HADMÉRNÖK

IX. Évfolyam 2. szám - 2014. június

NAGY Dániel
nagy.daniel@operculum.hu

VEZETÉK NÉLKÜLI SZENZORHÁLÓZAT SZIMULÁCIÓ

Absztrakt

A vezeték nélküli szenzorhálózatok sajátosságai speciális problémákat vetnek fel a fejlesztés és a finomhangolás terén. A kisméretű, nagyszámú node-ok célterületre történő kijuttatása után azokat begyűjteni a szenzorok méretétől és mennyiségétől függően körülményes vagy lehetetlen feladat. Nagy hangsúly tevődik tehát a hálózat működésének számítógépes szimulációjára. Több erre a célra kifejlesztett szoftver áll rendelkezésre. Jelen cikk e szoftverek működésének alapjait tárgyalja, illetve a vizsgálati feladathoz legjobban illő szimulációs szoftver kiválasztásához ad útmutatást.

Certain characteristics of wireless sensor networks bring up special problems in the development and fine tuning process of these systems. After deploying the small and numerous nodes to the target area, it is hard or impossible to gather these devices for further adjustments. This leads to a great importance for simulating the operation of these systems during the development. There are several software available for this task. The publication at hand discusses the basic principles of these software, and gives guidance for picking the one which serves the purposes of the project best.

Kulcsszavak: *szenzorhálózat, szimuláció ~ sensor network, simulation*

BEVEZETÉS

A vezeték nélküli szenzorhálózatok (Wireless Sensor Networks) az elmúlt évtizedek elektronikai és informatikai robbanásának szülöttei csakúgy, mint a globális helymeghatározás, mobilkommunikáció vagy maga az internet. Jelentősége természetesen nem összevethető a felsorolt három nagy vívmánnyal, ám az élet számos területén találkozunk velük, és minden bizonnyal a jövőben ez egyre gyakrabban így lesz. Ezek a rendszerek a haditechnika szülöttei, ma is aktívan használják őket felderítési célokra.

A szenzorhálózatok node-okból, más néven mote-okból állnak. Ezek kisméretű, kis teljesítményű számítógépek, amelyek egymással – tipikusan rádiós kapcsolaton keresztül – kommunikálni képesek. Az eszköz tartalmaz valamilyen szenzort vagy szenzorokat is, melynek segítségével adatokat gyűjt a környezetéből. Működését telep biztosítja, esetleg egyes esetekben a környezetből is képes energiát felvenni, tipikusan napelem segítségével. A node-ok ma cipős- vagy gyufásdoboz méretűek, azonban a fejlesztések már lényegesen kisebb eszközöket is lehetővé tesznek. A SmartDust[1] fantázianevű készülékek milliméteres nagyságrendbe eső méretükkel a fentebb felsorolt összes komponenst tartalmazhatják, és a miniatürizálás folyamata minden bizonnyal még ennél is elképesztőbb mérettartományt hoz a jövőben.

Ha a kis méret olcsó gyártási költséggel párosul, ezek a szenzorok több száz, de akár több ezer példányban is a célterületre juttathatók, ahol automatikusan megkezdik egymással összehangolt tevékenységüket. Adatokat gyűjtenek, amelyeket tipikusan egy úgynevezett gateway (átjáró) node-on keresztül juttatnak el a központba.

A szenzorhálózatok fejlesztése és finomhangolása során, ha azok katonai felhasználásúak, és különösen, ha olyan alkalmazásról beszélünk, amelyben nagyszámú szenzorról van szó, a szimuláció egyenesen elengedhetetlen fejlesztési eszköz. Ennek számos oka van. Egyfelől a telepített szenzorok (manuálisan kihelyezett, repülőgépről kiszórt, tüzérséggel átlőtt) telepítés utáni begyűjtése gyakorlatilag lehetetlen feladat. Ha a szenzorhálózatban szisztematikus hiba (tervezésből adódó) van, a szenzorhálózat nem fog működni. Nem éles, de azzal megegyező környezetet teremteni és ott tesztelni, ugyanúgy rendkívül költséges vagy lehetetlen a node-ok számából adódóan. Tesztelés alatt elsősorban nem a node-ok egyéni működését és megbízhatóságát kell érteni. Könnyen tesztelhető, hogy egy szenzor kibírja-e a telepítés jelentette mechanikus igénybevételt, képes-e ezután automatikusan üzemi állapotba kapcsolni, képes-e ellenállni az előrelátható időjárási és egyéb környezeti behatásoknak. A szóban forgó, nehezen vagy nem tesztelhető üzemelési kockázatok elsősorban a node-ok együttműködéséből adódnak.

Az együttműködés, azaz a node-ok kommunikációjának modellezése rendkívül komplex feladat lehet. Egyfelől a sok node között létrejövő számos lehetséges útvonal már önmagában egy igen nagy komplexitású rendszert jelent, de a jelek fizikai terjedésének sajátosságai legalább akkora, ha nem nagyobb kérdés a modellezés során.

SZIMULÁCIÓS SZOFTVER KIVÁLASZTÁSI SZEMPONTOK

Ahogyan fentebb írtam, szenzorhálózat fejlesztés során a szimuláció elengedhetetlen. Ehhez a feladathoz az Interneten számos szimulációs eszközt találunk. Jelen publikációban áttekintem a szenzorhálózat szimulációs szoftverekkel foglalkozó eddig megjelent írások egy részét, majd saját kutatásom és szempontrendszerem szerint vonok le következtetéseket a véleményem szerint releváns tényezők tekintetében.

[1]Okos Por. Rendkívül kisméretű, milliméter nagyságrendbe eső szenzorhálózat, vagy egyéb kisméretű eszközök

Absztrakciós szint

A szenzorhálózat szimulátornak, vagyis valójában az éppen terítéken lévő szimulációs feladatnak, az egyik legfontosabb tényezője, a szimuláció absztrakciós szintje. Más szavakkal fogalmazva „mennyire legyen részletes" a rendszer modellje. „Minél részletesebb, annál jobb" – gondolnánk elsőre, ezzel egydimenziós skálára téve a szimulátorokat, ám így túlegyszerűsítenénk a kérdést. Egy kevéssé részletes, az alsóbb szintek működését elnagyoló, leegyszerűsítő szimulátor célravezető lehet, ha magas szintű protokollokat szeretnénk tesztelni. Ilyen esetben egy nagyon részletes, alacsony absztrakciójú rendszer felesleges faktorokat visz be a vizsgálódás folyamatába. Ez nem csak felesleges munkát, időt, hibalehetőséget és számítógépes erőforrást jelent, hanem konkrétan rosszabb eredményt is hozhat. Erre lentebb példát mutatok. Másfelől ha a fizikai réteg, azaz a rádió működése, és média-hozzáférési protokoll a fejlesztés célterülete, éppen ezek a részletek válnak fontossá, ezen számítások nélkül csak nagyon általános, közelítőnek sem nevezhető eredményt kapunk. Mivel a szenzorhálózatok szimulációjával kapcsolatos kérdések gyakorta nagy mennyiségű node-ot feltételeznek, a legtöbb esetben a részletesség a skálázhatóság rovására megy, a szimulátort futtató számítógép- vagy gépek véges erőforrásai miatt.

A teljesen valóságos szimuláció nem lehetséges és nem is szükséges. Valahol meg kell húzni a szimuláció absztrakciós szintjét, és ez a döntés alapvetően befolyásolja a szimuláció kimenetelét és a felhasznált szimulációs szoftver kiválasztását is. Fontos tekintetbe venni, hogy szenzorhálózatok esetében sokszor az OSI[2] rétegek nem különülnek el olyan mértékben, ahogyan azt szokásos vezetékes hálózatoknál megszoktuk [1]. Ez azt eredményezi, hogy egy mélyen részletezett média-hozzáférési protokoll magával vonzhat egyéb tényezőket is, mint például ennek megfelelően elemeire bontott jelterjedési modellt is. Ennek viszont már csak akkor lehet értelme, ha maga a környezet is kellő részletességgel modellezve van.

A kérdést tehát úgy kell feltenni, hogy melyek azok a tényezők, amelyek bizonyosan nem befolyásolják a szimuláció sikerét, hasznosságát. Ez persze nem minden esetben dönthető el előzetes vizsgálódások nélkül. A választottnál nagyobb mértékű részletesség változtatna-e a szimuláció kimenetelén? Érdemes-e időt és erőforrást ölni egy jelterjedési modell kifejlesztésébe, amely az adott célhoz illik vagy sem? Általában célszerű a szimuláció szintjét olyan magasra (kevéssé részletesre) célozni, amennyire csak lehetséges [1]. És nem pedig fordítva. Olyan alacsonyra célozni, amit még a rendelkezésre álló számítási kapacitással és egyéb erőforrásokkal teljesíteni lehetséges. A fentieket az alábbi két példával szeretném alátámasztani.

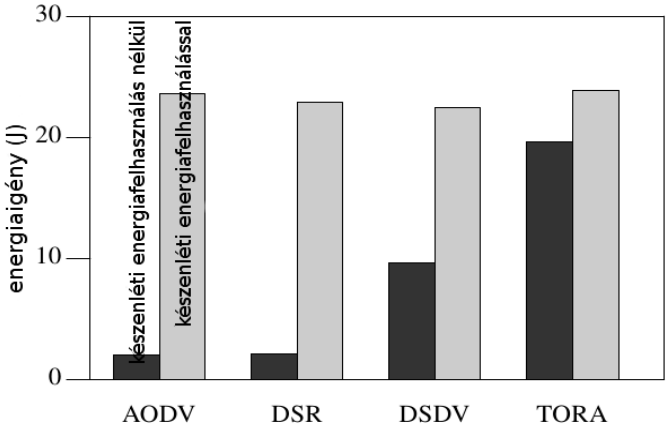
Routing protokoll energiaigény vizsgálat

Heidemann at.al.[1] érdekes javaslatot mutat az absztrakciós szint helyes meghatározásának fontosságára négy ad-hoc hálózatban alkalmazott routing protokoll energiaigényének esettanulmányán keresztül . A felsorolt protokollok mind vezeték nélküli rendszerekben használatos routing (útvonalválasztó) protokollok. A node-ok rádiójának energiaigénye többféleképpen (különböző részletességi szinteken) modellezhető:

- legegyszerűbben: a sikeresen fogadott vagy elküldött csomag energiafogyasztás egységet jelent;
- média-hozzáférési aktivitások is tekintetbe vehetők, azaz hibajavítások, újraküldések, kapcsolat felépítések;
- a rádióhoz kapcsolódó, és annak közvetett energiaigényét is tekintetbe vevő számítások.

[2] Open Systems Interconnection – Nyílt rendszerek összekapcsolása

A fenti három pont példaként mutat részleteiben növekvő tendenciájú szimulációt. Heidemann at.al.[1] megmutatta, hogy az alábbi négy vizsgált protokoll esetében éppen a készenléti idők energia felhasználása közötti különbség vált meghatározó faktorrá.



1. ábra. Négy routing protokoll energiaigénye. A sötéttel jelzet értékek a készenléti energia felhasználás nélküli, míg a világosabbal jelzett érték a készenléti energia felhasználást is magában foglalja. [1]

Ahogyan az 1. ábra mutatja, az AODV[4] és DSR[5] protokollok energia felhasználása lényegesen alacsonyabb az ábrán jobbra látható DSDV[6] és TORA[7] protokollokhoz képest. Ezek a különbségek azonban szinte eltűnnek, ha egy részletesebb energia modellt alkalmazunk, amely tekintetbe veszi a készenléti energia felhasználást is.

A fenti példa az én értelmezésben érdekesebb annál, mint amit Heidemann at.al. [1] következtetésként leszűrt. Nem csak arról van szó, hogy ebben az esetben a modellnek magában kell foglalnia a routing protokoll készenléti idejének energia igényét is. Mivel a készenléti állapot tekintetbe vételével az energiafogyasztások közel azonosak, így amennyiben a cél csak a protokollok összevetése, nem tévedünk sokat, ha valamilyen közelítő állandóval vezetjük be őket a modellbe.

Véleményem szerint a következtetés igen érzékletes példája a szimuláció-részletesség meghatározásának fontosságára. Ebben az esetben arról van szó, hogy amennyiben az energiafelhasználásban nem modellezzük a készenléti időt, akkor különböző protokollok esetén eltérő (hibás) adatokhoz jutunk. Tehát az absztrakció szintjét csökkenteni kell. Ugyanakkor felismerhető, hogy a megnövelt részletesség igen primitív összefüggést mutat, hiszen a szumma energiafelhasználás közel azonos. Vagyis éppen egy, a kiindulásnál primitívebb modellel is kielégítő eredményhez juthatunk. Így nem csak a lehető legmagasabb absztrakciós szintre kell törekednünk, de célszerűnek látom a megfelelő szint elérését magasról-alacsony (kevéssé részletestől a részletes felé) irányúnak választani és nem pedig fordítva. A részletesség szintjének és a várható eredmények hasznosságának összefüggését nem monoton függvény írja le.

Helymeghatározás példa

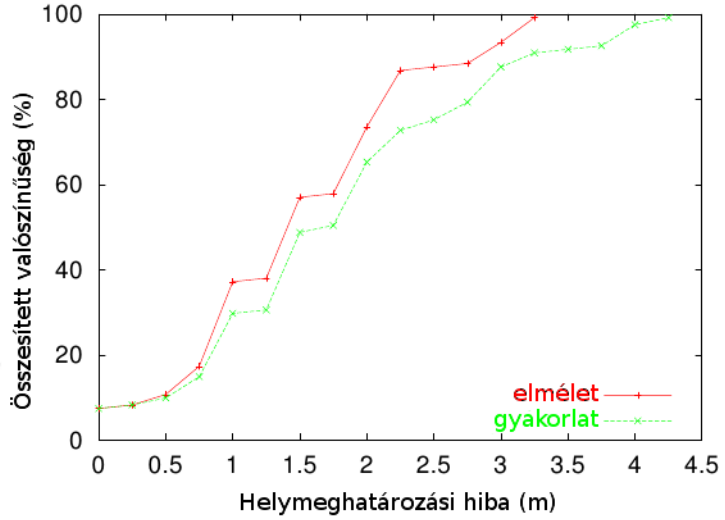
Nirupama at.al.[2] kísérletében node-ok egymáshoz viszonyított távolságát mérte, GPS nélkül, kizárólag az alapján, hogy egy adott node milyen erősségű jelet vett a többi node irányából. Ennek modellezésére ismét felállíthatunk néhány részletességi szintet:

[4]Ad hoc On-Demand Distance Vector – Alkalmi igényalapú távolságvektoros útvonalválasztás
[5]Dynamic Source Routing – Dinamikus forrásútvonal választás
[6]Destination-Sequenced Distance Vector routing – Célállomás sorrendű távolságvektoros útvonalválasztás
[7]Temporally-ordered routing algorithm – Átmeneti rendezettségű útvonalválasztó algoritmus

– az egyszerű modell csupán a két node távolságát veszi figyelembe, a jelerősség csökkentése csak egy egyszerű egyenlettel van modellezve;
– valamivel bonyolultabb modellek megkülönböztethetnek közelebbi és távolabbi node-okat, hogy a rálátásból és a visszaverődésekből kialakuló jelerősségeket modellezzék;
– statisztikai alapon árnyékolás modell is alkotható;
– nagy objektumok által okozott jel-elnyelődések és visszaverődések;
– egy nagyon részletes modell pedig olyan tényezőket is figyelembe vehet, mint antenna geometria (merre néz, milyen messze van a földtől), és akár hullám-követés szimulációval pontos elnyelődés és reflexió modellt alkothat, és az interferencia jelenségeket is figyelembe veheti.

Nirupama at.al.[2] kísérletében node-ok helyét határozta meg kizárólag rádió segítségével, néhány ismert referencia node alkalmazásával. Első lépésben egy rendkívül egyszerű modellt választott, ideális rádióval és gömb alakú jelterjedéssel, azzal a szándékkal, hogy majd ennek a szimulációnak a tanulságán továbbfejleszti a modellt. A szimulációt szembeállította valós körülmények között, valós node-okkal végzett kísérletekkel. (Egy parkolóban helyezett el néhány referencia node-ot, és azt, amelyik ezek jele alapján a relatív helyzetét meghatározta.) Tapasztalataik alapján ilyen környezetben a primitív modell is meglepően jó eredményt hozott. Az eredményeket a 2. ábra szemlélteti. Ebben az esetben bár a vizsgált tulajdonság azt sejtette, hogy részletes rádiómodellre lesz szükség, valójában a célt nagyon egyszerű modellel is szimulálni lehetett.

Ez tehát egy másik példa arra az alapvetésre vonatkoztatva, hogy a szimuláció absztrakcióját magastól-lefelé kell felépíteni és finomítani, még akkor is, ha az intuíció bizonyos helyzetekben ezt túlzónak ítélné meg.



2. ábra. A referencia-kísérlet és az egyszerű modellezett helymeghatározási hibája [2]

Program architektúra

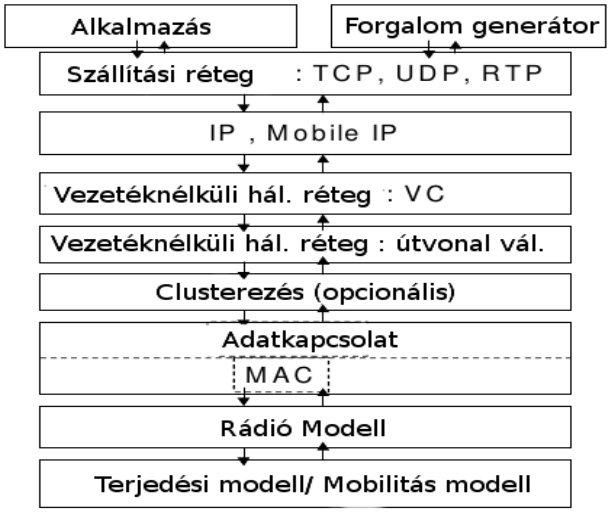
A szimulációs szoftver architektúrája, vagyis a rendszer kialakítására használt paradigma nagymértékben meghatározza a skálázhatóságot, azaz annak minőségét, hogy a szimulátorunk milyen teljesítményre képes kevés (<10), sok (~500) vagy nagyon sok (>5000) node esetén.

Az objektum orientált (OO) programszerkezet magától értetődőnek látszik szenzorhálózat szimuláció programozáshoz, hiszen ezt a valóságot igen egyszerűen képezhetjük le OO paradigmára. Kézenfekvő a node-ot egy osztályként megalkotni, amely magában foglal más osztályokat, mint a telepet, rádiót, mikrokontrollert szenzort és így tovább. Nem véletlen, hogy

zömmel a nagy teljesítményű, de OO paradigmára is képes C++ -t találjuk, ha a szimulációs szoftverek programozási nyelveire tekintünk.

David[3] azonban leírja, hogy éppen ez a programszerkezet teszi a szimulátort rosszul skálázhatóvá. Nagyszámú node szimulálása esetén exponenciálisan megnő a futtatókörnyezet erőforrás igénye. Ha minden node egy objektum a programban, amely más objektumokból épül fel, akárcsak a valóságban, a node-ok számával négyzetesen növekszik az azok interakcióját vizsgáló kódrészek hívásainak száma is.

E probléma kiküszöbölésére egyes szimulációs szoftverek, például a GloMoSim ugyan OO paradigmát, de célszerűen eltérő objektum-szerkezetet használ, a fentebb említett exponenciális összefüggés orvoslására. A GloMoSim architektúrájában egy objektum egy szintnek felel meg a hálózati kommunikációban, és az futtatja az összes node kódját [3]. Ezt komponens-alapú modellnek nevezhetjük.



3.ábra. A GloMoSim komponens-alapú architektúrája [3]

A komponens-alapú architektúrák lényegesen jobban skálázhatók, mint OO társaik, azonban nehezebb jól fejleszthető, moduláris rendszert készíteni [3].

Szimulátor vagy emulátor

A szenzorhálózatok modellezésére használatos szoftverek két nagy csoportba sorolhatók: a szimulátorok és emulátorok. Az emulátorok a valós hardvert implementálják programkóddal vagy akár valós node-ok bevonásával. Ilyen esetben, az emulátorban a valós hardver firmware-je és a valós hardverben futtatandó kód kerül. Az emulátorok alkalmasak inkább az alacsony, hardver-közeli működés vizsgálatára [4]. A szimulátor egy absztraktabb megközelítés, a hangsúly inkább azon vagy, hogy a rendszer 'mit csinál', nem azon, hogy 'hogyan.' Szimulátor tehát inkább egy fentről-lefelé nézetet ad a rendszerről, elsősorban a kommunikációs protokollok, topológia és adatgyűjtés vizsgálatára használatos [4].

Popularitás

A szimulációs szoftver kiválasztása tehát egy fontos lépés, és az is lehet, hogy ugyanazon rendszert akár több szimulációs szoftverrel is vizsgálnunk kell attól függően, hogy a rendszer milyen tulajdonságának tesztelése a szimuláció célja. Ezt a célszerűség orientáltságú kiválasztást hard-tényezőnek nevezem. Létezik azonban egy másik megközelítés is, amely legalább ennyire fontos, és soft-tényezőnek fogom nevezni. Ez pedig nem más, minthogy a kiszemelt szimulációs szoftver mennyire elterjedt, mióta készül, mennyi tapasztalat áll rendelkezésre. Lehet egy szimulátor ígéretes, funkcióleírásában céljainkhoz illő, összehasonlító

táblázatban egy másik elébe sorolt, azonban ez semmiképp sem garancia, hogy az egész szimulációs folyamat és az azon alapuló fejlesztést valóban legjobban szolgálja. Annyi szimulációs szoftver áll rendelkezésre, hogy azokat kipróbálgatni, valamelyikkel egy irányba elindulni, hogy aztán esetleg a folyamat derekán kiderüljön, hogy valami miatt mégsem alkalmas a céljainkra, kockázatos dolog. Nevezhetjük ezt megbízhatóságnak is, amelyet jobb híján a popularitáson keresztül tudhatunk meghatározni.

Ha hivatkozásokban fellelhető szimulációs szoftverek popularitását nézzük, lényegesen árnyaltabb képet kapunk arról, hogy melyikkel érdemes komoly munkába kezdeni. Az 1. táblázatban feltüntettem az általam eddig számba vett szimulációs szoftverek népszerűségét. A népszerűség mérésére egy egyszerű módszert választottam. A nevére rákerestem a Google keresőjével, és a találatok számát vettem a népszerűség mutatójának. Azért, hogy a listát ne torzíthassa az, hogy az elnevezés mennyire egyedi illetve esetleg más területen is előforduló szóról van szó, a keresési sztringhez hozzáillesztettem a „wireless sensor network" szókapcsolatot is. Így a végleges kereső kifejezés ekképp festett: „*név*" „wireless sensor network", ahol *név* az éppen vizsgált szoftver neve. Az idézőjel itt azért fontos, mert csak így keres pontos egyezésre a Google. Ha a név mezőnél például a SimX keresésekor elhagytam az idézőjelet, 283000 találatot kaptam, mert a Google egyenértékűnek vette a sims szóval. Idézőjel használata esetén azonban mindössze 62 találat született. Mivel az idézőjel nagyon szigorúvá teszi a keresést, a J-Sim és JSim írásmódok is különbözőek lettek. Ilyen esetekben több lehetséges írásmóddal is végrehajtottam a keresést és az eredményeket nagyvonalúan átlagoltam.

Ez a módszer sem ad teljesen megbízható eredményt, de mindenesetre egy irányvonalat biztosan. Továbbá elmondható, hogy bár a nagy számok nem feltétlenül jelentenek nagy popularitást (nem kizárható, hogy mégis összekeverte a kereső valamivel), a kis számok minden bizonnyal alacsony popularitást jelentenek. Magyarul felfelé lehet téves a táblázatban szereplő számok, de lefelé aligha.

Név	Találat (db)	Kapcsolat	Használat**
SENSE	460000*	COST DES alapú [3]	na.
NS2	120000	REAL alapú	42.80%
OPNET	74000	Eredetileg katonai célú [5]	7.60%
Avrora	48700	-	
OMNET++	38000	-	0.80%
NS3	30000	NS2 alapú, de nem kompatibilis vele	na.
GloMoSim	27000	-	1.60%
QualNet	24100	GloMoSim utódja	4.20%
TOSSIM	21800	-	na.
Prowler	9500	-	na.
WSNet	7850	-	na.
Castalia	7700	OMNET++ alapú [6]	na.
COOJA	4600	-	na.
ATEMU	4500	-	na.
SensorSim	3600	NS2 alapú [3]	na.

*az érték bizonyosan irreális, mert a szoftver neve egy értelmes szó

**[7] eredményei szerint a felhasználók 36,8%-a Matlabot használ a szimulációra

1. táblázat. Szimulációs szoftverek népszerűsége3

[3]2013-as saját statisztika, illetve Nurul I. Sarkar és Syafnidar A. Halim használati adatai [7].

Fejlesztői aspektusok

A szimulátor programok használata során gyakran kell valamilyen kódot is írni. Nem minden szimulátor rendelkezik GUI-val, de amelyiknek van, ott is szükség lehet új modulok, modellek, beállítás szkriptek létrehozására, amely kódolási feladat. Ezen túlmenően a rendelkezésre álló felhozatalból olyat kell választanunk, amelynek láthatóan folyik a fejlesztése, az eszköz mögött álló csapat frissíti, javítja a kódot, új verziókat ad ki, támogatja az adott szimulátort választó felhasználókat.

A szimulátor írására használt programnyelv is fontos tényező. A Javában írt program általában könnyen bővíthető és módosítható, fordítás és futás során is hasznos hibaüzeneteket kaphatunk a kód hibamentesítéséhez. A futtatásához szükséges virtuális gép azonban erőforrás igényessé válhat, különösen, ha nagyszámú node szimulálásáról van szó. A másik legpopulárisabb nyelv a C++. Magában hordozza az OO paradigmából fakadó előnyöket és a natív bináris által jelentett előnyöket egyaránt. Azonban minthogy közeli rokonságban van a C-vel, a futásidőben generálódó hibaüzenetek sokkal kevésbé beszédesek, mint Java esetében, illetve könnyebb nehezen felderíthető hibát véteni a kódban.

A 2. táblázatban az adott szimulátort szoftverfejlesztési szempontok szerint tüntettem fel. Látható hogy egyes, a forrásanyagokban említett programok fejlesztése már nem él vagy nem érhető el publikusan.

Név	Nyelv	Platform	Licenc	Legutóbbi verzió	Projekt honlap
SENSE	C++	multi	academic	3.1 2008.11.19.	http://www.ita.cs.rpi.edu/
NS2	C++	multi	academic	2.35 2011.11.02.	http://www.isi.edu/nsnam/ns/
OPNET	C, C++	multi	commercial	n.a.	http://www.opnet.com/
Avrora	Java	Linux	BSD	1.7.117 2013.08.21.	http://avrora.sourceforge.net/
OMNET++	C++	Linux	academic és üzleti	4.3.1. 2013.09.17.	http://www.omnetpp.org/
NS3	C++	multi	GNU GPLv2	3.18 2013.08.29.	http://www.nsnam.org/
GloMoSim	Parsec	multi	na	2000 óta nem frissül. Utódja: QualNet	http://pcl.cs.ucla.edu/projects/glomosim/
QualNet	C++	multi	commercial	n.a.	http://web.scalable-networks.com/content/qualnet
TOSSIM	nesC	multi	BSD	2008?	http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM
Prowler	Java/MAtlab	TinyOS	academic	na.	http://www.isis.vanderbilt.edu/projects/nest/prowler
Castalia	C++	Linux	academic és commercial	3.2 2011.03.30.	http://castalia.research.nicta.com.au/index.php/en/
COOJA	Java	multi	BSD	na.	http://www.contiki-os.org/start.html#start-cooja
ATEMU	C	AVR/ TinyOS	BSD	0.4 2004.03.31.	http://www.hynet.umd.edu/research/atemu/
SensorSim	C++	multi	academic	na.	http://nesl.ee.ucla.edu/projects/sensorsim/

2. táblázat. Szimulációs szoftverek fejlesztői jellemzői [saját gyűjtés]

Tulajdonságok

Végezetül a 3. táblázatba kigyűjtöttem az adott szoftverek érdekes, vagy említésre méltó tulajdonságait, amely tulajdonságok vízválasztók lehetnek felhasználásuk fontolásánál. Az

alábbi kijelentések a forrásanyagok konklúziói. A hivatkozással jeleztem, hogy melyik forrás ítéletét takarja egy-egy sor.

Név	**Tulajdonság**	**Lehetséges hátrányok**
SENSE	Az NS2-höz hasonló, de nem OO. [3] Támogatja a párhuzamosítást. [3] Jó egyensúly a modellezés és a hatékonyság között. [4] Memória takarékos, gyors, kiterjeszthető és újrahasznosítható. [4]	Nem kompatibilis az NS2 szkriptekkel. [3] Nem alkalmas WSN kutatáshoz. [4] Kevés teljes kidolgozottságú modell. [4] Nincs GUI. [4]
NS2	Sok (>100) node szimulálására közepesen alkalmas. [5] A legelterjedtebb szimulátor. [4] Nagy számú protokoll áll rendelkezésre, és könnyen bővíthető újakkal. [4]	A bővítéséhez programozói ismeretek szükségesek. [8] Nehézkes és időigényes a használata. [8] Vezeték nélküli szimulációhoz csak két MAC protokollt támogat: 802.11 és single-hop TDMA. [8]
OPNET	Támogat szenzor specifikus hardver szimulációs modulok használatát. [3] Saját csomagformátumok definiálhatók. [3] GUI-val rendelkezik. [3] Kiválóan alkalmas nagyszámú node esetén. [5] Erőssége a pontos rádiómodell. [5]	Zárt forrású. [5] Kevés előre beépített modell létezik a WSN-ökhöz. [5]
Avrora	10000 node-ot tartalmazó hálózatot is kezel. [4] Nagy hálózatok időfüggése is vizsgálható vele. [4]	Nem szimulálja az óra elcsúszást (clock drift). [4] 50%-al lassabb mint a TOSSIM. [4] Nem képes mobilitást kezelni. [4]
OMNET++	Nem kimondottan vezeték nélküli hálózatok szimulálásra lett kifejlesztve, de nagyon jól bővíthető keretrendszer. Magába foglal egy teljes integrált fejlesztői környezetet (IDE). [6]	
NS3	Az NS2 alapjaira épül, annak továbbfejlesztésének tekinthető. [8]	A megváltozott szkript-nyelv miatt a korábbi NS2 szkriptek nem alkalmazhatók, így gyakorlatilag nem kompatibilis az elődel. [8]
GloMoSim	Párhuzamos szimulációs képesség. [4] Kimondottan WSN céljára. [4] GUI elérhető. [4]	Gyakorlatilag csak IP protokollal használható, az alapfelépítése következtében. [4] Nincsenek új protokollok. [4]
QualNet	Könnyen használható, világos GUI. [8] Támogatja a multiprocesszoros-, sőt az elosztott rendszereket is. [8]	Drága [8]
TOSSIM	Nagyfokú pontosság. [4] GUI elérhető. [4]	A fordítás során elvész a kód részletes idő- és megszakítás tartalma. [4]
Prowler	Pontos rádió modell.[4]	Csak a TinyOS MAC protokollja érhető el. [4]
Castalia	Fizikai folyamatmodell, érzékelő eszköz zaj, node óracsúszás, és pár MAC protokollal rendelkezik. [4] Konfigurálható MAC protkollok. [4]	Nem szenzor specifikus. [4] Nem célszerű, ha egy adott szenzorplatform kódját akarjuk tesztelni. [4]
COOJA	A szoftvert és a hardvert is szimulálja. [4] Nagy node-számú hálózat hálózat megfigyelésére alkalmas. [4]	Nem hatékony. [4] Egyidejűleg csak korlátozott számú node működhet. [4] Az időfüggő szimulációk nehézkesek. [4]
ATEMU	Ciklus-pontos emulációja az AVR platformnak. [3] XML fájllal és GUI-val is paraméterezhető. [3]	Maximum 120 node szimulálására alkalmas. [3] Mivel minden rádió minden rádióval kapcsolatban van, skálázhatóság tekintetében exponenciálisan romlik .[3]
SensorSim	NS2 alapú, de tartalmaz egy energia modellt is, valamint képes interfészelni külső alkalmazásokkal. [3]	Jelenleg nem fejlesztik.

3. táblázat. Szimulációs szoftverek tulajdonságai és esetleges hátrányai

KONKLÚZIÓ

Bár számos szimulációs szoftver érhető el, az áttekintés után lehet konkrét javaslatot és kiindulópontot ajánlani, ha általános szenzorhálózat szimulálási célra keresünk megoldást. Az egyik legfontosabb feladat, a megfelelő absztrakciós szint kiválasztása. Az emulátorok inkább a csekélyebb absztrakciós feladatokra, a szimulátorok inkább magasabb absztrakciójú feladatokra alkalmasabbak.

A szenzorhálózat szimulációs program alkalmasságát nagymértékben meghatározza a programozási paradigma. Ha minden node egy objektum az OO programban, nagyszámú node esetében skálázhatósági problémák lépnek fel. Nagyszámú node szimulálására alkalmasabbak az olyan megoldások, amelyek hálózati rétegekként valósítanak meg objektumokat, amelyek megvalósítják a node-ok működését.

A választást nagymértékben befolyásolja, hogy mennyire élő fejlesztésről van szó. Egy felületes áttekintés több tíz szenzorhálózat szimulációs lehetőséget mutat, de valójában csak néhány olyan van, amely eléggé érett és élő ahhoz, hogy komolyabb munkába érdemes legyen egyáltalán belekezdeni.

Jó választásnak látszik az NS2-vel, de inkább az NS-3-al kezdeni, hiszen ez a legelterjedtebb, legnagyobb háttéranyaggal rendelkező megoldás. Ha konkrét célnak nem felel meg, érdemes megnézni, hogy valamelyik arra épülő megoldás esetleg közelebb van-e saját céljainkhoz. Az OPNET és QUALNET szintén elterjedt és jó hírnévvel rendelkező megoldások, de egyik sem nyílt forráskódú és nem oktatási célra fizetni kell a licencért.

ÖSSZEGZÉS

A vezeték nélküli szenzorhálózatok szimulálhatósága rendkívül fontos kérdés, hiszen nagyszámú szenzor kiszórása után a begyűjtés nehéz vagy lehetetlen, és a hálózat működése, annak tesztelése csak akkor releváns, ha a node-ok már ki vannak telepítve. Számos szimulációs szoftver érhető el, melyek nagyobb része ingyenes és nyílt forráskódú.

Jelen publikációban példákon keresztül megmutattam a helyesen megválasztott absztrakciós szint fontosságát, és bemutattam egyéb tulajdonságaikat. Felhívtam a figyelmet a popularitás fontosságára, amelyet friss statisztikával prezentáltam. A vonatkozó publikációk tanulmányozására támaszkodva felállítottam egy saját szempontrendszert a bemutatott 14 szimulációs szoftver tulajdonságainak összefoglalására és összehasonlítására, amelyet táblázatos formában közöltem. Konklúzióként konkrét szoftvert javasoltam általános célú szimulációkhoz.

Felhasznált irodalom

[1] John Heidemann, Nirupama Bulusu, Jeremy Elson, Chalermek Intanagonwiwat, Kun-chan Lan, Ya Xu, Wei Ye, Deborah Estrin, Ramesh Govindan: Effects of Detail in Wireless Network Simulation. SCS Communication Networks and Distributed Systems Modeling and Simulation Conference September 26, 2000

[2] Nirupama Bulusu, John Heidemann, Deborah Estrin: GPS-less Low Cost Outdoor Localization For Very Small Devices. IEEE PERSONAL COMMUNICATIONS MAGAZINE, OCTOBER 2000.

[3] David Curren: A Survey of Simulation in Sensor Networks. University of Binghamton, NY, 2005.

[4] Harsh Sundani, Haoyue Li, Vijay K. Devabhaktuni, Mansoor Alam, & Prabir Bhattacharya: Wireless Sensor Network Simulators A Survey and Comparisons. International Journal Of Computer Networks (IJCN), Volume (2) : Issue (5)

[5] Marko Korkalainen, Mikko Sallinen, Niilo Kärkkäinen, Pirkka Tukeva: Survey of Wireless Sensor Networks Simulation Tools for Demanding Applications. 2009 Fifth International Conference on Networking and Services

[6] Milos Jevtić, Nikola Zogović, Goran Dimić: Evaluation of Wireless Sensor Network Simulators. 17th Telecommunications forum TELFOR 2009

[7] Nurul I. Sarkar and Syafnidar A. Halim," A Review of Simulation of Telecommunication Networks: Simulators, Classification, Comparison, Methodologies, and Recommendations", Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), March Edition, 2011

[8] Mrs. Poonam Chhimwal, Dhajvir Singh Rai, Deepesh Rawat: Comparison between Different Wireless Sensor Simulation Tools. IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) e-ISSN: 2278-2834,p- ISSN: 2278-8735. Volume 5, Issue 2 (Mar. - Apr. 2013), PP 54-60

[9] Fei Yu: A Survey of Wireless Sensor Network Simulation Tools. [Online] http://www1.cse.wustl.edu/~jain/cse567-11/ftp/sensor/index.html last access: 2013.11.01.