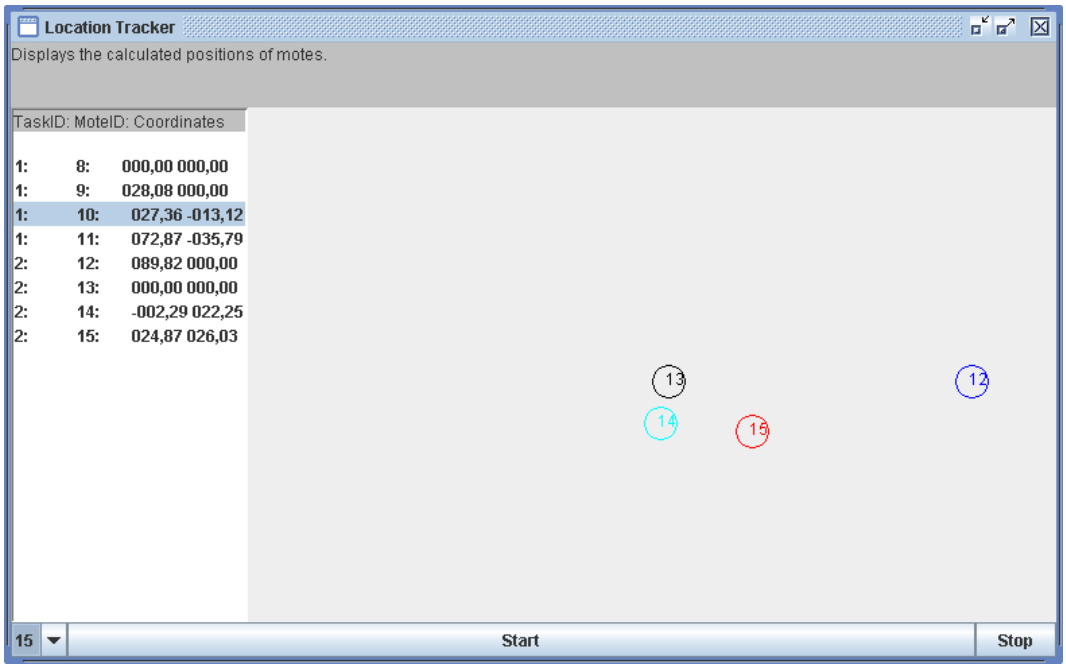
Körülmények leírása

A rendszert irodai székekre szerelt cricketek segítségével teszteltem. A külön rendszerek a folyosón és a laborban működtek, majd fokozatos közelítéssel

kerültek egymás hatókörébe. A rádióhullámokat a rendszerek közt lévő fal nem árnyékolta le, több beállítással és távolsággal is próbálkoztam. Végül sikerült a megfelelő körülményeket megteremteni a teszt számára (Mind a rendszer kialakulása alatti zavarás, mind a teljes átfedés túlzottan lassú működést eredményez).

Eredmények ismertetése

A rendszerek észlelték egymást, és az algoritmus segítségével sikerült fenntartaniuk az egymás melletti működést. Az 5. ábra működés közben, egymás hatókörében mutatja a két rendszert. A taszkok megjelenítésekor a koordináták állandóan, a topológia kirajzolása felváltva történik.



5. ábra

A futtatáshoz tartozó üzenetküldési napló részlete:

2043428 PO2POMsg 12 MSG_DATA 3 1 15 112 1 0 1 1 12
2043458 PO2POMsg 11 MSG_DATA 3 1 8 56 1 0 1 0
2043478 BeaconMsg 8 2 12.130669 20.95146 0.0
2043508 PO2POMsg 8 MSG_ACK 3 1 8 56 1 1 1 1 8

```
2043538  PO2POMsg   13 MSG_HALTPREVPO_SENDING   2 1 13 0 1 1 0 0
2043558  PO2POMsg   8 MSG_ACK                  3 1 8 56 1 1 1 1 8
2043588  PO2POMsg   8 MSG_DATA                 4 4 8 56 1 0 1 0
2043608  PO2POMsg   12 MSG_HALTPREVPO_SENDING   3 1 12 0 1 1 0 0
2043628  PositionMsg 9 -2.6207674 29.774647 0.0
2043658  PO2POMsg   9 MSG_ACK                  4 4 9 56 1 4 1 1 9
2043678  PO2POMsg   12 MSG_DATA                3 1 15 112 1 0 1 1 12
2043698  PO2POMsg   9 MSG_ACK                  4 4 9 56 1 4 1 1 9
2043728  PO2POMsg   9 MSG_DATA                 1 1 8 57 1 0 1 0
2043899  BeaconMsg  11 1 105.186356 0.0 0.0
2043939  PO2POMsg   11 MSG_ACK                 2 1 11 57 1 1 1 1 11
2043989  PO2POMsg   12 MSG_RESTARTPREVPO_SENDING 3 1 12 0 1 1 0
0
2044009  PO2POMsg   11 MSG_ACK                 2 1 11 57 1 1 1 1 11
2044039  PO2POMsg   12 MSG_HALTPREVPO_SENDING   3 1 12 0 1 1 0 0
2044099  PO2POMsg   9 MSG_STOPPED              4 4 9 0 1 4 1 0
2044149  PO2POMsg   11 MSG_DATA                3 1 8 57 1 0 1 0
2044189  BeaconMsg  8 2 10.9201355 21.615093 0.0
2044209  PO2POMsg   8 MSG_ACK                  3 1 8 57 1 1 1 1 8
2044259  PO2POMsg   13 MSG_HALTPREVPO_SENDING   2 1 13 0 1 1 0 0
2044279  PO2POMsg   8 MSG_ACK                  3 1 8 57 1 1 1 1 8
2044299  PO2POMsg   12 MSG_HALTPREVPO_SENDING   3 1 12 0 1 1 0 0
2044329  PO2POMsg   8 MSG_DATA                 4 4 8 57 1 0 1 0
2044339  SPPMsg     12 MSG_PROCESSOR_REQUEST    3 0
2044449  PO2POMsg   8 MSG_DATA                 4 4 8 57 1 0 1 1 8
2044680  PO2POMsg   8 MSG_DATA                 4 4 8 57 1 0 1 1 8
2044710  PO2POMsg   13 MSG_RESTARTPREVPO_SENDING 2 1 13 0 1 1 0
0
2044730  PO2POMsg   12 MSG_RESTARTPREVPO_SENDING 3 1 12 0 1 1 0
0
2045010  PO2POMsg   8 MSG_STOPPED              3 1 8 0 1 1 1 0
2045040  PO2POMsg   9 MSG_JOINREQUEST          4 4 9 0 1 4 1 0
2045271  BeaconMsg  13 0 0.0 0.0 0.0
2045291  PO2POMsg   9 MSG_DATA                 1 0 9 0 1 0 1 0
2045331  PO2POMsg   13 MSG_ACK                 1 1 13 113 1 1 1 1 13
2045361  PO2POMsg   13 MSG_DATA                2 1 15 113 1 0 1 0
2045381  BeaconMsg  12 1 96.81452 0.0 0.0
2045411  PO2POMsg   12 MSG_ACK                 2 1 12 113 1 1 1 1 12
2045431  PO2POMsg   9 MSG_HALTPREVPO_SENDING   1 1 9 0 1 1 0 0
2045461  PO2POMsg   11 MSG_HALTPREVPO_SENDING   3 1 11 0 1 1 0 0
2045481  PO2POMsg   12 MSG_ACK                 2 1 12 113 1 1 1 1 12
```

```
2045501   PO2POMsg    11 MSG_HALTPREVPO_SENDING    3 1 11 0 1 1 0 0
2045531   PO2POMsg    12 MSG_DATA                  3 1 15 113 1 0 1 0
2045661   PO2POMsg    12 MSG_DATA                  3 1 15 113 1 0 1 1 12
2045892   PO2POMsg    12 MSG_DATA                  3 1 15 113 1 0 1 1 12
```

Ez a részlet tartalmaz a rádiózás megosztására vonatkozó üzenetküldéseket. A futtatás célja a folyamatos egymás mellett működés demonstrálása volt.

10. Irodalomjegyzék

[1] A. Meretei, Z. Palotai, A. Lőrincz: *„Systems and methods for sensing physiologic parameters of the human body and achieving a therapeutic effect”* - United States Patent 20070043591

[2] www.google.com

[3] www.wikipedia.org

[4] www.xbow.com

[5] www.tinyos.net

[6] Zia, T.A., and Zomaya, A.Y., *„An Analysis of Simulations and Programming in Wireless Sensor Networks*", In the proceedings of the International Workshop on Sensor Networks and Applications

[7] Alec Woo *„The Mica Sensing Platform",* Jan 15th, 2002 NEST Retreat

[8] Kozma L., Varga L.: *A szoftvertechnológia elméleti kérdései"* ELTE Eötvös Kiadó, 2007, ISBN 963 463 648 9

[9] Hawoong Jeong, Albert-László Barabási, Réka Albert, *„Scale-free characteristicsof random networks: the topology of the world-wide web",* Physica A 281 (2000), 69_77.

[10] Erdős, P. and Rényi, A.: *"On Random Graphs"*, Publicationes Mathematicae, Vol. 6, pp. 290-297, 1959.