VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATION

LOKALIZAČNÍ PROTOKOL PRO WSN S PODPOROU MOBILITY UZLŮ

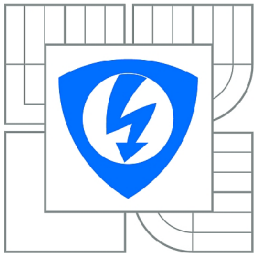
Localization protocol for WSN with support of node mobility

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — BC. MARTIN VOTAVA
AUTHOR

VEDOUCÍ PRÁCE — ING. PAVEL HOLEŠINSKÝ
SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Martin Votava	***ID:***	78409
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Lokalizační protokol pro WSN s podporou mobility uzlů

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh lokalizačního protokolu, který bude uvažovat mobilitu uzlů.
Nejdříve je nutné proniknout do problematiky lokalizace bezdrátových sítí a následně se zaměřit na problémy související s mobilitou lokalizovaných uzlů.
Úvahy by měly směřovat k návrhu lokalizačního protokolu optimalizovaného pro mobilitu senzorových uzlů.

DOPORUČENÁ LITERATURA:

[1] BULUSU N., JHA S., Wireless sensor networks.
Boston: Artech House, 2005, 326 stran. ISBN: 978-1580538671.

[2] Bachrach J. and Taylor C., Handbook of Sensor Networks, Wiley, 2005. 552 stran, ISBN: 978-0-471-68472-5.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Pavel Holešinský

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Prohlášení

Prohlašuji, že svou diplomovou práci na téma Lokalizační protokol pro WSN s podporou mobility uzlů jsem vypracoval samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne . . . .

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Pavlu Holešínskému za vedení při psaní diplomové práce a cenné rady pro její úspěšné dokončení.

V Brně dne . . . . . .

. . . . . .

podpis autora

ANOTACE

Práce je zaměřena na bezdrátové senzorové sítě. Hlavní tématem je mobilita a lokalizace v těchto sítích. Práce popisuje různé metody lokalizace a možnosti mobilních schémat a vytvoření nového lokalizačního protokolu pro bezdrátové senzorové sítě. V neposlední řadě je zde obsažena simulace nového protokolu v prostředí Network Simulator 2 a zhodnocení získaných výsledků.

Klíčová slova: bezdrátové senzorové sítě, ZigBee, lokalizace, mobilita, sběr dat, Network Simulator 2, dynamická lokalizace, statická lokalizace

ABSTRACT

Work is focused on wireless sensor networks.The main theme is the mobility and localization in these networks. This work describes different methods and capabilities of mobile and localization schemes. This work describes a new localization protocol for wireless sensor networks. Finally, there is included a new protocol in the simulation at Network Simulator 2 and evaluate the results obtained.

Key words: wireless sensor networks, ZigBee, localization, mobility, data collection, Network Simulator 2, dynamic localization, static localization

Obsah

Seznam obrázků

1. Úvod

Cílem mé diplomové práce je seznámení čtenáře s problematikou bezdrátových senzorových sítí a provedení následné simulace této sítě v prostředí Network Simulator 2. Toto téma jsem si zvolil, protože v průběhu mého studia na VUT mě nejvíce zaujala tvorba sítí a senzorové sítě jsou jedním z odvětví tohoto oboru.

V teoretické části mého projektu se zaměřím na vlastnosti sítě Zigbee, pro kterou v praktické části navrhnu lokalizační protokol. Dále chci čtenáři představit možnosti lokalizačních schémat, které je možno využít v síti ZigBee. Dalším tématem, kterým se chci v teoretické části zabývat, je podpora mobility v bezdrátových senzorových sítích, která bude v lokalizačním protokolu také obsažena. V závěru teorie chci vysvětlit vlastnosti programu Network Simulátor 2, ve kterém budou později prováděny veškeré simulace.

V praktické části se pokusím navrhnout lokalizační protokol, který bude podporovat mobilitu uzlu. Protokol by měl obsahovat více lokalizačních a mobilních schémat, aby byl schopen všestranného využití v bezdrátových senzorových sítích.

Věřím, že se mi podaří naplnit mé cíle tím, že čtenáři přiblížím možnosti lokalizace a mnou navržený protokol bude využitelný pro lokalizaci mobilních uzlů v bezdrátové senzorové síti.

2. Bezdrátové senzorové sítě

V dnešní době se stále více firem zabývá vývojem bezdrátových senzorových sítí. Důkazem toho může být vývoj sítě ZigBee, na němž se podílí několik nadnárodních společností, mezi ně patří například: Motorola, Honey-well, Philips, Samsung, Invensis a další. ZigBee technologií se budeme zabývat níže. Výhodou bezdrátových senzorových sítí je to, že mohou být instalovány kdekoli v terénu, není nutné řešit jejich napájení (standardně jsou napájeny z baterie). Senzorové sítě mohou být použity k monitorování kvality vody, životních funkcí člověka, seismické aktivity a ve spoustě dalších odvětví. Všechna data mohou být posílána do řídícího centra, kde jsou dále zpracovávána. Data ze všech možných odvětví lidského života jsou zasílána do základnové stanice, ta je přeposílá do řídícího centra, kde jsou data vyhodnocena. Nyní se pojďme podívat na technologii ZigBee, na níž si vysvětlíme, jak síť prakticky funguje.

3. ZigBee

ZigBee je definována standartem IEEE 802.15.4. Tento standard definuje malé bezdrátové sítě, které jsou označovány WPAN (Wireless Personal Area Networks). Bezdrátová síť ZigBee vyniká stabilitou, flexibilitou, jednoduchou konfigurací a nízkou cenou.

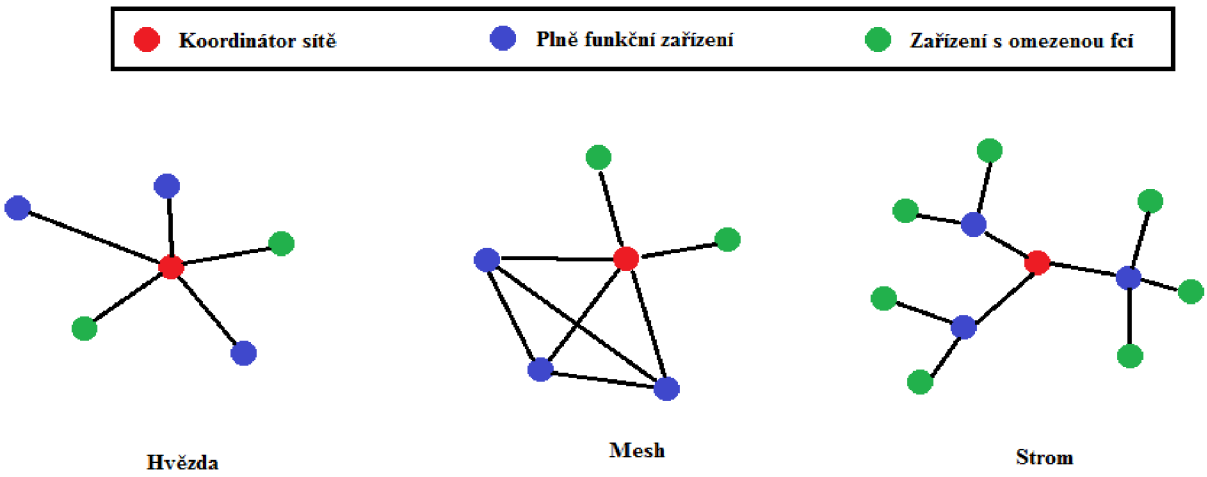
3.1. Topologie podle standardu ZigBee

Standard ZigBee specifikuje tři typy zařízení. Prvním je **koordinátor sítě.** Každá síť má tento prvek pouze jednou. Jeho základní funkce je, uchovávat informace o síti a stanovovat optimální cestu mezi jednotlivými uzly. Druhé zařízení figurující v této síti je tzv. **plně funkční zařízení**. Toto zařízení implementuje kompletní protokolový rámec a zajišťuje veškeré služby, které specifikace ZigBee stanovuje. Také je schopno předávat data z jiných zařízení. Posledním prvkem je **zařízení s redukovanou funkčností.** Je schopné komunikovat s koordinátorem sítě či s plně funkčním zařízením, ale není schopno předávat data z jiných zařízení.

Pro adresaci jednotlivých zařízení se používá binární kód, který může mít dvě podoby. Dlouhou, která má 64 bitů, a zkrácenou, ta má bitů 16. Pokud použijeme zkrácený kód, maximální počet stanic v síti je 65 535.

Každá síť je dále definovaná pomocí PAN ID, tento identifikátor má 16 bitů a slouží k rozlišení překrývajících se sítí.

Standard ZigBee definuje tři povolené topologie. Základním typem je **hvězda**. V tomto rozložení jedno zařízení funguje jako koordinátor sítě a všechna ostatní zařízení jsou s ním přímo spojena a zastávají funkci koncových zařízení. Mohou to být jak plně funkční zařízení, tak zařízení s omezenou funkcí. Druhou formou je topologie typu **strom**. Zde opět jedno zařízení zastává funkci koordinátora sítě, ostatní zařízení jsou koncovými zařízeními, ale na rozdíl od hvězdy nemusí být přímo spojeny s koordinátorem sítě. Poslední topologií je topologie typu **síť** (mesh), která kombinuje oba předchozí typy. Síťová topologie vykazuje největší funkčnost. Všechny výše popsané topologie je možné vidět na obrázku 2.1.



Obr. č. 1 Topologie sítě ZigBee

3.2. Radiové vlastnosti ZigBee

Standardu ZigBee byla přidělena tři radiová pásma tak, aby byl akceptovatelný po celém světě.

a) Pro globální použití pásmo ISM 2,4 GHz s 16 kanály a přenosovou rychlostí 250 kb/s

b) Pro území Ameriky a Austrálie pásmo 915 MHz s 10 kanály a přenosovou rychlostí 40 kb/s

c) Pro Evropu pásmo 868 MHz s jedním kanálem a přenosovou rychlostí 20kb/s

Přístup k přenosovému médiu je řešen metodou CSMA/CA. Ta říká, že stanice, která chce vysílat, určitý čas naslouchá, zda je médium volné a pokud ano, může začít vysílat. Pokud je médium obsazeno, počká, až se kanál uvolní. Tato metoda nijak neřeší problematiku kolizí.

Standart IEEE 802.15.4. definuje komunikační protokol, který je založen na přenosu rámců. Máme čtyři typy těchto rámců. Prvním z nich je **Datový rámec** (Data Frame). Tento rámec má délku 104 bytů a přenáší užitečná data. Druhým typem je **Potvrzovací rámec** (Acknowledgement Frame), jak už z jeho názvů vyplývá, slouží k přenosu potvrzovací informace. Využívá se na úrovni MAC při potvrzované komunikaci. Je vysílán ihned po přenosu paketu. **Příkazový rámec** (MAC Command Frame), slouží ke konfiguraci sítě a zařízení v síti. Posledním rámcem je **Beacon** (Beacon Frame), je využíván k synchronizaci sítě. A využívá se k uvádění koncových uzlů do režimu spánku, který prodlužuje jejich životnost.

3.3. Synchronizace

Jak už bylo výše uvedeno, pro synchronizaci zařízení v síti ZigBee se používá rámec Beacon. Celý proces synchronizace řídí koordinátor sítě. Ten vysílá k ostatním uzlům synchronizační sekvence, tedy rámec Beacon. Ostatní uzly se podle nich synchronizují s koordinátorem sítě. Díky tomu je možné koncové zařízení na dlouhou dobu uvést do režimu spánku. Interval synchronizačnich sekvencí lze nastavit v rozsahu 15 ms až 15 minut. Přenos je potom uskutečněn pomocí tzv. superrámce, který začíná rámcem Beacon. Po něm následuje interval, kdy jednotlivé stanice soutěží o přístup k médiu. Poté následuje interval s rezervovanými časovými sloty, sloužící k prioritnímu přenosu.

3.4. Spotřeba energie

Standard ZigBee je navržen tak, aby koncové stanice měly co nejmenší spotřebu energie. Obvykle jsou totiž napájeny bateriemi. Dále se doporučuje, aby koordinátor sítě nebyl napájen z baterie, protože na jeho funkčnosti je závislá celá síť. Využívání rámců Beakon zajišťuje snížení spotřeby na minimum. Protože stanice neposílá data, je v režimu spánku, tudíž v režimu minimální spotřeby. Velké nároky jsou kladeny na koordinátora sítě, který můsí být schopen uložit všechna data z jednotlivých uzlů.

3.5. Struktura protokolu ZigBee

Protokol je navržen tak, aby mohl být implementován do jednočipových mikrokontrolerů, které jsou málo výkonné. Dvě nejspodnější vrstvy jsou definovány standardem IEEE 802.15.4.

Nad nim i je síťová vrstva NWK (Network layer), která se stará o připojování a odpojování k síti. Dále má na starosti směrování a zabezpečování paketů. U koordinátora sítě se stará o start sítě a přiřazování adres novým uzlům.

Aplikační vrstva je rozdělena na několik podvrstev. Jednou z nich je pomocná aplikační podvrstva APS (Aplication Support Sublayer), která se stará o párovací tabulky, které rozčleňují zařízení na základě poskytovaných služeb. Další podvrstvou je objekt ZigBee ZDO (ZigBee Device Object). Ten definuje roli zařízení v síti (zda se jedná o koncové zařízení či koordinátor) a také objevuje nové zařízení.

Poslední vrsta je určena profilem, který stanovuje vlastnosti zařízení, která mohou být obsažena v síti, dále udává formát zpráv tak, aby tvořil smysluplnou aplikaci. Profil ZigBee je určen 16 bitovým identifikátorem, který je vydáván společností ZigBee Allience.

4. Mobilní sbírání dat

Mobilita je důležitou součástí senzorových sítí. Na problematiku mobility se lze dívat z několika úhlů pohledu.

4.1. Mobilní základnová stanice MBS (Mobile Base Station)

Při této variantě se základnová stanice pohybuje během provozu a získává data od senzorů. Senzory tedy data uchovávají v paměti jen krátkou dobu. Pokud základnová stanice není mobilní, může docházet k tomu, že uzly v její blízkosti budou mít vysokou spotřebu energie, protože nepřenáší jen informace, které získají oni samy, ale také informace od ostatních stanic, protože tvoří prostředníka mezi nimi a základnovou stanicí. Existuje několik řešení toho, jak se má základnová stanice pohybovat.

4.1.1. Relokace základnové stanice

Při tomto řešení dochází k tomu, že se základnová stanice pohybuje tak, aby spotřeba všech uzlů sítě byla vyvážená. Provádí se to tak, že čas je rozdělen na kola, v jejichž průběhu je základnová stanice stacionární a po skončení jednoho kola je vypočítána další pozice, kam se má základnová stanice přesunout .

4.1.2. Spojení mobility a směrování

Základnová stanice se bude pohybovat po kruhové trajektorii. Uzly, které jsou uvnitř tohoto kruhu, posílají data nejkratší cestou do základnové stanice. Uzly mimo kruhovou trajektorii data posílají na trajektorii základnové stanice. Tím se šetří energie a prodlužuje životnost sítě.

4.1.3. Pohyb a zastavení

Další možností, jak prodloužit životnost sítě, je rozčlenění senzorů do mřížky. Základnová stanice se vždy zastaví v bodě mřížky, kde jsou uzly umístěny. A odtud pokračuje do dalšího bodu. Doba přechodu mezi jednotlivými body je zanedbatelná.

4.2. Mobilní "sběrač" dat MDC (Mobile Data Colector)

Základem této varianty je mobilní sběrný uzel, který prochází sítí a od každého uzlu získává informace přímo. V praxi to tedy vypadá tak, že každý uzel nasbíraná data ukládá do své paměti, dokud k němu nepřistoupí sběrný uzel a data si nenahraje. Tento přístup se používá pro sběr dat na rozsáhlých plochách. Například při měření provozu ve velkých městech, kde postačí několik málo senzorů na několika vzdálených místech. Bylo by velmi ekonomicky náročné propojit všechny uzly a data sbírat najednou. Proto se volí toto řešení a data jsou od každého uzlu nahrávána individuálně. Existuje několik vzorů, jak se může sběrač pohybovat, a to náhodně nebo předem stanovenou cestou a nebo řízeně v reálném čase.

4.2.1. Datová "mula"

Takto je nazýván sběrač dat s náhodným přístupem. Postup, jakým jsou data sbírána, vypadá následovně. Senzory sbírají data ze svého okolí a ukládají je do své paměti. Mobilní sběrač "mula" prochází náhodně jednotlivé senzory a nahrává si od nich data přímou cestou. Sebraná data jsou posílána bezdrátově do přístupového bodu, kde jsou dále dle potřeby zpracovávána. Náhodný pohyb sběrače je založen na Markovovu modelu, který závisí na velikosti paměti, počtu přístupových bodů a počtu sběračů.

4.2.2. Předvídatelný sběr dat

Princip tohoto modelu si můžeme ukázat na senzorech, které jsou rozmístěny ve městě. Sběrač senzoru je na autobuse. Senzor tedy ví, kudy a jak dlouho autobus pojede. Ví také, kdy dojde k přenosu dat a podle toho se může uvést do režimu spánku a tím šetřit energii.

4.2.3. Plánování mobility

Pokud je využíváno MDC může dojít k přetečení paměti koncových uzlů, protože je nenavštívil sběrný uzel. S toho důvodu jsou sběrným uzlem nejdříve obslouženy stanice, u nichž hrozí přetečení paměti. Bere se ale také v potaz vzdálenost mezi stanicemi.

4.3. Rendezvous bod

Řešení pomocí rendezvous bodů je kombinací obou výše zmíněných. Využívá se v několika oddělených sítích. Každá síť má jeden uzel, do jehož paměti se shromažďují veškerá posbíraná data. Z tohoto uzlu jsou poté data přehrána do MDC. V praxi to vypadá tak, že v rámci sítě data sbírá MBS a mezi jednotlivými sítěmi MDC.

4.3.1. Přenášený sběr dat

Tento přenos probíhá následovně. Mobilní sběrač dat projíždí senzorovou síť, když vstoupí do prostoru nějakého senzoru, dojde k přenosu dat. Jednotlivé koncové stanice zjišťují zda se k nim MDC přiblíží, pokud ne, posílají svá data přes ostatní uzly do jeho cesty. Existují různé algoritmy, které určují, ve kterých oblastech má sběrač zpomalit či úplně zastavit a ve kterých naopak zrychlit tak, aby došlo k co nejefektivnějšímu sběru dat.

-

5. Lokalizace

Lokalizace je proces, při kterém se určuje poloha uzlu v bezdrátové senzorové síti. Určení polohy uzlu je důležité pro správné fungování sítě. Bez správného určení polohy nastává problém s geografickým směrováním, zaměřováním objektů a s detekcí událostí s určením polohy. Pro určení polohy se nabízí využití systému GPS, který je v dnešní době velice rozšířen. Jeho velkou nevýhodou je vysoká cena, zvyšování spotřeby energie při jeho nasazení a nutnost spojení s družicí, což činí problém v budovách a ve členitém terénu. Z toho důvodu se v senzorových sítích využívají algoritmy, pomocí nichž si uzel změří vzdálenost ke svému sousedovi a z dostupných informací určí svoji polohu, bez použití systému GPS.

5.1. Lokalizace pomocí majáků (kotev)

Tento princip lokalizace je založen na tom, že máme v síti rozmístěno nekolik uzlů, tzv. majáků či kotev, které znají svoji pozici a zbylé uzly díky nim dokáží určit svoji pozici. Majákové systémy můžeme rozdělit na dva typy: těsně spárované systémy a volně spárované systémy.

5.1.2. Těsně spárované systémy

Tyto systémy mají majáky, které jsou umístěny na přesně stanovených místech a jsou přímo propojeny s centrálním řízením. Vykazují velkou přesnost a jsou schopny pracovat v reálném čase. Nevýhodou tohoto řešení je přesně definovaná pozice majáků a pevné spojení s centrálou, zpravidla kabelové. To má za následek, že tyto systémy nelze nasadit v každém prostředí.

5.1.3. Volně spárované systémy

Jejich součástí jsou majáky, které jsou bezdrátové a decentralizované, zpravidla jejich funkci plní některé uzly v síti. Nasazení těchto systémů je jednodušší a můžeme je použít ve velkém množství. Na druhé straně tyto systémy jsou méně přesné. Majáky jsou vybaveny

zařízením GPS. Díky němu zjistí svoji polohu a tu posílají po síti. Další uzly díky této informaci vypočítávají svojí pozici.

5.2. Nasazování majáků

Pro správně fungující lokalizaci je nutné zvolit správný počet majáků a jejich správnou polohu. Zvolení velkého počtu majáků může vést k příliš velkému provozu na síti a velkým ekonomickým nákladům. V opačném případě, při malém počtu majáků, dochází k nedostatečnému pokrytí sítě tedy k nepřesné lokalizaci. Vznikají tu dva přístupy, jak zvolit správný počet a polohu majáků.

5.2.1. Využití haldy (heap)

Tento přístup je vhodný do sítí, které mají malou hustotu. Je založen na myšlence, která říká, že je lepší nasadit menší počet majáků a hledat jejich ideální umístění, než nasadit velké množství majáků. Funguje tedy na inkrementaci majáků (je jich málo, najdu ideální místo a přidám další). Tento algoritmus předpokládá tři typy uzlů a to maják, klasický uzel a nasazovač. Jeho princip vypadá následovně:

- Majáky mezi sebou komunikují a hledají vhodného kandidáta pro nový maják. Ze svého okolí potom vyberou jednoho kandidáta a jeho polohu pošlou nasazovači.
- Uzly přijímají kandidáty navrhované sousedními majáky, zvolí si jednoho a posílají tuto informaci nasazovači.
- Nasazovač všechny přijaté informace vyhodnotí a odstraní všechny kandidáty, kteří nesplňují předem stanovené podmínky pro to, aby se staly novým majákem. Poté zvolí body pro nasazení majáků.

5.2.2. Výběrový algoritmus STROBE

Využívá se v sítích s hustým pokrytím. Je přesným opakem algoritmu halda. To znamená, že máme na začátku velký počet majáků a z nich vybíráme ty, které mají být aktivní. Tím prodlužujeme životnost celé sítě. Při výběru majáků je cílem to, abychom snížili

počet aktivních majáků na minimum, ale nesnížili přesnost lokalizace. Majáky se mohou nacházet ve třech různých stavech: volající, označený a spánek.

Každý maják začíná ve stavu volající. V tomto stavu posílá informace o své pozici a přijímá informace o svých sousedech. V tomto stavu zůstává, dokud nevyprší časovač T_v. Po vypršení tohoto časovače se maják na základě získaných informací rozhodne, zda přejde do stavu spánku nebo do stavu označený. Pokud přejde do stavu spánku, informuje o tom své okolí. A zapne si časovač T_S , po jehož uplynutí se opět probouzí a přechází do stavu volající. Pokud přejde do stavu označený, posílá do svého okolí zprávy o své poloze ale žádné zprávy nepřijímá. Tím šetří energii. V tomto stavu se nachází do uplynutí časovače T_D. Poté se opět vrací do stavu aktivní. Životnost sítě tedy ovlivňuje nastavení časovačů T_D, T_V a T_S.

6. Navrh lokalizačního protokolu

Jak už bylo zmíněno výše, lokalizace je schopnost senzoru určit svoji aktuální fyzickou polohu. Tento proces je velice důležitý pro agregaci uzlů v senzorové síti a pro směrování dat sítí.

Při návrhu lokalizačního protokolu je důležité si uvědomit, že lokalizace probíhá v několika krocích a v několika úrovních. Pokud se podíváme na počáteční definici lokalizace, tak je v ní velice důležité slovo aktuální poloha uzlu. Protože pokud bude k datům připojována poloha uzlu, kde se uzel už nenachází, tak se pro nás data mohou stát bezcennými. Z toho důvodu je důležité vyřešit aktualizaci lokalizačních dat. Touto problematikou se zabývají různá mobilní schémata. Můžeme využít statické či dynamické metody aktualizace. Další důležitou rovinou lokalizace uzlu je technika lokalizace. Ta může být buď přímá, pomocí GPS zařízení a nebo nepřímá, například s využitím triangulace. A v neposlední řadě je velice důležité, jak bude protokol do bezdrátové senzorové sítě začleněn. Všechny tyto parametry lokalizačního protokolu jsou jeho nedílnou součástí a jejich špatné zvolení by mohlo vést k nefunkčnosti protokolu. Proto je nutné si tyto části lokalizačního protokolu přiblížit.

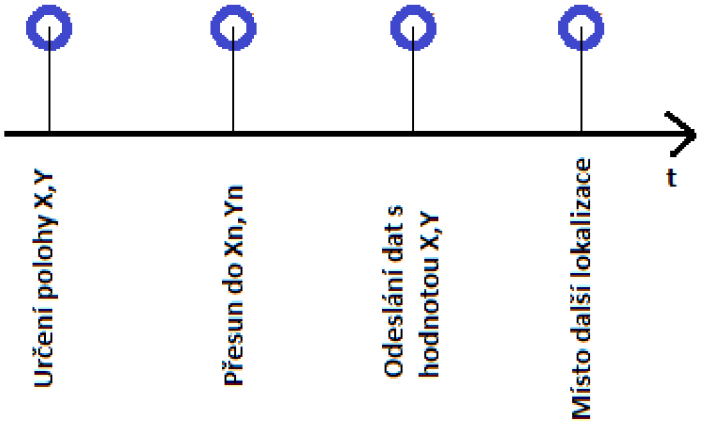
6.1. Mobilní schémata

Mobilní schémata jsou zcela nezávislá na tom jakým způsobem je lokalizace uzlu prováděna. Aktualizace dat je důležitá z toho důvodu, že pokud je interval, v kterém jsou data aktualizována příliš krátký a rychlost uzlu je velká, tak potom informace o poloze přiložená k datům není aktuální, viz obrázek číslo 2, v případě, že je lokalizační interval příliš krátký, dochází ke zbytečnému vybíjení baterie. Mobilní schémata mohou být buď statická, dynamická nebo prediktivní. Další možností, jak podporovat mobilitu uzlu, je uzel vybavit doplňkovým zařízením accelerometrem, který zaznamenává změny pohybu.

6.1.1. Statické mobilní schéma (Static Fixed Rate)

Lokalizace je prováděna periodicky v pevně stanovené periodě T. Chyba určení polohy, která nám zde vzniká je v důsledku toho, že určím polohu uzlu X,Y. Uzel se

pohybuje dál a ještě před vypršení periody T, odešle data s polohou X,Y. Jeho poloha je ovšem již jiná. Tato chyba se zvyšuje s rychlostí uzlu. Při nízkých rychlostech je zase spotřeba energie vyšší než je nutné.



Obr. č. 2 Chyba vzniklá z neaktuálních dat

6.1.2. Dynamické mobilní schéma (Dynamic Velocity monotonic)

Tento algoritmus mění dynamicky lokalizační frekvenci v závislosti na rychlosti uzlu, kterou určuje z předchozích lokalizací. Na počátku lokalizace je nutné zvolit počáteční lokalizační frekvenci. U tohoto protokolu volíme parametr α, který udává maximální chybu, tedy jaký úsek může uzel urazit bez toho, aniž by došlo k novému určení polohy. Pomocí tohoto parametru a rychlosti uzlu jsme schopni spočítat čas další lokalizace. Využití tohoto algoritmu je ideální v podmínkách, kde mají uzly přibližně konstantní rychlost. V případě, že se bude rychlost uzlu náhle měnit, například bude uzel půl periody stát a pak se začne pohybovat rychlostí *v*, bude jeho rychlost spočtena jako $\frac{v}{2}$ a lokalizační frekvence bude určena špatně. Bude nám vznikat větší chyba, než je zvolena parametrem α. Obecně lze k tomuto algoritmu říci, že čím bude rychlost uzlu větší, tím bude menší lokalizační perioda a čím bude rychlost menší, tím bude perioda větší. Může ovšem nastat situace, pokud bude uzel stát na místě, že bude lokalizační perioda spočtena jako nekonečno. Z toho důvodu je možné nastavit horní a dolní hranici lokalizační periody.

6.1.3. Prediktivní mobilní schéma (Mobility Aware Dead Reckoning Driven)

Součástí tohoto algoritmu je mobilní schéma senzoru, které se používá pro určení budoucího pohybu. Lokalizace je spuštěna v případě, že rozdíl mezi aktuální hodnotou uzlu a predikovanou hodnotou uzlu překročí chybovou hranici. Na rozdíl od dynamického protokolu, kde je vždy určen čas další lokalizace, zde k určení polohy nedochází, pokud se uzel pohybuje dle očekávání. Lokalizační frekvence je tedy velice nízká. Tento algoritmus lze ovšem použít pouze tehdy, pokud lze pohyb uzlů předpokládat, tedy vytvořit mobilní schéma.

6.1.4. Využití accelerometru

Všechna výše uvedená schémata s sebou nesou, nutnost aktualizace polohy i v případě, že se uzel pohybuje konstantní rychlostí přímým směrem a nebo stojí na místě. Toto zvyšuje spotřebu baterií a snižuje životnost sítě. Tuto problematiku řeší vybavení uzlu accelerometrem. Princip funkce je potom následující. Uzel je začleněn v síti a v případě, že zaznamená změnu směru nebo rychlosti, tak provede lokalizaci. Spustí časovač a po jeho vypršení provede druhou lokalizaci. Ze získaných informací určí směr uzlu a jeho rychlost. Pokud uzel nemění směr ani rychlost, tak lze polohu uzlu určit pomocí metody postupného výpočtu. Takže v případě, že je uzel požádán o data, spočítá si svoji polohu bez toho, aniž by komunikoval s ostatními uzly. To velice šetří baterie.

Tento princip je vhodný do takových sítí, kde se uzly pohybují konstantní rychlostí a bez prudkých změn pohybu. V takových případech bude vykazovat velice nízkou spotřebu energie. Jeho nevýhodou je nutnost dalšího zařízení, což má za následek větší finanční náklady na síť.

6.2. Algoritmy pro určení polohy uzlu

Algoritmy pro určení polohy vycházejí z následujícího principu. Pomocí některé z loklizačních metod, viz kapitola 6.5. , určí vzdálenost či směr k sousedním uzlům a poté tyto data zpracují pomocí matematických operací k určení své polohy, viz kapitola 6.6.

Jak už bylo zmíněno výše, máme pro lokalizaci dva typy technik. Techniky přímé a techniky nepříme. Techniky přímé určí svoji polohu bez toho, aniž by potřebovaly

informace o pozici okolních uzlů. Nepřímé techniky vyžadují kontakt s okolními uzly, které již znají svoji polohu, pomocí níž spočítají polohu svoji.

6.3. Přímá lokalizace

6.3.1. Volba polohy

Nejjednodušší metodou, jak určit polohu uzlu, je uzly v síti napevno umístit a jeho polohu zaznamenat do uzlu. Tento uzel musí být samozřejmě statický, protože má implicitně zvolenou polohu, kterou nemůže měnit. Tato metoda se může použít například v budově, kde umístíme tyto uzly a ostatní pohyblivé uzly pomocí nich svoji polohu určí. Využití tohoto typu lokalizace je energeticky a ekonomicky nenáročné, protože nepotřebujeme žádné přídavné zařízení, které by nám energii spotřebovávalo. Je ovšem nutné každý uzel vzít a umístit na jeho přesné místo a zaznamenat jeho polohu.

6.3.2. GPS (Global Positionning systém)

Je to družicový systém, pomocí něhož lze určit polohu objektu kdekoliv na zemi s přesností na centimetry. V případě, že chceme tento systém zařadit do bezdrátové senzorové sítě, je nutné uzly vybavit zařízením GPS, což zvyšuje náklady a životnost sítě. Energetické nároky na zařízení vybavené GPS se podstatně zvětšují. Proto je nutné mít k uzlům přístup a v případě potřeby jim vyměnit baterie. Z toho důvodu se zdá vhodnější variantou předchozí způsob, při kterém každý bod navštívíme jednou při zakládání sítě, ale poté se o něj již nemusíme starat. Praktickým příkladem využití v senzorových sítích je ZebraNet, který se využívá k monitorování pohybu zeber pro určení jejich životních cyklů. Senzory jsou umístěny na zebrách a každé tři minuty posílají informace o jejich poloze. Tato síť tedy využívá statickou aktualizaci určení polohy, což s sebou nese problémy, které byly popsány výše.

6.3.3. Spotlight systém

Funguje na jednoduchém principu, pro jehož chod postačí senzor vybavit detektorem světla. Senzory jsou náhodně rozmístěny v terénu, například vyhozením z helikoptéry. Potom

pro určení jejich polohy, stačí terén osvětlovat v předem definovaném časovém schématu. Jakmile senzor detekuje, že byl osvícen, posílá tuto informaci s časem osvícení do koordinátoru sítě, ten má zabudované časové schéma osvětlování a z něj určí polohu uzlu. Ta je uzlu poslána. Nasazení tohoto systému je podstatně levnější oproti GPS a má menší energetické nároky. Vyžaduje ovšem nasazení v otevřeném terénu, aby bylo možné všechny uzly osvítit a zařízení, z kterého bude k osvěcování docházet, např. helikoptéru či vyvýšenou plošinu.

6.4. Nepřímá lokalizace

Všechny nepříme lokalizační algoritmy, využívají pro určení polohy uzlů, které znají svoji polohu tzv. majáků. Pomocí matematických operací z jejich polohy určí polohu svoji.

6.4.1. TDOA (Time difference of arrival)

Tento mechanizmus funguje na přímém spojení mezi uzlem, který určuje svoji polohu a mezi kotevními uzly (majáky). Kotevní uzly, se kterými je v kontaktu, musí být minimálně tři. Signál přijatý z kotevního uzlu je demodulován a je spočítán rozdíl mezi časem odeslání zprávy a čase jejího přijetí. Situace ovšem není tak jednoduchá, jak se může zdát. Protože jsou většinou uzly umístěny někde v budovách nebo v krajině, dochází k mnohocestnému šíření signálu, což má za následek vznik chyby. Signál, který je přijat, vychází z následující rovnice:

$$S_{AM}(t) = \sum_{n=1}^{N} \sum_{k=0}^{K} A_{nk} \cdot S_n(t - t_n - \tau_{nk})$$

Vzorec č. 1 Výpočet přijatého signálu

Kde: A_{nk} *je amplituda k-tého odraženého signálu n-tého vysílače v přijímači*

t_n *je zpoždění přímého signálu n-tého vysílače v přijímači*

τ *je zpoždění k-tého odraženého signálu n-tého vysílače v přijímači*

Pro určení vzdálenosti je nutné najít t_n. To může být v důsledku mnohocestného šíření velice obtížný úkol. Pro jeho splnění se používají různé filtry a matematické výpočty. I přes jejich nasazení může metoda vykazovat velkou chybovost.

6.4.2. DV-Hop

V některých systémech není možné, aby byly kotvy distribuavány všude po síti a aby byly všechny uzly s kotvami přímo spojeny. Proto byl vyvinut mechanizmus DV-Hop. Ten počítá počet skoků mezi uzlem a kotvou pro určení vzdálenosti od kotvy. To probíhá ve dvou fázích.

V první fázi všechny kotvy pošlou všesměrově, rámec beacon, každý uzel si uloží počet skoků k nejbližší kotvě.

V druhé fázi, dojde k tomu, že kotvy obdrží beacon od jiných kotev. Určí fyzickou vzdálenost mezi kotvami a tu podělí počtem skoků, které je dělí. Tím určí vzdálenost jednoho skoku. Tuto hodnotu potom pošlou sítí. Uzly si spočítají vzdálenost od kotev. Jakmile mají vzdálenost minimálně od tří kotev, tak spočítají svoji polohu pomocí trilaterace. Tedy vytvoření kružnic se středem v kotvách o poloměru vzdálenosti ke kotvě. Kružnice se nám nestřetnou v jednom bodě v důsledku nepřesností. Proto se musí průnik zprůměrovat.

DV-Hop vykazuje dobré vlastnosti v sítích, kde jsou uzly rovnoměrně rozmístěny. V opačném případě musíme počítat s chybou při určení polohy. Tato metoda je tedy vhodná, tam kde nevyžadujeme velkou přesnost.

6.4.3. DV (Distanční metoda)

Je obdobou DV-hop algoritmu, jen na rozdíl od něj nepočítá počet skoků, ale reálnou vzdálenost mezi jednotlivými uzly. Stejně jako DV-Hop je dvoufázová.

V první fázi maják rozešle zprávy o své poloze, formát zprávy je následující: [id,Xi,Yi,Di], kde *id* je identifikátor majáku, *Di* je reálná vzdálenost mezi uzly a *Xi,Yi* jsou souřadnice majáku. Když uzel obdrží zprávu, uloží si ji a pomocí některé z lokalizačních metod určí vzdálenost od uzlu, od kterého zprávu obdržel. Tuto vzdálenost přičte k *Di* a zprávu posílá dál. Pokud obdrží zprávu s *id*, které má již uložené, porovná hodnoty *Di*. jestliže je *Di* v nové zprávě větší, zprávu zahazuje, v opačném případě zprávu ukládá a posílá dál. Ve druhé fázi je poté počítána průměrná délka skoku.Ta se určí podle vzorce číslo 2.

$$c_i = \frac{\sum \sqrt{(Xi - Xj) + (Yi - Zj)}}{\sum di}$$

Vzorec č. 2 Výpočet průměrné délky skoku

Kde: di je nejkratší vzdálenost mezi majáky

Pro určení polohy uzlu se opět použije trilaterace kružnic se středem v majících a o poloměru $r_i = c_i \cdot D_i$. Opět se kružnice nesetkají ve stejném bodě a bude se muset udělat korekce. Tato metoda je velice citlivá na chyby, které vznikají při měření vzdálenosti mezi uzly.

6.4.4. Dvoufázová metoda

Jak už název napovídá, tak tato metoda probíhá ve dvou fázích. Je používána tam, kde jsou kladeny velké požadavky na přesnost. První fáze je fází startovací a ve druhé dochází ke zpřesnění. Při startu tohoto algoritmu se použije metoda DV nebo DV-Hop. Kde každý uzel určí svoji polohu.

Poté začíná fáze druhá, pomocí určené polohy se spočítá vzdálenost mezi uzly. A metodou nejmenších čtverců se spočítají vzdálenosti nové, přesnější. Podmínkou pro fungování tohoto algoritmu jsou minimálně tři uzly v dosahu, bez nichž by metoda nejmenších čtverců nefungovala. Druhá fáze se může pro větší přesnost opakovat vícekrát.

6.5. Lokalizační metody

K tomu, aby mohli algoritmy pro určení polohy uzlu fungovat, je nutné sesbírat informace o vzdálenosti k okolním uzlům a k tomu se právě využívá lokalizačních metod. Lokalizační metody máme trojího typu, metody založené na měření času, který urazí signál mezi příjemcem a odesílatelem, další metody využívají měření úrovně signálu přijatého v přijímací stanici a poslední metodou je metoda vycházející z úhlu přijatého signálu.

6.5.1.Měření času

Při této metodě je měřena doba putování signálu mezi vysílačem a přijímačem, podrobněji je tato metda popsána u algoritmu určení polohy TDOA. Pro určení polohy nemusí být využit pouze čas letu, ale také fáze, v jaké signál dorazí.

Fázový posun signálu: V této metodě je porovnáván příchozí signál se signálem referenčním, který generuje přijímač. Vzdálenost se potom určí pomocí posuvu signálu přijatého a refernčního. Nevýhodou je nejednoznačnost, signál může být posunut o 2π.

6.5.2. Měření úrovně signálu

Je jednou z nejpoužívanějších metod. Využívá měření úrovně signálu. Jeho charakteristika je popsána rovnicí číslo 3. V ní ΔP je rozdíl mezi úrovní vysílaného a přijímaného signálu. Hodnota f udává nosnou frekvenci, c je rychlost světla a d je hledaná vzdálenost mezi uzly. V rovnici vystupují dvě konstanty α (frekvenční faktor) a β (vlastnosti prostředí). Tato metoda vykazuje větší chybu než metoda měření času.

$$\Delta P[dB] = 10\alpha \log\left(\frac{f}{c}\right) - 10\beta \log(4\pi d)$$

Vzorec č. 3 Měření úrovně signálu

6.5.3. Měření úhlu signálu

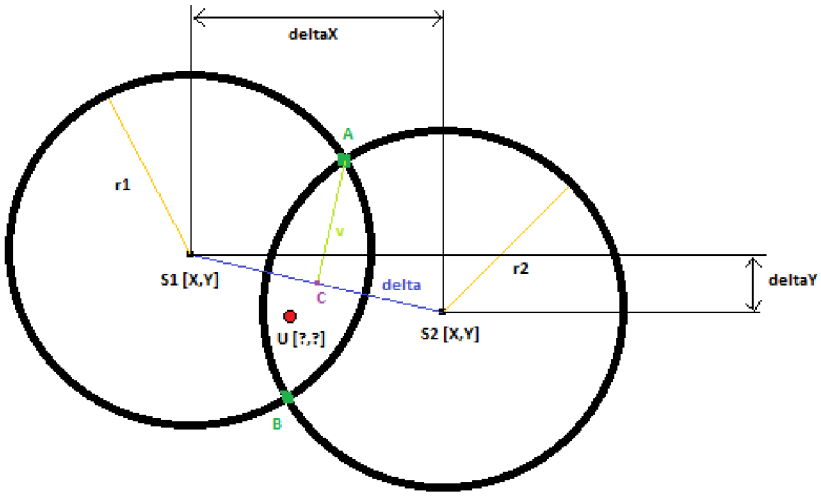
Metoda, která je založená na měření úhlu příchozího signálu, využívá se antén se směrovými charakteristikami, pro výpočet polohy se poté využije triangulace, tato metoda se příliš v bezdrátových senzorových sítích nepoužívá, protože pro funkčnost této metody jsou nezbytné anténní soustavy, které lze na malé uzly bezdrátové sítě jen těžko umístit.

6.6. Matematické řešení lokalizace

Data sesbíraná lokalizačními metodami, jsou zpracovávána v algoritmu pro určení polohy právě matematickými postupy. Mezi matematické řešení lokalizace patří trilaterace, průnik kružnic či metoda otisku.

6.6.1. Průnik kružnic

Průnik kružnic není příliš přesnou lokalizační metodou, poloha je určena jako plocha průniku kružnic. V případě, že máme pouze dvě kružnice pro určení polohy uzlu, je plocha, kde by se mohl uzel vyskytovat poměrně velká. Proto, čím více majáků pro určení máme, tím bude náš výpočet přesnější. Celá situace je vidět na obrázku číslo 3.



Obr. č. 3 Průnik kružnic

Je vidět, že každý maják je definován souřadnicemi a poloměrem, tedy sílou signálu. Pro určení polohy uzlu U je důležité určit body průniku A a B. K těmto bodům dospějeme pomocí následujících rovnic.

Nejprve je nutné určit vzdálenost majáků S1 a S2. K tomu musíme určit vertikální a horizontální vzdálenost uzlů. (deltaX a deltaY). K té dospějeme pomocí vztahů číslo 4.

$$deltaX = S1X - S2X$$

$$deltaY = S1Y - S2Y$$

Vzorec č. 4 Určení vertikální a horizontální vzdálenosti

Nyní můžeme spočítat absolutní vzdálenost obou majáků, vzdálenost delta. Vztah č.5:

$$delta = \sqrt{deltaX^2 + deltaY^2}$$

Vzorec č. 5 Určení absolutní vzdálenosti

Nyní máme hodnotu vzdálenosti obou majáků. Tuto úsečku rozdělíme na dvě části *m* a *n*, přičemž bod, ve kterém úsečku rozdělíme budeme označovat *C* a bude bodem výšky trojúhelníků *S1, S2, A a S1, S2, B.* Velikost úsečky m potom bude:

$$m = \frac{r1^2 - r2^2}{2 \cdot delta} + \frac{delta}{2}$$

$$n = delta - m$$

Vzorec č. 6 velikosti úseček m a n

Nyní vypočítame velikost výšky *v* v bodě C. Tato výška nám vytvoří dva pravoúhlé trojúhelníky *S1, A, C a S2, A, C*. Výška je určena vztahem č. 7:

$$v = \sqrt{r1^2 - m^2}$$

Vzorec č. 7 velikost výšky v

Poté, co jsme určili výšku *v*, nám pro výpočet bodů průniku chybí už jen poloha bodu C. Tuto hodnotu dopočítáme následující rovnicí:

$$Cx = Ax + \frac{m}{delta} \cdot (Bx - Ax)$$

$$Cy = Ay + \frac{m}{delta} \cdot (By - Ay)$$

Vzorec č. 8 Vypočet souřadnic bodu C

Nyní už máme všechny potřebné informace k tomu, abychom určili body průniku obou kružnic. K výpočtu použijeme vztah č. 9:

$$Ax = Cx + \frac{v}{delta} \cdot (S1y - S2y)$$

$$Ay = Cy - \frac{v}{delta} \cdot (S1x - S2x)$$

$$Bx = Cx - \frac{v}{delta} \cdot (S1y - S2y)$$

$$By = Cy + \frac{v}{delta} \cdot (S1x - S2x)$$

Vzorec č. 9 Body průniku kružnic

6.6.2. Trilaterace

Této matematické techniky se využívá například v GPS. Pro správné určení polohy jsou zapotřebí minimálně tři body se známou polohou. Výpočet polohy je poté řešen pomocí tří rovnic o třech neznámých:

$$\sqrt{(X_1 - X)^2 + (Y_1 - Y)} = |r_1|$$
$$\sqrt{(X_2 - X)^2 + (Y_2 - Y)} = |r_2|$$
$$\sqrt{(X_3 - X)^2 + (Y_3 - Y)} = |r_3|$$

Vzorec č. 10 Výpočet trilaterace

6.6.3. Metoda postupného výpočtu

Metoda postupného výpočtu je jednou z nejstarších matematických výpočtů polohy. Byla využívána v námořnictvu, pro navigování letadel či automobilů.

Vychází ze známého výchozího bodu se souřadnicemi X_0, Y_0. Dalšími známými hodnotami jsou úhel α, pod kterým se objekt pohybuje a rychlost v. Výpočet polohy uzlu se řeší pomocí rovnice č. 11:

$$X_1 = X_0 + L\cos\alpha$$
$$Y_1 = Y_0 + L\cos\alpha$$

$$L = v \cdot \Delta t$$

Vzorec č. 11 Metoda postupného výpočtu

6.6.4. Metoda otisku

Využívá ve svůj prospěch mnohocestné šíření signálu v uzavřených prostorách, pomocí nějž dojde k popisu prostoru. Určení polohy se děje dvoufázově. V první fázi je pro každé místo prostoru změřena úroveň signálu. Ze získaných dat je poté sestavena databáze. Ve druhé fázi jsou rozmístěny senzory, ty přijmou signál určité úrovně a z databáze popisu prostoru určí svoji polohu.

6.6.5. Metoda nejmenších čtverců

Je aproximační metodou, která spočívá ve výpočtu funkce f, pro kterou je součet čtverců odchylek vypočítaných hodnot od naměřených minimální. Nejjednoduší závislost dvou veličin je závislost lineární. Ta je pro jednu závislou a jednu nezávislou proměnnou reprezentována rovnicí přímky: $f(x) \equiv y = ax + b$. Symbol y označuje závislou proměnou, x označuje nezávislou proměnou, a je směrnicí přímky a b je počáteční hodnota. Pro výpočet a a b se potom použijí následující vzorce:

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)}$$

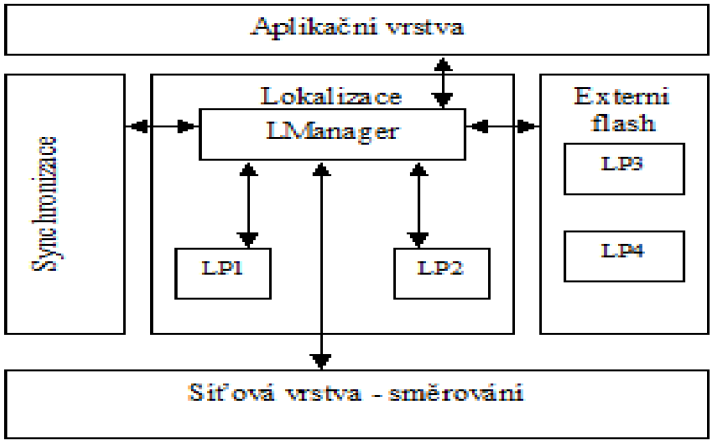
$$b = \frac{\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i\right) - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

Vzorec č. 12 Metoda nejmenších čtverců

Takto lze aproximovat i pro složitější závislosti, například exponenciální nebo logaritmickou.

6.7. Začlenění protokolu do bezdrátové senzorové sítě

Pro začlenění lokalizačního protokolu do bezdrátové senzorové sítě je nutné doplnit do každého uzlu jednotlivé lokalizační mechanizmy, tak aby mohl každý uzel zvolit správný mechanizmus pro určení své polohy. Součástí každého uzlu musí být tedy i řízení, které bude vhodný mechanizmus vybírat. Struktura takového uzlu je vidět na obrázku číslo 4.



Obr. č. 4 Struktura uzlu

Důležitými prvky uzlu jsou lokalizační manager (LManager), lokalizační protokol (LP) a rozhraní, kterým je začleněn do struktury sítě. Dále je zde obsažen blok synchronizace, který zajišťuje, že bude lokalizace probíhat u všech uzlů současně. Dalším blokem je externí flash disk, na kterém jsou nahrány lokalizační protokoly. A důležitou roli hraje také napojení na síťovou vrstvu, díky níž může uzel komunikovat se svými sousedy.

6.7.1. Lokalizační rozhraní

Funguje jako spojení mezi lokalizačním managerem a protokolem. Jeho součástí jsou příkazy, pro řízení protokolu a události o stavu managera.

Příkazy, které může využívat, jsou setLocation (nastavení polohy), používá se v případě, že uzel určí svoji polohu. Dalším příkazem je getStatus pro zjištění, v jakém stavu se systém nachází.

Stavy, v kterých se může manager nacházet, jsou: Initializing, Executing a Completed.

6.7.2. Lokalizační protokol

Je implemetací protokolu do senzoru, je zcela závislý na konkrétním protokolu. Může být využito GPS, DV-Hop protokolu nebo kteréhokoli jiného protokolu. Lokalizační protokol může být přímo implementován v uzlu a nebo může být nahrán na flash disku, viz obrázek číslo 4.

6.7.3. Lokalizační manager

Je zodpovědný za výběr lokalizačního protokolu. Tento protokol vybírá tak, aby byl shodný s okolními uzly. Výsledky lokalizačního protokolu shromažďuje, případně přes síťovou vrstvu posílá dál.

6.8. Průběh lokalizace

6.8.1. Lokalizace řízená seshora

Základní myšlenkou lokalizačního protokolu je to, že lokalizace bude řízena se shora tedy od koordinátora sítě. Ten bude říkat, kdy bude lokalizace probíhat a v jakých intervalech. Koordinátor sítě bude využívat dynamicé mobilní schéma. Bude tedy vyhodnocovat rychlost jednotlivých uzlů, z které spočítá průměrnou rychlost, pomocí níž bude určovat lokalizační periodu. Loalizace bude probíhat dvoufázově. V první fázi vyšle koordinátor sítě lokalizační rámec beacon. Uzly, které budou mít implementovánu možnost přímé lokalizace, tedy GPS, Spotlight nebo určená poloha, provedou určení své polohy. V druhé fázy uzly, které již znají svoji polohu, ji pošlou všesměrově ostatním uzlům a ty buď metodou Dv, DV-Hop, TDOA

nebo dvoufázovou metodou spočítají svoji polohu. Tím se lokalizují všechny uzly. Tento proces se bude opakovat každou lokalizační periodu. Celý postup je vidět na diagramu viz. příloha č. 1.

Jak celá situace probíhá v uzlu, který určuje svoji polohu, je popsáno v příloze č. 2. Uzel příjme rámec beacon, zjistí zda je schopen určit svoji polohu některou z přímých metod, pokud ano, lokalizační manager zvolí vhodný protokol a pokusí se o lokalizaci. V případě, že proběhne lokalizace bez problému, posílá uzel informaci o své poloze všesměrově ostatním uzlům. Pokud se lokalizace nezdaří, začne uzel naslouchat. Jakmile uzel získá informace k tomu, aby mohl použít některou z nepřímých lokalizačních metod, tak lokalizační manager spustí příslušný lokalizační protokol a spočítá svoji polohu. Poté uzel spustí časovač a po jeho vypršení pošle informace o svojí poloze ostatním uzlům. Časovač je spouštěn z toho důvodu, aby uzly, které zatím neurčily svoji polohu, měly nejprve šanci využití majáků, které je přesnější, než výpočet z polohy uzlu, která byla určena nepřímou metodou.

6.8.2. Lokalizace řízená uzlem

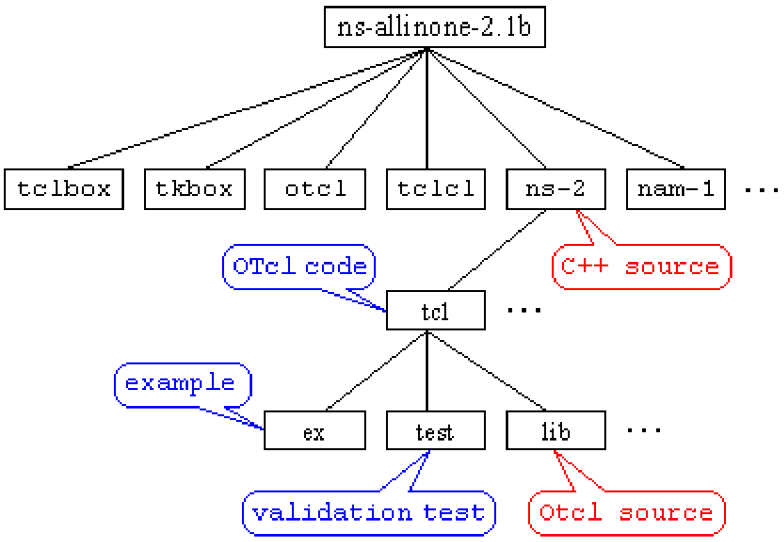
Při této variantě musí být všechny uzly vybaveny accelerometrem. Každý uzel si potom v případě změny směru zažádá o určení jeho polohy. Jak celý proces probíhá v jednotlivých uzlech je znázorněn v příloze č. 7. Uzel zaznamená změnu pohybu, některým z lokalizačních algoritmů je určena poloha uzlu. Uzel si spustí časovač a po jeho vypršení provede lokalizaci znovu, tím určí svůj směr a rychlost. V případě, že je potom koordinátorem sítě požádán o data, svoji polohu spočítá pomocí metody postupného výpočtu a spočtenou polohu přiloží k datům. V případě, že bude uzel stále na stejném místě a nebo se bude pohybovat konstantní rychlostí v přímém směru, nemusí docházet k energeticky náročnému určování polohy, poloha se spočítá.

7. Network Simulator 2

Je simulační nástroj, jehož základem jsou dva programovací jazyky. Objektově orientovaný simulátor napsaný v jazyce C++ a překladač OTcl. Jazyk C++ nám dovoluje vykonávat velice rychle a efektivně jednotlivé simulace. Diky jazyku OTcl jsme schopni vytvářet jednotlivé topologie. Jazyk OTcl je rozšířením jazyka Tcl o objektově orientovné programování. Jeho syntaxe je velice jednoduchá a díky tomu se dá snadno integrovat do jiných jazyků. NS2 má velice bohatou knihovnu sítí a protokolů. Tu lze dále rozšiřovat o vlastní protokoly.

7.1. Implementace vlastního protokolu do NS2

Ještě než se pustíme do implementace vlastního protokolu, bude nejlepší, když se seznámíme se strukturou programu NS2. Na obrázku č. 5 je vidět rozmístění jednotlivých zdrojových kódů po adresářích.



Obr. č. 5 Adresářová stuktura NS2

V adresáři tcl je umístěn adresář lib, v němž jsou obsaženy základní stavební prvky sítě, které jsou již v simulátoru implementovány. Například to jsou agenti, kteří reprezentují

protokoly, linky, pakety, adresování a směrování. Další dva podadresáře jsou ex a test. První obsahuje příklady různých simulačních skriptů. Druhý obsahuje testovací skripty.

Chceme-li implementovat nový protokol do NS2, například s názvem MyAgent, je nutné napsat kód v jazyce C++ a ten navázat na OTcl. Prvním krokem pro implementaci nějakého protokolu je vytvoření třídy Agenta v jazyce C++ například s názvem MyAgent, který dědí vlastnosti z třídy Agent. Poté je potřeba vytvořit instanci objektu v jazyce OTcl. To se provede tak, že nadefinujeme vazbu v objektu MyAgentClass, který dědí ze třídy TclClass. Výše popsané úkony jsou vidět na následujících řádcích.

```
class MujAgent : public Agent {
public:
        MujAgent();
protected:
        int command(int argc,const char*const* argv);
private:
        int my_var1;
        double my_var2;
        void MyPrivFunc (void);
};
static class MyagentClass : public TclClass {
public:
        MyAgentClass() : TclClass ("Agent/MyagentOtcl") {}
        TclObject* creat(int, cons char*const*) {
                return(new MyAgent());
        }
} class_my_agent;
```

Při prvním spuštění Network Simulátoru dojde k vytvoření konstruktoru statické proměnné *class_my_agent* a tím je instance *MyAgentClass* vytvořena. V tomto procesu se vytvoří třída *agent/MyAgentOtcl* a všechny její příslušné metody v prostoru OTcl. Pokud se uživatel snaží vytvořit instanci tohoto objektu, použije příkaz *new Agent/MyAgentOtcl,* čímž vyvolá *MyAgentClass::create* , který vytvoří instance *MyAgent* a vrací adresu.

Námi vytvořený objekt *MyAgent* má dva parametry proměnných *my_var1 a my_var2.* Ty lze snadno měnit pomocí OTcl simulačního skriptu. Abychom to mohli provést, použijeme funkci *bind* pro každou proměnnou třídy C++, kterou chceme exportovat. Funkce *bind* vytvoří nového člena proměnné, který má stejné jméno jako proměnná ve třídě *Agent/MyAgentOtcl* a vytvoří obousměrnou vazbu mezi proměnnou třídy OTcl a proměnnou v C++, jejíž adresa je specifikována jako druhá proměnná. Příklad použití funkce *bind* je vidět níže.

```
Myagent::MyAgent () : Agent (PT_UDP) {
        bind ("my_var1_otcl", &my_var1);
        bind ("my_var2_otcl", &my_var2);
}
```

Funkce *bind* umístíme do kódu námi definovaného agenta *MyAgent,* tak aby se provedla funkce *bind* při vytváření instance objektu. *Bind* nám dovoluje využívat čtyři odlišné funkce *bind* pro pět různých proměnných:

bind () : pro realné a celočíselné proměnné
bind_time () : pro časové proměnné
bind_bw () : prošířku pásma
bind_bool () : pro proměnnou boolean

Takto lze měnit parametry a proměnné síťových komponent, které jsou implementovány v C++.

Pokud chceme mít kontrolu nad objektem napsaným v C++, je dobré nadefinovat si funkci *command* v našem objektu *MyAgent,* který funguje jako příkazový překladač jazyka OTcl. Na následujících řádcích je vidět využití funkce *command* pro našeho agenta *MyAgent*.

```
int MyAgent::command(int argc, const char*const* argv) {
        if (argc == 2) {
                if (strcmp (argv[1], "call-my-priv-func") == 0) {
                        MyPrivFunc ();
                        return(TCL_OK);
                }
        }
```

```
        return(Agent::command(argc, argv));
}
```

Pokud istance OTcl odpovídá objektu MyAgent, který je vytvořený v prostoru OTcl "*set myagent [new Agent/MyagentOTcl]*" a uživatel zkouší volat funkci tohoto objektu *"$myagent call-my-priv-func*" OTcl nejprve hledá funkci v objektu OTcl, pokud ji nemůže nalézt, tak vyvolá *MyaAgent::command* a prochází uživatelské OTcl funkce a argumenty v argc/argv formátu. Pokud je volaná funkce nalezena, vykoná se a vrátí výsledek. Pokud ne, je funkce hledána, dokud není nalezena. Pokud nelze funkci nalézt, je vypsáno chybové hlášení.

Po implementaci nového objektu v C++, můžeme vykonat OTcl příkaz z C++ objektu. Níže je uvedena implementace *MyPrivFunc* do funkce *Myagent*, který tvoří OTcl interpreta, který vypíše hodnoty proměnných *my_var1* a *my_var2*.

```
void MyAgent::MyPrivFunc (void) {
        Tcl& tcl = Tcl::instance();
        tcl.eval ("puts \"Message From MyPrivFunc\"");
        tcl.evalf ("puts \"      my_var1 =  %d\"", my_var1);
        tcl.evalf ("puts \"      my_var2 =  %d\"", my_var2);
}
```

Pro vykonání OTcl příkazu z C++, je potřeba udělat referenci na *Tcl::instance(),* která deklaruje, že se jedná o statickou proměnnou. K tomu zde máme několik funkcí .

Pro zkompilování všech výše uvedených kroků, dáme uvedené zdrojové kódy do jednoho dokumentu, který nazveme MyAgent.cc, který ještě rozšíříme o tři hlavičkové soubory a to

```
#include <stdio.h>

#include <string.h>

#include "agent.h"
```

Takto vytvořený soubor nakopírujeme do složky, kde máme network simulator 2. Poté v souboru Makefile, který se nachází v NS2 připíšeme na konec listu objektů náš objekt MyAgent.o.

Pomocí příkazu *make* zrekompilujeme NS2. Poté napíšeme skript, který nám ověří, zda vše bylo správně zkompilováno. Ten má následující tvar:

set myagent [new Agent/MyagentOtcl]
$myagent set my_var1_otcl 1
$myagent set my_var1_otcl 4.5
$myagent call-my-priv-func

První řádek vytváří našeho agenta, druhý a třetí nastavují hodnoty proměnných a poslední je příkaz pro *MyAgent.* Námi vytvořený skript uložíme pod názvem test.tcl a poté spustíme z konzolz příkazem *ns test.tcl*.

V konzole se nám vypíše následující text.

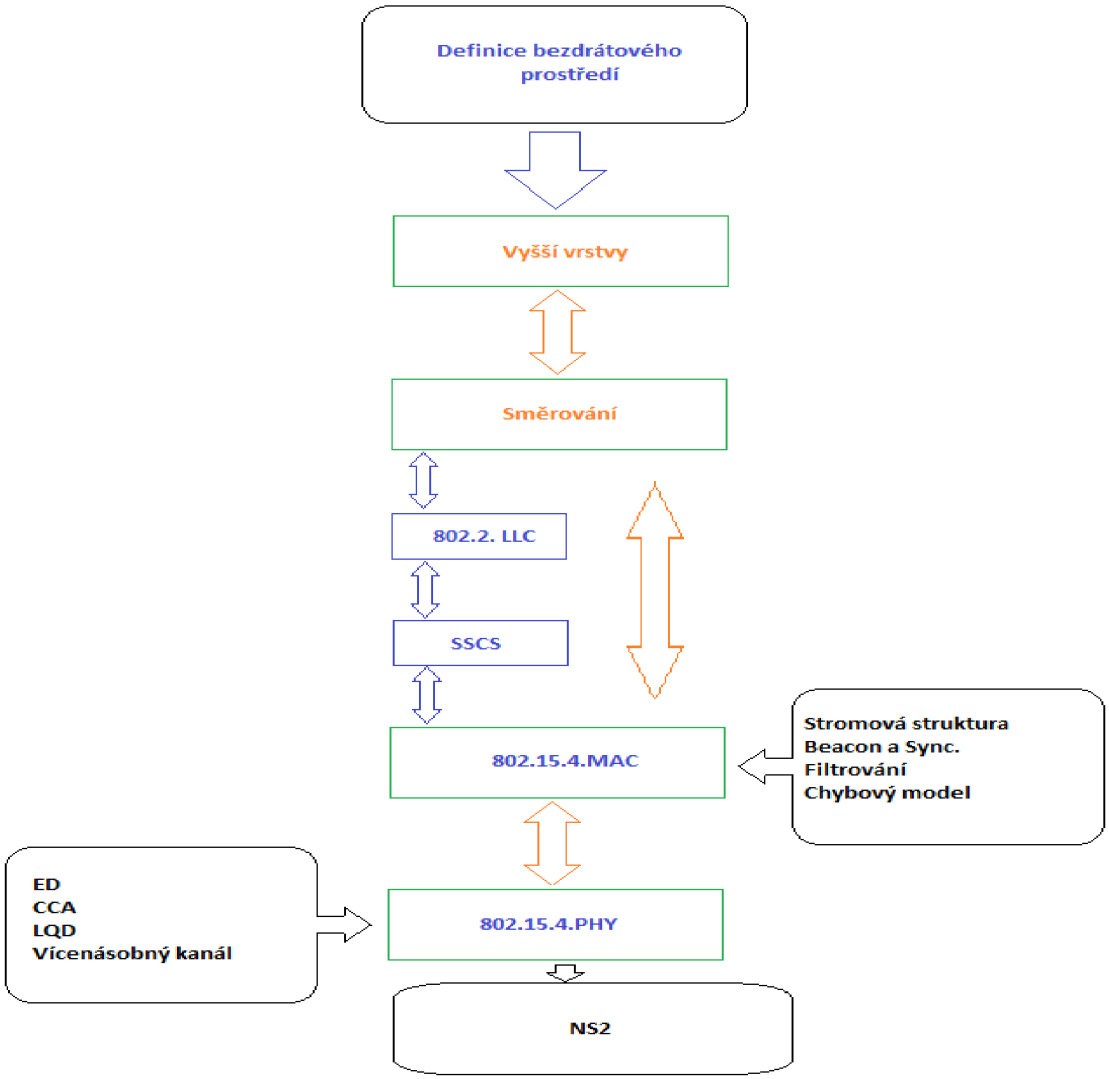
```
student@debian:~/Desktop$ ns test.tcl
warning: no class variable Agent/MyAgentOtcl::my_var1_otcl

        see tcl-object.tcl in tclcl for info about this warning.

warning: no class variable Agent/MyAgentOtcl::my_var2_otcl

Message From MyPrivFunc
     my_var1 = 1
     my_var2 = 4.500000
student@debian:~/Desktop$
```

Obr. č. 6 Výpis konzole po spuštění testovacího kódu

7.2. Protokol ZigBee v NS2

Výše bylo popsáno, jak se obecně implementuje nový protokol do NS2. My budeme v našich simulacích využívat již implementovaný protokol ZigBee, a proto si vysvětlíme, jak je tento protokol začleněn do NS2.

Na obrázku číslo 7 je popsána stukruta protokolu ZigBee a její navázání na NS2. Protokol ZigBee pracuje jen na dvou nejnižších vrstvách. Proto jsou vybírány ze struktury NS2 další protokoly na vyšších vrstvách, aby mohla celá simulace fungovat. Nyní se podívejme na strukturu jednotlivých adresářů protokolu 802.15.4.



Obr. č. 7 Struktura protokolu ZigBee

Pokud bychom vytvářeli úplně nový protokol, bylo by nutné nejprve v adresáři NS2, vytvořit adresář s názvem našeho protokolu, do kterého budeme ukládat námi vytvořené soubory se zdrojovými kódy, které budou specifikovat náš protokol. Pro nejzákladnější protokol by nám postačilo pět souborů, které by měly názvy:

nazev_protokolu.h - To bude hlavičkový soubor, ve kterém budou specifikovány časovače a směrovací agent.

nazev_protokolu.cc - v tomto souboru jsou implementovány všechny časovače, směrovací agenti a spojení s jazykem Tcl.

nazev_protokolu_pkt.h - Zde jsou definovány pakety, které bude využívat náš protokol.
nazev_protokolu_rtable.h - Hlavičkový soubor, ve kterém je definována naše směrovací tabulka.
nazev_protokolu_pkt.cc - Implementace naší směrovací tabulky.

Takto bude vypadat rozložení souborů a adresářů. Nyní se podívejme na jejich logické spojení. Pro implementaci protokolu do NS2 je nutné vytvořit agenta, který dědí z třídy Agent, tato třída je propojena s jazykem Tcl a díky tomu můžeme řídit vytvořený směrovací protokol pomocí skriptů v jazyku Tcl.

Náš agent bude obsahovat vnitřní stavy a směrovací tabulky. Vnitřní stavy mohou být reprezentovány jako nová třída a nebo mohou být definovány přímo v novém agentu. Směrovací tabulky budou obsaženy v souboru **nazev_protokolu_rtable.**

V novém protokolu musí být samozřejmě definován paket, který bude přenášet data. Tato definice bude umístěna v souboru **nazev_protokolu_pkt.h**. Pokud budeme chtít námi definované pakety posílat v nějakých intervalech, je dobré k tomu využít třídu **Timer**. Časovače jsou využívány i ve spoustě dalších případů, jako je doba života paketu, časovač pro simulaci zpoždění atd. Další důležitou třídou, kterou budeme využívat, je třída **Trace** (trasovací). Tato třída zaznamenává informace o průběhu simulace, které lze později využít pro její vyhodnocení.

Takto by vypadala struktura nového jednoduchého protokolu. Nyní se podívejme na strukturu protokolu 802.15.4. v NS2. Tento protokol je umístěn ve složce WPAN. V této složce se nachází podstatně více souborů, než bylo uvedeno výše pro nový základní protokol. Jsou tam například navíc soubory *p802.15.4fail.cc* a *p802.15.4fail.h*, které definují chyby a výpadky na lince. Dále je tam soubor *802.15.4field.h*, který definuje pole rozložení uzlů v síti a některé další. Jsou tam samozřejmě i soubory podobné výše popsaným.

My se nyní blíže podíváme na soubor *802.15.4pkt.h*. V tomto souboru je definován tvar paketu. Jsou zde veškerá makra, konstanty a datové struktury potřebné pro náš paket. Pokud se blíže podíváme na jednotlivé řádky souboru, uvidíme, že hned na začátku je definován náš soubor. Poté jsou vypsány hlavičkové soubory, které bude náš paket využívat. Mezi nimi je i soubor *packet.h*, který je základem NS2 a popisuje základní stavbu paketu. První řádky souboru potom vypadají následovně:

```
#ifndef p802_15_4pkt_h
#define p802_15_4pkt_h
```

```
#include <packet.h>
#include "p802_15_4const.h"
#include "p802_15_4field.h"

#define HDR_LRWPAN(p)  (hdr_lrwpan::acces(p))
```

Na dalších řádcích je definována podoba jednotlivých paketů či rámců, protože v senzorových sítích není přenášen jen jeden typ paketu. Máme zde definovány následující typy paketů a rámců: paket s informací o adrese uzlu, rámec beacon, datový rámec, potvrzovací paket ACK a paket nesoucí příkazy v rámci sítě. Poté je zde definován tvar záhlaví paketu.

My se nyní podíváme na strukturu rámce beacon, který je používaný koordinátorem sítě k uvádění uzlů do režimu spánku. Na tomto rámci si vysvětlíme funkci jednotlivých řádků.

```
struct lrwpan_beacon_frame
{
UINT_32        SHR_PreSeq;          // Preamble 32 bit
UINT_8         SHR_SFD;             //Startovací sekvence 8 bit
UINT_8         PHR_FrmLen;          //Délka rámce 8 bit
UINT_16        MHR_FrmCtrl;         // řídící část rámce 16 bitů
                                        // --(012):       Typ rámce
                                        //        --(210)=000:  Beacon
                                        //        --(210)=001:  Data
                                        //        --(210)=010:  Ack
                                        //        --(210)=011:  MAC příkazy
                                        //        --(210)=ostatní:       Reservované
                                        // --(3):         Povolení ochrany
                                        // --(4):         Očekávaný rámec
                                        // --(5):         Požadované ACK
                                        // --(6):         Uvnitř PAN
                                        // --(789):       Reservované
                                        // --(ab):        Cílový adresovací mód
                                        //        --(ba)=00:     PAN ID a Adresové pole
                                        //        --(ba)=01:     Reservované
                                        //        --(ba)=10:     16-bit short address
                                        //        --(ba)=11:     64-bit rozšířená adr.
                                        // --(cd):        Reservovaný
                                        // --(ef):        Zdrojový adresovací mód
UINT_8                 MHR_BSN;             // Číslo sekvence rámce beacon 8 bit
panAddrInfo    MHR_SrcAddrInfo;     // Zdrojové adrsení informace
UINT_16        MSDU_SuperSpec;      // Specifikace super rámce 16 bit
                                        // --(0123):      Příkay Beacon
                                        // --(4567):      Příkaz Superrámce
```

```
                                        // --(89ab):     Final CAP slot
                                        // --(c): Prodloužení život. baterie
                                        // --(d):        Reservované
                                        // --(e): PAN Koordinátor
                                        // --(f): Povolení združování
GTSFields      MSDU_GTSFields;   // Pole GTS
PendAddrFields MSDU_PendAddrFields;  //Adressové pole
                                              // --(012):      # pro short addressing
                                              // --(3):        Reservované
                                              // --(456):      # pro rozšířené adresování
                                              // --(7):        Reservované
UINT_16                MFR_FCS;          // Kontrolni sekvence
```

Funkce každého řádku je popsána v komentáři. Složitější části jsou rozepsány. Je vidět, že při specifikaci rámce se jen vypisuje, z jakých částí bude rámec složen a kolik bitu která část zabere. Všechna pole, které rámec obsahuje, jsou popsána v některém z hlavičkových souborů. Výše popsaný rámec beacon je zobrazen na obrázku číslo 8.

Bity	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	**Preamble**																															
32	**Startovací sek.**												**Délka rámce**																**Řídící informace**			
64	**Číslo sekvence**												**Zdr. Adr. Inf.**																**Specifikace superrámce**			
96	**Pole GTS**												**Adresové pole**																**Kontrolní sekvence**			

Obr. č. 8 Rámec Beacon

Takto vytvořenou strukturu je nutné nějak připojit k třídě Paket. K tomu, abychom to mohli udělat, je nutné si uvědomit, že hlavičky paketů jsou uchovány v podobě pole znaků. Abychom k těmto polím mohli přistupovat, je nutné vytvořit offset, ve kterém je pole umístěno.

Tento offset je definován na konci souboru *802.15.4pkt.h*. Je statický a obsahuje funkci pro přístup k hlavičce a funkci, která vrací hlavičku paketu, která je k paketu připojena.

```
static int offset_;
        inline static int& offset() {return offset_;}
        inline static hdr_lrwpan* access(const Packet* p)
        {
                return (hdr_lrwpan*) p->access(offset_);
                }
```

Na začátku kódu je definováno makro, které tutu funkci využívá.

```
#define HDR_LRWPAN(p)  (hdr_lrwpan::acces(p))
```

Nyní se podívejme na to, jak je naše hlavička paketu navázána na jazyk Tcl. Abychom to zjistili, musíme přejít do dalšího souboru a tím je soubor 802_15_4mac.cc. V tomto souboru je zpřístupněna hlavička paketu 802.15.4. jazyka Tcl. K tomuto zpřístupnění slouží následující řádky:

```
int hdr_lrwpan::offset_;
static class LRWPANHeaderClass : public PacketHeaderClass
{
public:
        LRWPANHeaderClass() : PacketHeaderClass("PacketHeader/LRWPAN",
                                                sizeof(hdr_lrwpan))
        {
                bind_offset(&hdr_lrwpan::offset_);
        }
} class_hdr_lrwpan;
```

Obdobně je napsán kód pro vrstvu fyzickou. Jeho kódy jsou umístěny v souboru *802_15_4phy.cc*

Nyní, když víme, jak je protokol ZigBee začleněn v prostředí NS2, můžeme se pustit do vytvoření vlastní simulace sítě ZigBee a k ověření vlastností lokalizačního protokolu, který jsme vytvořili.

7.3. Vytvoření simulace bezdrátové senzorové sítě

Vytváření simulací pro simulátor NS2 se provádí ve kterémkoliv textovém editoru. Vytvořený zdrojový kód se musí uložit s koncovkou .tcl.

My vytvoříme simulaci bezdrátové sítě typu ZigBee, která se rozkládá na ploše 500 x 500 metrů. Obsahuje 16 senzorů, které tvoří mřížku 4 x 4. Bude probíhat komunikace mezi dvěma uzly. Jeden představuje koordinátora sítě a druhý klasický koncový uzel. Zdroj vysílání, tedy klasický uzel, se v průběhu simulace bude pohybovat.

Prvními řádky našeho zdrojového kódu vytvroříme novou simulaci a otevřeme trasovací soubory, do kterých je ukládán průběh simulace.

```
set ns_          [new Simulator]
set tracefd     [open WSN.tr w]
set namtrace    [open WSN.nam w]
$ns_ trace-all $tracefd
```

$ns_ namtrace-all-wireless $namtrace 100 100
$ns_ puts-nam-traceall {# nam4wpan #}
Mac/802_15_4 wpanNam namStatus on

Poslední řádek výše zmíněného kódu nastavuje, že se bude jednat o bezdrátovou síť standardu 802.15.4. Pokud bychom tento řádek vynechali, simulace by nefungovala, protože je tento parametr standardně nastaven na vypnutý.

V dalším kroku vytvoříme prostor pro naši topologii. Jak už bylo zmíněno výše, bude se jednat o plochu 500 x 500m

set topo [new Topography]
$topo load_flatgrid 500 500

Do simulace vložíme objekt *god*, který uchovává informace o stavu bezdrátové sítě, o jejich uzlech, počtu skoku pro doručení zprávy do cíle a další důležité informace.

set god_ [create-god 15]

Již máme vytvořenou plochu, na které se bude vyskytovat náš model, nyní vytvoříme uzel se všemi potřebnými parametry.

```
$ns_ node-config  -adhocRouting AODV \
                  -llType LL \
                  -macType Mac/802_15_4 \
                  -ifqType Queue/DropTail/PriQueue \
                  -ifqLen 50 \
                  -antType Antenna/OmniAntenna  \
                  -propType Propagation/TwoRayGround \
                  -phyType Phy/WirelessPhy/802_15_4 \
                  -topoInstance $topo \
                  -agentTrace OFF \
                  -routerTrace OFF \
                  -macTrace ON \
                  -movementTrace OFF \
                  -energyModel "EnergyModel" \
                  -initialEnergy 1 \
                  -rxPower 0.045 \
                  -txPower 0.076 \
```

-sleepPower 0.00000006 \
-idlePower 0.0012 \
*-channel $chan_1_*

Vysvětleme si ty nejdůležitější parametry pro nastavení našeho uzlu. Význam většiny parametrů lze odvodit z jejich názvů. První parametr *adhocRouting* říká, jaký typ směrovacího protokolu bude použit pro naši AdHoc síť. My jsme zvolili AODV(Ad hoc On-Demand Distance Vector). Další parametr udává typ linkové vrstvy. Pro typ MAC vrstvy jsme zvolili standard 802.15.4., který definuje síť ZigBee. Dalšími parametry se určuje typ fronty, jakým budou řazena data, velikost fronty a anténa, kterou mají uzly používat. Pak přiřadíme uzlům topologii, do které patří. Určíme, která data se mají trasovat a která ne. Dalšími parametry jsou energetické vlastnosti uzlů. Posledním parametrem přidělíme kanál, na kterém budou uzly komunikovat. Tím jsme popsali uzly, které se budou v síti nacházet a nyní je vytvoříme a vložíme do sítě.

```
for {set i 0} {$i < 16} {incr i} {
        set node_($i) [$ns_ node]
        $node_($i) randem-motion 0
    }
```

For cyklus nám vytvořil všech šestnáct uzlů a nazve je node_(0 až 15). Druhý řádek v cyklu for vypíná náhodný pohyb uzlů. Uzly jsme tedy vytvořili a teď je rozmístíme.

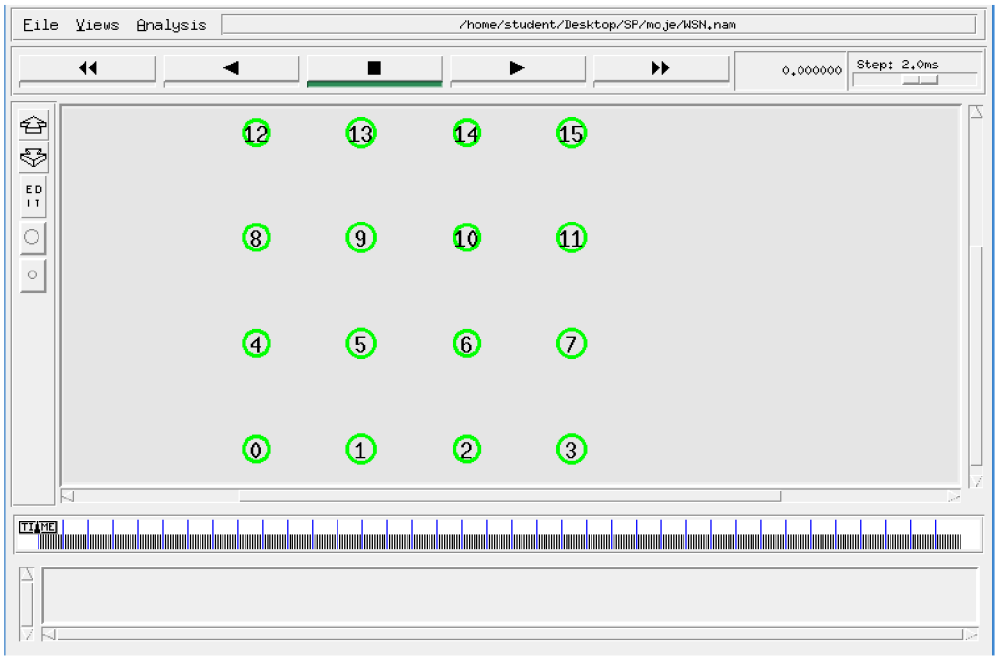
```
for {set i 0} {$i < 16} {incr i} {

  $node_($i) set X_ [expr ($i*20)%(80)]
  $node_($i) set Y_  [expr ($i/4)*(20)]
}
```

Pro rozmístění uzlů použijeme opět cyklus for, kde první řádek určuje polohu uzlu na ose X a druhý polohu na ose Y. Nyní uzel, který bude představovat náš zdroj informace uvedeme do pohybu.

```
$ns_ at 3.0 "$node_($src) setdest 50.0 45.0 5.0"
```

Zápis nám říká, že v čase 3s se uzel *src* začne přemísťovat do polohy X = 50 a Y = 45. Pohyb bude probíhat rychlostí 5 m/s.

Pokud bychom tak to vytvořenou simulaci spustili, vytvořilo by se šestnáct uzlů, ty by se nám uspořádaly do mřížky, ale nedocházelo by zatím k žádné komunikaci. Simulaci zatím bez provozu lze vidět na obrázku 9.



Obr. č. 9 Rozmístění uzlů v simulaci

Provoz v síti vytvoříme v následujících řádcích. Pro komunikaci použijeme konstantní bitový tok CBR (Constant Bit Rate), který bude pro tuto jednoduchou simulaci zatím nejlepší.

set udp($src) [new Agent/UDP]

$ns_ attach-agent $node_($src) $udp($src)

set sink($dst) [new Agent/UDP]

$ns_ attach-agent $node_($dst) $sink($dst)

$ns_ connect $udp($src) $sink($dst)

set cbr($src) [new Application/Traffic/CBR]

$cbr($src) attach-agent $udp($src)

$cbr($src) set type_ CBR

$cbr($src) set packet_size_ $size

$cbr($src) set interval_ 0.018

$cbr($src) set maxpkts_ 8000

$cbr($src) set random_ false

První odstavec nám vytvořil agenta UDP (User Datagram Protocol), ke kterému je ve druhém odstavci připojen konstantní bitový tok CBR, tím je vytvořen zdroj vysílání. Stejně tak je vytvořen i příjemce. Zdroj a příjemce nejsou zatím přiřazeni k žádnému uzlu, ale jen k proměnné *src* a *dst*. K těmto proměnným na závěr přiřadíme konkrétní uzly. Tento postup se volí z toho důvodu, že kdybychom chtěli později změnit například zdroj vysílání, provedeme to jen na jednom místě, a to u proměnné *src* a nemusíme procházet celý program.

Posledním krokem pro zřízení provozu v naší simulaci je spuštění konstantního bitového toku.

```
$ns_ at $start "$cbr($src) start"
```

V poslední části našeho kódu musíme celou simulaci ukončit. To provedeme následovně:

```
for {set i 0} {$i < 16} {incr i} {

    $ns_ at $stop "$node_($i) reset";

}

$ns_ at $stop "stop"
$ns_ at $stop "puts \"NS EXITING...\" ; $ns_ halt"

proc stop {} {

    global ns_ tracefd
    global ns_ namtrace
    $ns_ flush-trace
    close $tracefd
    close $namtrace
    exec nam WSN.nam &
    exit 0

}

$ns_ run
```

Nejprve restartujeme všechny uzly a poté zavřeme pomocí procedury stop všechny trasovací soubory. Poslední řádek spouští celou simulaci.

Simulace by nám v tuto chvíli ještě nešla spustit, protože nemáme definované všechny proměnné. Ty nadefinujeme následovně, text dáme na začátek našeho kódu.

set start 1.0

set stop 100.0

set src 1

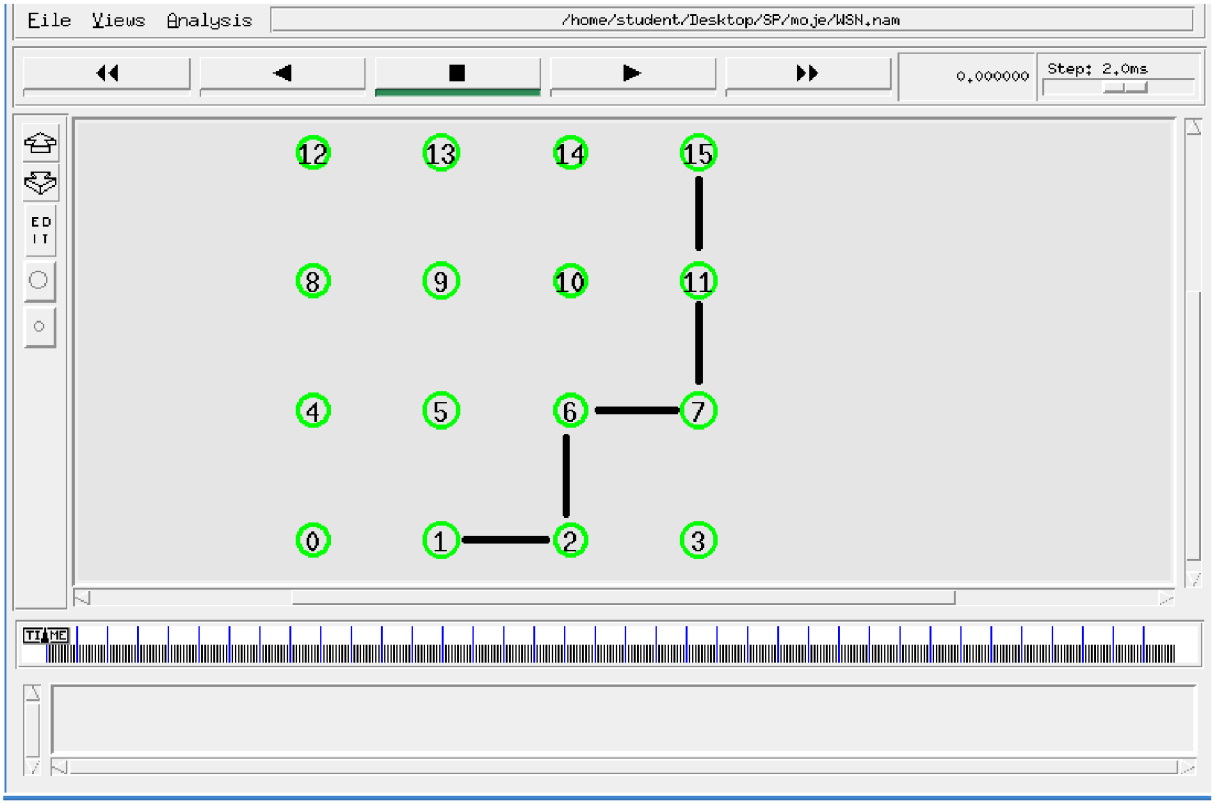
set dst 15

set size 10

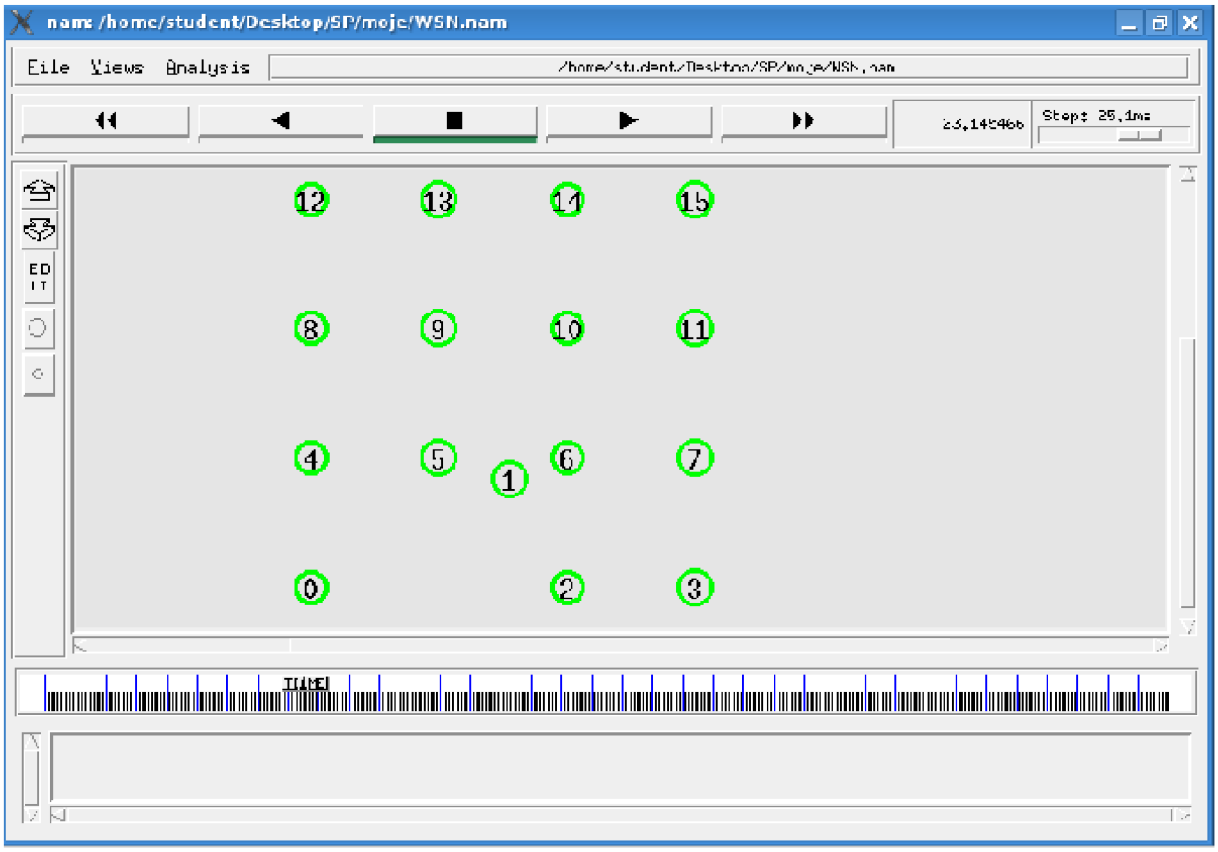
Začátek provozu tím nastavíme na jednu sekundu, konec na sto sekund. Jako zdrojový uzel jsme zvolili uzel 1 a cílový je uzel 15. Velikost paketů je 10 kbit. Tímto jsme dokončili celý zdrojový kód naší simulace a nezbývá nic jiného, než ji spustit příkazem:

ns NázevSouboru.tcl

Výslednou simulaci můžeme vidět na následujících dvou obrázcích 10 a 11. Na prvním je vidět, jakou cestou probíhá komunikace, a na druhém je vidět, že došlo k přesunu zdrojového uzlu.



Obr. č. 10 Provoz v simulaci



Obr. č. 11 Pohyb uzlu v simulaci

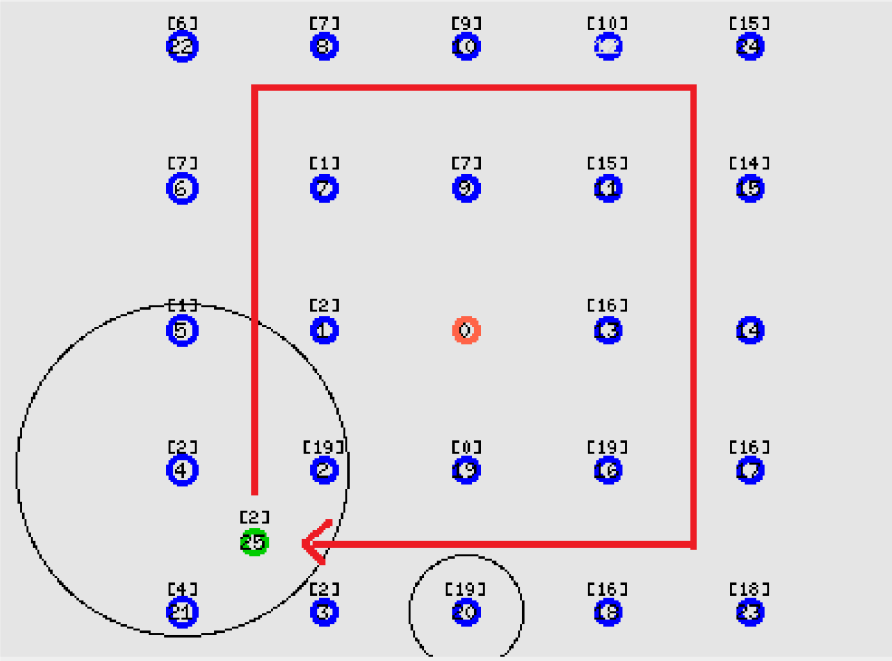
Námi navržená simulace zatím nepodporuje pohyb uzlu. Takže jakmile se náš uzel dostane mimo dosah uzlu 2, přes který posílá zprávy, dojde k tomu, že se komunikace ukončí.

7.4. Simulace mobilních schémat

Výše popsanou simulaci poupravíme tak, aby podporovala mobilitu uzlu a rozšíříme ji ještě o další uzly tak, abychom mohli simulovat a vyhodnotit různé varianty mobilních schémat.

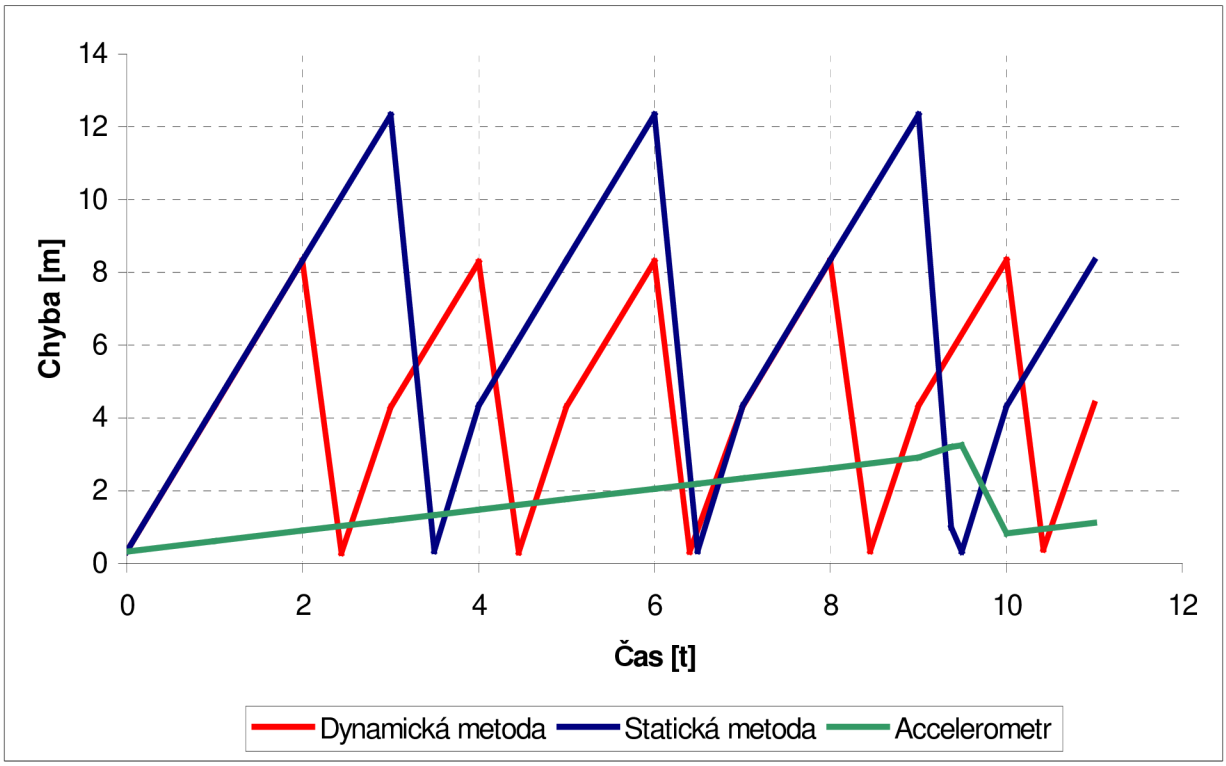
Zaměříme se na statické a dynamické mobilní schémata a na využití accelerometru. Všechna schémata budeme testovat v síti ZigBee, která je v NS2 již implementována, viz kapitola 7.2. Rozmístění uzlů v simulaci je vidět na obrázku číslo 12. V naší simulaci je celkem 26 uzlů, jeden vystupuje jako koordinátor sítě, 24 uzlů je statických a funguje jako síť majáků a jeden uzel se v této síti pohybuje. Směr jeho pohybu je zřetelný z obrázku. Mobilní

uzel se v první simulaci pohybuje rychlostí 4 m/s. Aktualizace dat pro statické mobilní schéma bude probíhat každé tři sekundy. U dynamické lokalizace si mezní chybu nastavíme na 8m. Celkový úsek, který uzel urazí, je 150 m. První simulaci zaměříme na vyhodnocení chyby polohy uzlu v závislosti na čase.

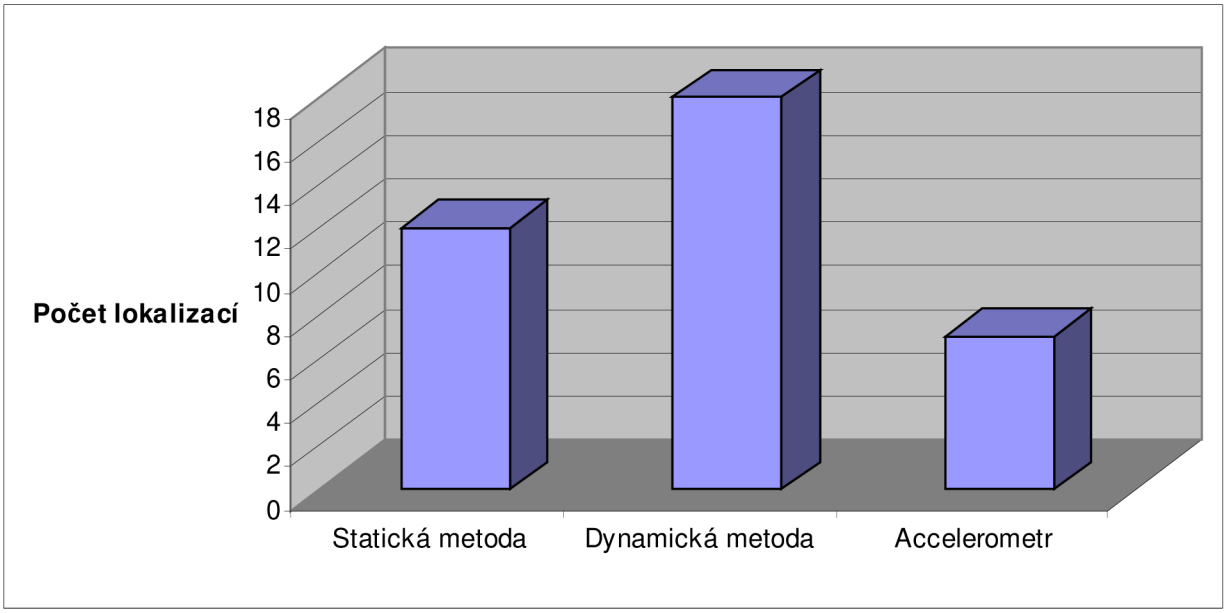


Obr. č. 12 Rozmístění uzlů v simulaci

Výsledky simulace jsou zobrazeny v grafu č. 13. Na začátku simulace všechny metody určí svoji polohu, proto mají všechny počátek ve stejném bodě, poté už se každá řídí svým algoritmem. Je zřetelné, že nejlépe ze všech metod vychází využití accelerometru. Chyba, kterou vykazuje použití accelerometru, je způsobena chybou vzniklou při lokalizaci uzlu a šíření této chyby v rámci výpočtu pozdější polohy uzlu. Nejhůře ze všech schémat vychází statické mobilní schéma. To vykazuje chybu v nejhorším bodě, více jak 12 metrů. Dynamické mobilní schéma má nastavenou hodnotu chyby na osm metrů a tu dodržuje. V grafu číslo 14 je znázorněno, kolikrát jednotlivé metody provedly aktualizaci polohy uzlu v průběhu celé simulace. Tyto hodnoty jsou velice důležité pro životnost baterie. Nejlépe opět vychází accelerometr, který při každé změně směru provede dvě lokalizace, aby byl schopen určit směr a rychlost uzlu. Nejhůře dopadla dynamická lokalizace, která v průběhu 150 metrů, které uzel urazil, provedla 18 aktualizací.



Obr. č. 13 Chyba lokalizace v závislosti na čase pro rychlost 4m/s



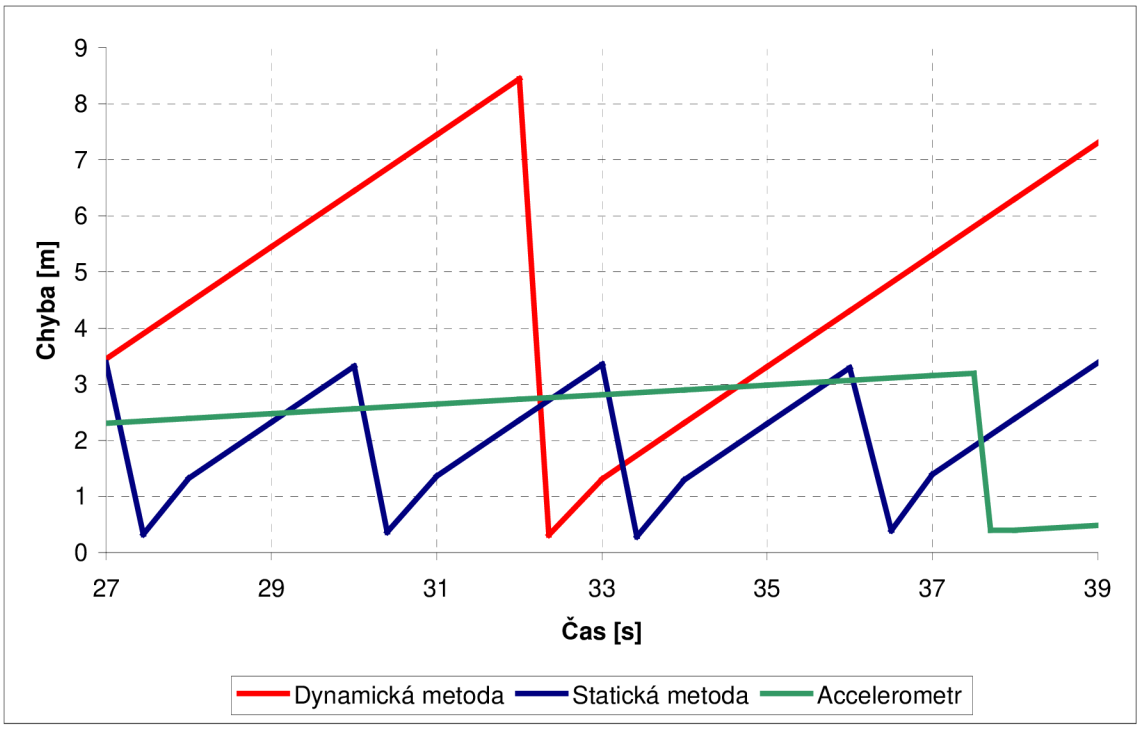
Obr. č. 14 Počet lokalizací při rychlosti 4 m/s

Abychom zjistili, jak se jednotlivá schémata chovají při nižších rychlostech, snížíme rychlost uzlu na 1 m/s, dráhu uzlu a všechny ostatní parametry ponecháme stejné.

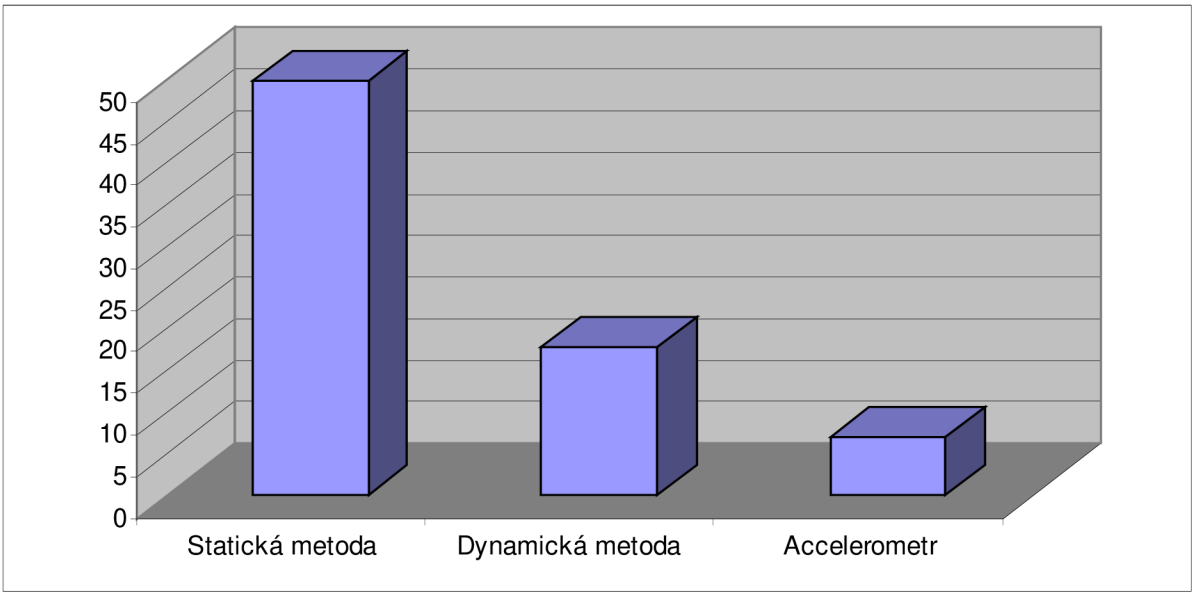
Doba celé simulace se nám zvýší na 150 sekund. Výsledky jsou opět zobrazeny ve dvou grafech a to pro chybu v závislosti na čase a počet lokalizací.

Největší chybu vykazuje dynamická metoda, ale to jen z toho důvodu, že máme nastevenou maximimální chybu na 8m. Pokud bychom ji snížili, dosáhli bychom tak nižší chyby. Metoda statická a metoda dynamická mají obdobné výsledky.

Co se týče počtu aktualizací, tak nejhůře dopadla statická metoda, která v průběhu celé simulace provedla padesát aktualizicí polohy, což by mohlo mít za následek brzké vybití baterií. Dynamická metoda má stejné, výsledky jako v předchozí simulaci a to osmnáct aktualizací. Je to způsobeno tím, že počet aktualizací je závislý na úseku, který uzel urazí a ne na jeho rychlosti. Stejně tak využití accelerometru nám dává stejný výsledek, protože aktualizace se provádí jen při změně směru nebo rychlosti.

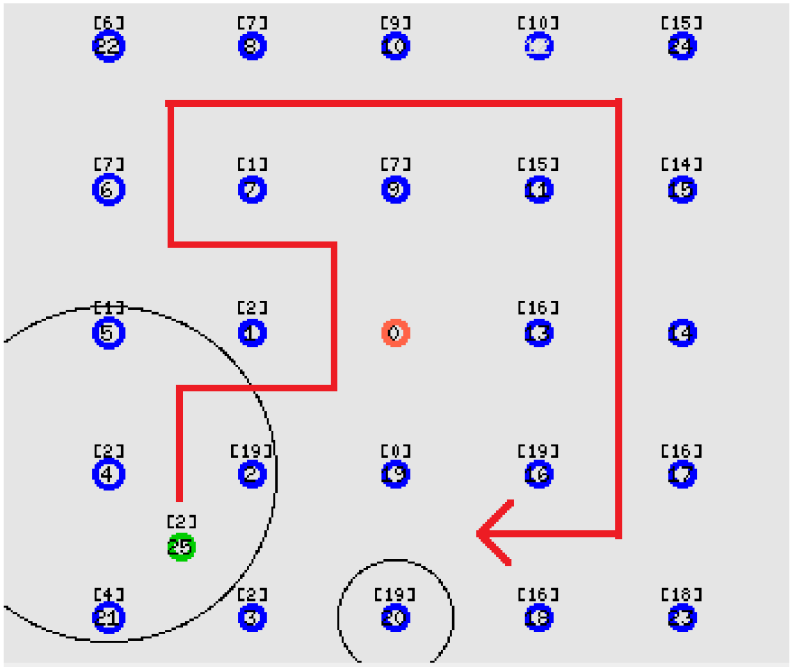


Obr. č. 15 Chyba lokalizace v závislosti na čase pro rychlost 4m/s



Obr. č. 16 Počet lokalizací při rychlosti 1 m/s

Ze všech simulovaných metod nám zatím nejlépe vychází metoda s použitím accelerometru. V další simulaci prověříme jednotlivá mobilní schémata na častou změnu směru a rychlosti. Rychlost budeme měnit v rozsahu 1 m/s až 4 m/s a uzel se bude pohybovat tak, jak je znázorněno na obrázku.

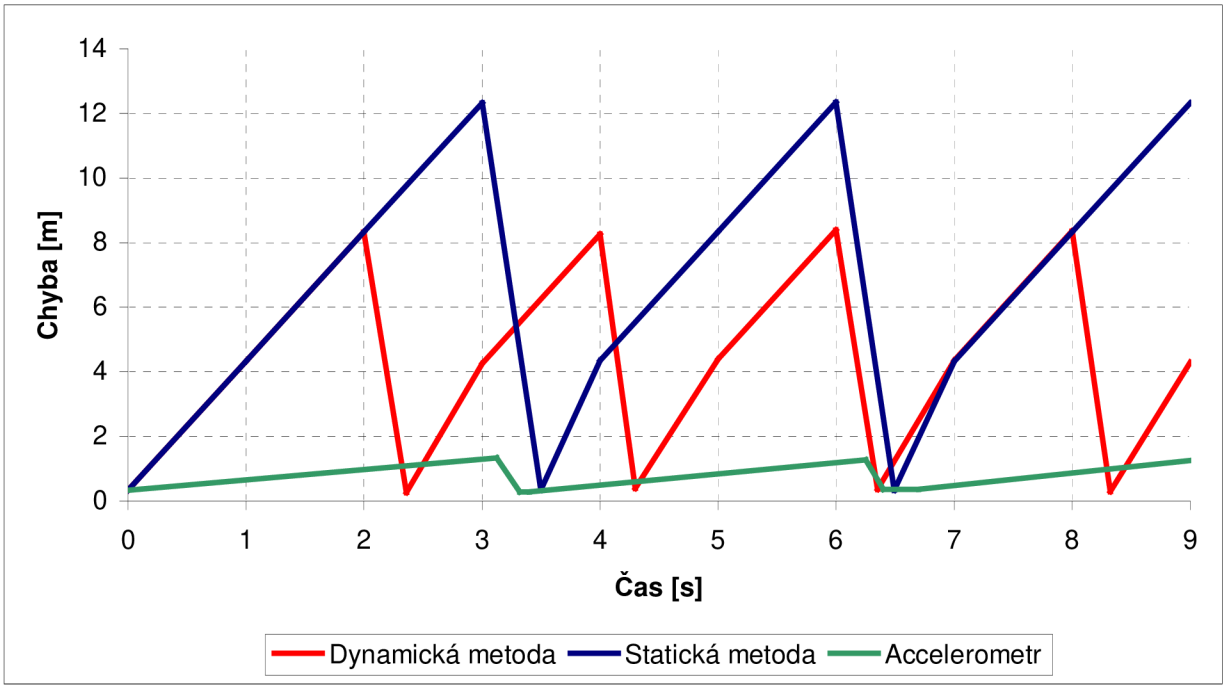


Obr. č. 17 Pohyb uzlu(změny směru a rachlosti)

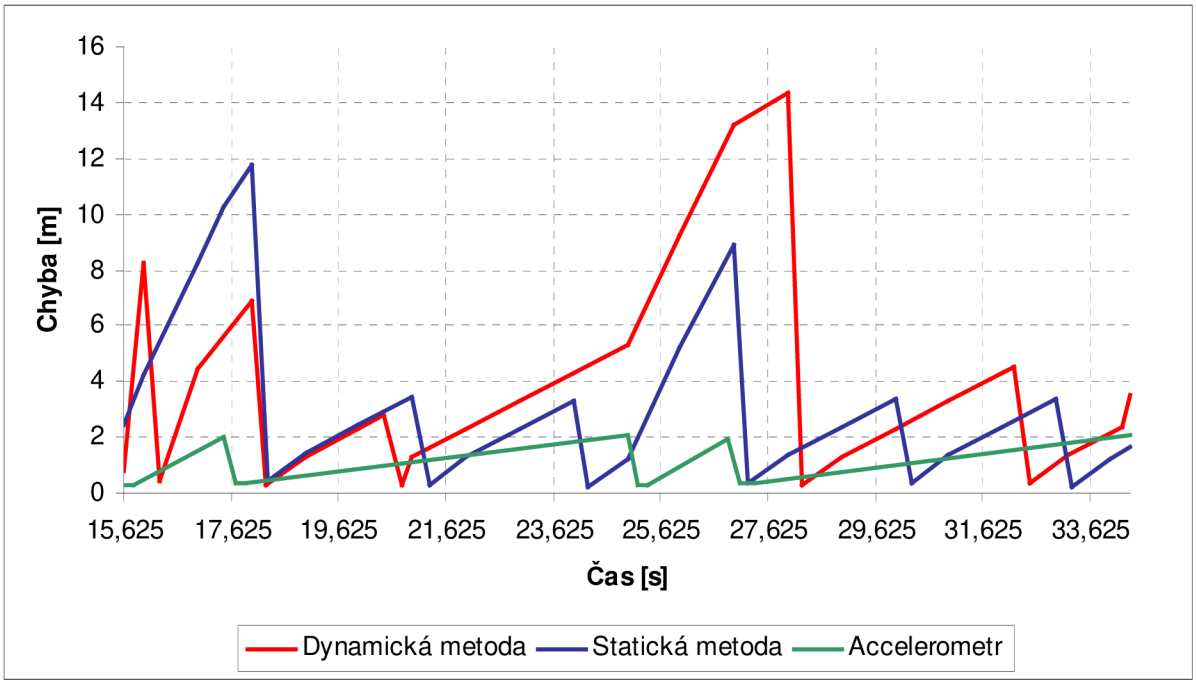
V krátkých úsecích simulace má uzel konstantní rychlost. V dlouhých úsecích se uzel zrychluje a zpomaluje.

Výsledky simulace jsou zobrazeny ve třech grafech. V prvním grafu je zobrazena chyba v závislosti na čase v prvním úseku, kde dochází pouze k časté změně směru. Ve druhém grafu je zobrazena opět chyba, ale tentokrát v úseku, kde dochází ke změně rychlosti. Ve třetím grafu je zobrazen počet aktualizací polohy v průběhu celé simulace.

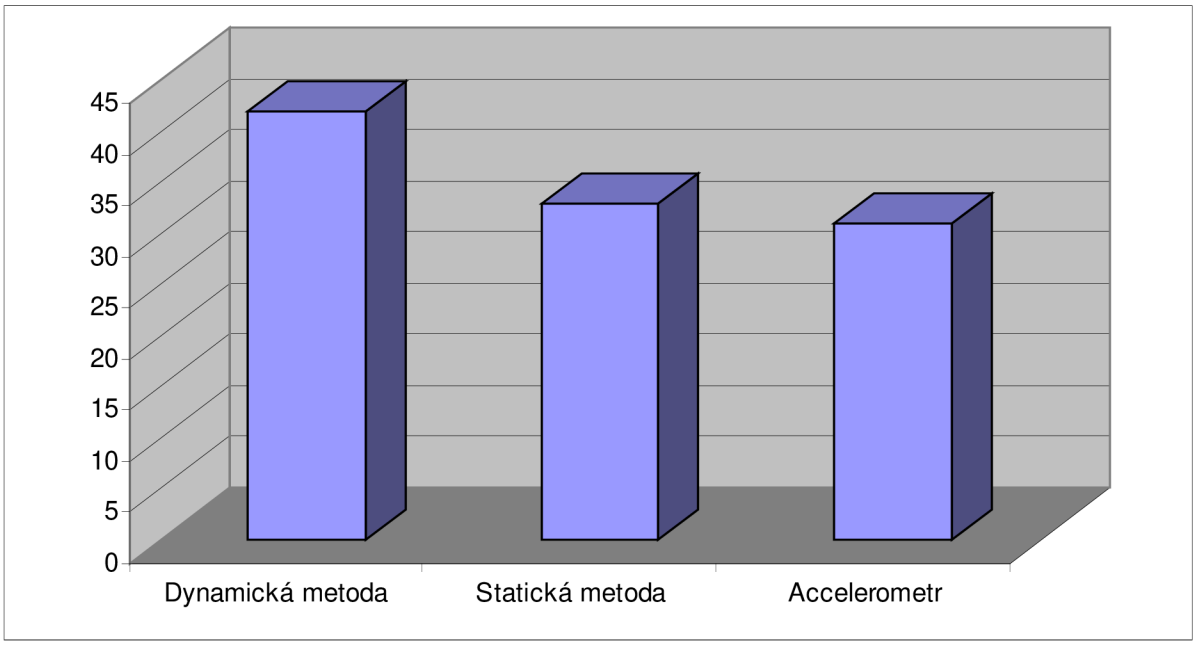
Z prvního grafu vyplývá, že v případě změny směru, která proběhne na úseku 62,5 metrů čtyřikrát, vychází nejlépe využití accelerometru. Druhou nejlepší metodou je využití dynamické metody a nejhůře vychází metoda statická. V druhém grafu je již situace jiná, accelerometr má stále nejlepší výsledky, ale dynamická metoda nestíhá rychle reagovat na změny rychlosti a tím nám vzniká chyba, větší než jsme si nastavili jako maximální. V jejím maximu má hodnotu více jak 14 metrů. Co se týče počtu aktualizací, tak téměř stejného výsledku dosáhla statická metoda a accelerometr. Dynamická metoda měla aktualizací nejvíce. Je důležité ještě zmínit, že ve druhém úseku uzlu, kde docházelo k časté změně rychlosti, měl nejvíce aktualizací accelerometr.



Obr. č. 18 Chyba lokalizace v závislosti na čase pro změnu směru



Obr. č. 19 Chyba lokalizace v závislosti na čase pro změnu rychlosti



Obr. č. 20 Počet lokalizací při rychlosti změně směru a rychlosti

7.4.1. Zhodnocení výsledků simulace

Výsledky výše zmíněných simulací ukazují, že každé z mobilních schémat má své klady i zápory a hodí se do různých typů sítí. Výhody a nevýhody jednotlivých schémat jsou shrnuty v tabulce číslo 21. Podrobně jsou výsledky rozepsány v následující kapitole.

Metoda	Výhody	Nevýhody
Statická metoda	Jednoduchá implementace	Chyba roste s rychlostí Neovlivnitelná velikost chyby
Dynamická metoda	Volba max. chyby Přizpůsobení se změně rychlosti	Nevhodné pro velké změny rychlosti
Accelerometr	Velká přesnost Malý počet lokalizací	Nutnost dalšího zařízení

Obr. č. 21 Přehled mobilních schémat

Statická metoda:

Statická metoda v simulacích měla na svoji jednoduchost poměrně dobré výsledky. Ze simulací vyplývá, že chyba této metody roste s rychlostí pohybu uzlu. Tato metoda je tedy vhodná do sítí, kde nepožadujeme velkou přesnost lokalizace. Co se týče energetické náročnosti této metody, tak ta je závislá na tom, jak nastavíme lokalizační periodu. Čím bude perioda větší, tím bude životnost sítě delší. Podstatně se nám ale snižuje přesnost lokalizace. Je tedy vždy nutné, najít kompromis mezi přesností, kterou požadujeme a životností sítě.

Ideální sítí pro nasazení statické metody je síť, která nevyžaduje velkou přesnost a uzly v ní se pohybují nízkou rychlostí.

Dynamická metoda:

Tuto metodu bych doporučoval do sítí, kde uzly nemění svoji rychlost a nebo je změna rychlosti plynulá. Pokud by tomu tak nebylo, metoda nestíhá dostatečně rychle reagovat na změny rychlosti, a tím nám může vznikat chyba větší, než požadujeme. Dalším důležitým aspektem je to, že na přesnost metody nemají velký vliv změny směru. Z tohoto pohledu je tato metoda výhodnější než využití accelerometru.

Pokud se budeme na tuto metodu dívat z pohledu životnosti sítě, tak nejdůležitějším aspektem dynamické metody je parametr α, který udává maximální chybu. Při volbě parametru α je nezbytné si uvědomit, že čím větší přesnost budeme požadovat, tím bude

menší lokalizační perioda, a tím bude nižší životnost sítě. Proto je nutné volit tento parametr velice obezřetně.

Využití accelerometru:

Ve všech simulacích, mělo využití accelerometru nejmenší chybu. Ta vznikala v důsledku nepřesnosti některé z lokalizačních metod, tedy přesných hodnot pro výpočet polohy v čase. Pokud by se uzel pohyboval v jednom směru až do nekonečna, byla by i výsledná chyba nekonečná. Proto je tento systém vhodný tam, kde se uzly pohybují v jednom směru maximálně stovky metrů, protože jinak se nám neúměrně zvětšuje chyba. Pokud chceme tuto metodu použít i pro delší úseky, je dobré ji nakombinovat se statickou metodou.

Z energetického pohledu je tato varianta velice výhodná tam, kde nedochází k časté změně směru nebo rychlosti. V tomto případě je tato metoda ze všech nejvýhodnější.

Velkou nevýhodou je to, že každý uzel je nutné vybavit dalším zařízením. To nám zvyšuje náklady na vybudování sítě a energetické náklady.

Využití accelerometru se nám tedy vyplatí, pokud snížíme počet lokalizací na tolik, aby to minimálně pokrylo jeho energetické náklady. Ideální sítí pro nasazení accelerometru je síť s uzly s konstantní rychlostí, které minimálně mění směr.

8. Závěr

V této práci jsem seznámil čtenáře s vlastnostmi bezdrátových senzorových sítí. Konkrétně jsem vše přiblížil na síti ZigBee, která je v současnosti stále ve vývoji.

Teoretické části jsem věnoval prvních pět kapitol. Ve druhé a třetí kapitole se zaměřuji na vlastnosti bezdrátových senzorových sítí, respektive ZigBee. Ve čtvrté kapitole jsem popsal mobilní sběr dat. Pátá kapitola je zaměřena na využití majáků pro lokalizaci a jejich nasazování.

Na teoretickou část navazuje část praktická. Šestá kapitola je zaměřena na návrh lokalizačního protokolu s využitím různých způsobů lokalizace a různých mobilních schémat. Navrhl jsem dva lokalizační protokoly, jeden využívá accelrometru, druhý je založen na několika lokalizačních algoritmech. V poslední sedmé kapitole je popsána funkce programu Network Simulátor 2 a začlenění protokolu ZigBee do jeho struktury. V závěru této kapitoly jsou v tomto prostředí simulována různá mobilní schémata.

Z výsledku simulací lze usoudit, že neexistuje žádné ideální mobilní schéma. Je vždy nutné volit mobilní schéma podle vlastností navrhované sítě.

Věřím, že tato práce dostatečně čtenáři osvětlí problematiku lokalizace a mobility v bezdrátových senzorových sítích.

9. Použitá literatura

[1] Information Sciences Institute, "The Network Simulator - ns-2", June 2004, [online] URL:<http:// www.isi.edu/nsnam/ns/>.

[2] S. R. Gandham *et al.*, "Energy Efficient Schemes for Wireless Sensor Networks With Multiple Mobile Base Stations," *Proc. IEEE GLOBECOM*, 2003.

[3] D. Niculescu and B. Nath, "DV-based positioning in adhoc networks," *Telecommunication Systems*, 2003.

[4] ZigBee Alliance, [online] URL:<*http://www.caba.org/standard/zigbee.html*>.

[5] Perkins C., Belding-Royer E., Das S., "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC3561, July 2003.

[6] Silva R., Silva J., Simek M., Boavida F., "Why should multicast be used in WSNs", June 2008

[7] D. Culler and W. Hong., "Wireless sensor networks, Special Issue", 2004.

[8] Jae Chung, Mark Claypool: NS by Example, [online] URL:<http://nile.wpi.edu/NS/>

[9] K. Kar, M. Kodialam, T.V. Lakshman, and L. Tassiulas, "Routing for Network Capacity Maximization in Energy-constrained Ad-hoc Networks," *Proc. IEEE INFOCOM*, 2003.

[10] R. Stoleru, J. A. Stankovic, and S. H. Son, "Robust node localization for wireless sensor networks," in *Proceedings of IEEE Workshop on Embedded Networked Sensors (EmNetS)*,2007.

[11] D. Niculescu and B. Nath, "DV-based positioning in adhoc networks," *Telecommunication Systems*, 2003.

[12] Sameer Tilak, Vinay Kolar, Nael B. Abu-Ghazaleh, Kyoung-Don Kang, "Dynamic localization control for mobile sensor networks*", 2005*.

[13] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proc. of the 1st ACM WSNA*, 2002.

[14] C. Chevallay, R. E. Van, and D. T. A. Hall, *"*Self-organization protocols for wireless sensor networks*," In In Thirty Sixth Conference on Information Sciences and Systems, 2002.*

[15] A. Choksi, R. P. Martin, B. Nath, and R. Pupala, "Mobility support for diffusion-based ad-hoc sensor networks*," Technical report, Department of Computer Science, Rutgers University, 2002.*

[16] L. Hu and D. Evans, "Localization for mobile sensor networks. MobiCom '04*", 2004.*

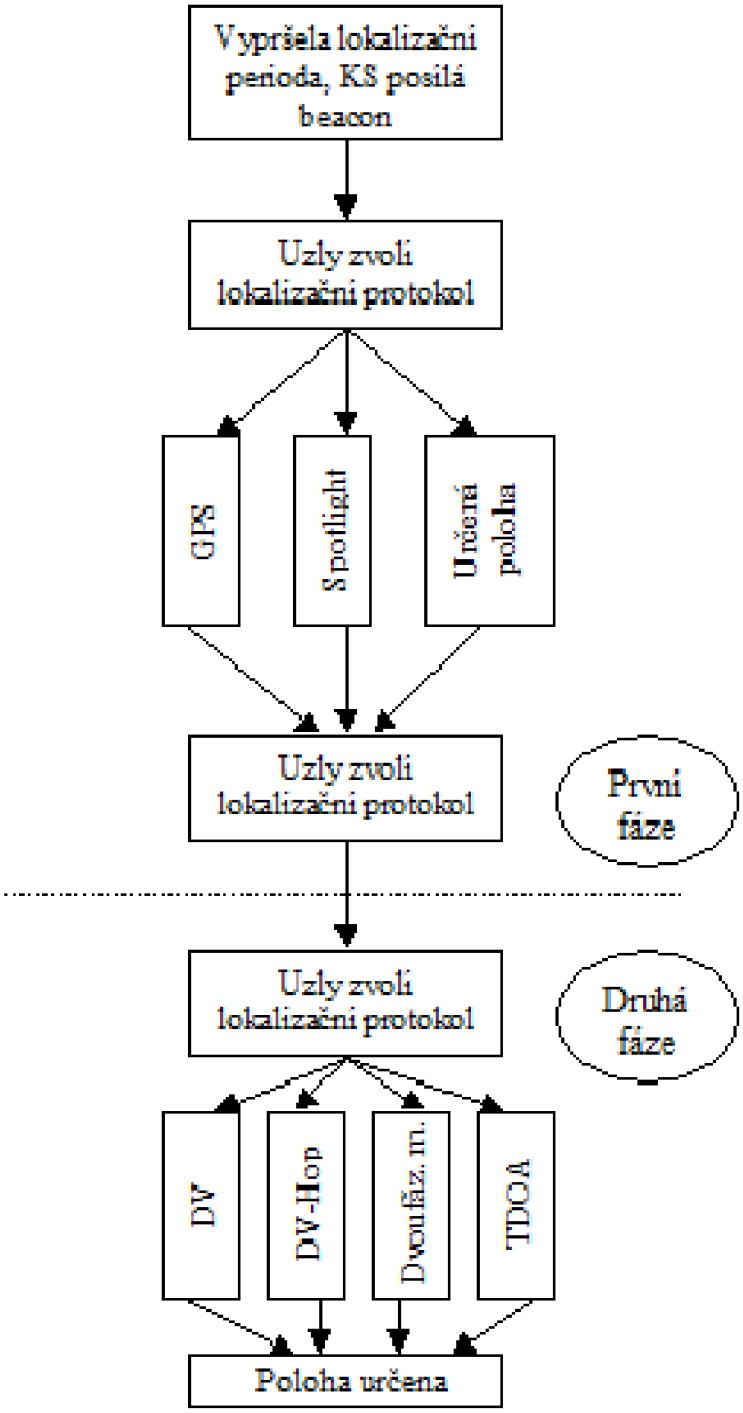
[17] S. Capkun, M. Hamdi, and J.-P. Hubaux, "GPS-free Positioning in Mobile Ad-Hoc Networks," *Cluster Computing Journal*, vol. 5, 2002.

Přílohy

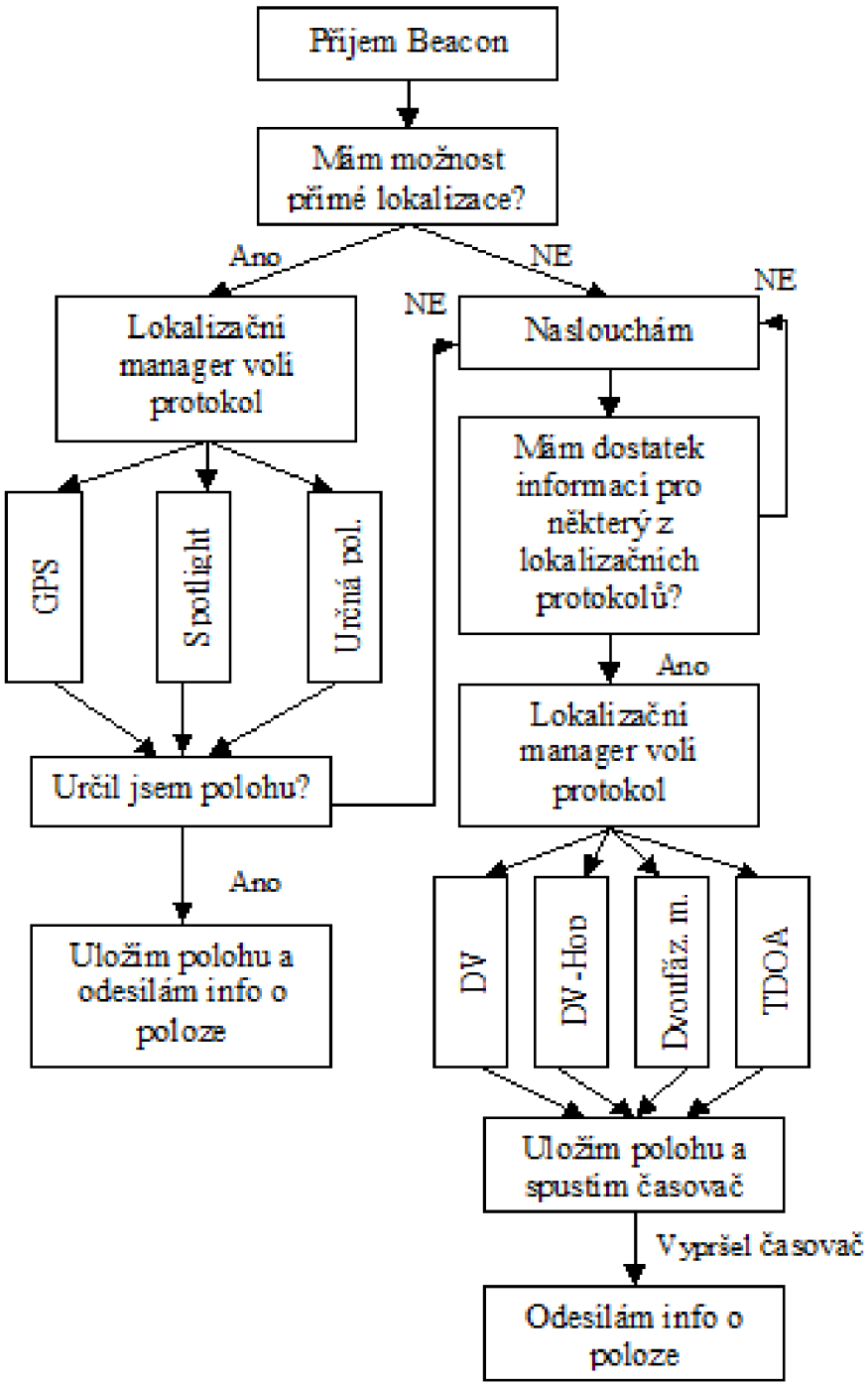
Seznam příloh:

1. Průběh lokaliazace
2. Průběh lokalizace v uzlu
3. Využití accelerometru

1. Průběh lokalizace



2. Průběh lokalizace v uzlu



3. Využití accelerometru