HADMÉRNÖK

VII. Évfolyam 1. szám - 2012. március

Nagy Tibor István
nagy.tibor@nik.uni-obuda.hu

SZENZORHÁLÓZATOK SZOFTVERFEJLESZTÉSI KÉRDÉSEI

Absztrakt

A felügyelet nélküli szenzorhálózatok a civil szférában és a szárazföldi harcászati felderítésben egyaránt fontos szerepet játszanak. A legtöbb informatikai eszközhöz hasonlóan a szenzorok működéséhez a hardverösszetevőkön túl szoftverekre is szükség van, melyek a szenzorok vezérlését, a hálózat kiépítését, illetve a megbízható működést szabályozzák.
Ez a publikáció áttekintést ad a felügyelet nélküli rendszereknél leggyakrabban használt operációs rendszerekről, programozási nyelvekről, tervező- és fejlesztőeszközökről.

Unattended ground sensor networks are important in civil society and in ground reconnaissance. Like most of the devices in information technology, sensors also need hardware and software components to operate the sensors, build up the network between them and ensure reliable operation.
This paper gives a review of the operating systems, programming languages and development tools mostly used with unattended sensors.

Kulcsszavak: *szenzor, felügyelet nélküli szenzorhálózat, intelligens szenzorhálózat, tervezőeszközök, fejlesztőeszközök, IDE ~ sensor, unattended sensor network, intelligent sensor network, design tools, development tools, IDE*

1. BEVEZETŐ

A civil életben, a honvédelemben, rendvédelemben és a katasztrófavédelemben egyaránt találunk sok olyan tevékenységet, amelyeknek emberek általi végrehajtása nem megoldható, mert a célterület nem megközelíthető, túl nagy kockázattal jár, vagy nagypontosságú méréseket igényel. Ezen tevékenységek nagy részéből természetesen nem hagyható el az ember, de bizonyos részeit gépekre, elektronikára, mesterséges intelligenciára lehet – és célszerű is – bízni. Az emberi erőforrás nélkül megvalósítható tevékenységek közé általában az adatgyűjtés, különböző jellemzők mérése, vegyi- meteorológiai-, geológiai folyamatok megfigyelése tartoznak.

A lehetséges alkalmazási területek például az erdőtüzek megfigyelése, megfékezése, épületen belüli tűzvédelmi feladatok ellátása, árvizek, belvizek esetén adott terület geológiai, statikai jellemzőinek mérése, vulkáni tevékenység során lezajló folyamatok nyomon követése, harctéri ellenséges csapatmozgások, ellenséges erők megfigyelése, orvlövészek detektálása.

Az adatgyűjtést, jellemzők mérését szenzorok, érzékelők segítségével lehet elvégezni. Sokféle fizikai elven működő szenzor létezik, amelyek az általuk alkalmazott mechanizmus felhasználásával különféle fizikai paraméterek változásait képesek érzékelni. A szenzorok általában „egy csomagban” vannak az áramforrással, illetve vezérlő- és előfeldolgozó egységgel, amelyek a működésüket biztosítják. Ha nagy területet kell átfogni, figyelni, erre egyetlen ilyen csomag – más néven node – nem alkalmas az érzékelők korlátozott hatótávolsága, és energetikai paraméterei miatt, ezért legtöbbször ezekből többet kell a megfigyelt területen elhelyezni, amelyek egymással kapcsolatban állnak, és összehangoltan, egymás képességeit kiegészítve, a feladatokat egymás között megosztva működnek, azaz hálózatba szerveződnek. Az összehangolt működéshez a node-ok közti kommunikációra is szükség van, ezért a node-ok kapcsolattartást lehetővé tevő egységeket is tartalmaznak.

A szenzorhálózatok katonai alkalmazási területi, lehetőségei igen széleskörűek; leginkább az elektronikai felderítésben, és az elektronikai támogatásban bizonyulhatnak hatékony eszköznek, de kisebb változtatásokkal akár az elektronikai ellentevékenység terén is használhatóak.

Elsődleges felhasználási területként a harcászati felderítést lehet említeni, ahol az ellenséges csapatok mozgásával, létszámával, összetételével kapcsolatban lehet létfontosságú adatokat gyűjteni segítségükkel. Nagy előnyük, hogy nem kell a veszélyes területre felderítő katonákat küldeni és így értékes emberéleteket lehet megóvni, ami a további harcok sikerességét jelentősen befolyásolhatja.

A megfelelő vezérlés megvalósításához, a node-ok közti kapcsolat felépítéséhez és fenntartásához, a begyűjtött adatok átalakításához, továbbításához a hardverelemeken kívül természetesen szoftverekre is szükség van. Az optimális működéshez meg kell határozni a hardver- és szoftverösszetevők megfelelő arányát, a feldolgozást, átalakítást, adatfuzionálást ellátó szoftverelemek node-ok és bázisállomások közti megosztásának arányát, figyelembe kell venni a szoftverek futtatását lehetővé tevő operációs rendszer és a szoftverek elkészítéséhez használható tervező-, fejlesztő eszközöket, programozási paradigmákat, és programozási nyelvi sajátosságokat.

Ebben a cikkben a szenzorhálózatok szoftverfejlesztési kérdéseivel foglalkozom. Bemutatom a hálózatok tervezésének kérdéseit általában és a szenzorhálózati tervezés speciális feladatait. Végül csoportosítom a szenzorhálózatoknál használt szoftverelemeket, operációs rendszereket és fejlesztő eszközöket.

2. VEZETÉK NÉLKÜLI HÁLÓZATOK TERVEZÉSE

2.1. Hálózat fogalma [1]

Tanenbaum szerint a számítógépes hálózat olyan rendszer, amelyben a feladatokat „sok-sok különálló, de egymással összekapcsolt számítógép látja el”. [1]

Sokféle hálózati architektúra létezik, függően a felhasználás céljától, helyétől. A legegyszerűbb esetben két számítógép áll (közvetett vagy közvetlen) kapcsolatban egymással, és mindkét gép képes egymás szolgáltatásait igénybe venni (pont-pont kapcsolat). A két gép tekinthető egyenrangúnak, és mindkettőnél ül egy felhasználó, aki a saját, illetve a másik számítógépét használja (pl.: számítógépes játékok, bluetooth kapcsolat két gép között, stb.). A másik lehetőség szintén két gép közvetett, vagy közvetlen kapcsolata, de itt az egyik gép általában jóval nagyobb teljesítményű a másiknál. A kisebb teljesítményű gép használja a nagyobb teljesítményű szolgáltatásait (kliens-szerver architektúra).

Hálózatot természetesen nem csak számítógépek, hanem elektronikai eszközök (pl.: szenzorok) is alkothatnak, vagy akár vegyesen számítógépek és különböző elektronikai eszközök is. A rengeteg különböző igény, sokféle különböző eszköz miatt a hálózatok kizárólag elektronikai gyártással történő megvalósítása nem lehetséges. Emiatt megalkottak úgynevezett hálózati hivatkozási modelleket, amelyek különböző, egymástól elkülönülő rétegeket definiálnak, melyek egymástól függetlenül valósíthatóak meg, ezzel biztosítva az alkalmazási területek végtelen sorát.

Mindegyik modell lényege, hogy az egyes rétegek csak a közvetlenül alattuk, illetve felettük levő rétegekkel kommunikálhatnak meghatározott interfészeken keresztül. A rétegek fekete dobozként működnek, azaz elrejtik mások elől adataikat és működésük részleteit.

Minél magasabb szintű rétegről van szó, annál inkább tolódik el a működést biztosító technológia a hardver irányából a szoftver felé, illetve annál bonyolultabb, összetettebb funkciókat biztosít, és annál bonyolultabb, nagyobb adategységekkel képes dolgozni.

2.1.2. OSI hivatkozási modell:

A legrégibb, illetve legáltalánosabban használható modell. A többi modell általában ennek a specializált, konkrét igényekre átszabott változata. Hét réteget definiál:

- *Fizikai réteg:* Feladata a bitenkénti adattovábbítás megvalósítása az átviteli közeg felhasználásával. Ez a réteg az átvitel mechanikai, elektronikai kérdéseivel foglalkozik. A főszerepet itt a vezetékek, csatlakozók, áramkörök játsszák.
- *Adatkapcsolati réteg:* Az adatátviteli egység itt az adatkeret, amely néhány száz, vagy néhány ezer bájt. A réteg feladata a keretek helyes sorrendben és hibamentesen történő eljuttatása a küldőtől a fogadóig.
- *Hálózati réteg:* Feladata az adatcsomagok eljuttatása a küldőtől a fogadóig, illetve az ehhez használandó útvonal kiválasztása, csomagtorlódások megakadályozása, hálózati forgalom vezérlése. Ez a réteg általában a hálózati szolgáltatást nyújtó szolgáltató routerein működik. A legelterjedtebb protokollja az IP (Internet Protocol).
- *Szállítási réteg:* A viszonyrétegtől érkező információkat darabolja szét megfelelő méretű és szerkezetű adategységekké. A szállítási réteg hasonló funkciót biztosít mint a hálózati réteg, azzal a különbséggel, hogy általában a küldő, illetve a fogadó számítógépen működik. A szállítási rétegben leggyakrabban használt protokollok a TCP és az UDP.
- *Viszony réteg:* A réteg feladata két gép közötti viszony (session) létrehozása és kezelése. A viszonyban az adás jogának kiosztása, kritikus műveletek végrehajtási jogának szabályozása, kommunikáció szinkronizálása.

- *Megjelenítési réteg*: Bonyolultabb, illetve különböző típusú adatszerkezetek használatát és átvitelét teszi lehetővé.
- *Alkalmazási réteg:* A felhasználói programok által a hálózat lehetőségeinek használatához szükséges protokollokat tartalmazza. A leggyakrabban használt ezek közül a weblapok és egyéb webszervereken található erőforrások lekéréséhez használható http, de ide tartoznak az FTP a fájlok átviteléhez, elektronikus levelezés protokolljai (például az SMTP), stb.

2.1.3. TCP/IP hivatkozási modell:

Ez a hivatkozási modell az internet születésekor, annak hálózati modelljeként funkcionált, és a mai napig is ezt a szerepet tölti be. Több, különböző méretű, típusú, különböző technológiákat használó hálózat biztonságos, megbízható összekötését képes megteremteni és vezérelni.

- *Hoszt és hálózat közötti réteg:* A TCP/IP hivatkozási modellben ez a réteg nincs kidolgozva. Nem definiál semmilyen áramkört, átviteli közeget, fizikai kapcsolati módot, tehát a modell legalsó rétege tulajdonképp a következő internet réteg.
- *Internetréteg:* A TCP/IP hivatkozási modell központi rétege. Az elküldendő adatokat képes csomagokra felosztani és azt bármilyen típusú hálózatban található címzetthez eljuttatni. Protokollja az IP (Internet Protocol).
- *Szállítási réteg:* A réteg feladata az elküldendő adatok üzenetekké alakítása és az internetréteg felé történő továbbítása, és az adatátvitel sebességének szabályozása a küldő és fogadó sebességkülönbségétől függően. Protokollja a TCP (Transmission Control Protocol), amely az adatok biztonságos, és megfelelő sorrendű átvitelét biztosítja, illetve az UDP (User Datagram Protocol), amely nem nyújt biztonságos átviteli szolgáltatást, viszont gyors és kapcsolat nélküli módon valósítja meg az átvitelt.
- *Alkalmazási réteg:* Az OSI modell Megjelenítési- és viszonyrétege a TCP/IP modellben nincs külön rétegként definiálva. Ezek feladatait is az alkalmazási réteg látja el. Itt találhatóak azok a protokollok, amelyek a különböző felhasználói alkalmazások működéséhez szükségesek (FTP, HTTP, SMTP, TELNET, DNS, …).

2.2. Hálózattervezés főbb feladatai, célja, rendeltetése

Általánosságban nagyon tág a hálózattervezés fogalma. Az alsóbb rétegekben a hordozó média, topológiák tekintetében már kialakult rendszerek vannak, amelyeket a hálózattervezők használnak, és amelyet a hálózati eszközt gyártó cégek az eszközbe beépítve kínálnak (pl.: topológiák terén leginkább busz, illetve csillag topológiát alkalmaznak, UTP kábelt használnak a gépek és hálózati eszközök összekötéséhez, stb.).

Leginkább a futtatott szoftverek tekintetében lehet nagyobb feladatról beszélni, vagyis a felsőbb rétegek szoftvereinek megtervezése és elkészítése jelentik a hálózattervezők feladatának nagyobb kihívást jelentő részét, ide értve a meglévő protokollok használatát, vagy esetleg új protokollok kidolgozását is.

Általánosságban léteznek olyan hálózattervezési kérdések, amelyek egy hálózat működési paramétereit, minőségét meghatározzák. Ezeknek a rétegeknek a megvalósításával a hivatkozási modell egyes rétegei foglalkoznak.

Általános hálózattervezési kérdések [1]:

- Címzés: az üzenet fogadójának és küldőjének azonosítása;
- Hibavédelem: hibajavító kódok alkalmazása az átvitelnél okozott adatvesztés, vagy hibák kijavítására, üzenetek megfelelő sorrendjének biztosítása;
- Forgalomszabályozás: a különböző sebességű adók és vevők szinkronizálása;

- Multiplexelés: egy csatornán több kommunikáló számítógéppár üzenetei összefésülve kerülnek átküldésre;
- Forgalomirányítás: annak meghatározása, hogy az adatok milyen útvonalon jussanak el a feladótól a címzettig, hogy a leggyorsabban, legkisebb hálózati terheléssel, legbiztonságosabban érkezzenek meg.

2.3. Szenzorhálózatok hálózattervezésének sajátos feladatai

A szenzorhálózatok esetén is az OSI modell rétegei, illetve ezek összevonásából kialakult új rétegek használatosak az egyes feladatok különböző szinten történő megvalósítására. A specialitást itt az adja, hogy a node-ok esetében figyelembe kell venni néhány olyan jellemzőt, amelyek a hagyományos számítógép-hálózatoknál nem okoznak problémát. Ilyen például a szűkösen rendelkezésre álló energia, a node-ok esetleges helyváltoztatásának, illetve működésképtelenségének problémája, illetve a feladatok egyes node-ok közti elosztásának módja. Ezekre a specialitásokra alkották meg a WSN protokoll-vermet.

WSN (Wireless Sensor Network) protokoll-verem [2]:

Egy kockaként ábrázolható, amely síkokra, illetve rájuk merőleges rétegekre van osztva. A síkok a szenzorhálózat működésének legkritikusabb, egymástól elkülönülő feladatcsoportjait jelenítik meg, amelyek esetén az egyes rétegeknél más és más tervezési szempontokat kell figyelembe venni. Ezek a síkok biztosítják azt is, hogy az egyes rétegek ne csak a közvetlenül alattuk lévő rétegekkel tudjanak kommunikálni, hanem a többi réteg által előállított, de az aktuális réteg által igényelt adatokkal is tudjanak dolgozni. Ez az OSI modell eredeti szándékát – miszerint a rétegek jól elszeparált egységek, melyek működésüket elrejtik a külvilág elől, és csak a közvetlenül alattuk, illetve fölöttük lévő rétegekkel kommunikálhatnak – nem veszi figyelembe, de könnyebbé teszi a konkrét szenzorhálózati megvalósítások elkészítését. Többféle protokoll-vermet is definiáltak, ezek közül itt kettőt írok le: az egyik három, míg a másik négy síkot definiál.

Három síkot definiáló változat:

- Energia-menedzsment sík: Itt a legfontosabb figyelembe veendő szempont, hogy az egyes node-ok minél kevesebb energia felhasználásával tudják ellátni feladatukat, ezzel minél hosszabb ideig legyenek képesek működni. Ez biztosítható minél jobb akkumulátorok használatával, olyan kiegészítő eszközökkel, amelyek a környezetből képesek energiát felvenni, illetve az egyes node-ok közötti feladatok elosztásával az akkumulátorok töltöttségi szintjeitől függően.
- Mobilitás-menedzsment sík: léteznek olyan szenzorhálózatok, melyekben az egyes node-ok képesek a telepítést követően változtatni helyzetüket, esetleg néhány közülük elromolhat, szándékos rongálás következtében elpusztulhat. A telepítés jellegétől függően közvetlenül a telepítés után szükség lehet a node-ok helyzetének felderítésére, hogy a hálózati kapcsolatot ki lehessen közöttük építeni, ami szintén ennek a síknak a feladata.
- Feladat-menedzsment sík: a feladatok kiosztása, adattovábbítás, adatfeldolgozás, tömörítés, kódolás, feladatok párhuzamosítása, illetve elosztott feladat-végrehajtás tartoznak ennek a síknak a feladatkörébe.
- Négy síkot definiáló változat [3]:
- Energia-menedzsment sík: megegyezik a három síkot definiáló változatnál ismertetettel.
- Biztonsági sík: az adatok titkosítása, kódolás, dekódolás a feladata.
- Időszinkronizációs sík: az egyes node-ok időbeli összehangolása tartozik a feladatkörébe.

- Szomszédfelderítő sík: egy node szomszédjainak felderítését, a szomszédok megkeresését, lekérdezését teszi lehetővé.

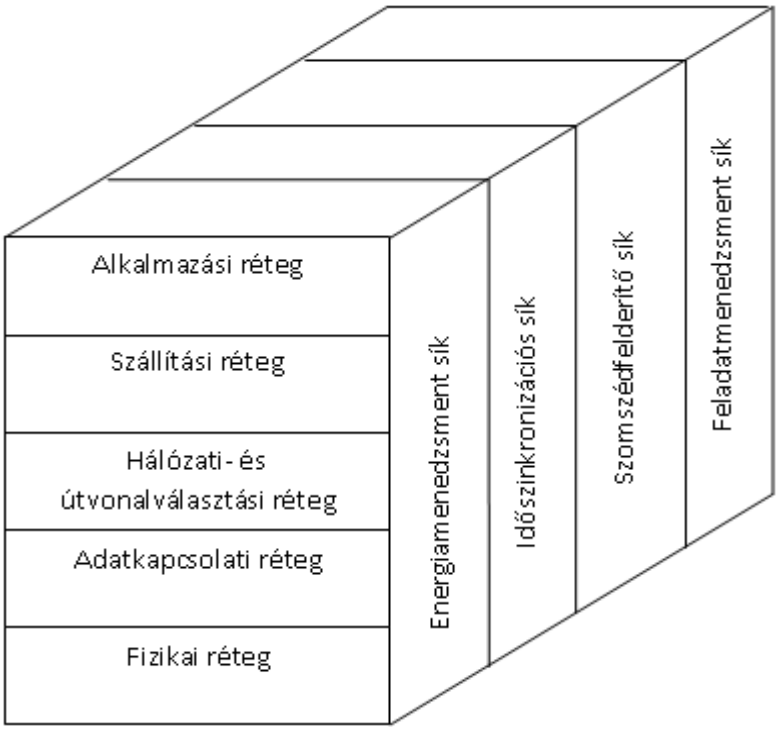
A második protokoll-verem még ezen kívül egy negyedik dimenziót is bevezet, amely szétválasztja egymástól az adatok kezelését és a vezérlést a különböző síkok és rétegek metszeteiben.

A rétegek pedig:

- Fizikai réteg: ez a réteg a kommunikáció fizikai paramétereivel foglalkozik, mint a frekvencia kiválasztása, modulációs mód, jelfelismerés, jelátalakítás, kódolás. Ezek a jellemzők leginkább a hardver szintjén megvalósíthatók.
- Adatkapcsolati (MAC) réteg: a hálózat kiépítését végzi, meghatározza a hálózati topológiát, az üzenetküldés módját, biztosítja a megbízható hálózati kapcsolatot, illetve az átküldött adatokban keletkezett hibák kijavítását.
- Hálózati és útvonal-választási réteg: legfőbb feladata az útvonalválasztás, ahol is az egyes node-ok megcímzése attribútum központúan történik, az optimális energiafelhasználás, illetve a lehető legjobb jelerősség biztosítása mellett.
- Szállítási réteg: feladata a különböző típusú hálózatok összekötése lenne. Erre a feladatra a TCP / UDP protokollok a legmegfelelőbbek, ezeknek, vagy ezeknek a speciális igényekhez igazított változatai használhatóak.
- Alkalmazási réteg: a réteg feladatai nagyrészt hasonlóak az OSI modellnél ismertetettekkel. Az itt alkalmazható protokollok fejlesztés alatt állnak, több kutatási projekt is foglalkozik ezzel a kérdéssel. Az egyik ilyen protokoll az SMP (Sensor Management Protocol), amely a szenzorok adminisztrátorok általi, távoli (például internetes kapcsolaton keresztüli) vezérlését teszik lehetővé. A másik a TADAP (Task Assignment and Advertisement Protocol), amely a felhasználók számára érdekes adatok, illetve node-ok kiválasztását teszik lehetővé, a harmadik pedig az SQDDP (Sensor Query and Data Dissemination Protocol), melynek segítségével lekérdezések formájában lehet hozzájutni a kívánt adatokhoz az SQLT (Sensor Query and Tasking Language) segítségével.

Felmerülhet a kérdés, hogy a protokoll-verem megvalósítások közül melyik lenne a legjobb a katonai alkalmazások esetén – különös tekintettel a kutatási témára: az ad hoc szervezésű, szárazföldi harcászati szenzorhálózatokra –, vagy lehetne-e a két változat ötvözetét használni. Véleményem szerint a síkok uniója nem lenne megfelelő modell, hiszen hat különböző sík, plusz még az adatok és a vezérlés szétválasztása átláthatatlanná és nehézkessé tenné a tervezést. A biztonságos adattovábbításnak fontos szerep jut, de szerintem nincs szükség rá, hogy több réteg is foglalkozzon ezzel a kérdéssel. A mobilitás-menedzsment csak olyan hálózatoknál fontos, ahol a node-ok helyet tudnak változtatni. A szinkronizáció az összadatforrású felderítés miatt lényeges, hiszen egy célpont észleléséhez szükséges a különböző típusú szenzoroktól érkező adatokról pontosan tudni, hogy mikor mérték őket. A repülőgépről történő kiszórással, vagy tüzérségi eszközzel történő telepítés esetén ad hoc hálózat alakul ki, melyben a szomszédok felderítésének nagy a jelentősége. Az energia-menedzsment természetesen nem elhagyható, fontos szempont, ezért ennek a síknak a megtartása is lényeges lehet. A feladatok szétosztása szintén olyan tevékenység, melyről több réteg is kell, hogy tudjon, hiszen például az alkalmazás réteg csak akkor tudja a megfelelő node-oknak továbbítani a kiválasztott feladatrészt, ha tudja, hogy mely node-ok képesek a működőek közül végrehajtani azt.

Ez alapján az általam javasolt WSN protokoll-verem az 1. ábrán látható.



1. ábra. Javasolt WSN protokoll verem

3. SZOFTVERTERVEZÉS SZENZORHÁLÓZATOKBAN

3.1. Szoftverfejlesztés szerepe szenzorhálózatokban

A node-ok érzékelőket, kommunikációs részegységeket, memóriát és az ezek vezérlését végző processzort tartalmaznak, vagyis a node-ok speciális számítógépek. Mint ilyenek, természetesen nemcsak hardvert, hanem szoftvert is tartalmaznak. Bizonyos funkciók előre huzalozottan, a gyártó által beépítve állnak rendelkezésre, más funkciókat pedig szoftverek segítségével kell megvalósítani. Az alacsonyabb szintű szoftveres funkciókat operációs rendszerek biztosítják, míg a magasabb szintű, bonyolultabb, összetettebb feladatokat az operációs rendszer által futtatható programok látják el.

A szoftver- és hardverelemeknek három fő feladatcsoportot kell ellátniuk:

- Információszerzés: adatok begyűjtése a szenzoroktól;
- Adatfeldolgozás: a szenzoroktól begyűjtött adatok továbbítható formára alakítása, megfelelő adatszerkezetekbe foglalása, a mért jellemzők jellegének és a mérés idejének rögzítése, hibajavító kódok alkalmazása, adatok titkosítása;
- Adattovábbítás: a begyűjtött és előfeldolgozott adatok továbbítása egy gyűjtőcsomóponthoz, meghatározott útvonalon, figyelembe véve az optimális energiafelhasználást.

3.2. Operációs rendszerek

Az operációs rendszerek biztosítják a programok futtatását és a hardver programból történő elérésének lehetőségét. A node-okon többféle operációs rendszer is használható, amelyet a következőkben sorolok fel:

- *Contiki:* nyílt forrású, eseményvezérelt, többfolyamatos és egyszerűsített szálkezeléses operációs rendszer, amely hálózatba kötött beágyazott rendszereken és szenzorhálózatokon működik. Tipikusan 2 kbyte RAM-ot és 40 kbyte ROM-ot igényel, vagyis kevés memóriával rendelkező eszközökön képes működni. IPv4 és IPv6 kommunikációt biztosít alacsony energiafelvétel és szoftveres

energiamenedzsment mellett. A TCP/IP hivatkozási modell egyes rétegeiben saját fejlesztésű protokollokat alkalmaz. Saját fájlrendszert használ a node-okon történő adattároláshoz. A szoftverfejlesztés C nyelven történik, a kész szoftverek tesztelésére pedig szimulátorok állnak rendelkezésre. [4]

- *LiteOS:* dinamikus memóriakezelésű, többszálas programokat futtatni képes operációs rendszer, amely UNIX-szerű hierarchikus fájlrendszert használ. Lehetővé teszi az alkalmazások telepítés utáni frissítését / újratelepítését. A node-okhoz terminálparancsok segítségével lehet hozzáférni. A programok fejlesztése C nyelven történik, az AVR Studio fejlesztőkörnyezet pedig megkönnyíti a programozó munkáját. Különlegessége, hogy minden protokoll egy-egy szálként van megvalósítva, így a beépített protokollok könnyen lecserélhetők sajátra, vagy akár több protokoll is megfér egymás mellett. [5]
- *NanoQplus:* nyílt forrású, szálkezelés-központú operációs rendszer. A szálak ütemezése preemptív „round-robin" módszerrel történik. A memóriatakarékosságot egy speciális lapozó technikával valósítja meg, amely a működés sebességének a csökkenésével járhat. Kernel szinten tartalmaz olyan funkciókat, amelyek az útvonalválasztást, node-ok felderítését valósítják meg. Léteznek hozzá grafikus fejlesztőeszközök az automatikus kódgeneráláshoz, távoli monitorozáshoz, illetve távoli szoftverfrissítéshez. [6]
- *TinyOS:* nyílt forrású, komponensalapú, eseményvezérelt operációs rendszer. Nem támogatja a dinamikus programfuttatást / telepítést, programok frissítését és a dinamikus memóriakezelést. A programok nesC nyelven íródnak, komponensekből tevődnek össze, amelyek egymáshoz interfészeken keresztül kapcsolódnak. A végrehajtási modellje nem blokkoló, azaz a hosszabb időt igénybe vevő feladatok aszinkron hajtódnak végre. Egyszerű folyamatkezeléssel rendelkezik. Az eseményvezérelt, aszinkron működés és az egyszerű folyamatkezelés miatt az összetettebb feladatokra programot készíteni nehézkes. A szenzorhálózatoknál ez a legelterjedtebb operációs rendszer, ezért sokféle node-ot támogat. [7]

Ezen operációs rendszerek mindegyike megvalósítja a hálózati hivatkozási modell rétegeit vagy már létező, vagy saját protokollok segítségével.

A felsoroltakon kívül még sok szenzorhálózatoknál használható (Erika Enterprise, SOS, Nano-RK, …), illetve általános beágyazott rendszereken működő (Windows CE, embedded Linux, VxWorks, eCos, SymbOS, …) operációs rendszer létezik, de a fentiek a legelterjedtebbek és paramétereiket tekintve leginkább illeszkednek a szenzorhálózatok által támasztott igényekhez.

A megfelelő operációs rendszer kiválasztásánál több szempontot is érdemes figyelembe venni: a kiszemelt node típust támogatja-e, mennyire kényelmes a fejlesztés (fejlesztőeszközök, fejlesztőkörnyezetek), mennyi RAM, ROM, flash memória szükséges a működéséhez, illetve az elkészített programok frissíthetőek-e. Az első szempontnak a TinyOS felel meg a leginkább, hiszen ez a legrégebben használt, legelterjedtebb operációs rendszer, viszont a komponens alapú, eseményvezérelt működés és a túl egyszerű folyamatkezelés miatt nem biztosít kényelmes és nem rugalmas szoftverfejlesztést. A felsoroltak közül mindegyik kimondottan kevés erőforrással rendelkező eszközökre lett optimalizálva, ezért nincs nagy eltérés közöttük a memóriahasználat tekintetében. Az elkészített programok frissíthetőségét a Contiki és a LiteOS lehetővé teszi, ezen kívül mindkettőhöz létezik kényelmesen használható fejlesztőeszköz, illetve szimulátorok a tesztelés megkönnyítésére. A fejlesztőeszközök tekintetében a TinyOS sem marad le, hiszen az Eclipse-hez telepíthetőek kiegészítők, melyekkel TinyOS-re lehet programot fejleszteni, illetve a NanoQplus-hoz is léteznek fejlesztő, telepítő és monitorozó eszközök. A fentiek alapján véleményem szerint a legjobb operációs rendszerek a felsoroltak közül a Contiki és a LiteOS.

3.3. Információszerző szoftverelemek

A szenzorok működtetésének alapfunkcióit huzalozottan valósítják meg. A szenzoroktól érkező adatok elérését, az energiamenedzsment funkciókat az operációs rendszerbe építik be. Az egyes operációs rendszerek más és más módon valósítják meg ezeket, és a felhasználói alkalmazások ezeket rendszerhívásokon keresztül érhetik el. Az operációs rendszerek többsége biztosít lehetőséget a beépítetten megvalósított módszer sajátra történő lecserélésére.

3.4. Adattovábbító szoftverelemek

Huzalozottan csak a rádió adóvevő működtetését készítik el, az útvonalválasztást, adatok eljuttatását egyik node-tól a másikig, egy adott node-tól a gyűjtő csomópontig (sink node) az operációs rendszer valósítja meg egy meghatározott protokoll szerint, amely operációs rendszerenként különböző lehet. Az operációs rendszerek többsége biztosít lehetőséget a beépítetten megvalósított módszer sajátra történő lecserélésére.

3.5. Adatfeldolgozó szoftverelemek

A szenzortól érkező adatok előfeldolgozásához (konverzió, tömörítés, titkosítás, megfelelő struktúrába szervezés, …) sem huzalozott, sem operációs rendszerbe beépített megoldások nem léteznek. Erre szoftvert kell készíteni. Operációsrendszer-szinten az adatok tárolásának lehetősége biztosított valamilyen egyedi fájlrendszer-megoldással, vagy például a Contiki operációs rendszerben egy node-okon használható adatbáziskezelő rendszerrel.

A bázisállomásokon történő adatfeldolgozáshoz, adattároláshoz egyrészt használhatóak a kereskedelmi forgalomban kapható általános célú adatbáziskezelő rendszerek egy megfelelő szoftver-kiegészítéssel, amely összekapcsolja ezt a rendszert a szenzorhálózattal. Képfeldolgozáshoz, képfelismeréshez, adatfúzióhoz saját szoftvert kell készíteni.

Az adatok megjelenítéséhez, vizualizálásához a legtöbb operációs rendszer tartalmaz távoli parancssori elérési lehetőséget, amellyel a node-ok, illetve szenzoraik állapotai és az általuk szolgáltatott adatok lekérdezhetők. Ezen kívül van olyan operációs rendszer, amely grafikus felhasználói felülettel rendelkező alkalmazást is rendelkezésre bocsájt ugyanezen feladat ellátására.

4. KONKLÚZIÓ

A szenzorhálózatok az élet legkülönbözőbb területein felhasználhatóak az ipari gyártástól az intelligens otthon vezérlésén keresztül a harctéri felderítési feladatok ellátásáig. A hagyományos számítógépekhez, illetve ezeket összekötő hálózatokhoz hasonlóan itt is szükség van szoftverekre a megfelelő működés biztosításához, illetve a rugalmas, skálázható, bővíthető rendszer kialakításához. Ez a cikk áttekintést adott a hálózatok tervezésének általános, és szenzorhálózatoknál alkalmazandó technikáiról, tervezési elveiről, módszereiről, és a különböző feladatcsoportok esetén rendelkezésre álló szoftverelemekről.

Felhasznált irodalom

[1] A. S. TanenBaum: Számítógép-hálózatok, Második, bővített kiadás, Panem Könyvkiadó Kft., 2004. ISBN 963 545 384 1

[2] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci: Wireless sensor networks: a survey, Computer Networks 38, 2002, pp 393-422. http://www.larces.uece.br/~celestino/RSSF%20I/Wireless%20Sensor%20Networks%20a%20Survey.pdf; (letöltés: 2012. 01. 30.)

[3] Siddhart Ramesh: A Protocol Achitecture for Wireless Sensor Networks, University of Utah. http://www.cs.utah.edu/~sramesh/attachments/sensornet_archi.pdf; (letöltés: 2012. 01. 30.)

[4] A Contiki OS hivatalos weboldala, http://www.contiki-os.org/; (letöltés: 2012. 02. 24.)

[5] Q. Cao, T. Abdelzaher, J. Stankovic, T. He, The LiteOS Operating System: Towards Unix-like Abstractions for Wireless Sensor Networks, IPSN '08: Proceedings of the 7th international conference on Information processing in sensor networks, pp 233-244. IEEE Computer Society, 2008.

[6] S.C. Kim, H. Kim, J.K. Song, M. Yu, P. Mah: NanoQplus : A Multi-Threaded Operating System with Memory Protection Mechanism for WSNs, Proceedings of the CKWSN, 2008, http://nano-os.tistory.com/attachment/cfile23.uf@135FD6484DC6900E2BC27E.pdf; (letöltés: 2012. 02. 26.)

[7] P. Levis: TinyOS 2.0 Overview, 2006, http://wiesel.ece.utah.edu/redmine/projects/wasp/repository/revisions/f4407f25ded5a962bd0ed34c950b49f29a88a4a3/raw/doc/pdf/overview.pdf; (letöltés: 2012. 02. 26.)