Budapesti Műszaki és Gazdaságtudományi Egyetem
Távközlési és Telematikai Tanszék
Nagysebességű Hálózatok Laboratóriuma (HSNLab)

Forgalomfüggő Bluetooth hálózat optimalizáló algoritmus tervezése és szimulációja

TDK dolgozat

Félegyházi Márk
V. évfolyam, villamosmérnök szak

Konzulensek: Dr. Halász Edit, docens, Távközlési és Telematikai Tanszék
Miklós György, Rácz András, Ericsson Magyarország Kft.

Budapest, 2000. október 19.

Tartalomjegyzék

1. Bevezetés

Mindennapi életünkben egyre több és több elektronikus készüléket használunk. A mobiltelefonok mellett egyre elterjedtebbek a különböző hordozható számítógépek, személyi segédeszközök (*Personal Digital Assistant, PDA*) is. Bár eszközeink mindegyikét más és más célra használjuk, számtalanszor lenne hasznos, ha a különböző készülékek között kapcsolatot tudnánk létesíteni. Például az elektronikus naptárunkból előkeresett telefonszámot a mobiltelefon azonnal hivni tudja, anélkül, hogy a gombokat megnyomnánk. Amikor hordozható számítógépünkre elektronikus levél érkezik, akkor a küldő adatait azonnal rögzítené személyi segédeszközünk. Mindezek a kapcsolatok vezetékes átvitellel is megvalósíthatóak, de lényegesen hasznosabb és kényelmesebb egy vezeték nélküli, rövid hatótávolságú rádiós vagy infravörös átviteli megoldás.

Az eszközök csatlakoztatására napjainkban a legelterdjettebb megoldás az infravörös átvitel alkalmazása. Az infravörös technológiának azonban több hátránya is van. Egyrészt túlságosan rövid hatósugarú átvitelt biztosít (tipikusan 1-2 m), másrészt a kapcsolat létrejöttéhez az eszközök között közvetlen rálátás szükséges, vagyis nem lehet köztük semmilyen akadály. További hátrányt jelent, hogy infravörös porton keresztül csak 2 eszköz kommunikálhat egyszerre, így ez a technológia hálózat kialakítására nem alkalmas.

1.1. A Bluetooth megoldás

Az igazi megoldást várhatóan az egyre inkább előtérbe kerülő, kis hatósugarú rádiós technológiák nyújtják majd. Ezen technológiák közül a Bluetooth [1, 2, 3], mint globálisan egységes, szabványosított és számos ipari cég által támogatott rádiós rendszer indul a legjobb esélyekkel arra, hogy egy globális szabvány szerepét betöltse. A Bluetooth technológiát eredetileg rövid kábelek helyettesítésére tervezték, de mára láthatóvá vált, hogy több alkalmazási területen is új lehetőségeket teremt. Segítségével nemcsak a számítógépet csatlakoztathatjuk a különböző egységekhez, mint

például a nyomtató, az egér vagy a monitor, hanem több más területen is az eszközök „intelligens” hálózatát alakíthatjuk ki a segítségével. A Bluetooth eszközök hatékonyan alkalmazhatóak például otthoni környezetben, ahol egy eszközzel, például a mobiltelefonnal irányítható a televízió, a mikrohullámú sütő vagy a hifi berendezés. Ugyanilyen típusú alkalmazás képzelhető el az autóban, ahol a vezető személyét érzékelve például az ülések és tükrök beállítódnak a megfelelő pozícióba. Nagyon érdekes felhasználási terület az elektronikus kereskedelem, hiszen itt a Bluetooth chipek újabb lépést jelenthetnek az elektronikus fizetés megoldása felé. Vásárláskor egyszerűen kiszámolná az összeget a bevásárlókosár, majd ezt az információt a „pénztárcánk” felé továbbítva a kasszánál már csak az összeg átutalását kellene engedélyeznünk.

Jelentős szerepet kaphat a Bluetooth olyan felhasználásoknál, amelyek például egy konferencia során kerülnek előtérbe. A konferencia résztvevői közül egyesek közvetlen módon kapcsolatba szeretnének lépni egy másik résztvevővel, míg egyesek a globális hálózathoz szeretnének csatlakozni. A Bluetooth mindkét csatlakozási mód megvalósításában szerepet kaphat.

A Bluetooth tehát egységes csatlakozási pontot jelent a különböző eszközök számára, amelyen keresztül képesek gyorsan kapcsolatot teremteni egymással. A Bluetooth eszközök kis mérete, alacsony ára és kis fogyasztása is elősegíti, hogy széles körben felhasználhatóak legyenek.

1.2. Ad hoc hálózatok

A Bluetooth technológia ad hoc hálózatok kialakítása során is jelentős szerephez juthat. A Bluetooth egyike lehet azon rádiós rendszereknek amelyek az ad hoc hálózatok fizikai rétegeként szolgálhatnak. Az ad hoc hálózatok kiépüléséhez nincs szükség egy előre telepített hálózati infrastruktúrára, mert az eszközök kapcsolódási pont nélkül alakítják ki egymás közt az összeköttetéseket. Az így létrejött hálózatban nincs kitüntetett szerepű eszköz, hiszen az eszközök fizikailag megegyező

felépítésűek. Az adatforgalom irányítását is maguk az eszközök végzik. Az IETF MANET [4] munkacsoportja az ad hoc hálózatokat érintő kérdésekkel foglalkozik. A munkacsoport az útvonalválasztó algoritmusok kidolgozására fókuszál. Ilyen útvonalválasztó algoritmusok a DSDV [5], DSR [6] vagy az AODV [7].

1.3. A dolgozat tartalma és felépítése

A Bluetooth szabvány lehetővé teszi egy nagyobb kommunikációs hálózat felépítését, de azt nem határozza meg, hogy az eszközök milyen módon építsék fel ezt a hálózatot úgy, hogy ez a topológia hatékony legyen. A dolgozatomban bemutatom a konzulensemmel közösen tervezett algoritmust, amely egy meglévő tetszőleges topológiát egy a kapacitás szempontjából előnyösebb hálózati topológiává alakít át. Az algoritmus tuladonságainak vizsgálatára ad hoc szimulációs környezetet hoztunk létre. A szimulátorban megvalósítottam az általunk tervezett algoritmust implementáló protokollt.

Dolgozatomban az alábbi fejezetek mutatják be munkánkat. Először röviden bemutatom a Bluetooth technológiát (2. fejezet), majd a 3. fejezetben rátérek a hálózatépítépítés problémájára. Ezután a 4. és 5. fejezetben bemutatom a fent említett algoritmust. A 6. fejezet ismerteti a PLASMA szimulációs környezetet különös figyelmet fordítva az ad hoc modulra. A 7. fejezet a kettőnk által tervezett algoritmus általam létrehozott implementációjának szimulációs kiértékelését tartalmazza. Végül bemutatjuk munkánk folytatásának lehetséges szempontjait és összegezzük dolgozatunkat.

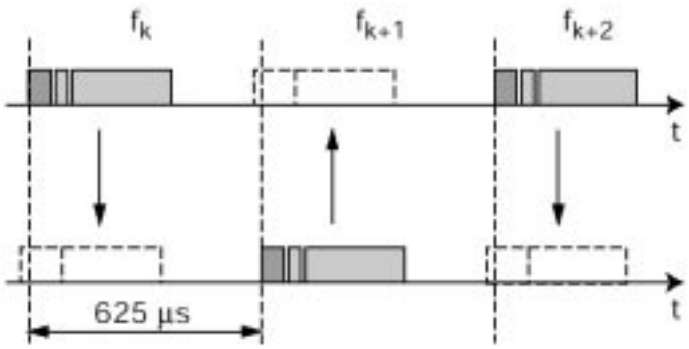
2. A Bluetooth vezetéknélküli technológia

A Bluetooth technológia a 2.4-2.5 GHz, úgynevezett *Industrial Scientific Medicine (ISM)* frekvenciasávban működik, ami frekvenciaengedély nélkül használható. Ezt a frekvenciasávot más rádiós technológiák és más eszközök is használják. Az így adódó zavarok hatásának csökkentése céljából a Bluetooth technológia a frekvenciaugrásos technikát (*Frequency Hopping Spread Spectrum, FHSS*) alkalmazza. A frekvenciaugrásos technika alkalmazásakor az információs egységeket időben váltakozva más-más frekvencián visszük át, így ha egy frekvencián zavar lép fel, akkor csak az adott egységnyi információ veszik el. A szabvány a rendelkezésre álló frekvenciasávot 79 kommunikációs csatornára osztja, amelyek között a frekvenciaugratás egy álvéletlen kódsorozat szerint történik .

A Bluetooth technológiában a kommunikáció master-slave viszonyra épül. A master illetve slave egy logikai állapot. Minden eszköz betöltheti mind az egyik, mind a másik szerepet. A half-duplex adatátvitel master és slave között időosztásos módon (TDD) valósul meg. A páros időrésekben a master→slave kommunikáció zajlik, míg a páratlan időrések a slave→master adatátvitelre szolgálnak. Minden egyes master→slave időrés végén az a slave adhat a következő slave→master időrésben, amelynek a master által küldött csomag szólt. Minden egyes időrés más frekvencián kerül átvitelre a frekvenciaugrási sorozatnak megfelelően (1. ábra). Az időrések 0.625 ms hosszúak, ebből 1600/s adódik a frekvenciaugrás sebességére.

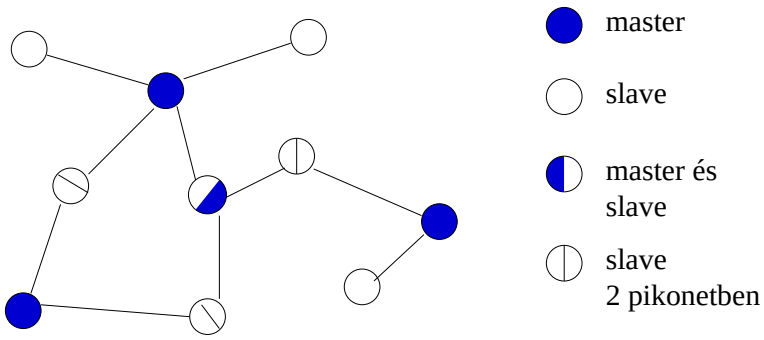
Egy master eszközhöz maximum 7 aktív slave csatlakozhat. A master és a hozzá kapcsolódó slave-ek együttesét *pikonet*nek nevezzük. A pikonetben a kommunikációt a master irányítja, ami azt jelenti, hogy ő határozza meg a frekvenciaugrás sorrendjét. A slave eszközök az ő órájára szinkronizálják a saját óráikat. Egy pikonetben mindig pontosan egy master van.

A pikonetek összekapcsolódásával egy nagyobb hálózat, *scatternet* (2. ábra) jöhet létre oly módon, hogy egy eszköz több pikonetben is részt vesz. A több pikonetben részt vevő eszköz időosztásos elven tartózkodik az egyik majd a másik pikonetben.



1. ábra. FHSS/TDD kommunikáció master és slave között

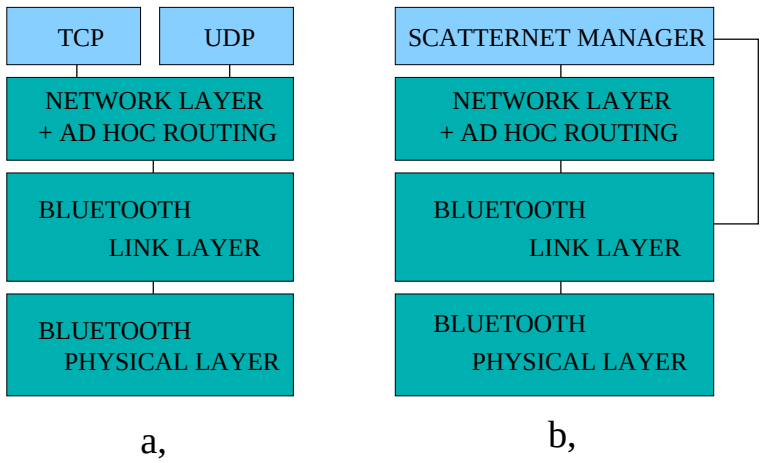
Annak az eszköznek, amely több pikonetben is részt vesz, a pikonetváltás során saját óráját szinkronizálnia kell annak a masternek az órájára, amelyik pikonetbe be akar lépni. Mivel a slave nem adhat akármikor egy pikonetben, ezért meg kell várnia a master engedélyét. Ha a pikonetben tartózkodása alatt esetleg nem kerül rá a sor, akkor feleslegesen tartózkodott a másik pikonetben. Mivel a neki szánt adatot nem sikerült megkapnia, így számára ez az idő kiesést jelent.



2. ábra. A scatternet felépítése

3. A Bluetooth hálózatéptés megoldandó problémái

A Bluetooth specifikáció lehetővé teszi, hogy a Bluetooth eszközökből hálózat épüljön fel, de nem definiálja, hogy milyen módon és milyen szempontok szerint. A Bluetooth technológiát azonban alapvetően nem hálózatok felépítésére tervezték, hanem pont-pont kapcsolatok biztonságos létrehozására. A Bluetooth hálózati alkalmazásokra való felhasználása új feladatok megoldását kívánja. Az alábbi ábrán a Bluetooth-ra épülő hálózati protokoll rétegeit ismertetjük.



3. ábra. A Bluetooth hálózati protokoll felépítése :
(a) felhasználói sík illetve (b) kontroll sík

A felhasználói sík (3a. ábra) felépítése mutatja, hogy a Bluetooth hálózaton a hálózati sík felett a megszokott forgalmakat visszük át. A nyitott kérdések a Bluetooth hálózat kontroll síkjával kapcsolatosak.

A kontroll síkban (3b. ábra) a hálózattal kapcsolatos karbantartás funkciókat a *scatternet manager* protokoll réteg valósítja meg. A scatternet manager feladatai közé tartozik a hálózat felépítése és esetleges módosítása. A scatternet manager gondoskodik arról, hogy egy új eszköz csatlakozni tudjon a hálózathoz illetve el tudja hagyni azt. Folyamatosan kezelnie kell a topológia változásait is, ami az eszközök mozgásából adódik. A scatternet manager megvalósíthat útvonalválasztást is az egyes pikonetek között. A scatternet manager feladata továbbá a masterhez

kapcsolódó slave eszközök kommunikációjának ütemezése. Az alábbiakban ezeket a feladatokat tekintjük át.

A hálózat topológiájának karbantartása

Az elsőként felmerülő kérdés, hogy hogyan szervezzük a scatternet hálózatunkat, vagyis, hogy mely Bluetooth eszközök között létesítsünk master-slave viszonyt, melyek legyenek a master-ek és melyek a slave-ek, hány pikonetet hozzunk létre. A [8] ezt a kérdést vizsgálja statisztikai, elméleti megközelítésből. Mindezen jellemzők helyes megválasztása azért fontos, mert nagyban meghatározza a scatternetben átvihető forgalom mennyiségét. A helytelenül kialakított scatternet többszörös hatékonyság csökkenést eredményezhet. A problémát nehezíti, hogy az adott körülmények között (adott forgalmi viszonyok, csomópontok elrendezése) hatékony scatternet egy másik helyzetben akár lehet a legkedvezőtlenebb is. Mindezekből következik, hogy a forgalom változásával, a felhasználók mozgásával folyamatosan kell a scatternet hálózatot karbantartani és az adott viszonyoknak megfelelően átszervezni. Az általunk tervezett algoritmus egy dinamikus scatternet optimalizáló eljárás, amely a master-slave viszonyok, Bluetooth kapcsolatok átrendezésével igyekszik a hálózatot a mindenkori körülményeknek megfelelően átrendezni, úgy hogy a hálózatban átvitt forgalom minél nagyobb lehessen. Az algoritmust részletesen az 5. fejezetben mutatjuk be.

Útvonalválasztási mechanizmus

A Bluetooth csomag 3 bit-es címmezője csak a master számára teszi lehetővé, hogy kijelölje a megfelelő slave-et, amelynek a csomag szól. A slave által küldött csomagban szükségtelen a címzettet megjelölni, hiszen az csakis a master lehet. Ebből a korlátozott címzési lehetőségből fakad, hogy Bluetooth csomagok nem továbbíthatóak több eszközön keresztül. Minden olyan esetben, amikor csomagokat több köztes csomóponton keresztül lehet csak a vevőhöz eljuttatni, az útvonalválasztás és csomagtovábbítás a felsőbb réte-

gekben kell megtörténjen. Az egyik kézenfekvő megoldás az ad hoc útvonalválasztó algorimust és ezáltal a scatternet szintű csomagtovábbítást az IP szinten megvalósítani. Útvonalválasztó algoritmusként működhet az IETF MANET munkacsoportja által kidolgozott ad hoc routing algoritmusok egyike. Ezek az algoritmusok azonban nincsenek a Bluetooth technológiára vannak optimalizálva, így kérdéses, hogyan működnek a Bluetooth környezetben.

Ütemezési feladatok

A pikoneten belül a master dönti el, hogy a következő slave→master időrésben melyik slave forgalmazhat. Az, hogy több slave esetén milyen módon állapítja meg a master a sorrendet, az meghatározó lehet a pikonet kapacitása szempontjából. A probléma megoldására egy ütemező algoritmust [9] kell alkalmazni (intra-piconet scheduling). A helyzet még bonyolultabb, ha több pikonet kapcsolódik össze és vannak olyan eszközök, amelyek több helyen is szerepelnek. Ekkor az ütemező algoritmusnak (inter-piconet scheduling) figyelembe kell venni, hogy az eszköz mikor tartózkodik az egyik pikonetben illetve mikor a másikban. Az eszköznek mindaddig várakoznia kell, míg meg nem kapja az adás jogát. Az ütemezés miatti időkiesés csökkenti az eszköz által átvihető forgalmat. Egy esetleges rossz ütemezés akár azt is okozhatja, hogy az eszköz nem tud adni egyik pikonetben sem, azaz az általa átvitt forgalom nullára csökken.

A fent említett problémák közül mi az elsőként említett problémával, a hálózat összeköttetéseinek kezelésével foglalkoztunk. Egy olyan algoritmust terveztünk, amely folyamatosan változtatja a hálózat felépítését annak érdekében, hogy a hálózat által átvihető forgalom a lehető legnagyobb legyen. Az algoritmus működésébe betekintést nyújt a következő fejezet. A 5. fejezetben részletesen bemutatjuk az algoritmus felépítését és működését.

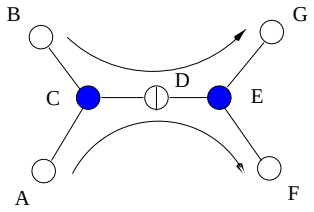
4. Egy példahálózat optimalizálása

Ebben a fejezetben egy példával illusztráljuk az hálózat otimalizáló algoritmus működését. A Bluetooth hálózatban nem elég, ha a hálózat felépítése során próbáljuk meg az új eszközöket a lehető legelőnyösebb módon csatlakoztatni a meglévő hálózathoz, hanem a forgalmi viszonyok változásából adódó igényeknek megfelelően a hálózatot periodikusan újra kell konfigurálni. Munkánk célja e feladat megoldása volt. Olyan algoritmust terveztünk, ami az egyes eszközök közti adatforgalom mérése alapján módosítja a hálózat felépítését, azaz dönt két eszköz között egy új kapcsolat felépítéséről, vagy egy kapcsolat lebontásáról. Ezen a két szabályon kívül bevezettünk egy a Bluetooth specifikációból adódó szabályt, nevezetesen, ha egy összeköttetésen mindkét eszköz úgy ítéli meg, akkor felcserélhetik a master-slave viszonyt.

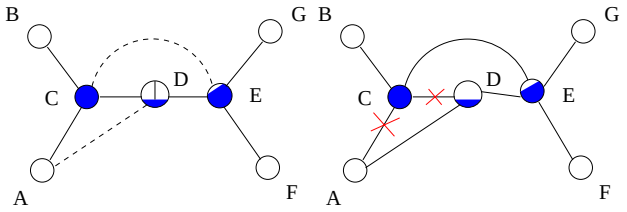
Az algoritmus tervezésekor peremfeltétel volt, hogy az algoritmus egyes lépései során az eszközök csak a szomszéd által küldött csomag információira és saját forgalmi méréseikre alapozva hozzák meg döntéseiket. Ez az elosztott módszer nem feltételezi egy központi irányító eszköz meglétét, ami robusztussá teszi az algoritmust. Az eszközöknek az összeköttetés módosító üzeneteken kívül nem kell más üzenetekkel terhelniük a hálózatot. Tovább egyszerűsíti az algoritmust, hogy egy esetleges csomagvesztés nem okoz nagy gondot, legfeljebb az adott hálózatmódosító lépés elmarad. Ez a tulajdonság lehetővé teszi, hogy a nyugtázással járó esetleg hosszadalmas folyamatot elkerüljük.

Az alábbi példa a szabályok működési mechanizmusát mutatja be .

A hálózat kezdeti felépítése során (4. ábra) A kommunikál F-el és B kommunikál G-vel. Mindkét forgalom több eszközön keresztül továbbítódik . C eszköz kérésére D egy új kapcsolatot épít ki A-hoz és ugyanezt történik D kérésére C és E eszköz között (4a. ábra). Ezen az új kapcsolaton C tölti be a master szerepét. Az A és C közti illetve a C és D közti kapcsolatokat lebontjuk (4b. ábra), mert a forgalom már az új összeköttetésen továbbítódik. Az összeköttetést felépítő szabály működésbe

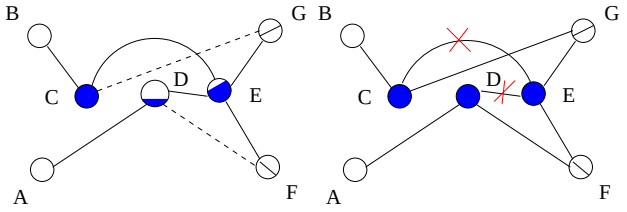


4. ábra. Egy egyszerű scattternet



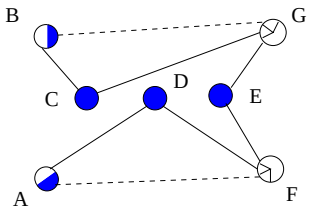
(a) Az első lépés (b) A második lépés

lép az E eszközben és a B és G közötti forgalom ezentúl az újonnan felépült C-G kapcsolatot használja (4c. ábra). Szintén E eszköz kérésére D is összeköttetést épít ki F-el. A forgalom kihasználja az új kapcsolatokat és a forgalom hiányában a C-E kapcsolat egyik végpontja úgy dönt, hogy lebontja a kapcsolatot (4d. ábra). Hasonló folyamat megy végbe D-E összeköttetés esetében. Az összeköttetést felépítő szabály

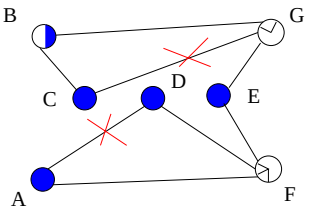


(c) A harmadik lépés (d) A negyedik lépés

hatására közvetlen kapcsolat épül fel A és F illetve B és G eszköz között (4e. ábra). Az A-D és C-G kapcsolat megszűnik (4f. ábra). Sorozatos master-slave szerepcserék során beáll a hálózat adott forgalmi tulajdonságoknál stabil állapota (4g. ábra). A szabályok működésének eredményeképp egy hatékonyabb topológiát kaptunk, mert

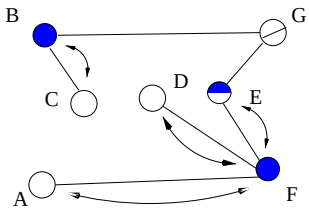


(e) Az ötödik lépés

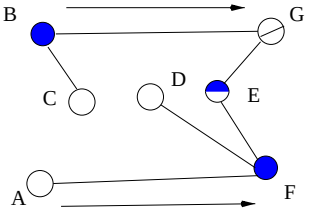


(f) A hatodik lépés

az A és F illetve B és G eszközök közti forgalom egy közvetlen kapcsolaton keresztül jut el a partnerhez (4h. ábra).



(g) Az utolsó lépés



(h) A stabil hálózati felépítés

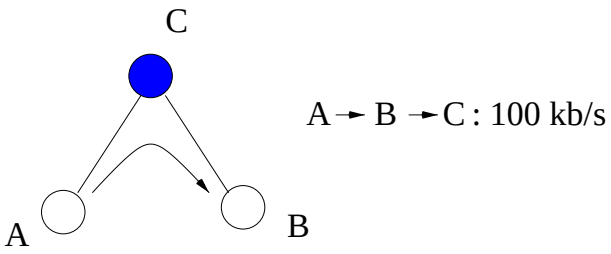
5. Forgalomfüggő hálózat optimalizáló eljárás

Ebben a fejezetben részletesen bemutatjuk az algoritmus és a hozzá kapcsolódó protokoll működését. A kapcsolatokat módosító szabályokat alkalmazva az eszközök üzenetekkel értesítik egymást arról, hogy számukra milyen előnyökkel vagy hátrányokkal jár a hálózat az adott szabály által végrehajtott módosítása (5.1. fejezet). Az összeköttetéseket és az egyes kapcsolatokhoz tartózó információkat minden eszköz tárolja a táblázataiban (5.2. fejezet). A táblázatok alapján bizonyos időközönként döntéseket hoznak arra vonakozólag, hogy végrehajtsanak-e egy topológia módosítást. A hálózat felépítése három szabály alkalmazásával változtatható meg. Felépíthetünk egy új kapcsolatot, vagy lebonthatunk egy meglévőt illetve megcserélhetjük a master-slave szerepeket. A 7. fejezetben azt vizsgáltuk meg, hogy a külön-külön alkalmazott szabályok együttesen előnyösebb topológiát hoznak-e létre. A hálózat optimalizáló algoritmust és az azt megvalósító protokollt mint egységes eljárást mutatjuk be az alábbiakban.

5.1. Az összeköttetések módosításához szükséges üzenetek

A kapcsolatok módosítását üzenetek segítségével hajtják végre az eszközök. Minden üzenet tartalmazza az üzenet típusát, a küldő (FROM) és a fogadó (TO) eszköz címét, így ezekre nem térünk ki minden üzenetnél. Az INFO mező tartalmazza a plusz információkat az adott szabállyal kapcsolatban. Ez a mező például kapcsolatfelépítés esetén hordozhatja azt az információt, hogy mikor történjen meg a fizikai kapcsolatfelépítési procedúra (paging). Ennek az információnak a birtokában a fizikai kapcsolatfelépítés gyorsabb lehet.

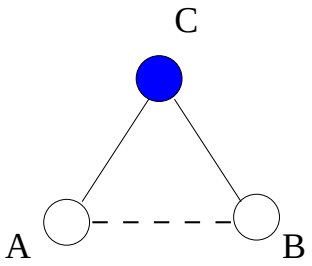
Példaként egy olyan egyszerű hálózat topológiájának módosítását mutatjuk be (5. ábra), amelyben a szabályok működése nyomon követhető.



5. ábra. Egy egyszerű pikonet

5.1.1. A kapcsolatfelépítés üzenetei

Ha egy eszköz (C) forgalmat továbbít két másik között (A,B) és ez a forgalom túllép egy meghatározott értéket, akkor a csomagtovábbító eszköz küld egy **LinkSetupSolicit** üzenetet közülük az egyiknek, például az A eszköznek (6. ábra).



6. ábra. Az összeköttetés felépítése

A **LinkSetupSolicit** üzenet BETWEEN mezője tartalmazza annak a két eszköznek a címét, akik között új kapcsolat kell hogy felépüljön. A WINNET által közölt forgalom jelzi, hogy a hálózat (C szerint) mennyi nyereségre tesz szert azzal, hogy az új összeköttetés felépül. Amint A megkapja ezt az üzenetet, eldönti, hogy számára mekkora plusz forgalmat jelent az új kapcsolat felépítése, hiszen ekkor újabb pikonet tagjává válik. Ha számára is nyereséges a kapcsolat felépítése, akkor küld egy **LinkSetupReq** üzenetet B-nek, amelybe beleírja a saját összes forgalmát (TRAFF) illetve a hálózat **LinkSetupSolicit** üzenetben kapott értékét (WINNET) továbbküldi. A M/S érték azt a szerepet jelzi, amely A számára előnyösebb lenne az új kapcsolatban. A WINMASTER és WINSLAVE mezőkben A elküldi, hogy mek-

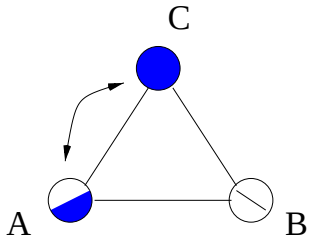
Sorrend a példában	Küldő	Vevő	Üzenet típusa	Paraméter neve	Paraméter értéke
1.	C	A	LinkSetupSolicit	BETWEEN WINNET	A-B 200
2.	A	B	LinkSetupReq	TRAFF M/S WINNET WINMASTER WINSLAVE	100 S 200 -50 -50
3.	B	A	LinkSetupAck	M/S INFO	S

1. táblázat. Az összeköttetés felépítés üzenetei

kora kapacitásváltozással jár számára a kapcsolat felépítése, ha master szerepet kap illetve ha slave lesz. Ha a **LinkSetupReq** megérkezése után B elfogadja az új kapcsolat felépítését, akkor válaszol egy **LinkSetupAck** üzenettel. Amennyiben nem fogadja el a kérést, akkor nem kell tennie semmit. A **LinkSetupAck** tartalmazza a M/S mezőt, amelyben B meghatározza A szerepét az új kapcsolatban.

5.1.2. Üzenetek a master-slave szerepcsere során

Ha az eszköz (A) slave szerepet tölt be egy kapcsolaton és számára előnyösebb volna, ha ő lenne a master, akkor ezt egy **LinkMSSwitchReq** üzenet küldésével jelzi a master eszköznek (C).



7. ábra. Master-slave szerepcsere

Az üzenet tartalmazza az eszköz forgalmát (TRAFF) és a szerepcsere által nyert forgalmi kapacitás értékét (WIN). Amikor C megkapja az üzenetet, akkor kiszámolja

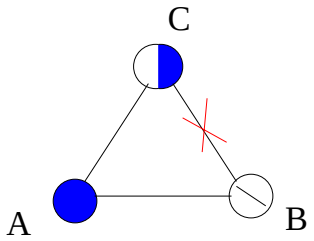
Sorrend a példában	Küldő	Vevő	Üzenet típusa	Paraméter neve	Paraméter értéke
1.	A	C	LinkMSSwitchReq	TRAFF WIN	100 50
2.	C	A	LinkMSSwitchAck	INFO	

2. táblázat. A master-slave szerepcsere üzenetei

saját nyereségét és ha számára is kedvező a csere, akkor válaszol egy **LinkMSSwitchAck** üzenettel. Ha számára nem kedvező a csere, akkor tétlen marad.

5.1.3. Az összeköttetés lebontás üzenetei

A forgalom átterelődik az új kapcsolatra, ezért B eszköz lebontásra ítéli a B-C összeköttetést.



8. ábra. Az összeköttetés lebontása

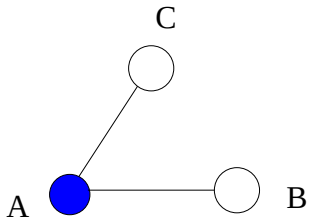
Sorrend a példában	Küldő	Vevő	Üzenet típusa	Paraméter neve	Paraméter értéke
1.	A	C	LinkTearDownReq	WIN	50
2.	C	A	LinkTearDownAck	INFO	

3. táblázat. Az összeköttetés lebontás üzenetei

Miután B kiszámolta, hogy kapacitása mennyivel növekszik (WIN) és hálózatnak a kapcsolat lebontásából származó plusz terhelését, elküldi a kapcsolatlebontás kérést (**LinkTearDownReq**) C eszköznek, amely üzenet tartalmazza B nyereségét.

C eszköz kiszámolja a saját nyereségét és a két érték alapján dönt, hogy elfogadja-e a kapcsolat lebontását. Amennyiben igen, úgy küld egy **LinkTearDownAck** üzenetet B-nek, ha nem, akkor nem tesz semmit.

Végeredményképpen egy kedvezőbb hálózati felépítés alakult ki, amelyen nagyobb mennyiségű adatot lehet forgalmazni (9. ábra).



9. ábra. Nagyobb hatékonyságú hálózat

5.2. Az összeköttetések állapotait tartalmazó táblázatok

Minden egyes eszközben két táblázat szolgál arra, hogy az összeköttetésekkel kapcsolatos információkat tárolja. A táblázatok tartalmazzák az eszköz által az egyes kapcsolatokon forgalmazott adatokat. A forgalomértékeket egy bizonyos időtartamra alkalmazott átlagolással számítjuk. Az átlagolás általunk alkalmazott módját a 6.2. fejezetben mutatjuk be részletesen.

5.2.1. Kapcsolatonkénti forgalom tábla

A kapcsolatonkénti forgalom táblában (4. táblázat) minden szomszédos eszközhöz tartozik egy sor. A táblázat olyan példaértékeket tartalmaz, amelyek nem kapcsolódnak az előzőekben megismert példahálózatokhoz, csak bemutatják a táblázat felépítését.

Az egyes sorok az adott szomszéddal felépített kapcsolat adatait tartalmazzák. A táblázatot nyilvántartó eszköz ezen információk alapján alkalmazza a master-slave szerepcserét végrehajtó és az összeköttetést lebontó szabályt.

Node	Role	Tr_{old}	Tr_{new}	T_{since}	$T_{teardown}$	$B_{teardown}$	T_{switch}	B_{switch}
B	M	160.23	84.567	85		1		1
C	S	40.1	214.76	5	3	2		1
D	M	12.43	47.3	17	3	8		1
E	IN	-	-					
F	S	190.568	35.77	21		1	3	2
G	OUT	-	-	63				

4. táblázat. Kapcsolatonkénti forgalom tábla

A táblázat első oszlopában az eszköz azonosítója található. A második oszlop megadja, hogy az eszköz milyen szerepet tölt be az adott kapcsolatban. *M* jelöli ha az eszköz master, *S* ha slave. Amennyiben a másik eszközzel már megszűnt a kapcsolat, akkor vagy belül van a rádiós tulajdonságokból adódó hatótávolságon (*IN*) vagy nincs (*OUT*). A harmadik és negyedik oszlop (Tr_{old} és Tr_{new}) az adott összeköttetésen zajló átlagolt illetve pillanatnyi forgalmat tartalmazza. Tr_{new} aktuális forgalom bizonyos időközönként hozzáadódik Tr_{old} értékhez. Az ötödik oszlopban a T_{since} időpont mutatja, hogy a létező kapcsolat mikor épült fel, vagy a lebontás mikor történt meg. Ezzel a bejegyzéssel megakadályozhatjuk, hogy egy kapcsolatot közvetlenül a létrejötte után lebontsunk vagy az épp lebontott kapcsolatot újra felépítsük. A $T_{teardown}$ és T_{switch} időpontok az utoljára elküldött kapcsolatlebontás- illetve master-slave szerepcsere kérés időpontját jelölik. A kéréseket egyre ritkábban küldjük, hogy ne terheljük feleslegesen a hálózatot. A kérések pillanatnyi sűrűségét a $B_{teardown}$ és B_{switch} értékek jeleznek (részletesen lásd. 5.3.fejezet).

5.2.2. Továbbított forgalom tábla

A továbbított forgalom táblázatban (5. táblázat) minden két eszköz közötti forgalomhoz egy bejegyzést rendelünk.

A *Between* mező a két eszköz címét tartalmazza. A Tr_{old} és Tr_{new} értékek hasonló szerepet játszanak, mint a kapcsolatonkénti forgalom táblázatban. A T_{setup} oszlop azt az időpontot tárolja, amikor az eszköz utoljára szólította fel a bejegyzéshez tar-

Between	Tr_{old}	Tr_{new}	T_{setup}	B_{setup}
B-C	194.35	32.65	25	2
C-E	32.55	223.83		1

5. táblázat. Továbbított forgalom tábla

tozó két eszköz egyikét a közvetlen kapcsolat felépítésére. B_{setup} a kapcsolatfelépítés gyakoriságát jelző érték.

5.3. A hálózat felépítését optimalizáló szabályok

A hálózat átkonfigurálását az alábbiakban részletezett három szabály látja el. A szabályok egymástól függetlenül hajtják végre a topológia módosítását. Az egyes szabályok periodikus időközönként megvizsgálnak egy összeköttetést és megpróbálják a kapcsolatot megváltoztatni. Ha ez a módosítás kedvező az eszköznek akkor az adott szabály alapján kezdeményezi azt. Ha nem, akkor legközelebb bizonyos idő múlva próbálkozik. Ezt az időt határozzák meg a B_{setup}, B_{switch} és $B_{teardown}$ (backoff counter) értékek. Ha a szabály sikertelenül próbálkozott, akkor növeli a megfelelő számláló értékét, hogy az általa generált kontroll üzenetek forgalma csökkenjen a sikertelen próbálkozások függvényében.

5.3.1. Az összeköttetést felépítő szabály

A kapcsolat felépítést az az eszköz kezdeményezi, amelyik forgalmat továbbít két másik között. Ezt a döntést a továbbított forgalom tábla alapján hozza meg az eszköz. Az algoritmus T_{setup} és B_{setup} értékeket veszi figyelembe. A $Tr_{minsetup}$ azt a minimális forgalmat jelöli, amelynél a kapcsolatfelépítés szabály életbe lép. P_{setup} az a minimális idő, aminek két üzenetküldés között el kell telnie. T_{down} jelöli a kivárási időt egy kapcsolatlebontás után, nehogy egy lebontott összeköttetést azonnal felépítsünk. T érték a pillanatnyi időt jelenti. A kapcsolat felépítésének kéréséről az alábbi módon dönt az algoritmus.

1. Ha két szomszéd között a forgalom kisebb, mint $Tr_{minsetup}$, akkor $B_{setup} = 1$ és állj.

2. Amennyiben $T_{setup} + P_{setup} \cdot B_{setup} <= t$, ami azt jelenti, hogy az ennek a párosnak küldött utolsó **LinkSetupSolicit** üzenet óta nem telt el a szükséges várakozási idő, akkor állj. Ha még nem küldtünk ilyen üzenetet, akkor T_{setup} legyen mínusz végtelen.

3. T_{setup} értéke legyen T és B_{setup} legyen $\min(B_{setup} * 2, B_{maxsetup})$. Küldjünk **LinkSetupSolicit** üzenetet a két forgalmazó eszköz közül az egyiknek. A WINNET értékbe a két eszköz között továbbított forgalom kétszeresét írjuk bele, mert az új összeköttetés a kezdeményező eszköznek két kapcsolatát tehermentesíti. Az iterációs ciklus legközelebb $T_{setup} \cdot B_{setup}$ idő múlva lép életbe.

Ha a forgalom egyik végpontja megkapja a **LinkSetupSolicit** üzenetet, akkor először ellenőrzi, hogy letelt-e T_{down} lebontás utáni kivárási idő. Ha nem telt le, akkor az eszköz nem tesz semmit. Amennyiben ez az idő letelt, akkor az eszköz kiszámolja, hogy mennyivel csökkenne a kapacitása, ha új kapcsolatot építene fel. Ezt az értéket meghatározza mind master (WINMASTER), mind slave (WINSLAVE) szerep esetére (5.1. alfejezet). Ha
WINNET + WINMASTER < 0 és WINNET + WINSLAVE < 0
teljesül, akkor az eszköznek nem előnyös az összeköttetés felépítése, és nem tesz semmit. Ellenkező esetben küld egy **LinkSetupReq** üzenetet annak az eszköznek, amelyikel kommunikál.

Ha a másik eszköz megkapja az üzenetet, akkor kiszámolja saját veszteségét mind master, mind slave esetre. Ha
GAINSLAVE = WINNET + WINMASTER(A) + WINSLAVE(B)
érték pozitív, akkor visszaküld egy **LinkSetupAck** üzenetet, ahol a M/S mezőt S értékre állítja. Ha
GAINMASTER = WINNET + WINSLAVE(A) + WINMASTER(B)

érték lesz pozitív, akkor a M/S mezőt M-ra állítja. Ha mindkét összeg pozitív, akkor a master eszköz kiválasztásához az alábbi két szabály alkalmazható.

- Válasszuk ki a nagyobb értéket GAINSLAVE és GAINMASTER közül, és a nagyobb érték szerint legyen ez az eszköz slave vagy master, a másik szerepet pedig közölje a másik eszközzel.
- Ha az eszközök közül az egyik már master egy más kapcsolatban, akkor legyen ő az új master. Ha mindketten rendelkeznek master szereppel, vagy egyikük sem, akkor nagyobb forgalom értékű eszköz legyen a master.

5.3.2. Master-slave szerepcsere szabálya

A master-slave szerepcserét mindig a slave eszköz kezdeményezi, mivel a master nem növeli a kapacitását a cserével. A döntést az eszköz a kapcsolatonkénti forgalom tábla alapján hozza meg. Az algoritmus a T_{switch} időt használja, mint az utolsó elküldött kérés idejét, P_{switch} értéket, amely két üzenetküldés közti kivárási időt jelzi és B_{switch} értéket, amely a szerepcsere kérésének gyakoriságát határozza meg. T a pillanatni időt jelzi. Az algoritmus lépései a következők:

1. Számoljuk ki a nyereség becsült értékét (WINS). Ha WINS < 0, akkor állj.
2. Ha $T_{switch} + P_{switch} \cdot B_{switch} <= t$, ami azt jelenti, hogy a P_{switch} által jelzett kivárási idő még nem telt le, akkor állj. Ha még nem küldtünk üzenetet, akkor T_{switch} legyen mínusz végtelen.
3. T_{switch} legyen T és B_{switch} legyen $\min(B_{switch} \cdot 2, B_{maxswitch})$. Küldjünk egy **LinkMSSwitchReq** üzenetet az összeköttetés másik oldalán lévő eszköznek. Az algoritmus leközelebb $T_{switch} \cdot B_{switch}$ idő múlva próbálkozzon szerepcserével.

Ha a master megkapja a **LinkMSSwitchReq** üzenetet, akkor szintén meghatározza a nyereségét (WINM). Ha WINS + WINM < 0, akkor állj. Ha az érték épp nulla, akkor fogadjuk el a kérést, ha a szomszéd forgalma nagyobb. Küldjünk egy **LinkMSSwitchAck** üzenetet válaszként.

5.3.3. Az összeköttetést lebontó szabály

A kapcsolatebontást szintén a kapcsolatonkénti forgalom tábla alapján döntjük el. $T_{teardown}$ jelöli az utoljára elküldött lebontáskérés időpontját, míg $B_{teardown}$ az üzenetküldés gyakoriságát. $P_{teardown}$ a lebontási kérések közötti kivárási idő. T_{up} időnek kell eltelnie az összeköttetés felépítése után, hogy le lehessen bontani. T az aktuális időt jelöli.

1. Számoljuk ki a kapcsolat lebontása által nyert kapacitás értékét és a hálózat veszteségét, amit az által szenved el, hogy az eddig egy kapcsolaten továbbított forgalmat hosszabb útra tereljük. A hálózat plusz terhelése:

$$WINNET = -L \cdot Tr$$

   A képletben szerepel az alternatív út hossza (L) és a lebontandó kapcsolat forgalma. Az alternatív út hosszának kettőt feltételezünk, mivel ez a minimális érték. Ha WIN + WINNET < 0, akkor állj.

2. Ha $T_{teardown} + P_{teardown} \cdot B_{teardown} <= t$, akkor még nem telt le a minimális idő két próbálozás között. Ez esetben állj. Ha még nem küldtünk lebontást kérő üzenetet, akkor $T_{teardown}$ értékét végtelennek feltételezzük. Ha $T_{since} + P_{up} <= t$, akkor csak nemrég épült fel a kapcsolat, állj.

3. Állítsuk a $T_{teardown}$ értékét T-re és $B_{teardown}$ számlálót $\min(B_{teardown} \cdot 2, B_{maxteardown})$ értékre. Küldjünk **LinkTearDownReq** üzenetet a szomszédnak. A szabály legközelebb $T_{teardown} \cdot B_{teardown}$ idő múlva lép életbe.

Ha a szomszéd megkapja a **LinkTearDownReq** üzenetet, akkor először megvizsgálja, hogy létezik-e alternatív út a szomszédhoz. Ha nem létezik, akkor állj. Erre a feltételre azért van szükség, nehogy egy kapcsolatlebontással a hálózat több részre szakadjon. Az útkeresés után már ismerjük az alternatív út hosszát (L), amit

beszorozva a kapcsolat forgalmával megkapjuk a hálózat kapacitásának csökkenését (WINNET). Ezt követően kiszámoljuk a saját nyereségünket és hozzáadjuk a szomszéd nyereségét és a hálózat veszteségét. Ha az összeg negatív, akkor állj. Ha az összeg nemnegatív, akkor küldjünk egy **LinkTearDownAck** üzenetet.

6. Ad hoc szimulátor a PLASMA környezetben

Munkánk során a szimulációs környezetben vizsgáltuk a hálózat optimalizáló algoritmus tulajdonságait, mivel a jelenleg elérhető Bluetooth eszközök még nem nyújtják az általunk feltételezett funkciókat, amelyek egy Bluetooth hálózat építéséhez szükségesek. A másik indok az volt, hogy szimuláció alkalmazásával tetszőleges nagyságú hálózatot alakíthatunk ki, míg a fizikai tesztelési megoldásban a rendelkezésre álló eszközök száma korlátozza a kiépíthető hálózat nagyságát.

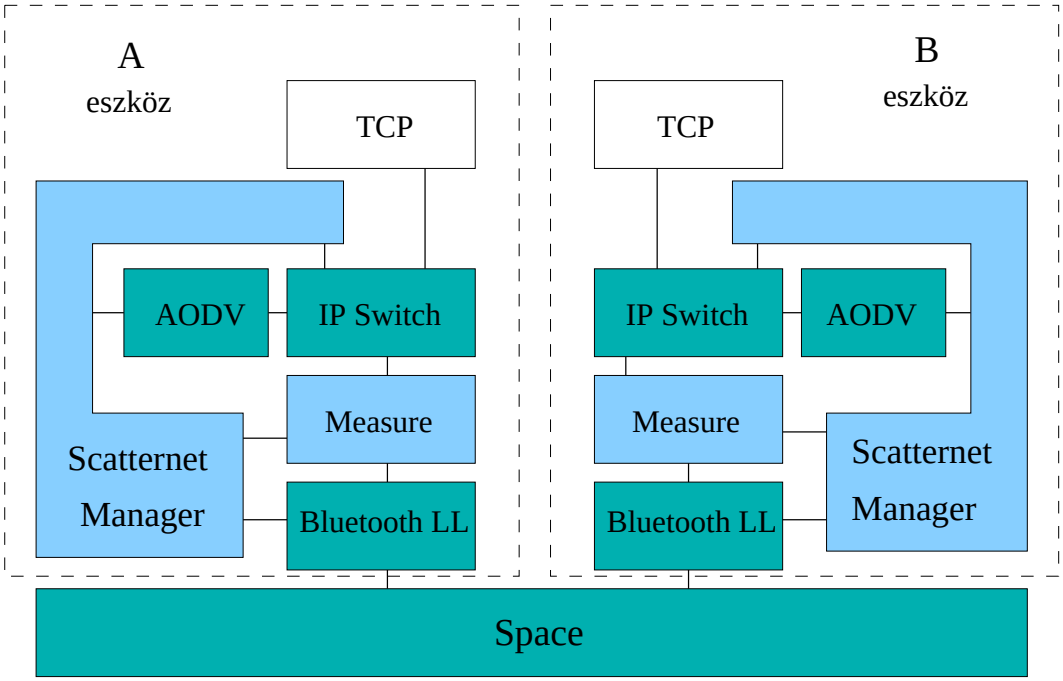
6.1. Az ad hoc modul általános bemutatása

Az általunk tervezett algoritmus vizsgálatát az Ericsson Traffic Lab. által kifejlesztett PLASMA hálózati szimulátor programmal végeztük el. A szimulátor moduláris szerkezetű, azaz külön modul valósítja meg az alapfunkciókat, a TCP/IP protokollt vagy a WWW forgalmat jelentő felhasználót. A PLASMA eddig nem tartalmazta az ad hoc hálózatokat megvalósító modult, ezért a TCP/IP protokollt megvalósító modult kiegészítettük egy ad hoc hálózati és azon belül Bluetooth technológiát megvalósító modullal.

Az ad hoc modul a Bluetooth technológia csomag szintű szimulációját valósítja meg, és tartalmazza a Bluetooth specifikációban [1] meghatározott jellemzőket, mint az időrések felépítése és a csomagok típusa, vagy az ARQ mechanizmus. A modul külön-külön objektumban valósítja meg a Bluetooth hálózat létrehozásakor felmerülő feladatokat. Ilyen feladatok a hálózat létrehozása és a topológia karbantartása vagy az ütemezés az egyes eszközök között. A csomagok továbbítását az AODV [7] ad hoc routing algoritmus végzi az IP réteg információinak segítségével. A fizikai réteg egy analitikus modellt alkalmaz a frekvenciaugrásból adódó ütközések és csomagvesztés meghatározására, amely figyelembe veszi az interferáló pikonetek számát és azok forgalmát.

Az ad hoc modul felépítését az alábbi ábra segítségével mutatjuk be.

A fehér színnel jelölt TCP objektum a szimulátorban már korábban meglévő ob-



10. ábra. Az ad hoc modul a PLASMA hálózat szimulátorban

jektumot jelöli, amelyet változtatás nélkül lehetett hozzákapcsolni az ad-hoc modul IP réteghez (az ábrán sötétebb színnel jelölve).

Az IpSwitch a hálózati rétegben az IP funkciókat valósítja meg. Az IpSwitch objektumot módosítottuk, hogy csatlakoztatni lehessen hozzá a MANET által kidolgozott AODV ad hoc routing protokollt megvalósító objektumot. A csomagtovábbítás csomagonként zajlik, azaz az IpSwitch minden egyes csomagot átad az AODV-nek, amely amennyiben nem ismeri a célhoz vezető utat, akkor megkeresi azt, és átadja a csomagot a célhoz vezető úton következő eszköznek. Ha az AODV objektum ismeri az utat, akkor a cache-ből válaszol az útvonalkeresési kérésre.

A Bluetooth LL objektum a Bluetooth kapcsolati réteget foglalja magába. Ebben az objektumban valósítottuk meg az olyan Bluetooth funkciókat, mint a kapcsolatok felépítése és bontása fizikai szinten (page, detach [1]). Az objektum magában foglal egy round-robin működési elvű pikoneten belüli ütemezési algoritmust illetve egy egyszerű pikonetek közti ütemező algoritmust. A Bluetooth LL objektum gon-

doskodik az IP csomagok Bluetooth csomagokká való átalakításáról. Az objektum támogatja a specifikáció által meghatározott csomagtípusokat.

A Space objektumban tároljuk a fizikai réteghez kapcsolódó információkat. A Space objetum figyelembe veszi a rádiócsatorna által okozott csomagvesztést is. Ezen veszteség hatása azonban nem számottevő a scatternet kialakítása során fellépő egyéb veszteségekhez képest, mint például a pikonetek közti csomagtovábbítás, vagy az ütemezés [10, 11]. A Space tartja számon az egyes eszközök fizikai távolságát is, amelyet a mobilitás modell határoz meg. Az egymas jelenlétét felfedező inquiry procedúrát úgy modelleztük, hogy az adott eszköz Bluetooth LL objektuma kérdezi meg a Space objektumtól a tartózkodási helyét és a látótávolságban lévő szomszédokat.

A Measure és Scatternet Manager objektum (az ábrán világos színnel jelölve) szorosan összetartoznak, mivel a Measure objektum méri az eszköznél végződő illetve az eszközön áthaladó forgalmat és ezt a forgalommérést használja fel a Scatternet Manager arra, hogy kitöltse az összeköttetésekhez tartozó táblázatokat (5.2. fejezet). A Scatternet Manager tartalmazza a 5.3. fejezetben részletesen bemutatott hálózat optimalizáló protokollt. A Scatternet Manager kapcsolatban van a Bluetooth LL objektummal is, mert így közli az adatkapcsolati réteggel, ha a fizikai topológia módosítására van szükség. Implementációnkban a Scatternet Manager kapcsolódik az IpSwitch-hez is, mivel a protokoll megkívánja, hogy két nem szomszédos Bluetooth egység Scatternet Manager-e is tudjon kommunikálni, ami Bluetooth szinten nem megoldható. Így a scatternet átkonfiguráló protokoll üzeneteket IP csomagokban küldjük el. Ezáltal elérjük azt is, hogy a protokoll üzenetek a normál adatforgalommal együtt kerülnek átvitelre, így hűen modellezzük ezen csomagok átvitelét. Az AODV objektumhoz azért csatlakozik a Scatternet Manager, mert az útvonalválasztó algoritmust értesítenie kell a kapcsolatai módosulásáról. Ez akkor történik, amikor az eszköz egy új kapcsolatot épít fel vagy egy meglévő kapcsolatot bont le. Az AODV ekkor az új kapcsolatnak megfelelően módosítja az útvonalválasztási bejegyzéseit.

6.2. A hálózat optimalizáló algoritmus implementációja

A konzulensemmel közösen tervezett hálózat optimalizáló algoritmust két külön objektumban valósítottam meg. Különválasztottam az algoritmus forgalmat mérő részét, ezt a Measure objektum tartalmazza, míg a Scatternet Manager objektum a mért forgalom alapján hajtja végre a hálózatmódosító szabályokat.

A Measure objektum feljegyzi az eszköz által elküldött vagy kapott csomagok hosszát és erről értesíti a Scatternet Managert. A Scatternet Manager a csomag hosszával növeli az adott összeköttetésen mért forgalom értékét. Az aktuálisan mért fogalom értékeket minden egyes kapcsolathoz tartozó bejegyzés esetén T_{check} időközönként hozzáadjuk az átlagolt forgalom értékhez. Az összeköttetéseket módosító szabályok az átlagolt forgalomértékek alapján hozzák meg döntéseiket. Ez azt jelenti, hogy a T_{check} érték meghatározza, hogy az algoritmus mennyire friss információkkal rendelkezik a forgalmat illetően. Ez az érték befolyásolja az algoritmus forgalomváltozásra való reagálásának sebességét.

A forgalmi értékek átlagolását az alábbi képlet szerint végzi az algoritmus.

$$Tr_{old} = P \cdot Tr_{old} + (1 - P) \cdot \frac{Tr_{new}}{t}$$

A Tr_{old} és Tr_{new} változók az átlagolt illetve az aktuális forgalmat jelzik. A P paraméter azt adja meg, hogy az átlagolás során milyen súllyal esnek latba a régebbi forgalmi értékek. Gyakorlatilag ez az érték határozza meg az algorimus „emlékezetét”. Ha az érték közel 1, akkor az átlagba nagyobb súllyal számítanak a korábbi forgalmi értékek („hosszú időre emlékezik”). Ha P közel 0 értékű, akkor jelentősebb az aktuális forgalom („gyorsan felejt”). Az aktuális időt a t változó jelzi.

Az algoritmus reagálásának gyorsaságát a T_{check} értéken kívül befolyásolja a P átlagolási tényező is. A fenti képlet egy általunk választott átlagolási megoldás, amelynek hatása független kell legyen az algoritmus alaptulajdonságaitól. Méréseink során a P paraméter értékét közel nullának választottuk, hogy az algoritmus az aktuális forgalmi értékek alapján döntsön.

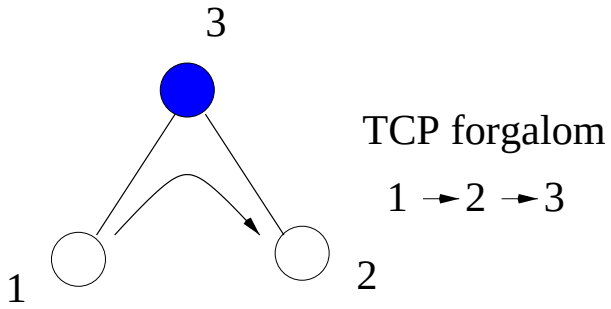
Az algoritmus a topológiát módosító szabályait külön-külön alkalmazza az egyes összeköttetésekre. A kapcsolatokat bizonyos időközönként újra megpróbálja módosítani az algoritmus az adott szabály szerint. Ezt az időtartamot a $T_{execution}$ értéke határozza meg. Implementációnkban ez az érték mindhárom szabályra vonatkozik, azonban az egyes szabályokra külön-külön is meghatározható az újra próbálkozás idejének az értéke.

Az új összeköttetés felépítéséhez kapcsolódik a $Tr_{minsetup}$ érték, amely megadja azt a forgalomértéket, ami felett az összeköttetést felépítő szabály működésbe lép. A $Tr_{minsetup}$-ban meghatározott forgalom felett új összeköttetés, azaz új pikonet épül fel.

A pikonetek közti váltás és ütemezés miatt az eszköz által átvitt forgalom csökken (bridging overhead). A forgalomkiesés értékének 50 kb/s állandó értéket tételeztünk fel. A $B_{overhead}$ értékét meghatározhatjuk bonolultabban is, hiszen figyelembe vehetjük a környező pikonetek forgalmát vagy azt, hogy az eszköz mennyi pikonetben vesz részt jelenleg, mennyire hatékony az ütemező algortimus, stb. A $B_{overhead}$ viszonylag pontos ismerete fontos az algoritmus működése szempontjából. Ha ugyanis az algoritmus $B_{overhead}$ értékét a valóságosnál nagyobbra állítja, akkor az eszköz úgy ítéli majd meg, hogy számára nagy kapacitáscsökkenést jelent egy új pikonetbe való belépés. A magas érték beállításával tehát az eszköz kevésbé reagál az összeköttetést felépítő kérésekre. Ha $B_{overhead}$ értékét alábecsüli az algoritmus, akkor kis forgalom esetén is új kapcsolatok felépítését fogja kezdeményezni, amely eredményeképpen rendkívül gyakran épül fel új kapcsolat. A $B_{overhead}$ értékével tehát szabályozni tudjuk, hogy mennyi pikonet alakuljon ki.

7. A protokoll működésének vizsgálata

Az általunk tervezett algoritmus müködésének hatékonysága a 6.2. fejezetben ismertetett paraméterek beállításától függ. Ebben a fejezetben bemutatjuk, hogy az algoritmus és az általam készített implementáció milyen módon módosítja a hálózat felépítését és ismertetjük néhány paraméter hatását az átkonfigurálás folyamatára. A szimuláció során megvizsgáltuk, hogy egy adott forgalom mellett az algoritmus milyen gyorsan konfigurálja át a Bluetooth hálózatot egy kezdeti, kedvezőtlen topológiából kiindulva. A folyamatos kommunikációt a két eszköz közötti TCP forgalom jelentette. A TCP forrás egy elméletileg végtelen hosszú file átvitelt hajtott végre, követve természetesen a TCP protokoll jól ismert szabályrendszerét. A TCP kezdeti tranziens hatásának kiküszöbölése érdekében a szabályok alkalmazása minden esetben 60s-nál kezdődött. A szimulációs mérések során kiindulásként a legegyszerűbb olyan esetet használtuk, ahol az átkonfigurálás jelentős kapacitásváltozásokat okozhat. Ez az eset az elméleti ismertető során megismert példahálózat átkonfigurálását jelenti (11. ábra). Az algoritmus a 5. fejezetben megismert előnyösebb topológiát hozta létre, amint azt az átvitt forgalom növekedése is jelzi a 12. és 14. ábrán.

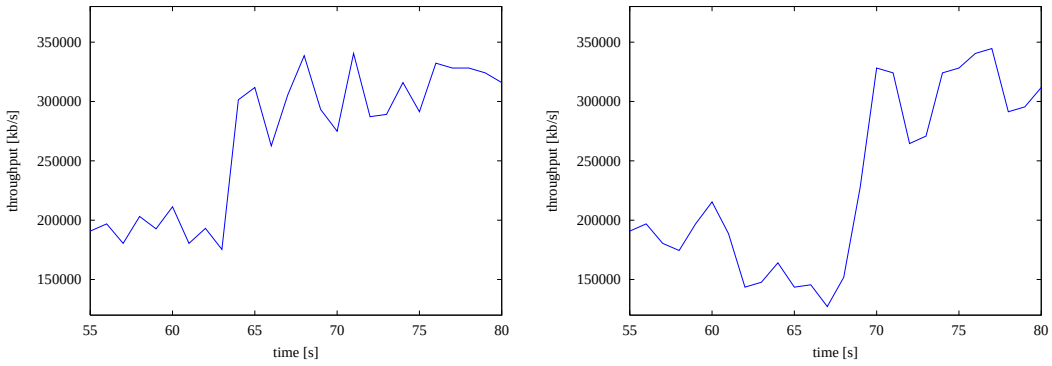


11. ábra. A mérési konfiguráció

7.1. A T_{check} értékének hatása

Az elsőként vizsgált T_{check} paraméter határozza meg, hogy a folyamatos mérés során milyen gyakorisággal frissítjük a forgalmat nyilvántartó táblázatok értékeit. Ha

gyakran frissítjük, akkor az algoritmus gyorsabban reagál a forgalom változására, ami gyorsan változó hálózatban indokolt. Az időkonstans értékét akkorára kell választani, amekkora időskálájú forgalomváltozást követni akarunk a topológia módosításával. Ha a paraméter értékét nagyra választjuk, akkor az algoritmus reackióideje megnő, viszont ezzel csökkentjük a szükséges számolási feladatok gyakoriságát, ami kíméli a mobil eszközben az erőforrások használatát. A 12. ábrán látható a szimulált hálózat átvitele $T_{check} = 1$s érték esetén (a,), ami a forgalomértékek gyakori aktualizálását jelzi illetve $T_{check} = 4$s értéknél (b,), ami ritka ellenőrzést jelent. A T_{check} alapbeállítása 1s volt. Az (a,) esetben a hálózat viszonylag rövid idő alatt (3s) átrendeződött, míg a lassúbb beállításnál ez majdnem a dupláját vette igénybe (7s). Az első esetben az időegységre jutó számítási kapacitás nagyobb volt, de a számítás miatti erőforrásfelhasználás olyan kicsi, hogy a különbség elhanyagolható.



(a) $T_{check} = 1s$ (b) $T_{check} = 4s$

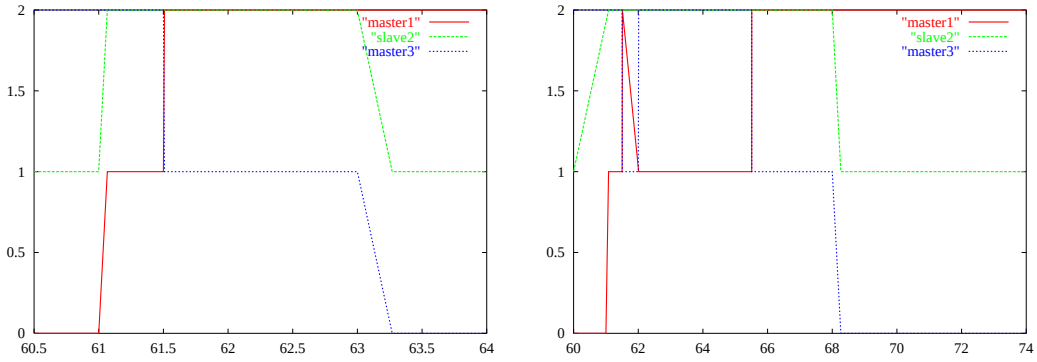
12. ábra. A T_{check} érték hatása a kapacitásnövekedésre

A 13. ábrán a hálózat felépítésének folyamata látható, ahol a példahálózat eszközeinek szerepeit ábrázoltuk az idő függvényében. Az ábrán az 1 és a 3 eszköz master szerepszámának változását mutatjuk meg illetve a 2 eszköz slave szerepszámának változását. A szerepek számainak változásából kiolvasható, hogy az algoritmus melyik szabályt hajtotta végre.

A 13a. ábrán 61 s-nél *master1* és *slave2* értékének egyidejű növekedése azt mu-

tatja, hogy 1 és 2 eszköz között felépült egy új összeköttetés. 61.5s-nál *master1* értékének növekedése és *master3* csökkenése master-slave szerepcserét jelez 1 és 3 eszköz között. Végül 63s időpontban *master3* és *slave2* egyidejű csökkenése mutatja, hogy 2 és 3 eszköz között megszűnt az összeköttetés. Hasonlóan a többi ábrán is végigkövethető az algoritmus helyes működése. A többi ábra esetében a részleteket mellőzve mutatjuk be az algoritmus működését.

A 13b. ábrán 61s-nál felépül az összeköttetés 1 és 2 között, aztán 61.5s-nál 1 eszköz szerepet cserél a 3-hoz tartozó kapcsolatán, de ez a két eszköz számára még nem előnyös, ezért azonnal visszacserélik a szerepeket. Majd csak 5s múlva válik számukra előnyössé a szerepcsere. A hálózat felépítésének módosítását a 2-3 összeköttetés lebomlása zárja. A szabályok működésének kezdete óta 7s telt el.



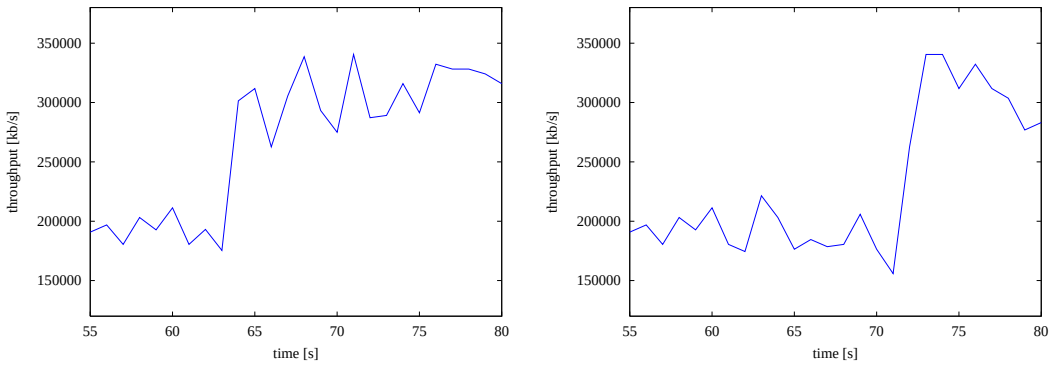
(a) $T_{check} = 1s$ (b) $T_{check} = 4s$

13. ábra. A hálózat kapcsolatainak változása T_{check} változásakor

7.2. A $T_{execution}$ értékének hatása

A második paraméter, ami az algoritmus reakciós képességét meghatározza, az a szabályok végrehajtásának gyakorisága. Ezt a mi protokoll implementációnkban a $T_{execution}$ érték határozza meg. A szabályok közül a leghosszabb időt a kapcsolatot felépítő szabály végrehajtása veszi igénybe, mivel több üzenetet kell a hálózaton

célba juttania. Az szabály alkalmazási idejének periódusát a forgalmi viszonyok határozzák meg. Méréseink során a szabályok közül a „leglassabb”, az összeköttetést felépítő szabály végrehajtása is befejeződött 0.5 s alatt, így $T_{execution}$ értékét egészen 1s-ig csökkentettük. Ha a forgalmi viszonyok ezt nem teszik lehetővé, például a hálózatban valahol torlódás miatt várakozni kell a csomagoknak, akkor ez az érték nem csökkenthető ilyen mértékben. A $T_{execution}$ értékének választás szorosan összefügg a T_{check} érték választásával, hiszen nem érdemes alacsony $T_{execution}$ értéket választani, ha az algoritmus ritkán frissíti a forgalmi adatokat, mert új információk híján a szabályok ugyanazokat a döntéseket hozzák meg. Érdemes ezért a két időváltozó értékét hasonlóra választani. A 14. ábra a $T_{execution}$ paraméter beállításától függően mutatja be az átkonfigurálási idő változását. A T_{check} értéket mindkét ábrán 1s-ra választottuk. Az ábra (a,) részében $T_{execution}$ értékét 1s-ban határoztuk meg. Ez rendkívül gyors (3s) átkonfigurálást tett lehetővé. Amikor $T_{execution}$ értéke 4s volt (b,), akkor a topológia módosításának ideje 11s lett. A nagy érték azzal magyarázható, hogy a szabályok nagy időközönként követték egymást, hiszen a 11s érték több szabály végrehajtási idejét takarja.

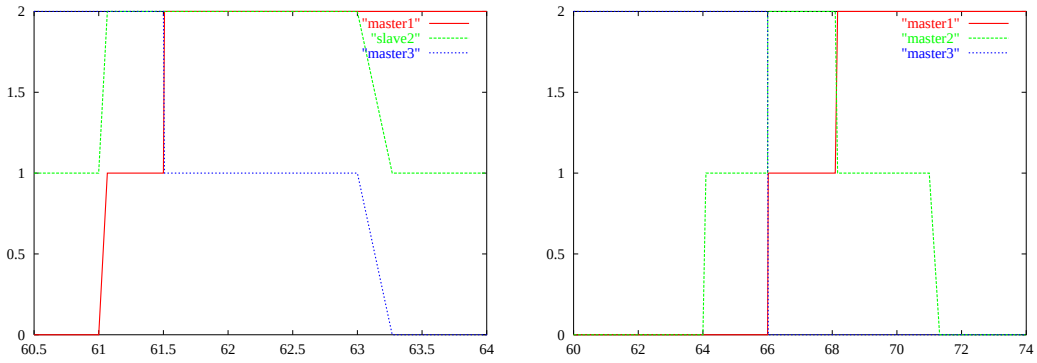


(a) $T_{execution} = 1s$ (b) $T_{execution} = 4s$

14. ábra. A $T_{execution}$ érték hatása a kapacitásnövekedésre

A 15a. ábra példája megegyezik a 13a. ábrán bemutatott alapbeállítások mellett végzett szimulációval. A 15b. ábra $T_{execution}$ értékének nagy beállításánál mutatja

az algoritmus működését. Az első linkfelépítés a nagy időállandó miatt csak 64s-nál kezdődik. 66s-nál mind a 1-es mind a 2-es eszköz szerepet cserél 3-al, aki eddig mindkettő számára master volt. 68s-nál 1 és 2 eszköz is felcseréli a master-slave viszonyt. Végül 71s-nál lebomlik a 2 és 3 eszköz közti kapcsolat és beáll a stabil topológia.



(a) $T_{check} = 1s$ (b) $T_{check} = 4s$

15. ábra. A hálózat kapcsolatainak változása T_{check} változásakor

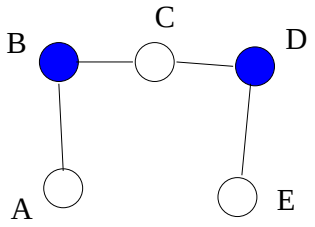
8. Jövőbeni tervek

Munkánkat a témán belül két területen folytatjuk tovább. Egyrészt újabb szimulációkat végzünk el az elkészült szimulátor segítségével, másrészt az eredmények alapján továbbfejlesztjük a tervezett algoritmust.

A szimuláció terén méréseket végzünk gyorsan változó forgalom esetére, ahol a forgalom változékonyságának függvényében megvizsgáljuk a paraméterek beállításait. A vizsgált hálózatok méretét növelve hosszabb ideig tart a hálózat átkonfigurálása és ez a beállítások módosítását követelheti. Méréseket végzünk azzal kapcsolatban, hogy milyen alsó határa van az algoritmus reakcióképességének, vagyis milyen gyorsan változtathatjuk a körülményeket úgy, hogy az algoritmus még képes legyen követni őket. Ennek érdekében megtervezzük az eszközök bonyolultabb mobilitás modelljét és az eszközök mobilitásának függvényében vizsgáljuk az algoritmus teljesítőképességét.

A szimuláció alapján vizsgáljuk, hogy az algoritmus mely tulajdonságai javíthatóak illetve milyen új tulajdonságokat tudunk megvalósítani. Néhány megoldás, amelyektől az algoritmus tulajdonságainak javulását várjuk:

- Az összeköttetés felépítés kérése ne csak szomszédos három eszköz között történhessen meg, hanem hosszabb úton is (16).



16. ábra. Összeköttetés felépítése több eszközön keresztül

Az ábrán A eszköz csak akkor létesíthet közvetlen kapcsolatot E eszközzel, ha például látja C eszközt, ugyanis ez egy lehetséges első lépés az E-hez való kapcsolatfelépítés során. Ha ez nem teljesül, akkor A és E között esetleg

nem épülhet ki kapcsolat. Ezt csak úgy lehet megoldani, hogy a hálózatban folyamatosan kéréseket továbbítunk, ami nem hasznos terhelést jelent.

- Az algoritmus a forgalom mérések és változások alapján adaptívan változtathatja a paraméterek beállítási értékeit. Például egy olyan hálózatban, ahol a forgalom gyorsan változik a paramétereket úgy módosítaná, hogy gyorsan reagáljon változásokra, míg egy ritkán változó topológiában a paraméterek értékeit nagyra választaná, így kevesebb kontroll üzenettel terhelve a hálózatot. Ez a paraméter módosítás hosszabb ideig tartó mérések alapján következhet be.

- Amikor az eszközök a kapacitás nyereségüket számolják, akkor több szempontot is figyelembe vehetnek. A master-slave viszonyokon kívül beszámíthatják az erőforrások állapotát vagy a forgalmak prioritását (QoS).

- Ha az eszköznek vannak szabad erőforrásai, akkor elfogadhat olyan kérést, amely számára előnytelenebb helyzetbe juttatja, remélve, hogy ez a döntés a jelenlegi topológiát egy globálisan kedvezőbb felé mozdítja.

9. Összegzés

A Bluetooth hálózat optimalizálására egy forgalommérések alapján működő algoritmust terveztünk. Ezzel olyan problémára nyújtottunk megoldást, amelyhez az irodalomban eddig nem volt ismert megoldás. Az algoritmus egyszerű szabályok segítségével - kapcsolat felépítése és bontása illetve master-slave szerepcsere - lokális információk alapján egy előnyösebb hálózati topológiát hoz létre. Az algoritmus megvalósítására protokollt terveztünk, amely tulajdonságait szimulációs környezetben vizsgáltuk. A Bluetooth technológia vizsgálatára egy ad hoc modult alkottunk meg a PLASMA hálózat szimulátorban. A szimuláció során megmutattuk, hogy a paraméterek beállításától függően az algoritmus alkalmazkodik a forgalmi viszonyok változásához.

Hivatkozások

[1] Bluetooth baseband specification version 1.0 B, http://www.bluetooth.com/

[2] Jaap Haartsen, „Bluetooth - The universal radio interface for ad hoc, wireless connectivity", *Ericsson Review No.3*, 1998, pp. 110-117

[3] Gy. Miklós, „Bluetooth: Olcsó, drótnélküli helyi összeköttetés", Magyar Távközlés, 2000. szeptember

[4] Mobile Ad-hoc Networks Working Group of IETF, *http://www.ietf.org/html.charters/manet-charter.html*

[5] C. Perkins and P Bhagwat, „Highly dynamic destination-sequenced distance vector routing (DSDV) for mobile computers", in *ACM SIGCOMM*, October 1994

[6] J. Broch, D. B. Johnson, D. A. Maltz, „The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks", *IETF Internet Draft draft-ietf-manet-dsr-01.txt*, December 1998

[7] C. E. Perkins and E. M. Royer, „Ad-hoc On-Demand Distance Vector Routing", *Proceedings of 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[8] Gy. Miklós, A. Rácz, Z. Turányi, A. Valkó, P. Johansson, „Performance Aspects of Bluetooth Scatternet Formation", INFOCOM 2001, beadva

[9] N. Johansson, U. Körner, P. Johansson, „Performance Evaluation of Scheduling Algorithms for Bluetooth," *Proceedings of IFIP TC6 Fifth International Conference on Broadband Communications'99*, Hong-Kong, November 10-12, 1999.

[10] P. Johansson, N. Johansson, U. Körner, J. Elg, G. Svennarp, „Short Range Radio Based Ad-hoc Networking: Performance and Properties," *Proceedings of ICC'99*, Vancouver, 1999.

[11] S. Zürbes, W. Stahl, K. Matheus, J. Haartsen, „Radio Network Performance of Bluetooth," *to appear in the Proceedings of ICC'2000*, New Orleans, 18-22 June, 2000.