Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

TDK Dolgozat

nRF alapú intelligens otthon hálózat tervezése és építése

Bana Szabolcs Gábor
Béres Gábor

Konzulens: Dr. Fehér Gábor (egy. docens)

2014

Tartalomjegyzék

Abstract

Napjainkra az okos otthon fogalma valósággá vált, több különböző megoldás elterjedt, egyre nő az igény az intelligensen működő automatizált berendezésekre. A jelenleg létező számos technológia mellett folyamatosan jelennek meg újabb szereplők, a közeljövőben a területen rohamosan fejlődést jósolnak. A vezeték nélküliek közül a legjelentősebbek például a Bluetooth 4.0, a ZigBee, a Z-Wave. Ezek mellett létezik az előbbieknél olcsóbb és az okos otthonok elvárásainak megfelelő nRF adó-vevő. Ez egy 2.4 GHz-es frekvencián működő, alacsony fogyasztású IC, amely maximum 2 Mbit/s sebességgel képes kommunikálni, és a környezettől függően 20-40 méteres hatótávolsággal rendelkezik. Ezek a paraméterek tökéletesen megfelelnek az okos otthon hálózat által teremtett követelményeknek.

A célunk egy mindenki számára elérhető, könnyen bővíthető, megfelelő hálózati paraméterekkel rendelkező okos otthon megoldás létrehozása volt, amely rendelkezik a létező rendszerek főbb funkcióival és teljesítményével, de azoknál egyszerűbben telepíthető és olcsóbb.

A munkát egy multihop hálózat működéséhez szükséges protokoll megtervezésével kezdtük. A követelmények között a legfontosabbak a hálózat minél egyszerűbb beüzemelése, a jó skálázódás, megbízható működés és az alacsony válaszidő voltak. A rendszer képes sok, jelentősen eltérő tulajdonságokkal rendelkező egység kezelésére.

Megvizsgáltunk néhány szenzorhálózatban használt kommunikációs protokollt, majd az azokból tanultak és az nRF adottságai alapján megterveztünk egy saját protokollt. A különböző okos otthon funkciók megvalósításához szükséges intelligencia egy saját fejlesztésű központi egységben található. Az internet és a hálózatba kapcsolt eszközök közötti összeköttetés is ide kerültek.

Implementáltuk a protokollt, és elkészültek a hálózati eszközök tervei is, valamint megépültek az első prototípusok. Hardverileg némileg különbözőek, készült RGB LED vezérlő, motor vezérlő, és különböző szenzorokkal felszerelt egység is. A különböző hardverekhez különböző szoftvereket is készítettünk

Az elkészült hálózaton méréseket végeztünk. Megmértük a maximális átviteli sebességet, a késletetést, és a csomagvesztést különböző beállítások és körülmények között. A mért eredmények alapján az elkészült rendszert összehasonlítottuk más, már létező megoldásokkal.

1. Bevezetés

Ha szóba kerül az „intelligens otthon” kifejezés, a tipikus asszociációk okos telefonnal vagy weboldalról kapcsolható lámpa, termosztát, esetleg a kerti locsoló vagy riasztórendszer szokott lenni, amit távolról irányíthatunk és elfogadjuk, hogy ettől okos lesz az otthonunk. Ennél azonban messzemenően többről van szó.

1.1. Elhelyezés a világban, más termékek

Az átfogó kép érdekében szükséges feltérképezni a jelenlegi paci szereplőket a kategóriában. Tudatosítani, hogy ma milyen termékekre és szolgáltatásokra mondjuk, hogy intelligens otthoni rendszer. Ennek érdekében vegyük szemügyre lényeges gyártókat.

1.1.1. Philips Hue

Vezeték nélküli, színes, alacsony fogyasztású LED lámpákat kínál melyek egy bridge-en keresztül a helyi hálózathoz csatlakoztatva wifi keresztül irányíthatóvá válnak egy dedikált applikációval. [1]
14 000 – 19 000/ darab.

1.1.2. Belkin WEMO

Az otthoni WiFi routeren keresztül irányítja a csatlakoztatott eszközöket. [2]
Csatlakoztatható eszközök:

- Vezeték nélküli, távolról irányítható LED körte
- WiFi kamera
- Lámpakapcsoló, melyet szintén távolról irányíthatunk
- Okos főző edény, melyet okos telefonunkról irányíthatunk
- Mozgásérzékelő
- Párologtató
- Kávéfőző

1.2. Koncepció

Látható, hogy nagyon széles a paletta, szinte mindent rákötethetünk az internetre. Ez praktikus, de vannak szempontok, amelyeknek nem árt figyelmet szentelni. Az egyik ilyen, hogy ezek az eszközök jelenleg még elég drágák. A gyártók is kis kiszerelésekben árusítják őket, legfeljebb 3-5 darabot egyszerre. Viszont ha egy háztartásban sok okos eszköz van

(akár 100+) fontos szempont, hogy lehetőleg ne kapcsolódjon minden eszköz közvetlenül az otthoni routerünkhöz, ennek nem ez a funkciója. A tendencia pedig az, hogy az okos eszközök száma egyre csak növekszik háztartásonként.

Erre igyekszünk megoldást nyújtani az általunk tervezett rendszerrel, amellyel arra törekszünk, hogy a lehető legtöbb háztartásban jelen lehessenek az okos eszközök nem terhelve az eddig használatban lévő hálózatot, mindenki számára elérhető áron.

Többek között ezért döntöttünk úgy, hogy az nRF24l01+ adó-vevőt felhasználva fogunk hálózatot tervezni. Ami amellett, hogy szakmai kihívást is jelentett, mert ekkora mértékű és ilyen komplexitású hálózatra még nem használták, igen olcsón is kapható.

A dolgozatunk témája intelligens otthoni rendszerhez tervezett ad-hoc, multi-hop hálózat általunk készített, egyedi kommunikációs protokolljának ismertetése, és a hálózat áteresztőképességének (throughput) valamint késleltetésének szemléltetése méréseken keresztül. A hálózatunk a szenzorhálózatokkal szemben támasztott alábbi követelményeket teljesíti:

- rövid várakozási idő
- késleltetés
- alacsony energia felhasználás
- biztonságos kommunikáció
- ezeken felül törekedtünk a csomópontok előállítási - alapanyag - költségeinek minimalizálására

A megbízható kommunikáció csomagok küldése szempontjából értendő, az üzenetek titkosítására egyelőre csak terveink vannak. Ezekről a dolgozat során ugyan említést teszünk, de a fejlesztés következő fázisában fog csak sor kerülni rá.

A konkrét feladatunk tehát ATmega328p mikrokontroller és nRF24L01+ rádiós eszközökből épített multi-hop hálózat önszervező protokolljának kidolgozása, implementálása és tesztelése volt.

Részekre bontva:

1. Útvonalválasztó algoritmus készítése, tesztelése.
2. Szomszédság felderítés segítségével megbízható útvonalak kiválasztása.

3. Különböző lehetséges csomagtovábbítási stratégiák vizsgálata.
4. Hálózatbiztonsági megfontolások, algoritmusok elemzése és saját igényekhez igazítása.
5. Az eszközök megbízhatóságának tesztelése, működésük körülményeinek és feltételeinek felderítése, biztosítása.
6. Webes kliens készítése, amely kommunikál a szenzorhálózati átjáróval és internetről is irányíthatóvá válik a hálózat.

Mérések:

- A kész hálózat áteresztőképességének mérése és összehasonlítása más rendszerekével
- A hálózat késleltetésének mérése és összevetése más rendszerekével.
- A hálózat kiépülésének ideje bekapcsolástól számítva.

A mérésekkel az általunk választott rádiós kiszolgáló technológiát, valamint az erre tervezett protokollunkat fogjuk feltérképezni. Így átfogó képet kapunk arról, hogy a választott hardverek mire képesek a gyakorlatban, valamint a mi protokollunk hol helyezkedik el a manapság intelligens otthoni rendszerekben használt kommunikációs protokollokhoz képest, melyek az érdemi különbségek.

2. Elméleti háttér

Több létező protokoll is rendelkezésre állt, melyek tanulmányozás után végül alkalmatlannak bizonyultak a céljainkra. A következőben ezeket a szabványokat mutatom be. Tekintettel a nagyszámú létező otthon automatizálási szabványra, csak a számunkra relevánsakat értékelem, tehát azokat, amelyek vezeték nélküli rendszerek és hasonló jellemzőik vannak a mi hardverünkhöz adatátviteli sebesség, fogyasztás, frekvencia tartomány és hatótávolság tekintetében. Ezt hivatott az 1. táblázat szemléltetni.

2.1. ZigBee

A ZigBee Alliance fejleszti és az IEEE 802.15 szabványra épül. 2.4GHz –es frekvenciasávban működik, 16 csatornát használ, 250Kb/s a névleges maximális adatátviteli sebessége. AES-128 titkosítás, támogatja a biztonsági kulcs generálást. Rendelkezik

energiatakarékos móddal. 5-500 méter a hatótávolsága a környezeti adottságoktól függően. [3]

2.2. Z-wave

A dán ZenSys cég kezdte el fejleszteni, majd tőlük megvette a jelenlegi tulajdonos, az amerikai Sigma Design. Önszervező, és javító hálózatot épít ki. Útvonal táblákat és szomszédság táblákat használ az útvonalak kiépítéséhez. Különböző országokban más-más frekvencián működik: 868.42 MHz (EU), 908.42 MHz (USA), 919.82 MHz (Hong Kong), 921.42 MHz (Ausztrália). Valamint 9.6 kb/s és 100 kb/s között változhat az adatátviteli sebessége. [4]

2.3. Bluetooth 4.0

A Bluetooth SIG csoport rádiós fejlesztését IEEE 802.15.1 szabványként standardizálták. Ez egy vezeték nélküli, kültéren maximum 100 méter, beltéren 10 méter hatótávolságú pont-pont összeköttetésű ad-hoc hálózatot definiál, amely 2.4GHz es frekvencián üzemel és alkalmazás rétegben 237 Kb/s, fizikai rétegben 1Mb/s adatátviteli sebességgel rendelkezik. A biztonságról 128 bites AES CCM kódolást használ.[5]

2.4. WiFi

A WiFi, azaz IEEE 802.11 szabvány, az utóbbi évek de facto szabványa lett a szélessávú vezeték nélküli adatátvitel tekintetében. A legtöbb notebook beépített WiFi adó-vevővel rendelkezik, az okostelefonoknak alapkövetelménye. Tipikusan csillag topológiában van implementálva nem pedig hálóban, ami pont-pont kapcsolatot jelent. 50-100 méter között változhat a hatótávolsága, ez házon belül töredékére csökken a falak tompítása miatt. [6]

2.5. nRF24l01+

A Nordic Semiconductors által gyártott 2.4 GHz-n működő maximum 2Mb/s névleges adatátviteli sebességre képes adó-vevő. A maximum küldhető üzenet mérete 0-32 byte-ig változhat, támogatja a dinamikus üzenethosszt. Támogat alacsony fogyasztású alvó üzemmódot. [7]

2.6. Az összehasonlítás eredménye

					
Eszköz neve	Z-Wave	ZigBee	Bluetooth 4.0	WiFi	nRF24l01+
Eszközök maximális száma [darab]	232 (javasolt: 30-50)	65536	7	2^32	65536
RF [GHz]	0.868-0.921	2.4	2.4	2.4, 5	2.4
Adatátvitel [Kb/s]	9.6-100	250	237	11262-102400[1]	2048
Csomagméret [byte]	9-21	127 (802.15.4)	1024	576	33
Vételár[2] [HUF/db]	19000	9636	766	1000	264

1. táblázat - A lényeges szabványok összehasonlítása a saját adó-vevővel

A felsorolt technológiákra általánosan igaz, hogy otthoni alkalmazásra nem tudtak széles körben elterjedni, mert drágán hozzáférhetők. Ez nem feltétlenül a technológiából következik, inkább a piaci értékéből. Ezért egyelőre olyan megoldások vannak, ahol pár eszközről szól a vezérlés.

Ebből adódott az első szempontunk a választáshoz: **legyen megfizethető**.

A második szempont az **energiahatékonyság** volt, tehát az eszköz ne rendelkezzen nagyságrendekkel nagyobb erőforrásokkal, mint ami egy szenzorhálózat elemeinél szükséges. Itt esett ki a WiFi lehetősége, mivel ott a minimális adatátviteli sebesség 11 Mb/s,

[1] WiFi szabvány függő az átviteli sebesség: rendre 802.11a: 54 Mb/s, 802.11b: 5,5 vagy 11 Mb/s, 802.11g: 54 Mb/s, 802.11n: 600 Mb/s, 802.11ac: 1300 Mb/s.

[2] Az értékek konzisztensen ugyanabból a nemzetközi webshopból származnak (eBay)

ennyire nincs szükségünk és ez jócskán nagyobb fogyasztást is eredményezne, mint amit szeretnénk (elemről működés). A Bluetooth a pont-pont kapcsolat használata miatt esett ki a versenyből, a sebessége és hatótávolsága is megfelelő lett volna.

Ekkor véglegesítettük a döntést, hogy kommunikációs hardvernek az nRF024l01+ adó-vevőt fogjuk használni, amely meglehetősen olcsón beszerezhető, és teljesíti az összes többi követelményt. A ZigBee és a Z-wave eszközök többszörösébe kerülnek, így ezekről a szabványokról is lemondtunk.

Az is világossá vált, hogy a célhardverre célszerű lenne, de valójában elkerülhetetlen is saját protokollt és alkalmazás réteget tervezni és implementálni.

3. Tervezés, kivitelezés

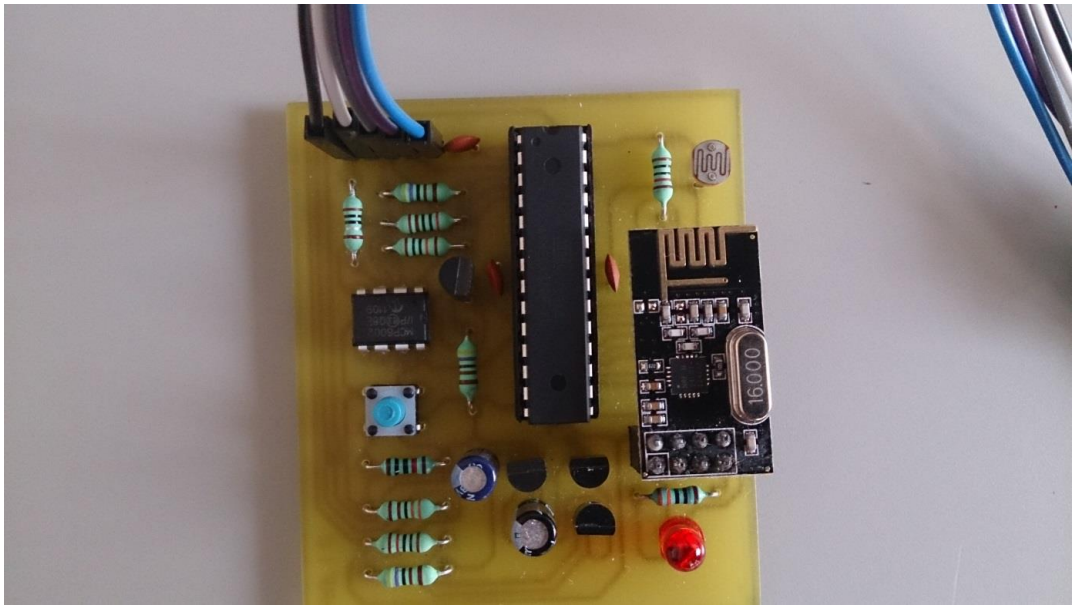
Egy intelligens otthon hálózatban sok, funkcionalitásban jelentősen különböző eszköz áll egymással összeköttetésben. Mivel az eszközök hálózatba kapcsolása általában egy extra szolgáltatás, ezért a megvalósításkor fontos, hogy minél kevesebb problémát okozzon a beüzemelése és működtetése, emellett olcsó is legyen. Ezekből adja magát, hogy ne minden résztvevő legyen egyenértékű. Amiből sok kell, az legyen minél egyszerűbb, a komplex kommunikációs feladatokat pedig egy központi egység végezze. Így egy központosított hálózatot kapunk. A központ a hálózat koordinátora, ami egyben az internetről elérhetőséget is biztosítja, ezért átjárónak (gateway) nevezzük.

3.1. Hardware

3.1.1. Csomópont

A csomópontok (felülnézetből az 1. ábrán látható) saját tervezésű hardveren működnek, melyek ATMega328P mikrokontroller segítségével NRF021L+ típusú adó-vevőn keresztül kommunikálnak egymással és az átjáróval. A mikrokontroller 32 kbyte programmemóriával és 2 kbyte RAM-mal rendelkezik. Az órajele maximum 20 MHz, mi viszont Arduino platformmal használjuk, amely 16 MHz-t támogat. Mind a programmemória, mind pedig a RAM szűkös, ezért a csomópontok szoftverébe csak az kerülhet bele, ami feltétlen szükséges.
Az adó-vevő a 2.4 GHz-s ISM sávban működik, maximum 2 Mbit/s sebességen. A fogyasztása küldéskor 7 mA és 11.3 mA között van az adás teljesítményétől függően, fogadáskor 12.6 mA

és 13.5 mA között a sebességtől függően. Ezek az értékek ideálisak a hálózatunk számára. A mikrokontrollerrel SPI-on keresztül kommunikál.



1. ábra – egy csomópont felülnézeti fényképe

3.1.2. Átjáró

A jelenlegi célhardver egy ARM architektúrájú SCB (Single Board Computer), aminek a szerepét egy Cubieboard tölti be. A számunkra legfontosabb jellemzői az 1 GHz-s dual-core CPU, 1 GB RAM, valamint egy 10/100 Ethernet inferface. Ez jóval nagyobb teljesítmény, mint amire szükség van, a jövőben egy lassabb, de olcsóbb alternatíva töltheti be a szerepét.

Az átjáróhoz az adó-vevőt kétféleképpen is illesztettük, az egyik irány az volt, hogy egy mikrokontrolleren (szintén ATMega328p) keresztül kommunikál vele. Ilyenkor a csomagok soros porton továbbítódnak az átjáró és a mikrokontroller között, valamint SPI-on a mikrokontroller és az adó-vevő között. Ennek előnye, hogy az átjárónak így nem kötelező az SPI interfész, ezért például egy PC-n is futhat. Hátránya, hogy egy speciális elem a hálózatban, ami plusz hibaforrás lehet, valamint a kontroller maximum 115200 baud sebességet támogat a soros porton, ami jelentősen lassabbá teszi így az átjárót, mint a többi csomópontot.

A másik lehetőség az adó-vevő közvetlen csatlakoztatása az átjáróhoz SPI-on. Ez a Cubieboard esetében könnyű, mivel 4 darabbal is rendelkezik a szükséges interfészből. Ennek előnye, hogy rendelkezésünkre áll a maximális sebesség, amire képes az adó-vevő, és

ugyan azt a meghajtó programot használhatjuk, mint a csomópontoknál. A fejlesztést viszont kicsit megnehezíti, mert különválik a fejlesztő és a célplatform.

3.2. A protokoll tervezése

A protokoll megalkotásának kiindulópontja a felé támasztott követelmények lefektetése volt. Ezek közül a legfontosabb, hogy legyen képes a közvetlenül nem szomszédos csomópontokkal is kommunikálni. Támogasson továbbá megbízható csomagküldést, legyen lehetőség egy sikertelen küldés érzékelésére, kezelésére. Az egyes csomópontok állapotának is lekérdezhetőnek kell lennie. Ezeken felül legyen egy könnyen használható interfésze a különböző alkalmazás profilok számára.

Összegezve tehát a protokoll funkciói a következők:

- multi-hop képesség
- lehetőség megbízható csomagküldésre
- hálózati diagnosztika
- alkalmazások könnyű illesztése

3.2.1. Multi-hop

A hálózat nagy előnye, hogy nem szükséges minden csomópontnak az átjáró hatókörében lennie. A protokoll több csomóponton keresztül is képes kiépíteni az útvonalakat és lehetővé teszi a kommunikációt törekedve a lehető legkisebb csomagvesztésre. Ezáltal minden csomópont el tudja érni az átjárót, az átjárón keresztül pedig bármely más csomóponttal kommunikálhat.

3.2.2. Megbízható csomagküldés

Két kommunikáló fél között a küldés megbízhatósága a kézbesítést visszaigazoló üzenettel (ack) és újraküldéssel növelhető. Mivel a hálózatban sok különféle eszköz található, nehéz meghatározni egy olyan maximális újraküldés számot, amely minden esetben megfelelő megbízhatóságot biztosít. Például egy hőmérséklet szenzor esetében általában nem probléma, ha néhány mérés elveszik, az viszont kellemetlen, ha egy lámpakapcsolót többször kell megnyomnunk csomagvesztés miatt. A legegyszerűbb megoldás, ha mindig a megengedett legtöbbször küldünk újra, amíg nem jön visszaigazolás. Ez azonban jelentősen terheli a hálózatot, valamint energiapazarlás is, ami ebben a

környezetben kifejezetten kerülendő. Ehelyett minden csomagra megadhatjuk, hogy mekkora megbízhatósággal szeretnénk kézbesíteni, és ez adja meg az újraküldések számát.

3.2.3. Diagnosztika

A hálózat üzemeltetéséhez elengedhetetlen, hogy ismerjük a csomópontok aktuális állapotát. A legfontosabb információk az elérhetőség (működik-e), a rajta keresztül elérhető csomópontok listája, valamint a szomszédai és azok megbízhatóságai. Ezeket az adatokat az átjáró gyűjti be, és ezek alapján végez olyan hálózat menedzsment feladatokat, amelyeket a csomópontok nem képesek a szűkös erőforrásaik miatt. Ezek közé a feladatok közé tartozik például a torlódás kezelés vagy az olyan útvonalak újratervezése, amelyek egy kieső csomópont miatt nem működnek tovább.

A csomópontok állapotának lekérdezésére a következő diagnosztikai eszközök kerültek implementálásra a protokollban:

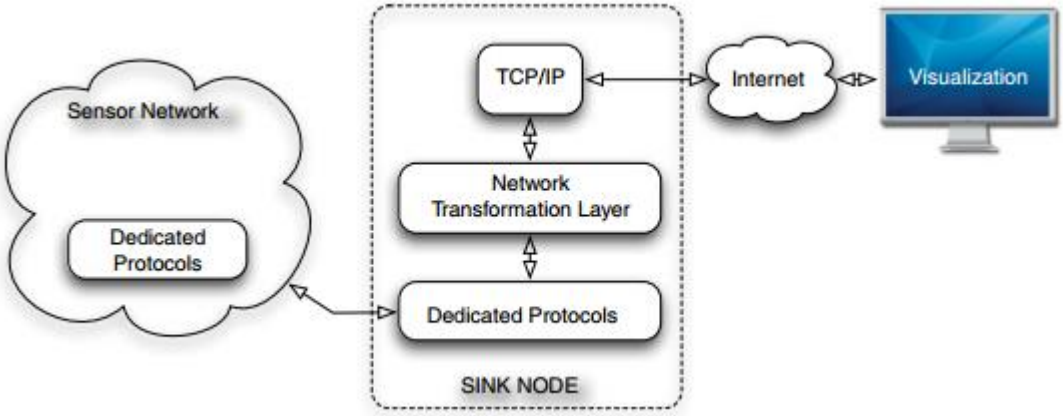
- Pingelés: Körülfordulási időt (RTT - Round Trip Time) mér a célcímként megadott csomópont és a küldő között.
- Routing Tábla lekérdezés: A csomópont elküldi, kik szerepelnek az útvonal táblájában.
- Neighborhood Tábla lekérdezés: A csomópont elküldi, kik szerepelnek a szomszédság táblájában.

3.2.4. Alkalmazások könnyű illesztése

A protokoll nem kizárólag okos otthon hálózatokban alkalmazható, hanem bármilyen olyan esetben is ahol sok, kis teljesítményű, viszonylag nagy területet lefedő hálózatra van szükség. Ez megkövetel egy egyszerű, de hatékony interfészt az alkalmazás és a hálózat réteg között. Az alkalmazásnak lehetősége van definiálni a saját üzenet típusait, megadni a protokoll működését befolyásoló paramétereket, továbbá egy üzenet kezelőt implementálni, ami az alkalmazás-rétegbeli üzeneteket kezeli. Esetünkben az okos otthon csomópont, és az átjáró egy-egy ilyen alkalmazás.

Az átjáró az alkalmazás rétegben található. Két végpontja van, az egyik egy HTTP-n kommunikáló webservice, a másik pedig az nRF24l01+ adó-vevő. Ezek segítségével vezérelhetővé válnak a helyi (LAN) hálózatról illetve az internetről (WAN) is amennyiben az

átjáró számára biztosított az internet kapcsolat. Ezek után az átjáró webservice-e könnyen irányítható külső szoftverekből HTTP GET és POST üzenetekkel. Ez képezi az alapját az android kliensünknek és a Djangoban írt front endnek is. A proxy alapú internetelérést szemlélteti az alábbi ábra (2. ábra), a mi rendszerünk is erre a logikára épül.



2. ábra – Proxy alapú internetelérés [8]

3.3. Specifikáció

A meghatározott funkciók alapján specifikálni tudjuk a részleteket a protokoll implementációjához.

3.3.1. Hardveres adottságok

A specifikáció során figyelembe kell venni a célhardverek képességeit. Az egyik a rádió adó-vevő, amely a fizikai réteget biztosítja, a másik a mikrokontroller, ami a protokollt futtatja. Az adó-vevő releváns paramétere a maximális csomag méret, ami 32 byte. A választott mikrokontrollernek a protokoll szempontjából a legfrontosabb paramétere a memória, amelyből 2 kbyte áll rendelkezésünkre. Ezek tekinthetők a protokoll minimális követelményeinek, ennél hosszabb csomagot küldő, vagy több memóriával rendelkező komponensekre könnyedén lehet váltani.

3.3.2. Az üzenetek fejléce

3.3.2.1. Adatkapcsolati réteg

Az adó-vevő által használt fejléc az alábbi ábrán (3. ábra) látható.

Preamble 1 byte	Address 3-5 byte	Packet Control Field 9 bit	Payload 0 - 32 byte	CRC 1-2 byte

3. ábra – az adó-vevő által használt fejléc

Preamble

Az értéke 01010101 vagy 10101010, a vevő ennek a segítségével kerül szinkronba az adóval.

Address

A cél cím. Esetünkben 5 byte, az első 3 értéke 0xF0F0F0, ez segít szinkronban tartani a kommunikáló feleket, az utolsó kettő pedig a hálózati rétegbeli cím.

Packet Control Field

Az első 6 bit a payload hosszát tartalmazza, amennyiben az dinamikusan változik. A következő kettő egy csomag azonosító, amely a többször fogadott csomagok többszöri feldogozásának megakadályozására szolgál. Az utolsó bit egy NO_ACK flag, amivel a hardver szintű autómatikus visszaigazolást lehet kikapcsolni csomagonként, például multicast küldés esetén hasznos.

Payload

A csomag felhasználó által megadott tartalma. Az adó-vevő támogatja a dinamikus payload hosszt, mi azonban fix 24 bájttal használjuk, mert a legtöbb esetben megközelítőleg ekkora a küldött adat, és kisebb memória és számítás igényű a kezelése, mint a dinamikusnak.

CRC

A hibásan fogadott üzenetek detektálására alkalmas ellenörző összeg. 2 byte méretűre állítottuk, mert így nagyobb megbízhatósággal működik. A kezdeti értéke 0xFFFF, a polynom a kövezkező: $x^{16}+x^{12}+x^5+1$

3.3.2.2. Hálózati réteg

A hálózati rétegben a csomagok 24 byte-ból állnak, ebből 13 a fejléc, a többi pedig a payload. A hálózati rétegben a címeket 2 byte-osnak választottuk meg. 1 byte az nagyon kevés lett volna, 3 esetén pedig a kioszthatő címek száma (~16 millió) jóval több, mint amennyit a rendelkezésre álló erőforrásokkal kezelni lehet. A fejléc felépítése a 4. ábrán látható.

Type + flags1	Source address	Destination address	Message id	Fragment offset	Reserved	Hop count + flgas2
1 byte	2 byte	2 byte	2 byte	1 byte	4 byte	1 byte

4. ábra – a hálózati réteg fejléce

Type and Flags1

A felső három bit flageket tartalmaz, a következő sorrendben:

- *FLAG_FRAGMENTED*:

Ha az üzenet több töredékből áll, akkor 1 az értéke

- *FLAG_IS_RESPONSE*:

Mivel a kommunikáció jelentős része kérdés-válasz párokból áll, ez a flag ezt dönti el. Ha 0, akkor ez üzenet egy kérés, ha 1 akkor válasz.

- *FLAG_NEED_TO_ACK*:

Ha az értéke 1, az jelzi a szállítási protokollnak, hogy visszaigazolás szükséges. Végpontok közötti, ha küldő nem kapja meg adott időn belül a visszaigazolást, akkor újra küldi a csomagot.

Az alsó öt bit megadja, hogy a payload részben milyen protokoll van használva. A 32 különböző érték közül jelenleg a következők vannak lefoglalva:

- *TYPE_ROUTE* = 1:
Útvonalválasztó protokoll, új útvonalak felvétele vagy törlés

- *TYPE_PING* = 2:
Diagnosztikai protokoll, a fogadó visszaküldi az üzenetet amint megkapja

- *TYPE_SEND_TABLE* = 3:
Diagnosztika protokoll, a csomópont routing táblájának és szomszéd listájának küldése

- *TYPE_ACKNOLEDGEMENT* = 4:
Szállítási protokoll, visszaigazoló üzenet

- *TYPE_ERROR* = 5:
Diagnosztika protokoll, hibák jelzése

- *TYPE_ROUTE_TABLE_EDIT* = 6:
Útvonalválasztó protokoll, meglévő útvonalak módosítása

- *TYPE_FRAGMENT_START* = 7:
Szállítási protokoll, jelzi, hogy töredezett üzenet fog érkezni a feladótól

- *TYPE_NEIGHBORHOOD_DISCOVERY* = 8:
Útvonalválasztó protokoll, a szomszédok és azoknak a megbízhatóságát deríti fel

Source address

A feladó végpont hálózat rétegbeli címe.

Destination address

A célvégpont hálózat rétegbeli címe.

Message identifier

Egy 2 byte-os érték, ami azonosítja az üzenetet. Segítségével elkerülhető például, hogy a csomópont kétszer feldolgozza ugyanazt, vagy összeegyeztethetők a kérdés-válasz párok. Az értékét a program futásának kezdete óta eltelt idő szerint kapja, abban a pillanatban, amikor az üzenet objektum létrejön, a processzoridő lesz a sorszám. Ezt igény szerint meg lehet változtatni

Fragment offset

A protokoll lehetőséget biztosít a payload méreténél hosszabb üzenetek küldésére is. Ebben az esetben ez a mező adja meg, hogy a töredék hol helyezkedik el a teljes üzenetben. Maximum 255 az értéke, ezért 256*11 = 2816 byte a leghosszabb, amit a küldeni lehet.

Reserved

Jelenleg nem használt terület. Továbbfejlesztéshez van fenntartva, például a biztonsághoz lehetne használni.

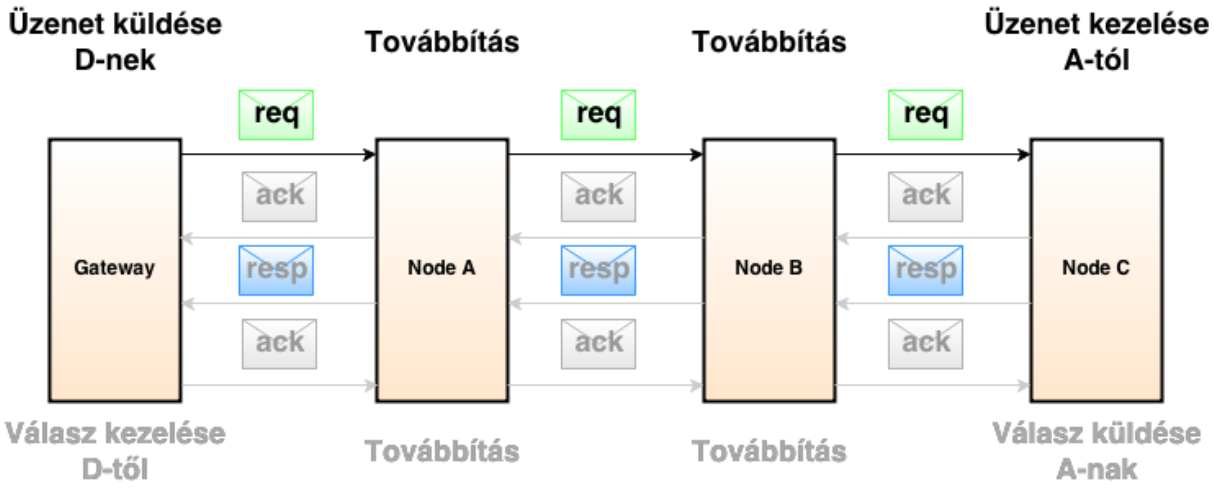
Hop count and Flags2

A felső két bit flageknek van fenntartva, jelenleg nem használtak. Az alsó hat az üzenet által megtett hop-okat számlálja. Habár a hálózatban lévő útvonalak körmentesek, előfordulhatnak hibák, amikor mégis körbe-körbe kering egy üzenet. Ennek a megakadályozására szolgál ez a mező. 0-ról indul, és minden hop-nál egyel, növekszik. Ha elér egy küszöb szintet, a csomag nem továbbítódik. Ez alapján meg tudjuk mondani, hogy egy csomag milyen messziről jött.

3.3.3. A kommunikáció folyamata

Az alábbi ábra (5. ábra) szemlélteti a kommunikáció folyamatát egy csomópont és az átjáró között. Mint az ábrán látható a kommunikációban további 2 csomópont is részt vesz, ezek a köztes csomópontok, akik csak továbbítják az üzeneteket, de mivel nem ők a címzettjei nem dolgozzák fel azt. Miután a címzett félhez eljut a kérő üzenet (request), a

fogadó fél visszaigazolást küld a kérőnek, hogy sikeres volt a küldés és megérkezett az üzenet. Ez a visszaigazolás az ack (acknowledgement). Miután a fogadó fél feldolgozta a kérést, elküldi rá a választ, melyre szintén várja a visszaigazolást. A választ ugyanúgy továbbítják a köztes csomópontok, mint a kérő üzenetet. Miután a válasz üzenet is megérkezik, a sikeres megérkezésről elküldi az teljes kérést kezdeményező csomópont a visszaigazoló üzenetet, ezzel le is zajlott egy kérés-válasz üzenetváltás.



5. ábra- egy kérés-válasz üzenetváltás visszaigazolással

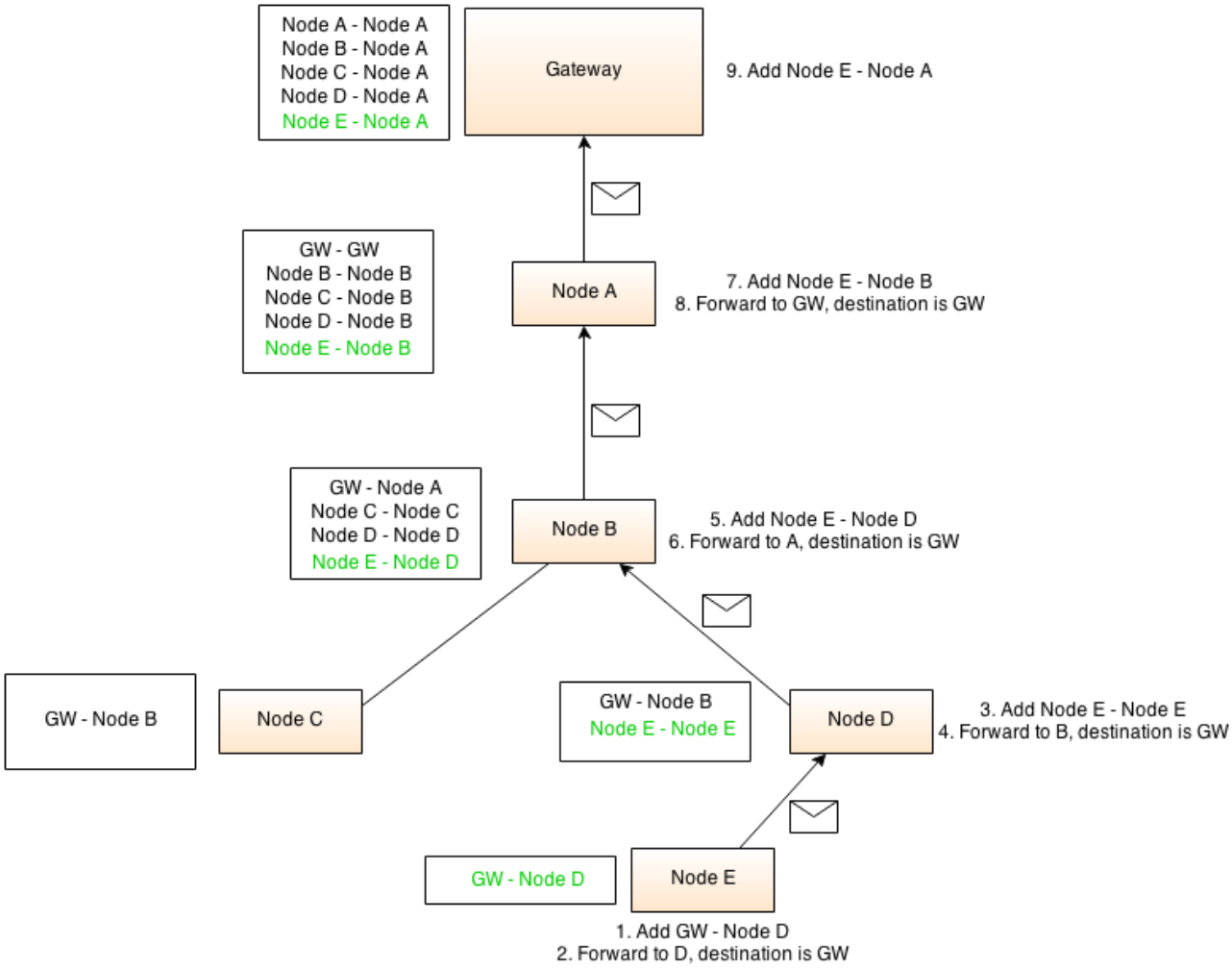
3.3.4. Multi-hop specifikáció

A hálózat a multi-hop képességét az útvonal választó protokollal valósítja meg. Ennek a feladata, hogy a csomagokat eljuttassa azokba a csomópontokba is, amik a küldő hatótávolságán kívülre esnek. A küldőnek elegendő azt tudnia, hogy ha egy adott cél címnek szeretne küldeni, akkor melyik szomszédjának továbbítsa a csomagot, amely továbbítani tudja azt a cél felé, a teljes útvonalat nem tudja.

Ehhez az útvonalválasztó protokoll minden csomópontban egy célcím - szomszéd cím összerendelésekből álló táblázatot tart nyílván. Mivel a hálózat központosított, minden csomópont rendelkezik egy bejegyzéssel a táblájában az átjáró felé, Továbbá minden olyan útvonal is egy-egy bejegyzés, amelyben szerepel a csomópont.

A hálózatban akkor keletkezik új útvonal, mikor bekapcsolódik egy csomópont. Ehhez első lépésként az új csomópont felderíti a szomszédságát. A felderítésre csak azok válaszolnak,

akik már a hálózat részei, tehát ismernek egy útvonalat az átjáróig. Ez után kiválasztja az egyik szomszédját (a választás folyamata a következő pontban kerül részletezésre), felveszi a táblájában az átjáróhoz vezető szomszédként, és rajta keresztül küld egy útvonal építő üzenetet, aminek az átjáró a címzettje. A csomag a szomszéd által ismert útvonalon eljut az átjáróig. Minden csomópont, ami továbbítja az útvonal építő üzenetet, felvesz egy bejegyzést a táblájába, amelynek a cél címe az üzenet feladója, a szomszéd címe pedig az, akitől közvetlenül kapta. így amikor beér az üzenet az átjáróba, kiépül az új csomópont és közte az útvonal. Ez látható a 6. ábrán.



6. ábra – útvonal kiépülése

Ezek alapján látható, hogy egy új hálózat kiépülése az átjárónál indul, és fokozatosan halad a távolabbi csomópontok felé, valamint a létrejövő utak körmentesek. Először csak az kaphat címet, aki az átjáró közvetlen szomszédja, mivel a címkérő üzenetet csak a már a hálózatban

lévő csomópontok továbbítják. Mikor az első csomópont megkapja a címét és felderíti a szomszédságát, csak az átjáró fog válaszolni neki, és ő és a csomópont között lesz az első útvonal. A továbbiakban egy kialakuló útvonal vagy közvetlen az átjáróhoz kapcsolódik, vagy egy olyan út folytatása, aminek a túlsó vége az átjáró. Egyik esetben sem lehet kör a kialakult út, ezért a hálózat is körmentes lesz.

3.3.5. Megbízhatóság specifikációja

3.3.5.1. Szomszédság felderítés (Neighborhood Discovery)

Ahhoz, hogy a lehető legmegbízhatóbb útvonalak épüljenek ki, egy csomópontnak tudnia kell, hogy melyik szomszédja mennyire megbízható, azaz az elküldött csomagok közül megközelítőleg hány fog megérkezni.

A szomszéd felderítés folyamata a következő:

1. A felderítést végző csomópont adott időn belül adott mennyiségű csomagot küld broadcast-en
2. A közelben lévő csomópontok megkapnak ebből valahány darabot. Az első csomag beérkezésekor megjegyzik a beérkezés idejét, és elkezdik számolni a megkapott csomagokat.
3. Miután a fogadó oldalon eltelt annyi idő az első csomag beérkezése után, mint amennyi ideig a felderítő csomópont küldi az üzeneteket, akkor már biztosan nem fog több csomag jönni. Ekkor, ha egy küszöbszintet elér a beérkezett csomagok száma, akkor visszaküldi a felderítőnek, hogy az elküldött csomagok hány százalékát kapta meg, valamint hogy mekkora megbízhatóságú az útvonala a gatewayhez, és felveszi a felderítést végző csomópontot a szomszédai közé.
4. A felderítő csomópont a küldések után várakozik a válaszokra. Mivel akár több tíz szomszédja is lehet, aki válaszolna, azonnali küldéssel nagy lenne az ütközés esélye, ezért azok különböző ideig várakoznak küldés előtt, minimalizálva ezzel az ütközés esélyét. A várakozási idő így összevethető a küldések idejével.

Mivel a felderítés implementálását jelentősen bonyolította volna, ha egy időben több is futhatna belőle párhuzamosan, és a programmemória szűk keresztmetszet, ezért

megszabtuk, hogy egyszerre csak egy csomópont végezhet szomszéd felderítést, és ezt az átjáró ütemezi.

Mivel a csomópont működéséhez elengedhetetlen, hogy az átjáróhoz legyen megbízható útvonala, ezért a címosztás és a szomszéd felderítés folyamata össze lett kapcsolva. A beérkező címkéréseket az átjáró egy várakozási sorba rendezi, és a sorban állók közül mindig azt szolgálja ki, amelyik kérés a legközelebbről jött (legkisebb a csomag hop count-ja), ezzel is segítve a hálózat középről kifelé felépülését. Egy cím kiküldése után addig nem szolgál ki újabb kérést a sorból, amíg arról a címről nem érkezik vissza, hogy végzett a szomszéd felderítéssel, vagy el nem telik annyi idő, amennyi után már biztos, hogy vége van. Ez garantálja, hogy egyszerre maximum egy legyen folyamatban.

A felderítés paramétereit úgy választottuk meg, hogy ne tartson túl sokáig, ne terhelje nagyon a hálózatot, de ehhez képest valós képet mutasson a szomszédok megbízhatóságáról. A cél az volt, hogy egy felderítés körülbelül egy másodperc legyen, és 50 szomszéd esetén még működjön.

A válaszok ütemezését 5 ms pontossággal próbáltuk egyenletesen elosztani, így a várakozási idő 250 ms lett. A küldési időpillanatot a csomópont címéből számoljuk, egy kicsi "véletlenszerű" eltolással az idő függvényében:

```
offset = (address modulo 50) * 5 + microseconds modulo 5
```

Az address a csomópont címe, a microseconds pedig a csomópont indulása óta eltelt mikroszekundumok száma. Az így kapott érték miliszekundumban értendő. Így a küldés időpontja:

```
Az első csomag beérkezése + felderítő küldésének maximális hossza + offset
```

Feltételezhetjük, ha egy csomópont az üzenetek első feléből nem kap meg egyet sem, akkor nem fog választ küldeni a felderítésre. Legrosszabb esetben a fogadó fél épp az utolsó n darab csomagot kapja meg, amivel még a küszöbérték felett lesz, tehát válaszolni kell a felderítőnek. Viszont nem tudja, hogy hányadik csomagot kapta meg, ezért végig kell várnia a felderítő maximális küldési idejét. A küldési idő kivárása után kiszámolja, hogy az válaszra szánt időrésben mikor küld, ez maximum 250 ms. Ebből következik, hogy a felderítőnek a

válaszokra szánt időrésem felül a küldési idő felét is ki kell kivárnia még egyszer (mivel lehet, hogy egy válaszoló épp a küldési idő felénél kapta meg az első csomagot, onnan viszont megkapott annyit, amennyi a válaszhoz kell), hogy minden választ megkapjon. Az egy másodperces felderítésből a várakozásokat levonva maradt tehát 500 ms. Ha egy üzenet elküldését átlagosan 1 ms-nek vesszük, és a hálózatot csak 10%-ban szeretnénk leterhelni, akkor 10 ms-enként küldhet a felderítő csomagot, ami 500 ms esetén 50 db küldést jelent. A tapasztalataink alapján ezzel jól lehet becsülni a tényleges megbízhatóságot.

3.3.5.2. Útvonalválasztó (Routing) algoritmus

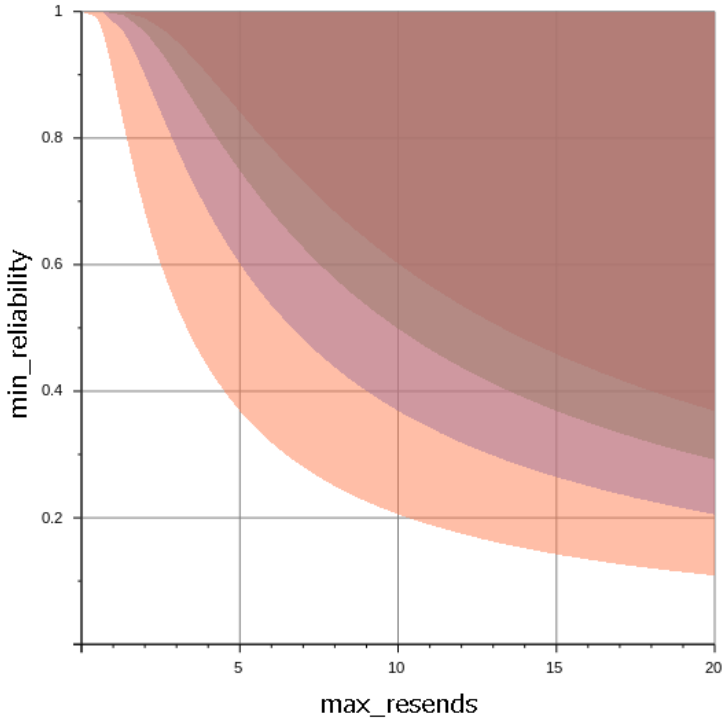
Az útvonalválasztás teljesen a megbízhatóságra épül. Annál jobbnak tekintünk egy útvonalat, minél kisebb rajta a csomagvesztés valószínűsége. Új útvonal a hálózatban egy szomszédság felderítés után keletkezik, amikor a felderítő kiválasztja, hogy melyik szomszédján keresztül fog kommunikálni az átjáróval. Egy csomópontban az átjáró megbízhatósága a csomópont és az átjáró között lévő úton a szomszédok között lévő megbízhatóságok szorzata. A felderítés végén a megbízhatósági küszöböt átlépő csomópontok elküldik, hogy ők mennyire megbízhatóan látják az átjárót, valamint a felderítőt. Az átjáró és a felderítő megbízhatóságának szorzata pontosan a felderítőtől az átjáróig vett út megbízhatósága lesz, ezek közül választja ki a maximálisat.

3.3.5.3. Megbízható csomagküldés

A rendszer több különböző megbízhatósággal képes üzenetet küldeni. Ezt csomagonként adhatjuk meg, méghozzá úgy, hogy megadjuk, hogy hány kilences valószínűséggel szeretnénk a csomagot sikeresen továbbítani. A sikeres csomagküldést két fő paraméter határozza meg: az újraküldések száma, és a minimum útvonal megbízhatóság, amit megengedünk a hálózatban. Egy adott valószínűséghez tartozó minimális értékeket a következő egyenlőtlenségből kaphatjuk meg:

$$p <= 1-(1-min\_reliability)^{max\_resends}$$

Ez grafikonon ábrázolva p <= 0.9, p <= 0.99, p <= 0.999 és p <= 0.9999 értékekre, a 7. ábrán látható képet kapjuk.



7. ábra – a megbízhatósági egyenlet ábrázolva

A maximális újraküldési számot 15-ben limitáltuk. Ekkor a minimum megbízhatóság, ami mellett már négy kilences valószínűséggel sikeres a küldés:

```
0.9999 <=1-(1-min_reliability)^15
```

amiből rendezés után a következő adódik:

```
min_reliability >= 0.45883
```

Egy útvonalnak minimum ekkora megbízhatósággal kell rendelkeznie, hogy kialakulhasson.

4. Mérések

Az elkészült rendszer viselkedésének megismeréséhez méréseket végeztünk rajta. Először megnéztük a hardver maximálisan mekkora sebességre képes. Ezt követően az általunk létrehozott protokoll képességeire voltuk kíváncsiak. Először a késleltetés alakulását néztük meg több hopon keresztül, majd a maximális átviteli sebességet. Lemértük továbbá a hálózat kialakulásának idejét, és a csomagvesztések alakulását több falon keresztül.

4.1. A hardverek maximális teljesítményének mérése

Ehhez először viszonyítási alapként megmértük, hogy a kiválasztott mikrokontroller és adó-vevő maximálisan mekkora átviteli sebességre képes a gyakorlatban. Egy olyan minimális, 2 mikrokontrollerből és 2 adó-vevőből álló mérő rendszert raktunk össze, ahol az egyik fél folyamatosan ad a lehető leggyorsabban, a másik pedig megnézi, hogy egy adott időintervallumban hány byteot képes így fogadni.

A legnagyobb sebesség eléréséhez célszerű a lehető legnagyobb csomagméretet választani, ezért az adó-vevőn a maximális, 32 byte-ot állítottuk be. Emellett kikapcsoltuk az automatikus visszaigazolást. Az adási sebesség az 2 mbit/s, az adási teljesítmény pedig a maximum, ez az összes teszt során így volt. Az időintervallumot 10 másodpercnek választottuk, majd négyszer lefuttattuk a vevő felet. Az eredményt fogadott csomagokban írta ki. amelyek a következők:

35473

35473

35470

35471

Az eredmények közötti különbségek elhanyagolhatók. A négy mérés összege 141887 csomag. Egy csomagban az adó-vevő fejléce, valamint a 32 byte payload volt. A fejlécben 5 byte-os címet és 2 byte-os crc-t használtunk, ezért összesen 73 bit volt a mérete. Payloaddal együtt 329 bit. Az átküldött bitek száma így 46680823. Ez 1167020 bit/s, ≈1.113 Mbit/s azaz az adó-vevő névleges teljesítményének több, mint a fele, ami nagyon jó eredménynek számít. Ezt valós környezetben természetesen nem lehet reprodukálni az összetettebb protokollok és programkód miatt.

4.2. A rendszerünk mérése

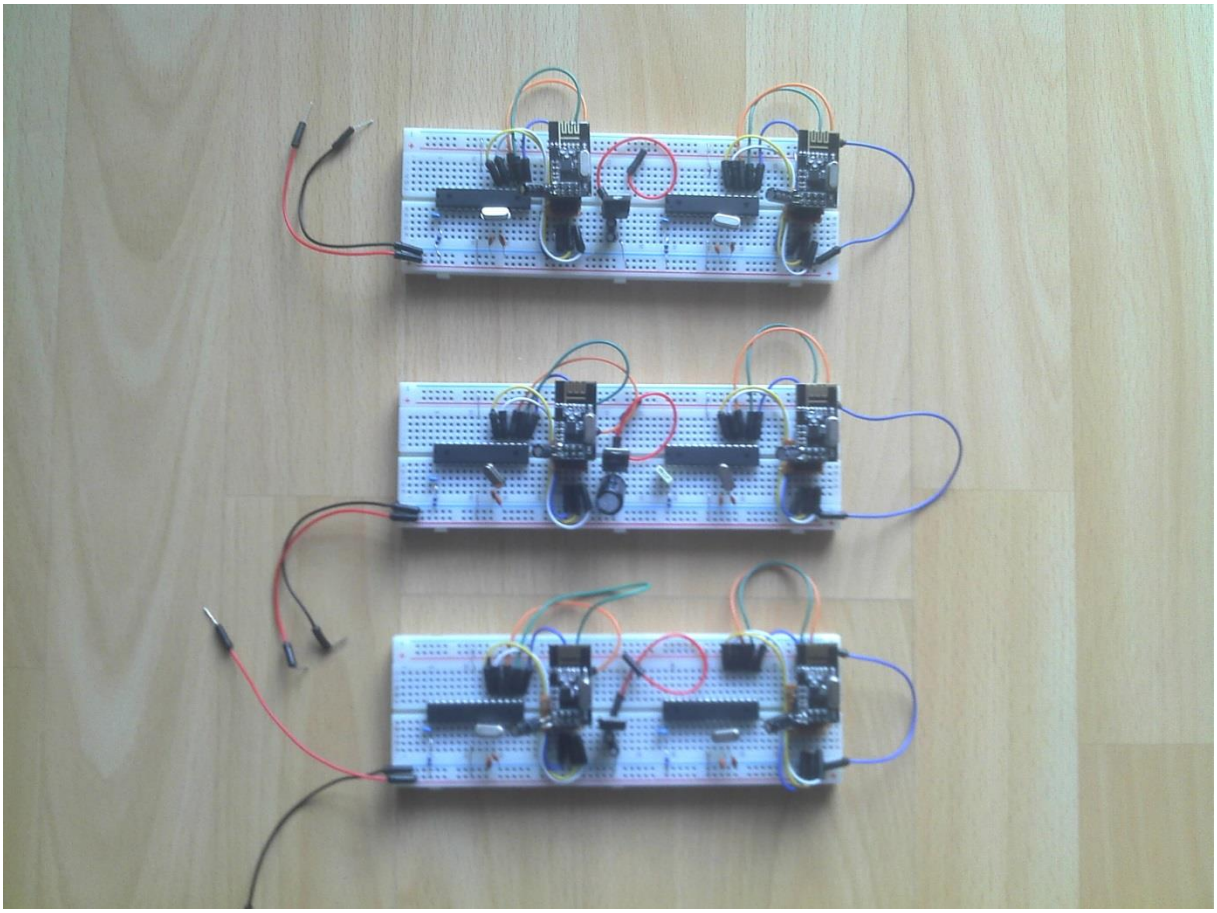
A mérések során fontos, hogy a lehető legkisebb overheaddel dolgozzunk, a mért értékek rögzítése minél kevésbé torzítsa el azokat. Emiatt mérés közben az eredményeket nem lehet soros porton keresztül kiküldeni, mivel 115200-as baud esetén is 10/115200, ≈ 86.8us egy karakter, egy 5 karakteres szám 434us, azaz kis híján 0.5 ms. Ehelyett azt választottuk, hogy az eredményeket a RAM-ban tároljuk, így minimum egy nagyságrenddel pontosabb eredményeket kapunk. Problémát okoz viszont a RAM szűkössége. Az eszköz indulása óta

eltelt mikroszekundumokat 4 byte-on lehet lekérdezni, ebből maximum 100 férne el. Ennek a felbontása 4 us, és kb. 70 percenként csordul túl. Viszont ha megelégszünk a 100us-es felbontással, és a jóval gyakoribb, kb 6.5 másodpercenkénti túlcsordulás sem probléma, akkor a mikroszekundumokat elosztva 100-al már 2 byte-on tárolhatjuk az eredményt. Esetünkben a 100us elfogadható, túlcsordulás pedig 6.5 másodpercenként következik be. 2 byte-tal 200 mérési eredményt tudunk eltárolni, ami a legtöbb esetben megvan a túlcsordulás előtt, de ha mégsem az sem probléma, utólag a túlcsordulási részhez hozzáadva 2^16-ont megkapjuk a pontos eredményt.

Fontos, hogy a mérések előtt legyen a mérendő eszközöknek egy közös időpillanata, amihez képest nézzük az eredményeket. Erre a célra egy broadcast üzenetet használtunk. Mivel azt egyszerre kapja meg az összes csomópont, ha eltárolják, akkor ahhoz képest nézve a csomagok beérkezését pontosan tudhatjuk, hogy mikor melyik eszközön járt.

A mérések megkezdése előtt lemértük 3 saját készítésű csomópontnak az órajelét egymáshoz képest, hogy mekkora különbség van köztük, és az nem okozhat-e mérési hibát. Ehhez egy gombot kötöttünk a három csomópontra, amivel szinkronizálni tudtuk őket, majd 10 miliszekundumonként megnéztük, hogy mennyi idő telt el rajtuk. Kiderült, hogy 10 másodperc alatt több, mint 35 ms csúszás is keletkezik, ami nem megengedhető. Az általunk készített eszközök tehát nem alkalmasak a mérésre, mert a belső órajelük nagyon pontatlan.

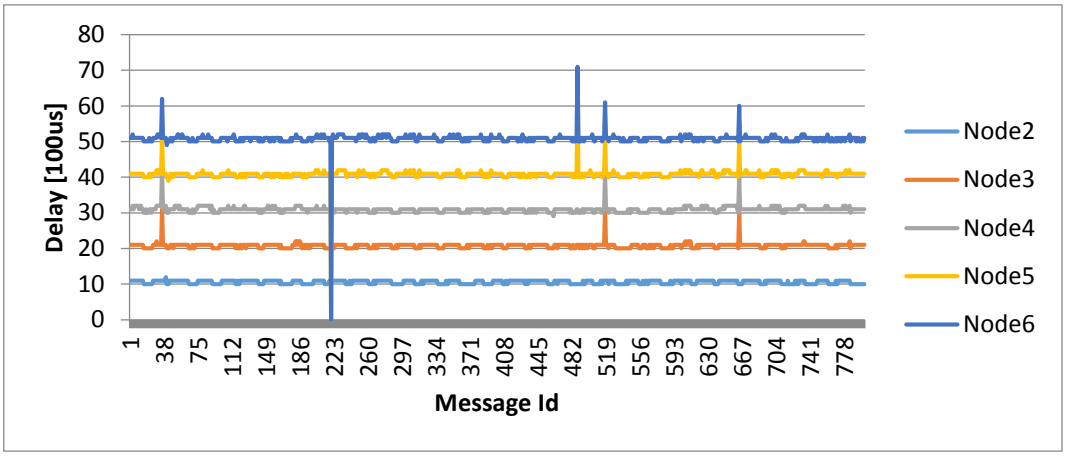
Ezért a méréseket breadboardon összerakott csomópontokon végeztük külső órajellel. A külső órajelek 30 ppm pontosságúak, így jóval hitelesebb lesz a mérés is velük. A mérési elrendezést a 8. ábra szemlélteti.



8. ábra - az 1. mérés elrendezése

4.2.1. Késleltetés vizsgálata

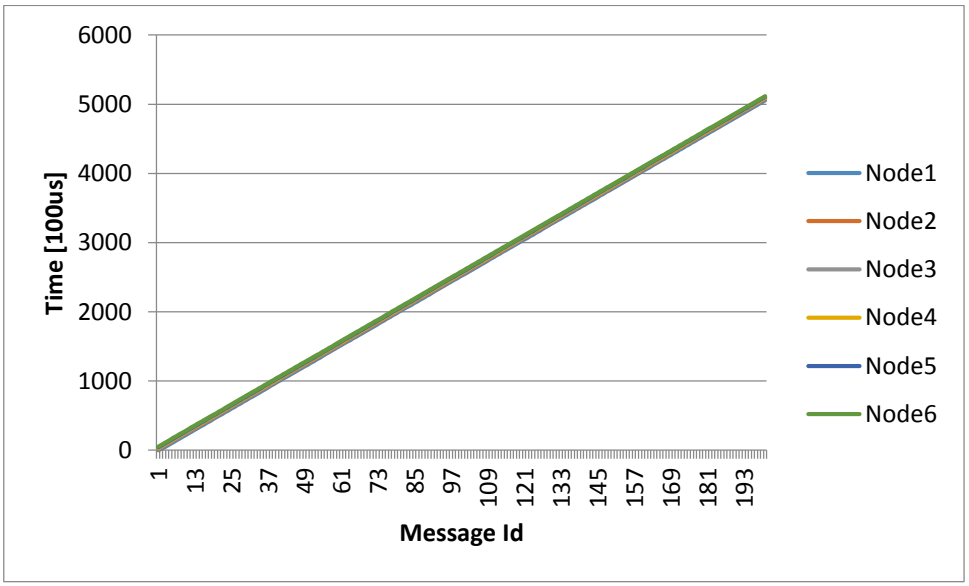
Ehhez a méréshez 6 csomópontot használtunk, Node1-től Node6-ig. Az üzenetek ütemezését a Node1 végezte, annak az idejét vettük nullának. Az ábráról jól leolvasható, hogy egy hop körülbelül 1 ms. Emellett a felfelé álló tüskék a hardveres újraküldést jelzik, a nullába tartók pedig az elvesztett csomagokat. A mérés során szoftveres visszaigazolás nem volt. A mérés eredménye grafikonon ábrázolva látható a 9. ábrán.
Az ábráról leolvasható az 5 csomóponton-on mért idők egymáshoz és a 0. referencia csomóponthoz viszonyított eltérése.



9. ábra - 1. mérés eredményei

4.2.2. Átviteli sebesség mérése

Az átviteli sebesség méréséhez késleltetés nélkül, folyamatosan küldtünk csomagokat, és megnéztük, mennyi idő alatt sikerül elküldeni adott számú csomagot. Itt 200 db 33 byte hosszúságú csomag 500 ms körül érkezett meg az utolsó nodera. Ez 400 csomagot jelent másodpercenként, ami 105.6 Kbit/s sebességnek felel meg. Ez látható a 10. ábrán.



10. ábra - a 2. mérés eredményei

A kapott 105.6 Kbit/s a rádió adó-vevő 2 Mbit/s-os elvi maximumához képest (ami a méréseink szerint valójában 1.113 Mbit/s) azért lett ennyire kevés, mert a felhasznált mikrokontrolleren (ATMega328p) a protokoll futása nem a leggyorsabb. Ezen az implementáció optimalizációjával lehet még gyorsítani. Ekkora sebességgel egy másodperc

alatt nagyjából 6x el tudja küldeni a RAM teljes tartalmát, ami 2KB. PCM kódolással modulált hangot már tudnánk rajta továbbítani, ami 64 Kbit/sec-et igényel, ami viszont csomagvesztés toleránsabb a mi rendszerünknél. A gyorsaság nem az elsődleges cél volt, mert megbízhatóságra törekedtünk, a szenzorokról ritkán olvasunk le értékeket, viszont amikor leolvassuk akkor nagyon fontos, hogy megérkezzen az átjáróhoz. Ezt a mérések alapján kifogástalanul teljesíti a hálózat.

4.2.3. Hálózat kiépülésének mérése

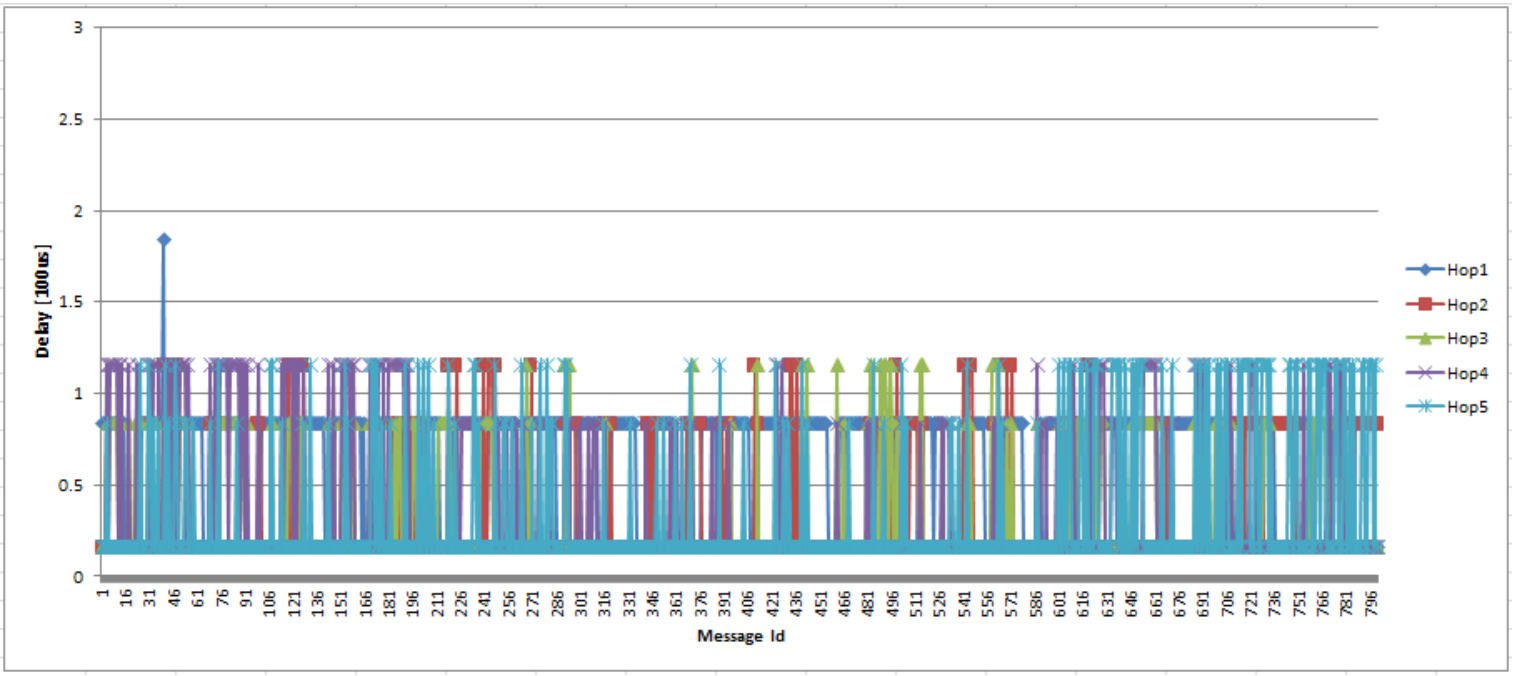
Itt a 3 breadboardot szétszórtuk egymástól 8 méter távolságra, falakkal elválasztva. Az átjáróhoz a legközelebb az 1-es és 2-es volt, középen a 3-as, 4-es, a lánc végén pedig az 5-ös és 6-os. Egyszerre bekapcsoltuk az összes csomópontot, ami kiépült a várakozásoknak megfelelően. A teljes kiépülés 15651 ms-ig tartott. A középső két csomópont közvetlenül az átjárót választotta a megbízhatósága alapján hozzá vezető útvonalnak, míg a lánc végén lévő kettő a középső kettőt.

4.2.4. Csomagvesztés falakon keresztül

A mérést két csomóponttal végeztük, melyek között először 2 majd 4 darab tíz centiméter vastag válaszfal volt, a falak között pedig 2-2 méternyi szabad tér. A méréseket megismételtük hardveres és hálózati rétegbeli visszaigazolás nélkül, majd csak hardveressel, végül mindkettőt engedélyeztük. Kettő fal esetén nem volt különbség a mérések között. Négynél viszont visszaigazolások nélkül 40% körüli csomagvesztés volt, hardveres visszaigazolással ez 20%-ra csökkent, ha mindekét fajta engedélyezve volt, akkor pedig nem volt csomagvesztés. Ez visszaigazolta a számításainkat, melyet a csomagok megbízható továbbítására és a csomagvesztés esélyére végeztünk adott újraküldés mellett.

4.2.5. Jitter mérése

Következőnek a késletetés ingadozását számoltuk ki. Ezt az 5 hop hosszú úton mért adatokból tettük. Először kiszámoltuk az átlagos késleltetést a hoppok között, ez 1.016 ms lett, majd vettük ennek és az összes késleltetésnek a különbségét. Ennek vettük az átlagát, amely 0.0404 ms lett. Várható volt, hogy alacsony lesz, mivel az adó oldalon a küldések között eltelt idő ingadozása kisebb, mint a mérés pontossága, a továbbítás során nincs véletlen várakozás, és az eszközök is egyformák és egyszerűek. Az ingadozást a hardveres újraküldések növelhetik. A mért késleltetéseket az alábbi ábra (2. táblázat) szemlélteti.



2. táblázat - Jitter mérése

5. Összefoglalás és kitekintés

A hálózat fejlesztés közben átfogó képet kaptunk mind hardveres, mind szoftveres szemszögből az intelligens otthon hálózatok állásáról. Megismerkedtünk már elterjedt protokollokkal, megtapasztaltuk, hogy mik egy ilyen hálózat fejlesztésének nehézségei. A mért eredmények alapján elmondható, hogy egy stabil, gyors rendszer lett az eredmény, amely megállja a helyét a megismert megoldások mellett, persze figyelembe véve a fejlesztés állapotát. A mérések sok helyen alátámasztották a számításainkat, az elvárt eredményeket hozták.

Mint a mérésekből is tapasztalható a hálózatunk 2 üzemmódban képes működni, az egyik a megbízhatóságra törekszik, mindenképp eljuttatva a csomagokat két csomópont között. Ez a működés az elvárt egy intelligens otthoni szenzorhálózattól is. A másik mód pedig ami az átviteli sebességet maximalizálja a protokoll és a felhasznált mikrokontroller keretein belül. Így a hálózat alkalmas lehet akár adatok streamelésére, mint a PCM modulált hangok, esetleg kisebb méretű képek átvitelére is.

A jövőben szeretnénk új hardvert tervezni, amely a jelenleginél kisebb méretű, és valós körülmények között is ki lehet próbálni. Ehhez még a protokollunkat is hangolni kell, valamint hosszú távon tesztelni.

Irodalomjegyzék

[1] http://www2.meethue.com/en-xx/what-is-hue/the-system/

[2] http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation/

[3] Kinney, Patrick. "Zigbee technology: Wireless control that simply works."Communications design conference. Vol. 2. 2003.

[4] Reinisch, Christian, et al. "Wireless technologies in home and building automation." *Industrial Informatics, 2007 5th IEEE International Conference on*. Vol. 1. IEEE, 2007.

[5] Gomez, Carles, Joaquim Oller, and Josep Paradells. "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology." *Sensors*12.9 (2012): 11734-11753.

[6] Jim Geier. „IEEE 802.11 Standard Overview.” 29.11.2000
http://www.informit.com/articles/article.aspx?p=19825

[7]https://www.nordicsemi.com/eng/content/download/2726/34069/file/nRF24L01P_Product_Specification_1_0.pdf

[8] Rodrigues, J. J. P. C. and Neves, P. A. C. S. (2010), A survey on IP-based wireless sensor network solutions. Int. J. Commun. Syst, 23: 963–981. doi: 10.1002/dac.1099