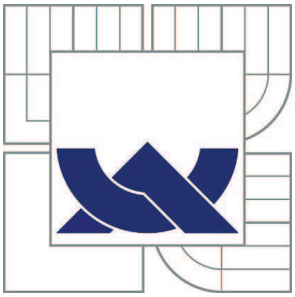
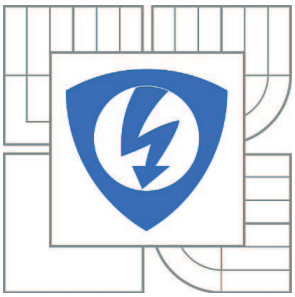


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE BEZKOTEVNÍ LOKALIZAČNÍ TECHNIKY DO SIMULAČNÍHO PROSTŘEDÍ NS2

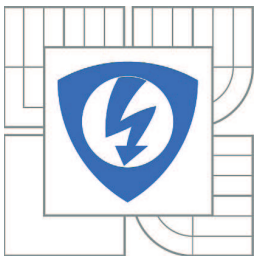
INVESTIGATION OF ANCHOR-FREE LOCALIZATION IN NETWORK SIMULATOR

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. TOMÁŠ MARTYNEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILAN ŠIMEK
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. Tomáš Martynek	***ID:***	78635
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Implementace bezkotevní lokalizační techniky do simulačního prostředí ns2

POKYNY PRO VYPRACOVÁNÍ:

Úkolem studenta je implementovat lokalizační algoritmus AFL do prostředí ns2 a pomocí tohoto modelu vyšetřit jeho efektivitu v simulačním prostředí se senzorovými uzly. Cílem práce je výzkum energetické náročnosti a přesnosti určení pozic jednotlivých uzlů tímto algoritmem. Vstupními parametry simulačního modelu je počet uzlů, plocha sítě, rádiový dosah uzlů a energetická zásoba každého uzlu. V rámci řešení diplomové práce, se student bude věnovat také výzkumu parametru RSSI a jeho závislosti na přenosovém prostředí a vzdálenosti mezi uzly. Výstupem práce je funkční simulační model, grafické závislosti a grafické zobrazení vypočtených souřadnic algoritmem AFL.

DOPORUČENÁ LITERATURA:

[1] Information Sciences Institute, "The Network Simulator - ns-2", June 2004, dostupné: <http://www.isi.edu/nsnam/ns/>
[2] Min-Xiou Chen, Yin-Din Wang, An efficient localization tracking structure for wireless sensor networks, Computer Communication, ISSN: 1495-1504

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

Vedoucí práce: Ing. Milan Šimek

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

UPOZORNĚNÍ:

Anotace

Diplomová práce se zabývá problematikou lokalizace v bezdrátové senzorové síti. Je zaměřena na implementaci bezkotevní lokalizační techniky do simulačního prostředí NS2 a následné vyhodnocení její efektivnosti. V teoretickém úvodu je krátce shrnuta technologie IEEE802.15.4 a její nadstavba ZigBee. Dále jsou popsány metody lokalizace s kotevními body (ABL) a bezkotevní lokalizace (AFL). Jsou uvedeny existující metody měření vzdáleností síťových uzlů a popsány jejich klíčové vlastnosti. Také je věnována pozornost simulačnímu prostředí NS2, především pak začlenění standardu IEEE802.15.4 do jeho struktury. V části věnované bezkotevnímu lokalizačnímu algoritmu je popsán způsob hledání a výběru jednotlivých kotevních bodů. Je popsána i metoda následné optimalizace měřením vzdálenosti sousedních uzlů za použití Mass-Spring algoritmu. Dále je popsáno začlenění nového protokolu s názvem AFLOCAL do struktury NS2 a detailně rozebrána jeho činnost. Je popsán proces lokalizace a výpočtu souřadnic. Popsán je způsob simulace a vyhodnocení výsledků lokalizace. V závěru jsou shrnuty dosažené výsledky a formulovány vlastnosti navrhnutého protokolu AFLOCAL.

Abstract

This thesis deals with the issue of localization in wireless sensor networks. It focuses on the implementation Anchor-Free localization techniques to NS2 and evaluation of its effectiveness. In the theoretical introduction is summarized IEEE802.15.4 and ZigBee technology. The following chapter describes the Anchor-Based and Anchor-Free localization methods. Furthermore, existing methods of measuring the distance of network nodes are described. Also pay attention NS2 environment, especially the inclusion of standard IEEE802.15.4. The section devoted to Anchor-Free algorithm describes how to search and select anchor modes. It also described the method of optimization by measuring the distance of neighboring nodes using the Mass-Spring Algorithm. Next chapter describe the inclusion of a new protocol called AFLOCAL into NS2 and detailed analysis of its function. Next described the process of localization and calculating the coordinates. Described was aslo a method of simulation and evaluation of results. In conclusion, the results are summarized and formulated the characteristics of designed protocol AFLOCAL.

Klíčová slova

IEEE802.15.4, ZigBee, Lokalizace, Kotevní uzel, WSN , AFLOCAL

Keywords

IEEE802.15.4, ZigBee, Localization, Anchor node, WSN, AFLOCAL

Bibliografická citace práce

MARTYNEK, T. Implementace bezkotevní lokalizační techniky do simulačního prostředí ns2. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 59 s. Vedoucí diplomové práce Ing. Milan Šimek.

Poděkování

Děkuji vedoucímu práce Ing. Milanu Šimkovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma **Implementace bezkotevní lokalizační techniky do simulačního prostředí NS2** jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a 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

Obsah

Prohlášení 5
1. Úvod 9
2. Teoretický úvod 10
2.1. Bezdrátové senzorové sítě 10
2.2. Technologie IEEE 802.15.4 11
2.2.1. Fyzická vrstva 11
2.2.2. MAC podvrstva 12
2.3. ZigBee 13
2.3.1. Síťová zařízení 14
2.3.2. Topologie 14
2.3.3. Superrámec 15
2.3.4. Síťová a Aplikační vrstva 16
3. Přehled problematiky lokalizace uzlu 17
3.1. Měření vzdálenosti v bezdrátových senzorových sítích 17
3.1.1. Fyzikální metody 17
3.1.2. Metody pracující s informacemi sítě 18
3.2. Lokalizační algoritmy 19
3.2.1. Lokalizace s kotevními uzly (Anchor-Based Localization) 19
3.2.2. Lokalizace bez kotevních uzlů (Anchor-Free Localization) 20
4. Zvolené řešení 21
4.1. Simulační prostředí NS2 21
4.2. Bezdrátová senzorová síť 21
4.2.1. Začlenění standardu IEEE 802.15.4 v NS2 21
4.2.2. Zpracování vysílaného rámce v NS2 22
4.2.3. Zpracování přijímaného rámce v NS2 23
4.3. Lokalizační algoritmus 23
4.3.1. Volba lokalizačního algoritmu 24
4.3.2. Popis zvoleného AFL algoritmu 24
4.3.3. Určení polohy jednotlivých uzlů 24
4.3.4. Optimalizace polohy 27

5. Realizace 29
5.1. Vytvoření nového protokolu 29
5.1.1. Formát přenášené zprávy 29
5.1.2. Třída Aflocal 29
5.1.3. Třída Aflocal_Timer 32
5.1.4. Třída Aflocal_table 33
5.2. Začlenění protokolu AFLOCAL do NS2 34
5.2.1. Soubor packet.h 34
5.2.2. Soubory cmu-trace.h a cmu-trace.cc 34
5.2.3. Soubor ns-packet.tcl 35
5.2.4. Makefile 35
5.2.5. Kompilace 36
5.2.6. Připojení k uzlu v Tcl skriptu 36
5.3. Popis lokalizačního procesu 36
5.3.1. Hledání kotvy 1 37
5.3.2. Hledání kotvy 2 39
5.3.3. Hledání kotvy 3 40
5.3.4. Hledání kotvy 4 41
5.3.5. Hledání kotvy 5 42
5.3.6. Výpočet souřadnic 43
5.3.7. Optimalizace souřadnic 44
5.4. Simulace v prostředí NS2 44
5.4.1. Simulační skript pro protokol AFLOCAL 45
5.4.2. Průběh simulace 45
5.4.3. Vyhodnocení simulace 45
6. Výsledky simulace 47
6.1. Hustota sítě 47
6.2. Velikost sítě 48
6.2.1. Chyba lokalizace 49
6.2.2. Spotřeba energie 50
7. Závěr 52
8. Použité zdroje 54

9. Přehled použitých zkratek a symbolů 55
10. Seznam příloh 56
11. Přílohy 57

1. Úvod

Obliba bezdrátových senzorových sítí v poslední době stoupá, a to nejen v oblasti automatizace domácností a budov, ale především v průmyslu. Není také divu. Bezdrátové senzorové sítě (WSN, Wireless sensor network) pro svou snadnou výstavbu, energetickou úspornost, vysokou spolehlivost a nízkou cenu poskytují vynikající prostředek pro zabezpečení budov a pracovišť, monitorování technologických procesů, přítomnosti nebezpečných látek nebo pro měření fyzikálních a chemických parametrů ve sledované oblasti bez nutnosti budování infrastruktury i při velkém množství senzorů.

Často také potřebujeme znát polohu zařízení v síti. Může jít například o určení polohy stroje ve výrobní hale nebo místa požárního poplachu. Možnosti určení polohy pomoci GPS jsou omezené cenou takovéhoto řešení i energetickou náročností pro bateriově napájené zařízení. Možnosti takovéto lokalizace jsou také omezené kvalitou signálu v budovách a zastavěných oblastech.

Právě na tuto problematiku se výborně hodí řešení v podobě lokalizace v bezdrátové senzorové síti. Takováto lokalizační síť je levná, snadno sestavitelná, energeticky nenáročná a hlavně umožňuje použití uvnitř budov a výrobních hal. Navíc může plnit i jiné úkoly. Může být zároveň součástí bezpečnostního systému budovy nebo například vybavena požárními detektory.

Tato práce zaměřena právě implementaci lokalizační techniky do simulačního prostředí NS2. Použita bude lokalizace bez předem známých kotevních bodů s následnou optimalizací pomoci měření síly přijímaného signálu RSSI. Navrhnuté řešení bude následně testováno v simulačním prostředí NS2 a vyhodnocena jeho účinnost.

2. Teoretický úvod

2.1. Bezdrátové senzorové sítě

Bezdrátová senzorová síť (WSN, Wireless sensor network) je tvořena množstvím samostatných senzorů. Senzory mohou sledovat fyzikální podmínky (teplota, tlak, vibrace apod.) či například přítomnost toxických látek. Prvotním impulsem k rozvoji senzorových sítí byly vojenské aplikace (sledování a monitorování bojiště). Dnes nacházejí uplatnění v mnoha odvětvích průmyslu, pro sledování technologických procesů, monitorování životního prostředí, zdravotnické aplikace nebo řízení dopravy.

Způsobů využití bezdrátových senzorových sítí je mnoho, ať už jde o sledování a kontrolu technologických procesů v průmyslu, monitorování pracoviště (pohyb osob a zařízení), bezpečnostní systémy, požární detektory. Typicky jsou pak jednotlivé senzory rozmístěny ve sledované oblasti a poskytují tak ucelený přehled. Pomoci senzorů rozmístěných uvnitř hlídaného objektu je pak například možno (na základě detekce pohybu, zvuku, vibrací apod.) zjistit přítomnost lupiče či útočníka. Senzory zjistí narušení a informují řídící prvek senzorové sítě, který pak následně vyhodnotí vzniklou situaci a provede potřebná opatření (spustí alarm, informuje hlídače, odešle zprávu internetem apod.). Bezdrátové senzorové sítě našly uplatnění i v zemědělství, kde se používají například k monitorování vlhkosti a teploty ve sklenících, automatizace a řízení zavlažování nebo i při skladování.

Každý senzor obsahuje mikrokontroler, komunikační rozhraní (rádiový vysílač a přijímač, bezdrátové optické rozhraní) a zdroj energie (baterie, solární články). Velikost senzoru je pak závislá hlavně od velikosti baterie (a tedy požadované výdrže), výkonu a rádiového dosahu. Senzory většinou slouží pouze ke sběru dat, o samotné vyhodnocení výsledků a případná opatření (spuštění alarmu, zastavení procesu nebo informování kompetentní osoby) se stará řídící uzel sítě.

Mezi základní vlastnosti bezdrátových senzorových sítí patři:

- Energetická úspornost, dlouhá výdrž na baterie
- Práce v nepříznivých klimatických podmínkách
- Dynamická topologie sítě, mobilita uzlů
- Bezobslužný provoz
- Široká oblast využití (Vojenství, Průmysl a Zemědělství, Lékařství)

Existuje několik standardů pro bezdrátové senzorové sítě. EnOcean je standard pro automatizaci budov (vypínače, detektory otevření, apod.) s velmi nízkou spotřebou a rozmanitými způsoby napájení (solární, pohybová energie, rozdíl teplot

prostředí) bez použití externích zdrojů. ZigBee je specifikace určená pro průmysl, sběr lékařských dat, spotřební elektroniku (dálkové ovládáni TV apod.) a automatizaci domácnosti. WirelessHART je rozšířením HART protokolu (digitální průmyslový automatizační protokol) a je speciálně navržen pro sledování technologických procesů a kontrolu. 6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) je IETF standard pro mapování IPv6 na standard 802.15.4. [1]

2.2. Technologie IEEE 802.15.4

Standard IEEE802.15.4 je podskupinou standardu 802.15: Wireless Personal Area Network pro nízké přenosové rychlosti s důrazem na nízkou spotřebu. Specifikuje fyzickou vrstvu a MAC podvrstvu (Řízení přístupu k médiu) pro low-rate bezdrátové osobní sítě (WPAN, Wireless Personal Area Network). Je to základ pro navazující specifikace jako ZigBee, WirelessHART a MiWi, které nabízejí kompletní síťová řešení vyšších protokolových vrstev. Alternativně může být použit 6LoWPAN standard a běžné internetové protokoly k vybudování bezdrátové IP sítě.

Standard IEEE802.15.4 klade důraz na velmi nízkou spotřebu blízkých zařízení bez nutnosti podpůrné infrastruktury. Základem je komunikační dosah 10 metrů s rychlostí 250kbit/s. S ohledem na spotřebu byly přidány rychlosti 20, 40 (specifikace 2003) a 100kbit/s (specifikace 2006). Dosah může být v závislosti na vysílacím výkonu a nastavení v rozmezí 1 až 100 metrů.

2.2.1.Fyzická vrstva

Fyzická vrstva zajišťuje mnoho funkcí, jako aktivaci radiového rozhraní, detekci energie v aktivním kanále, indikaci kvality linky, vyhodnocení obsazení kanálu pro CSMA-CA přístupovou metodu, výběr frekvence kanálu, vysílání a příjem dat. Udržuje také informace o aktivní WPAN.

K dispozici je několik frekvenčních pásem.

- 868,0~868,6MHz - Pouze Evropa, 1 komunikační kanál
- 902~928MHz – USA a Austrálie, 10 kanálů
- 2400~2483,5MHz – 16 kanálů s rychlostí 250kbit/s

K dispozici je tak celkem 27 kanálů, číslovaných postupně od 0 do 26 (0 pro 868Mhz, 1 až 10 pro 915MHz a 11 až 26 pro 2,4GHz).

V roce 2006 byly k dvěma existujícím fyzickým vrstvám přidány další dvě, které díky jinému kódování navyšují přenosovou rychlost u pásma 868MHz a 915MHz. K dispozici jsou tak celkem 4 verze PHY vrstvy. [2]

- 868/915MHz DSSS s BPSK modulací (20/40kbit/s)
- 868/915MHz DSSS s O-QPSK modulací (100/250kbit/s)
- 868/915MHz PSSS s BPSK a ASK modulací (250/250kbit/s)
- 2450MHz DSSS s O-QPSK modulací (250kbit/s)

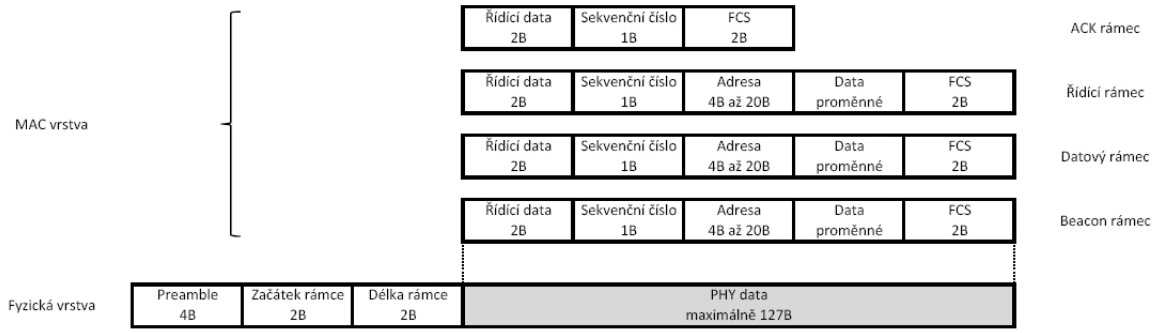
Tab.1. Modulační a přenosové parametry

PHY [MHz]	Frekvenční pásmo [MHz]	Modulační parametry		Přenosové parametry		
		Chip rate [kchip/s]	Modulace	Bitová rychlost [kb/s]	Symbolová rychlost [ksymbol/s]	Symbol
868/915	868,0-868,6	300	BPSK	20	20	Binární
	902-928	600	BPSK	40	40	Binární
868/915	868,0-868,6	400	ASK	250	12,5	20-bitový PSSS
	902-928	1600	ASK	250	50	5-bitový PSSS
868/915	868,0-868,6	400	O-QPSK	100	25	16-stavový Ortogonální
	902-928	1000	O-QPSK	250	62,5	
2450	2400-2483,5	2000	O-QPSK	250	62,5	

2.2.2.MAC podvrstva

MAC podvrstva poskytuje rozhraní mezi fyzickou vrstvou a navazujícími technologiemi (ZigBee, WirelessHART). Umožňuje přenos rámců po fyzickém kanále, asociaci k osobní síti (PAN), synchronizaci na přijímaný beacon rámec. Řídí přístup ke kanálu (přístupová metoda CSMA-CA) a případné generování rámce beacon. Přináší také podporu pro zabezpečení. MAC podvrstva poskytuje spolehlivou linku na MAC úrovni mezi zařízeními.

Na MAC vrstvě jsou definovány čtyři typy přenášených rámců. Jsou to datové rámce, potvrzovací (ACK) rámce, Beacon rámce a MAC příkazové rámce. Jednotlivé typy se liší velikostí záhlaví i typem přenášených informací. Rozdílné je samozřejmě také jejich určení. Beacon rámec je vysílán řídícím prvkem sítě (síťový koordinátor) a obsahuje informace o dané síti. Datový rámec je určen pro přenos dat vyšších vrstev. MAC příkazový rámec přenáší řídící informace nutné pro chod sítě. ACK rámec slouží k potvrzování úspěšného doručení rámce a jako jediný neobsahuje žádná data MAC vrstvy, pouze pořadové číslo a zabezpečení. [2]



Obr.1. Struktura MAC rámců

Komunikační kanál mezi sebou sdílí mnoho uzlů, je proto přístup k němu účinně řídit. Ve specifikaci 802.15.4 je zvolena přístupová metoda CSMA-CA (Carrier Sense Multiple Access with Collision Avoidance).

CSMA-CA je poddruhem přístupových metod s vícenásobným přístupem a naslouchání nosné (CSMA). Před samotným vysíláním paketu stanice určitý čas sleduje rádiová kanál a zjišťuje tak zda je volný. Pokud nikdo jiný nevysílá, může zahájit přenos. Jsou využity dvě modifikace této přístupové metody podle toho, zda se jedná o síť s beacon rámcem (slotted CSMA-CA) či bez beacon rámce (unslotted CSMA-CA). [2]

Pro PAN sít bez beacon rámce (unslotted CSMA-CA) čeká uzel před vysláním dat náhodnou dobu. Pokud je kanál volný, kontroluje stav kanálu ještě náhodný čas (backoff) a pak vyšle svá data. Je-li kanál během této doby obsazen jiným uzlem, celá procedura se vrací na začátek a uzel čeká opět náhodnou dobu, než se pokusí opět vysílat. Toto ale neplatí pro ACK rámce, které jsou vysílány bez této procedury.

U PAN sítě s vysíláním beacon rámce (slotted CSMA-CA) je použito časových slotů, ve kterých mohou uzly vysílat. Síť je synchronizována beacon rámcem, po němž následují jednotlivé sloty. Pokud je kanál volný, stanice zahájí vysílání v nejbližším dostupném slotu. Při obsazenosti kanálu čeká náhodný počet slotů před dalším pokusem o vysílání. ACK a Beacon rámce jsou vysílány bez této procedury.

2.3. ZigBee

Standard ZigBee je vyvíjen organizací ZigBee Alliance, která sdružuje mnoho nadnárodních firem a korporací, jako například Texas Instruments, Analog Devices, Cisco Systems, Freescale Semiconductors, Motorola a mnoho dalších. ZigBee je tedy jednoduchý bezdrátový komunikační standard umožňující komunikaci mnoha zařízení. Poskytuje spolehlivé, nízkopříkonové a nízkonákladové bezdrátové spojení pro kontrolní a řídící zařízení s dosahem signálu v desítkách metrů. Díky nízké

spotřebě a nenáročné implementaci nachází uplatnění v oblasti řízení budov, spotřební elektroniky a průmyslu. [3]

ZigBee standard si neklade za cíl konkurovat již zavedeným komunikačním standardům typu Bluetooth, ale spíše jako jejich doplněk který má rozšířit oblast nasazení. Není zaměřen na maximální přenosovou rychlost, ale na velmi nízkou spotřebu, jednoduchost, spolehlivost a nízkou cenu. Standard ZigBee je založen na využití fyzické a linkové vrstvy definované mezinárodním standardem IEEE 802.15.4. Tímto standardem jsou daná rádiová pásma, přenosové rychlosti i techniky přístupu k médiu. Vlastní standard IEEE 802.15.4 definuje komplexní komunikační protokol, který je založen na přenosu datových rámců.

2.3.1. Síťová zařízení

Standard ZigBee dělí zařízení na zařízení FFD (Full Functional Device) a RFD (Reduced Functionality Device). FFD zařízení implementují kompletní protokolový rámec a zajišťují veškeré služby, které standard ZigBee stanovuje. RFD zařízení implementují pouze nezbytné protokolové knihovny z důvodu maximálního omezení hardwarové náročnosti. [4, 5]

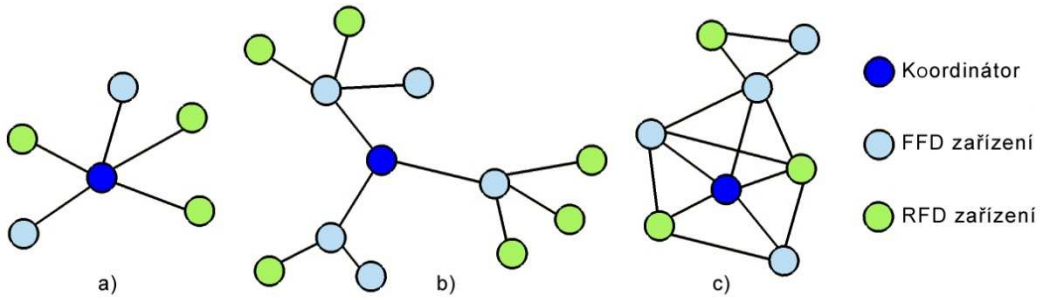
Pro adresaci jednotlivých zařízení se využívají binární adresovací kódy, které mohou být buď dlouhé (64 bitů), či zkrácené (16 bitů). Lokální adresa zkráceného adresovacího kódu umožňuje v jedné síti adresovat maximálně 65 535 zařízení. Každá sestavená síť je dále identifikována 16bitovým identifikátorem PAN ID, který slouží pro rozlišení překrývajících se sítí v případě, že v jednom prostoru dochází k vytvoření a sestavení více sítí standardu IEEE 802.15.4.

Každou síť s unikátním PAN ID zakládá a spravuje koordinátor (centrální stanice). Koordinátor je zařízení FFD, které vytváří Beacon rámec a řídí chod sítě. Funkci koordinátora může v případě nutnosti zastoupit jakákoliv jiná FFD stanice. Ostatní zařízení sítě se podle funkčnosti dělí na plně funkční zařízení (FFD), která mohou zastávat funkci koordinátora nebo směrovače i koncového zařízení, a na redukovaná zařízení RFD. Tato zařízení mohou pracovat pouze jako koncová. Mohou komunikovat pouze s koordinátorem sítě a jsou omezeny na hvězdicové uspořádání topologie (koncové větve).

2.3.2. Topologie

ZigBee používá tři různé síťové topologie. Základní topologií je Hvězdicová síť (star topology) s centrálním řídícím uzlem (Koordinátorem). Nevýhodou je, že všechny ostatní stanice působí v roli koncového zařízení a musí být v přímém dosahu řídícího uzlu. Druhým typem je Stromová struktura (tree topology), u které

nemusí být všechny stanice v dosahu koordinátora, ale pro komunikaci mohou využívat jiné koncové FFD zařízení ve funkci směrovače jako prostředníka. Díky tomu je možno zvětšit vzdálenost mezi koordinátorem a koncovým zařízením. Poslední topologií je topologie typu síť (mesh topology) u které je kombinováno vlastností topologie strom i hvězda. Vzniklá hybridní topologie umožňuje sestavit síť libovolným způsobem. [5, 6]



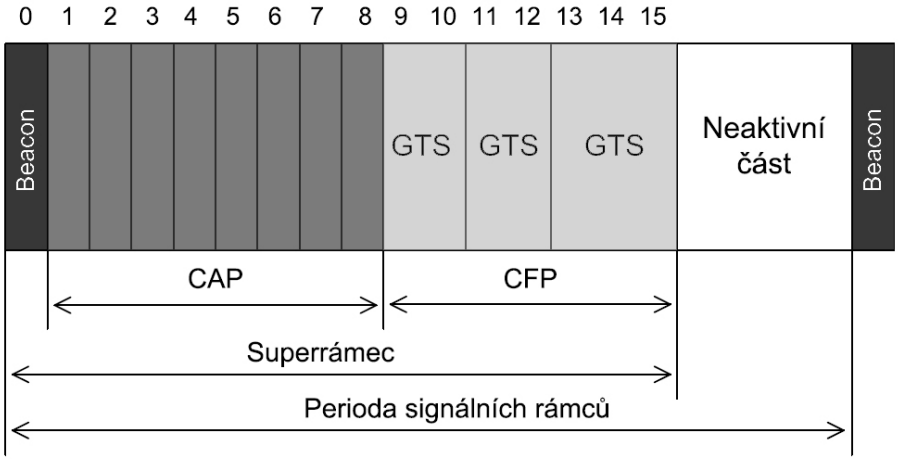
Obr.2. Topologie sítě ZigBee typu a) hvězda, b) strom, c) síť (mesh)

ZigBee síť umožňuje práci ve dvou režimech, v režimu řízeném rámcem Beacon (Beacon-enabled) a bez synchronizace na Beacon rámec (non-Beacon). V Beacon-enable ZigBee sítích je synchronizační autoritou koordinátor sítě, který v daných okamžicích vysílá synchronizační sekvence (beacon). Beacon rámec přijímají ostatní zařízení a synchronizují se podle nich s koordinátorem. Tento postup umožňuje koncová zařízení na dlouhou, předem definovanou dobu „uspat“, a značně tak snížit jejich spotřebu. V non-beacon síti koordinátor sice také vysílá beacon rámec, ale koncovým stanicím slouží pouze k jeho identifikaci. Koncová zařízení komunikují s koordinátorem pomocí požadavků na vysílání dat a potvrzovacích rámců.

2.3.3.Superrámec

ZigBee síť má možnost použít takzvaný superrámec, jehož struktura je definována koordinátorem sítě. Superrámec vysílaný koordinátorem je ohraničený beacon rámci a je rozdělen na 16 stejných slotů. V prvním je vysílán beacon rámec. Je určen pro synchronizaci, identifikaci sítě PAN a k popisu struktury superrámce. Ve zbývajícím čase může kterékoli zařízení na základě přístupové metody CSMA-CA komunikovat. Každý superrámec může mít aktivní a neaktivní část. Během aktivní části koordinátor komunikuje s příslušnou PAN a v neaktivní části může přejít do režimu spánku (low-power mode).

Aktivní část lze dále rozdělit na oblast CAP (Contention Access Period) a CFP (Contention Free Period). V době trvání CAP probíhá komunikace na základě CSMA-CA přístupové metody. Část CFP je složena z několika GTS (Guaranteed Time Slot), které mohou být vyhrazeny pro pomalá (low-latency) a prioritní zařízení. Koordinátor může vyčlenit až sedm GTS, přičemž jeden GTS může zabrat i více slotů, musí však zůstat dostatek slotů pro CAP. Po vyslání superrámce může koordinátor přejít do režimu spánku (low-power mode). Interval vysílání superrámce může být od 15ms do 252s.



Obr.3. Struktura Superrámce

2.3.4.Síťová a Aplikační vrstva

Síťová vrstva standardu ZigBee (NWK) má za úkol zabezpečení rámců a jejich směrování k cílovým uzlům. Hledá přímé sousední uzly a udržuje si o nich informace. Síťová vrstva koordinátoru zajišťuje přidělování adres novým zařízením. Síťová vrstva je pouze u FFD, u RFD již není implementována [6].

Aplikační vrstva je složena z pomocné aplikační vrstvy (APS - application support sub-layer), z objektů ZigBee (ZDO - ZigBee device object) a z aplikačních objektů definovaných výrobci [6]. Pomocná aplikační vrstva (APS) má za úkol udržovat vazební tabulky (binding table), které umožňují propojit dvě zařízení na základě jejich služeb a potřeb. Také přeposílá zprávy mezi vzájemně vázanými zařízeními. Objekt ZigBee (ZBO) určuje roli zařízení v síti, tedy jedná li se o ZigBee koordinátor nebo koncové zařízení. Dále řídí žádosti o spojení a zřizuje zabezpečené spojení mezi zařízeními sítě. Poskytuje také nástroje k hledání zařízení v síti a zjišťuje jimi poskytované služby.

3. Přehled problematiky lokalizace uzlu

Problematika lokalizace uzlu v bezdrátové senzorové síti (WSN) zahrnuje metody měření vzdáleností mezi uzly a metody výpočtu vzájemné polohy uzlů ze změřených vzdáleností. Podle použitého lokalizačního algoritmu je možno určit buďto relativní pozici uzlu v síti nebo i absolutní geografickou polohu uzlu.

3.1. Měření vzdálenosti v bezdrátových senzorových sítích

Problematiku měření vzdáleností v bezdrátových senzorových sítích můžeme podle odlišného fyzického přístupu rozdělit na fyzikální metody (RB, Range/angle-Based) využívající měření vzdáleností a úhlů, a na metody pracující pouze s informacemi sítě (RF, Range-Free). [7]

3.1.1. Fyzikální metody

Fyzikální metody (Range/angle-Based) pro svou činnost využívají měření vzdáleností nebo úhlů mezi jednotlivými uzly. Často tyto metody vyžadují odpovídající podporu v hardware jednotlivých uzlů. Existuje několik rozdílných RB metod, které využívají měření parametrů přímo souvisejících s fyzickou polohou uzlu.

Síla signálu (Signal Strenght)

Pro určení vzdálenosti mezi uzly je využito síly přijímaného signálu (RSSI, Received Signal Strength Indication), která teoreticky přímo závisí na vzdálenosti uzlů. Hodnota parametru RSSI pak reflektuje skutečnou vzdálenost od vysílače. Předpokladem samozřejmě je znalost vysílacího výkonu a parametrů antény. Teoreticky je pak vzdálenost uzlů možno jednoduše vypočíst. Měření RSSI je relativně levné a snadno realizovatelné. V praxi je ale vztah mezi sílou signálu a vzdálenosti ovlivněn několika faktory, jako měnící se podmínky prostředí, elektrickými komponenty, baterií a také samotnou anténou.

Úhel příjmu (AOA, Angle-of-Arrival)

Metoda založená na vyhodnocení směru příchozího signálu. Často se využívá společně s metodami pro měření vzdálenosti uzlů. Existuje několik způsobů realizace této metody. Jednou z nich je využití směrových antén, kdy uzel na základě znalosti jejich natočení určí příchozí směr signálu. Metoda AOA díky přídavným zařízením zvyšuje cenu zařízení a klade větší nároky na rozmístění uzlů.

Čas příjmu (TOA, Time of Arrival)

Pro určení vzdálenosti uzlů se využívá čas mezi vysláním signálu zdrojem a jeho přijetím příjemcem. Doba šíření signálu má lineární závislost na vzdálenosti uzlů. Ve vyslaném paketu je na straně zdroje přidána aktuální časová známka, a příjemce ji pak porovná s časem přijetí paketu. Po odečtení časů zpracování pak získáme potřebnou dobu šíření signálu. Problémem ale je přesná synchronizace časů v síti, která vzhledem k velké rychlosti šíření rádiového signálu často přesahuje možnosti jednotlivých uzlů. Často používanou modifikací je pak dvoucestné TOA měření, kdy je paket příjemcem poslán zpět odesilateli. Problémem metody TOA je hlavně krátký čas šíření signálu a vysoké nároky na přesnost.

Rádiová interferometrie (RI, Radio Interferometry)

Technika zvaná rádiová interferometrie využívá pro určení polohy dvou vysílajících uzlů, které vysílají na známých navzájem blízkých frekvencích. Vznikne signál s frekvenční obálkou s frekvencí rozdílu vysílaných kmitočtů. Další dva uzly přijímají tento výsledný signál a vzájemným porovnáním přijímaných fázi získají fázový posun, který reflektuje vzájemnou pozici těchto čtyř uzlů. Metoda RI poskytuje vysokou přesnost měření bez nutnosti přídavného hardware. Má však i své problémy, především synchronizace mezi sousedními uzly. Také vzdálenost je limitována kmitočtem frekvenční obálky.

3.1.2.Metody pracující s informacemi sítě

Range-free metody nedosahují takové přesnosti měření jako metody fyzikální, ale zato nevyžadují žádný přídavný hardware ani implementaci speciálních metod pro měření. Pro měření využívají dostupné informace sítě.

Konektivita (Connectivity)

Konektivita je informace, která vyjadřuje, jestli je uzel v komunikačním dosahu jiného uzlu. Podle takto zjištěných sousedních uzlů je možno určit polohu zařízení vzhledem k těmto uzlům. Metoda je vhodná především pro rovnoměrně rozložené senzorové sítě. Vzhledem k tomu, že jednotlivé uzly senzorových sítí si udržují informace o sousedních uzlech, je využití této metody jednoduché a bez nároků na přídavný hardware.

Počet skoků (Hop-Counting)

Další nefyzikální metoda je měření počtu skoků. Jde o měření počtu přeskoků, tedy počet mezilehlých uzlů, od počátečního uzlu. Jeden z uzlů (počáteční uzel) zahájí celý proces vysláním zprávy s hodnotou hop-count rovnou nule. Každý další uzel navýší počet hop-count a zprávu předá dále. Každý uzel sítě tak určí svou vzdálenost od počátečního uzlu jako počet přeskoků k němu. Každý uzel si udržuje tuto informaci, a při příchodu další zprávy nejprve porovná, zda je hodnota hop-count menší než ta dříve zjištěná. Pokud se skutečně jedná o kratší cestu, je hodnota aktualizována a zpráva předána dále. V opačném případě je zpráva zahozena.

3.2. Lokalizační algoritmy

Na základě polohy uzlu vzhledem k sousedním uzlům je nutné také určit jeho fyzickou pozici v celé síti, případně i vzhledem k okolnímu světu. K řešení této úlohy je možno využít vhodného lokalizačního algoritmu. Lokalizační algoritmy můžeme rozdělit na algoritmy s kotevními uzly (ABL, Anchor-Based Localization) a bez kotevních uzlů (AFL, Anchor-Free Localization).

ABL algoritmy využívají kotevních uzlů, u nich známe jejich pozici v síti. Pozice jakéhokoliv uzlu v síti lze pak určit na základě vzdálenosti od takovýchto kotevních uzlů. Příkladem takovýchto algoritmů jsou Centroid, RADAR, APIT, DV-hop, P-Grid. [7]

Oproti tomu AFL algoritmy nepoužívají žádných známých kotevních bodů, u nichž je definována poloha. Uzel zjistí svou relativní polohu k okolním uzlům bez návaznosti na fyzickou polohu.

3.2.1. Lokalizace s kotevními uzly (Anchor-Based Localization)

V senzorové síti jsou definovány kotevní (Anchor) uzly, u kterých je známá jejich poloha. Může jít o zařízení vybavené GPS přijímačem nebo statický uzel se zadanou polohou. Pro ostatní uzly pak tyto kotevní body slouží jako referenční. Pozice jakéhokoliv uzlu v síti lze pak určit na základě vzdálenosti od tří takovýchto kotevních uzlů. V dvojdimenzionálním prostoru stačí vzdálenosti od tří kotevních uzlů neležících na přímce, pro trojdimenzionální prostor pak vzdálenosti od čtyř kotevních uzlů neležících v rovině.

Lokalizace uzlu probíhá v několika krocích. Nejprve uzel zjistí svou vzdálenost k dostatečnému počtu kotevních uzlů některou z metod měření vzdáleností v WSN (popsáno výše). Následně uzel spočítá svou polohu ze známých vzdáleností a pozic kotevních uzlů.

Výhodou lokalizace s kotevními uzly je možnost získat absolutní geografickou polohu jakéhokoliv uzlu sítě, nevýhodou pak složitější budování sítě.

3.2.2. Lokalizace bez kotevních uzlů (Anchor-Free Localization)

Lokalizace bez použití kotevních bodů (AFL, Anchor-Free Localization) na rozdíl od ABL pro svou činnost nemá definované žádné uzly s předem známou polohou. Poloha uzlu je tak určována jako relativní k jiným uzlům sítě.

Samotná lokalizace uzlu probíhá ve dvou krocích. Nejprve uzel zjistí svou vzdálenost od okolních uzlů některou z měřících metod popsaných výše. Následně, na základě těchto vzdáleností, je spočítána relativní poloha k těmto uzlům.

Hlavní výhodou AFL algoritmu je jednoduchá stavba sítě, kdy není potřeba definování polohy žádného z uzlů. Nevýhodou je jen relativní určení polohy.

4. Zvolené řešení

4.1. Simulační prostředí NS2

Pro simulaci lokalizace uzlu v bezdrátové senzorové síti je zvoleno simulační prostředí Network Simulator 2 (NS2). Jedná se o objektově orientovaný nástroj určený pro simulaci sítě obsahující celou řadu síťových technologií, komunikačních a směrovacích protokolů. NS2 vychází z jazyka C++, simulační skripty jsou ale psány jako TCL skripty. Nástroj je šířen pod licencí GNU GPL [8] a je volně ke stažení z oficiálních stránek [9] včetně bohaté dokumentace.

Prostředí NS2 umožňuje budování rozsáhlých modelů sítě, vytváření spojení a generování provozu, modelování poruch, monitorování provozu a záznam událostí sítě. Výhodou tohoto simulačního nástroje je mimo jiné dostupnost zdrojových kódů. Díky tomu lze do simulačního prostředí implementovat vlastní protokol a následně simulovat jeho chování pro rozmanité podmínky.

Znázornění výsledků na základě simulačního skriptu TCL umožňuje nástroj NAM (Network Animator), který slouží k vizualizaci dokončené simulace. Jedná se o přehrávači podobnou aplikaci, která dovoluje s volitelnou rychlostí znázornit proběhnutou simulaci. Lze v něm zobrazit topologii sítě, provoz na jednotlivých linkách i jiné důležité události v simulaci, jako např. připojení uzlu nebo poruchy na lince. Pro zobrazení grafických závislostí lze využít nástroj xgraph, který slouží pro vykreslení zaznamenaných hodnot do grafu.

4.2. Bezdrátová senzorová síť

Pro lokalizaci uzlu je zvoleno prostředí bezdrátové senzorové sítě ZigBee, což je síť postavená na standardu IEEE 802.15.4. Standard ZigBee je určen pro bezdrátové osobní sítě (WPAN) s důrazem na velmi nízkou spotřebu energie. To umožňuje uzlům sítě dlouhodobou práci při napájení z baterií. Uzly sítě disponují rádiovým rozhraním s dosahem řádově desítky metrů. Maximální přenosové rychlosti se pohybují od 20kbit/s do 250kbit/s podle zvoleného kanálu a konfigurace. V prostředí simulátoru NS2 je standard IEEE 802.15.4 a ZigBee implementován.

4.2.1. Začlenění standardu IEEE 802.15.4 v NS2

Zdrojové kódy implementující technologii 802.15.4 do prostředí NS2 jsou umístěny v adresářové struktuře NS2 ve složce ns\ns-2.xx\wpan\. Nalezneme zde zdrojové soubory definující fungování fyzické a MAC vrstvy, formát posílaných rámců

a další specifikace. Z hlediska fungování sítě jsou nejzajímavější soubory p802_15_4phy.cc, p802_15_4csmaca.cc, p802_15_4mac.cc a p802_15_4pkt.h. Každý z těchto souborů (společně se stejnojmenným hlavičkovým souborem) specifikuje funkce příslušné vrstvy.

- **p802_15_4phy** realizuje fyzickou vrstvu definovanou standardem 802.15.4. Definuje metody pro probuzení uzlu při příjmu a příjem paketu z rádiového kanálu. Dále definuje metody spolupodílející se na vyslání dat, především pak v návaznosti na přístupovou metodu CSMA-CA.
- **p802_15_4csmaca** realizuje metody nezbytné pro použitou techniku přístupu ke kanálu CSMA-CA. Podílí se na vysílání dat, zajišťuje časování (náhodná zpoždění) pro detekci provozu na kanále a řídí přístup k němu.
- **p802_15_4mac** definuje funkce na úrovni MAC vrstvy. Obsahuje metody pro příjem rámce, jeho rozpoznání (datový, řídící, beacon, ACK). Řídí vysílání paketu, dává požadavky na přenos rámce a spouští metody CSMA-CA. Vykonává příkazy přijaté řídícím paketem. Synchronizuje uzel na přijímaný beacon rámec, zabezpečuje přenášená data proti chybám.
- **p802_15_4pkt.h** definuje struktury použitých rámců. Obsahuje definice datového, příkazového, potvrzovacího (ACK) i beacon rámce.

4.2.2. Zpracování vysílaného rámce v NS2

- Z vyšších protokolových vrstev je předán paket s požadavkem na vyslání voláním Mac802_15_4::recv(Packet *p, Handler *h).
- recv() pak na základě směru zavolá Mac802_15_4::MCPS_DATA_request().
- Pro vysílání paketu metoda MCPS_DATA_request() zavolá Mac802_15_4::csmacaBegin(pktType).
- csmacaBegin(pktType) přes metodu Mac802_15_4::csmacaResume() spustí metodu CsmaCA802_15_4::start(). Ta spočítá backoff time a spustí časovač macBackoffTimer.
- Po vypršení časovače je volána CsmaCA802_15_4::backoffHandler(). backoffHandler() voláním Phy802_15_4::PLME_SET_TRX_STATE_request(state) zapne přijímač a spustí sledování stavu kanálu Phy802_15_4::PLME_CCA_request().
- Po ověření stavu kanálu Phy802_15_4::CCAReportHandler() předá jeho zjištěný stav MAC vrstvě voláním Mac802_15_4::PLME_CCA_confirm(status).

- Pokud je kanál volný, ale ještě neproběhlo dostatečné množství měření, dekrementuje počet měření a spustí další. Po dostatečném počtu měření je volána Mac802_15_4::csmacaCallBack(status). Pokud kanál není volný, je volána metoda CsmaCA802_15_4::start() a čeká se na nový slot.
- csmacaCallBack() vrací řízení Mac802_15_4::MCPS_DATA_request(), která zapne vysílač voláním Phy802_15_4::PLME_SET_TRX_STATE_request().
- Mac802_15_4::PLME_SET_TRX_STATE_confirm() předá data metodě Mac802_15_4::txBcnCmdDataHandler(), která pomoci Mac802_15_4::sendDown() předá data fyzické vrstvě a metodě Phy802_15_4::recv(Packet *p, Handler *h).
- Na fyzické vrstvě metoda recv() volá Phy802_15_4::PD_DATA_request(). Metoda PD_DATA_request() použije metodu WirelessPhy::sendDown() čímž dojde k vyslání dat do Channel::recv().

4.2.3. Zpracování přijímaného rámce v NS2

- Channel::recv() předá kopii paketu každému uzlu přes WirelessChannel::sendUp() metodě Phy802_15_4::recv(Packet *p, Handler *h).
- Phy802_15_4::recv() použije metodu WirelessPhy::sendUp() a přijetí paketu je MAC vrstvě oznámeno přes volání Phy802_15_4::PD_DATA_indication(). Metoda Phy802_15_4::recvOverHandler() zahodí paket pokud není určen pro tento uzel. PD_DATA_indication() poté volá metodu Mac802_15_4::recv(Packet *p, Handler *h).
- Metoda recv() MAC vrstvy v případě, že došlo ke kolizi, paket zahodí. Pokud byl paket přijat správně, předá jej (podle typu rámce) příslušné metodě Mac802_15_4::recvData(). Ta voláním Mac802_15_4::MCPA_DATA_indication() předá data vyšším vrstvám.

4.3. Lokalizační algoritmus

Pro volbu optimálního lokalizačního algoritmu je potřeba ujasnit si požadovanou přesnost určení polohy a možnosti měření vzdáleností v síti. Také je třeba vzít v potaz schopnosti použitého hardware.

4.3.1. Volba lokalizačního algoritmu

Jako měřící metoda je vybrána metoda měření počtu skoků (Hop-count). Jde o metodu pracující s informacemi sítě (Range-free), nepotřebuje tedy pro svou činnost žádný přídavný hardware. Poskytují ale menší přesnost měření. Pro určení polohy uzlu v síti je zvolena metoda bez kotevních bodů (Anchor-Free Localozation).

Hlavní výhodou AFL algoritmu je jednoduchá stavba sítě, kdy není potřeba definování polohy žádného z uzlů. Nevýhodou je jen relativní určení polohy. Oproti tomu lokalizace s kotevními uzly dovoluje získat absolutní geografickou polohu jakéhokoliv uzlu sítě. Problematickým místem jsou kotevní body, které vyžadují přesné nastavení polohy nebo použití GPS modulů, což je ale omezeno energetickou náročností a stíněním v uzavřených budovách nebo zastavěné oblasti. Nevýhodou ABL je také složitější budování sítě, kdy je třeba dbát na vhodné rozmístění kotevních uzlů, přesnosti nastavení jejich polohy či zvýšené spotřebě díky GPS modulu. Lokalizace je také náchylná na případné poruchy kotevního uzlů.

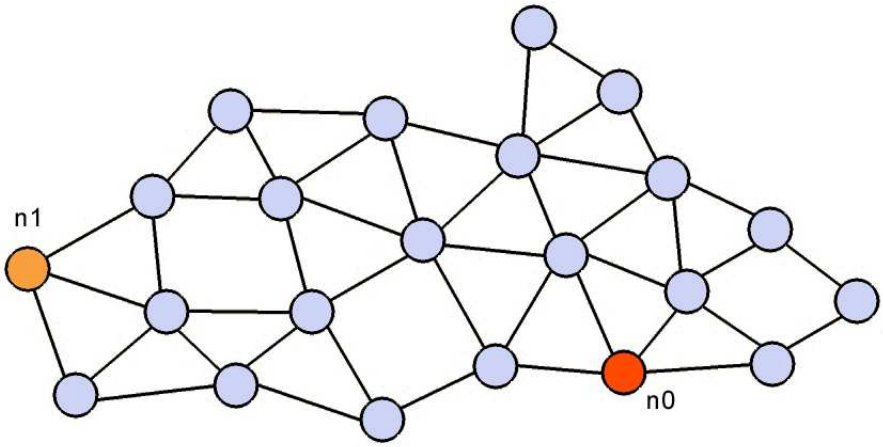
4.3.2. Popis zvoleného AFL algoritmu

Jde o algoritmus, který pro měření vzdáleností využívá měření hop-count. Pracuje ve dvou fázích. V první fázi je na základě daných pravidel určena pětice referenčních uzlů (4 po obvodu sítě a jeden středový), od kterých následně každý z uzlů změří své vzdálenosti. Z těchto vzdáleností jsou výpočtem určeny vektory vyjadřující pozici uzlu vzhledem ke středovému. V druhé fázi všechny uzly sítě provádí optimalizaci určené polohy vzhledem k poloze svých sousedů, což redukuje výslednou chybu určení polohy. Výsledkem jsou údaje o polohách uzlu odrážející fyzické rozložení uzlů sítě. [10]

4.3.3. Určení polohy jednotlivých uzlů

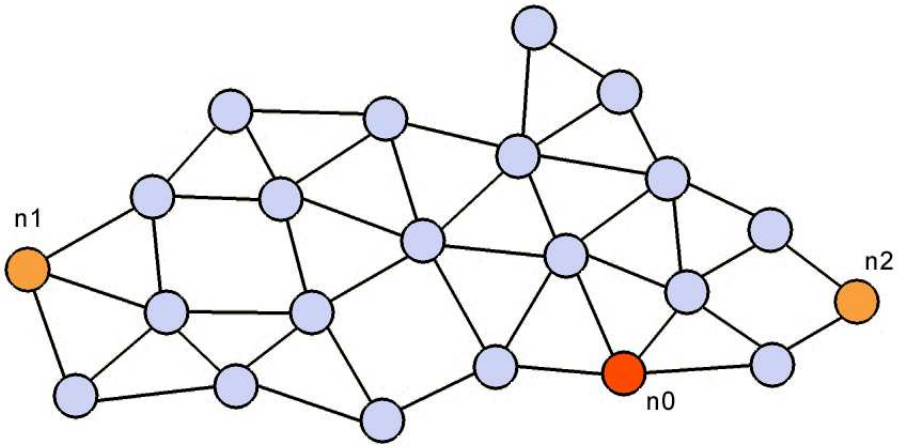
Mějme obecnou senzorovou síť, u níž neznáme polohu žádného z uzlů. Každý uzel může navázat komunikaci se svými sousedními uzly. Vzdálenost uzlů je měřena v počtu mezilehlých uzlů sítě. Jde tedy o počet skoků (hop-count). Vzdálenost dvou uzlů n_i a n_j budeme pro zjednodušení označovat $h_{i,j}$.

Celý proces je zahájen některým z uzlů, označme ho jako uzel *n0*. Ten zahájí prohledávání sítě s cílem nalézt nevzdálenější uzel (s největším počtem mezilehlých uzlů). Takto nalezený uzel je prvním kotevním uzlem, označme jej *n1*. Situace je vyobrazena na obrázku 4.



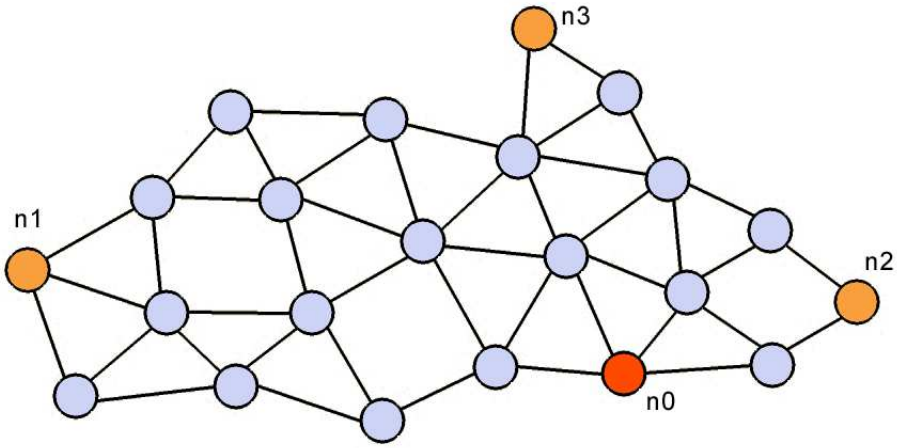
Obr.4. Krok 1 : Nalezení kotevního uzlu n1

V druhém kroku uzel *n1* zahájí hledání nejvzdálenějšího uzlu stejným způsobem jako v kroku jedna. Nalezený uzel označíme jako *n2* a jedná se o druhý kotevní uzel sítě (Obrázek 5).



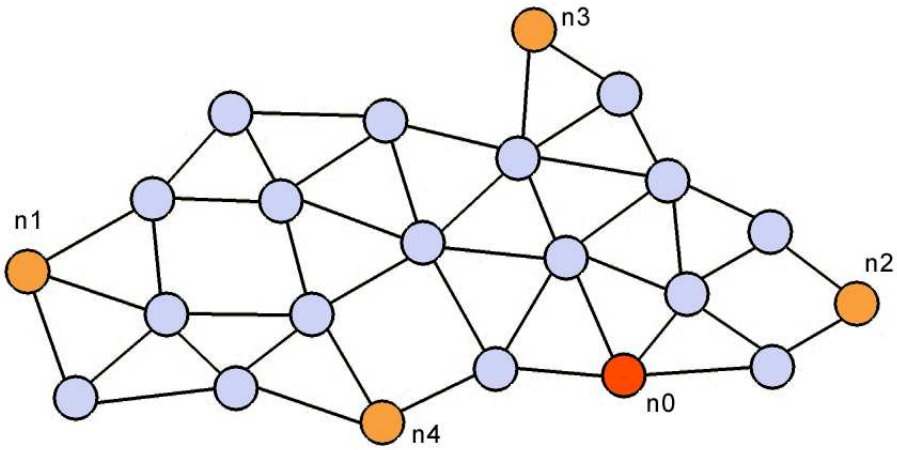
Obr.5. Krok 2 : Nalezení kotevního uzlu n2

Třetím krokem je nalezení kotevního uzlu *n3*. Jde o uzel s pokud možno stejnou vzdáleností od obou kotevních uzlů *n1* a *n2*. S této množiny vhodných uzlů je pak vybrán uzel s co největším součtem obou těchto vzdáleností. Hledání uzlu *n3* se proto skládá z výběru uzlů s minimální $|h_{1,3} - h_{2,3}|$ a následně volba uzlu s maximální $(h_{1,3} + h_{2,3})$. Situace je znázorněna na obrázku 6.



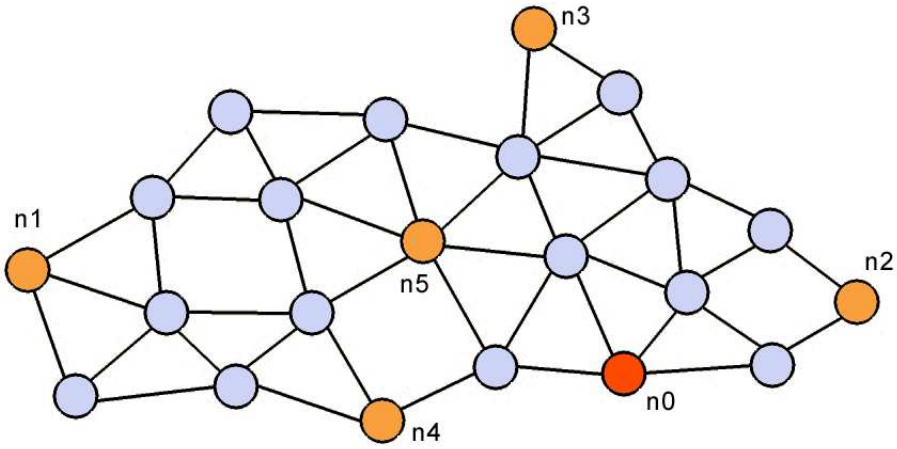
Obr.6. Krok 3 : Výběr kotevního uzlu n3.

Ve čtvrtém kroku je hledán uzel *n4*. Ten je vybrán jako uzel s pokud možno stejnou vzdáleností od obou kotevních uzlů *n1* a *n2*. S této množiny vhodných uzlů je pak vybrán uzel s co největší vzdáleností od *n3*. Hledáme tedy uzly s minimální $|h_{1,3} - h_{2,3}|$, ze kterých pak zvolíme uzel s maximální vzdáleností $h_{3,4}$ (Obrázek 7).



Obr.7. Krok 4 : Nalezení kotevního uzlu n4

Už zbývá jen určení posledního kotevního uzlu *n5*, který je opět volen jako uzel s pokud možno stejnou vzdáleností od obou kotevních uzlů *n1* a *n2*. S této množiny vhodných uzlů je vybrán jeden s pokud možno stejnou vzdáleností od kotevních uzlů *n3* a *n4*. Hledáme tedy uzly s minimální $|h_{1,5} - h_{2,5}|$, ze kterých pak zvolíme uzel s minimální $|h_{3,5} - h_{4,5}|$ (Obrázek 8).



Obr.8. Krok 5 : Stanovení středového kotevního uzlu n5

Když jsou určeny kotevní uzly, určíme polohu všech ostatních uzlů vzhledem k těmto uzlům. Pro každý uzel n_i změříme vzdálenosti $h_{i,1}$, $h_{i,2}$, $h_{i,3}$, $h_{i,4}$ a $h_{i,5}$. Výpočtem stanovíme polární souřadnice (ρ_i, θ_i) uzlu podle vztahu (4.01, 4.02) [10]. Jako jednotková vzdálenost R je použit maximální rádiový dosah.

$$\rho_i = R \times h_{i,5} \tag{4.01}$$

$$\theta_i = \tan^{-1}\left(\frac{h_{i,1}-h_{i,2}}{h_{i,3}-h_{i,4}}\right) \tag{4.02}$$

Takto spočítané souřadnice ale často příliš neodpovídají skutečnému rozmístění uzlů. Ve výpočtu není zohledněn vztah mezi sousedními uzly. Pro snížení výsledné chyby je proto třeba výsledky vhodně optimalizovat.

4.3.4.Optimalizace polohy

Pro optimalizaci je použit algoritmus Mass-Spring (Hmota a Pružina) [10]. V tomto algoritmu spojnice jednotlivých uzlů představují pružiny mezi dvěma hmotami (uzly). Délka pružiny je rovna jejich změřené vzdálenosti. Pokud se vzdálenost uzlů zvětší, pružina se natáhne a působí odpovídající přitažlivou silou. Pokud je vzdálenost uzlů menší než změřená, pružina se stlačí a na uzly působí odpudivou silou. V každém kroku optimalizačního procesu se uzel posune ve směru výsledné na něj působící síly. Pokud je síla nulová, proces optimalizace se pro takovýto uzel zastaví. Optimalizace celé sítě je hotová, až když se všechny síly vyrovnají.

Každý uzel sítě n_i zná svou polohu určenou vektorem $\vec{p}_i$ a změřenou vzdálenost $r_{i,j}$ od svých sousedních uzlů. Své sousední uzly také pravidelně

informuje o aktuální vypočtené poloze $\vec{p}_i$. Na základě těchto poloh je vypočten vektor $\vec{d}_{i,j}$ odhadované vzdálenosti uzlů.

Vektor $\vec{v}_{i,j}$ reprezentuje jednotkový vektor ve směru od $\vec{p}_i$ k $\vec{p}_j$. Síla $\vec{F}_{i,j}$ působící mezi uzly ve směru $\vec{v}_{i,j}$ je dána [10]:

$$\vec{F}_{i,j} = \vec{v}_{i,j}\left(\vec{d}_{i,j} - r_{i,j}\right) \tag{4.03}$$

Výsledná síla působící na uzel n_i je dána součtem jednotlivých sil, tedy [10]:

$$\vec{F}_i = \sum_j \vec{F}_{i,j} \tag{4.04}$$

Energie $E_{i,j}$ uzlu n_i a n_j je určena jako druhá mocnina rozdílu vypočtené a změřené vzdálenosti (4.05)[10]. Celková energie systému určuje součet těchto dílčích energií (4.06)[10].

$$E_i = \sum_j E_{i,j} = \sum_j \left(\vec{d}_{i,j} - r_{i,j}\right)^2 \tag{4.05}$$

$$E = \sum_i E_i \tag{4.06}$$

Energie uzlu E_i se snižuje při pohybu uzlu o velikost změn výsledné síly. Přesná velikost změny polohy uzlu je důležitá ze dvou hledisek. Za prvé je potřeba zjistit zda nová pozice uzlu bude mít menší energii než ta stávající. Za druhé pak zda tento posun nebude mít za následek uvíznutí algoritmu v lokálním minimu.

První podmínka je zaručena přepočítáním energie v nové pozici ještě před samotným posunem a posunutím pouze pokud bude nová energie uzlu nižší. Problémem ale je zaručení druhé podmínky, protože neexistuje snadná metoda pro zjištění, zda posun nepovede k lokálnímu minimu. Podle výsledků [10] je vhodné v jednom kroku měnit pozici uzlu o vzdálenost Δd_i (4.07).

$$\Delta d_i = \frac{\left|\vec{F}_i\right|}{2m_i} \tag{4.07}$$

Kde $\left|\vec{F}_i\right|$ je velikost výsledné síly a m_i je počet sousedních uzlů.

Možnost uváznutí algoritmu v lokálním minimu je ale podle [10] malá. Navíc takovéto lokální minimum způsobí chybné stanovené polohy pouze na omezené oblasti a postihne tak jen část sítě. Optimalizace celé sítě je dokončena když se celková energie rovná nule.

5. Realizace

5.1. Vytvoření nového protokolu

Pro zajištění potřebných funkcí aplikačního agenta a komunikaci mezi uzly je potřeba ve struktuře simulačního prostředí NS2 definovat nový protokol. Tento protokol je nazván ***AFLOCAL***.

Pro zdrojové kódy nového protokolu je vytvořen adresář *aflocal* v adresářové struktuře ns2. V tomto adresáři jsou vytvořeny zdrojové soubory agenta *aflocal.h* a *aflocal.cc*, formát přenášené zprávy je definován v souboru *aflocal_pkt.h*, tabulka uzlů pak v souborech *aflocal_table.h* a *aflocal_table.cc*.

5.1.1. Formát přenášené zprávy

Formát přenášené zprávy je definován v souboru *aflocal_pkt.h*. Je zde definována struktura *hdr_Aflocal*, tvořící vlastní zprávu obsahující několik položek. Pole *ac* přenáší kód požadované akce, *sn* je pořadové číslo zprávy. Pole *src_status* obsahuje status (roli uzlu) autora zprávy. Následující dvě pole *msg_src* a *msg_dst* přenáší informaci o zdrojové a cílové adrese zprávy. Pro předávání hodnot mezi komunikujícími uzly slouží trojice polí *i*, *x* a *y*. Pole *i* je celočíselného typu *int*, pole *x* a *y* pak typu *double*, tedy desetinné číslo. Struktura zprávy je naznačena na obrázku 9.

ac	**sn**	**src_status**	**msg_src**	**msg_dst**	**i**	**x**	**y**
1B	1B	1B	4B	4B	4B	8B	8B

Obr.9. Struktura AFLOCAL zprávy

5.1.2. Třída Aflocal

Agent protokolu je deklarován v souboru *aflocal.h*. Je zde deklarace třídy *Aflocal*, tvořící vlastního agenta. Tento agent běží na každém uzlu sítě a udržuje si pro něj všechny potřebné stavové proměnné a hodnoty. Proměnná *index* obsahuje adresu uzlu, trojice proměnných *node_sn*, *node_ack_sn* a *node_fwd_sn* udržují informace o pořadových číslech datových, potvrzovacích nebo předávaných zpráv. Dvourozměrné pole *kotva[6][2]* o velikosti 6x2 prvků patří k jedněm z nejdůležitějších. Obsahuje informace o adresách a vzdálenostech k jednotlivým kotevním (referenčním) bodům. Hodnota *node_status* určuje roli uzlu v síti (např. jde-li o kotevní bod). Proměnná *timer_status* je pomocnou proměnnou pro časovač, podle

níž se provede konkrétné akce po vypršení časovače. Dále deklaruje proměnné pro vypočtenou polohu uzlu a optimalizaci. Pro uchovávání informací o síti a sousedních uzlech slouží struktura lineárně vázaného seznamu *tabulka_hc* a *sousede* tvořeného třídou *Aflocal_table* (viz dále).

Třída *Aflocal* obsahuje také jednotlivé metody agenta. Pomocí nich agent posílá zprávy jiným uzlům, zpracovává přijaté zprávy, realizuje potvrzování a výpočty. Metody třídy *Aflocal* jsou definovány v souboru *aflocal.cc*.

Popis jednotlivých metod a jejich základní charakteristika:

- **Aflocal()** je konstruktor třídy *Aflocal*. Je spuštěn při vytvoření agenta. Provádí pouze počáteční inicializaci a nastavení.
- **command(int, const char*const*)** je metoda děděná od třídy *Agent*. Zajišťuje rozhraní mezi Tcl skriptem a C++ objektem, umožňuje z prostředí Tcl skriptu definovanými příkazy spouštět metody C++ objektu.
- **recv(Packet*, Handler*)** je druhou metodou děděnou od třídy *Agent*. Metoda je volána kdykoliv uzel přijme zprávu protokolu ***AFLOCAL***. V této metodě je na základě hodnoty *ac* přijaté zprávy rozhodnuto o volání příslušné metody.
- **timer_expire()** je metoda volaná při vypršení časovače. Podle hodnoty stavové proměnné *timer_status* spouští odesílání zpráv, čekání na potvrzení nebo výpočty.
- **fwdData(Packet *p)** je metoda, která slouží k přeposlání zprávy šířené všesměrovou cestou dále k adresátovi. Je využita při opakovaném vysílání, které je šířeno touto cestou.
- **startN0()** provede posloupnost příkazů pro startovní uzel. Je spuštěna na začátku lokalizačního procesu. Pošle všem uzlům všesměrovou zprávu se svou adresou, svou rolí a příkazem k hledání nejvzdálenějšího uzlu (viz dále). Spouští se z Tcl skriptu příkazem „aflocal_start“.
- **startN1()** metoda nastaví uzel do role kotevního uzlu 1 a zahájí jeho činnost. Je vyslána všesměrová zpráva oznamující celé síti adresu a roli kotevního uzlu. Pomoci této zprávy ostatní uzly sítě určí i vzdálenost od kotevního bodu.
- **startN2()** povýší uzel do role kotevního bodu 2. Uzel tuto skutečnost oznámí celé síti všesměrovou zprávou nesoucí informaci o jeho adrese a roli. Uzly tak také zjistí svou vzdálenost.
- **startN3()** metoda převezme další řízení algoritmu, povýší uzel do role kotevního bodu 3 a všesměrově rozešle zprávu o své adrese a roli. Příjemci také zjistí vzdálenost k tomuto uzlu.

- **startN4()** je obdobná metoda pro kotevní uzel 4. Změní jeho status a rozešle informaci o své adrese a roli.
- **startN5()** inicializuje kotevní uzel 5, rozešle informace o jeho adrese a roli. Tím je prohledávání sítě ukončeno a všechny síťové uzly znají pětici kotevních bodů a vzdálenosti k nim.
- **sendHello(u_int8_t ac)** slouží k rozeslání všesměrové hello zprávy. Jednotlivé kotevní uzly metodu používají pro oznámení své adresy a role ostatním uzlům. Parametr *ac* určuje požadovanou akci. Na jeho příjemce předá zprávu k příslušnému zpracování. Před samotným odesláním je ještě inkrementována hodnota sekvenčního čísla zprávy. Jednotlivé položky zprávy jsou předány metodě *sendMsg* (viz dále), která zajistí sestavení zprávy a její odeslání. Pro měření vzdálenosti od kotevního uzlu je nastavena položka *i* vytvářené zprávy na nulu.
- **recvHello(Packet *p)** metoda zpracuje přijatou hello zprávu. Slouží také pro zjištění vzdálenosti od odesilatele. Inkrementuje v příchozí zprávě celočíselnou hodnotu *i* a porovná ji s předchozí zjištěnou. Pokud je nově přijatá vzdálenost nižší, aktualizuje uložené údaje a přepošle hello zprávu svým sousedům (samozřejmě s novou hodnotou *i*). Druhým případem kdy je uložená vzdálenost aktualizována je přijetí hello zprávy s vyšším sekvenčním číslem (pro případ opakovaného spuštění algoritmu). Na základě role zdrojového uzlu je nastaven příslušný časovač a teprve po jeho vypršení je provedena požadovaná akce. Uzel tak získá čas pro přijetí hello zpráv od svých sousedů a je tak zaručeno, že zjištěná vzdálenost je nejmenší možná. Pokud přijme od souseda hello zprávu se vzdáleností dvě a horší (to znamená, že soused má špatnou vzdálenost), vytvoří a odešle znovu hello zprávu se svou vzdáleností.
- **recvHelloNeighbor(Packet *p)** metoda slouží pro sběr dat od okolních uzlů. Z přijaté zprávy uloží datové položky (*i*, *x* a *y*) společně s adresou odesilatele do seznamu *sousede*. Pokud pro příslušnou adresu záznam již existuje, jsou pouze aktualizována data.
- **recvData(Packet *p)** metoda je využita u kotevních uzlů ke shromažďování zaslaných dat. Uloží přijatá datové položky do seznamu *tabulka_hc* a odešle potvrzení odesilateli. Nakonec nastaví časovač, po jehož vypršení lze předpokládat, že obdržel všechny odpovědi.
- **recvDataAck(Packet *p)** metoda, jak již název napovídá, slouží pro přijetí potvrzovací zprávy. Potvrzení zastaví časovač a zamezí tak například opakování přenosu.

- **setTimer(int ns_code)** nastavuje a spouští časovač. Podle vstupního parametru *ns_code* je vybrána příslušná časová hodnota a pro pozdější identifikaci vypršení časovače je tento kód uložen do proměnné *timer_status*.
- **setTimer(double time)** je metoda názvem identická s předchozí, jde o přetíženou metodu se vstupním argumentem typu *double*. Provede přenastavení časovače na zadanou hodnotu *time*.
- **stopTimer()** je metoda volaná pro zastavení časovače.
- **sendMsg(nsaddr_t cil, u_int8_t ac, u_int8_t sn, u_int8_t src_status, nsaddr_t msg_src, nsaddr_t msg_dst, int i, double x, double y)** metoda zajišťuje sestavení paketu a jeho poslání nižším vrstvám. Vstupními parametry jsou *cíl* jakožto adresa použitá jako adresát paketu a dále postupně všechny položky AFLOCAL zprávy popsané výše. Jako cílová adresa lze použít adresu konkrétního uzlu nebo hodnotu IP_BROADCAST pro všesměrové vysílání.
- **opt_newpoz(double x, double y)** je metoda pro optimalizaci vypočtené polohy. Na základě zadané polohy spočítá z informací uložených v seznamu sousedních uzlu *sousede* energii v zadané poloze a doporučený posun uzlu. Návratovou hodnotou je vypočtená energie.
- **printTable()** je metoda použitá pro výpis seznamu *tabulka_hc*. Je ji možno spustit i z Tcl skriptu příkazem „print_table“.
- **printSousede()** je metoda pro výpis seznamu *sousede*. Je spustitelná z Tcl skriptu příkazem „print_sousede“.
- **printKotvy()** vypíše do jednoho řádku všechny adresy kotevních bodů, vzdálenosti k nim a vypočtenou polohu uzlu. Lze ji spustit z Tcl skriptu příkazem „print_kotvy“.

5.1.3.Třída Aflocal_Timer

V souboru *aflocal.h* je deklarována i třída časovače *Aflocal_Timer*. Ta dědí vlastnosti třídy *TimerHandler* existující ve struktuře NS2. Jeho metoda *virtual void expire(Event* e)* volá metodou *timer_expire()* třídy *Aflocal*.

5.1.4.Třída Aflocal_table

V souborech *aflocal_table.h* a *aflocal_table.cc* je definována třída *Aflocal_table*. Hlavním úkolem této třídy je ukládání, udržování a spravování dat o jiných uzlech. Informace jsou uloženy do struktury obousměrně vázaného lineárního seznamu.

Pro vkládání dat je určena metoda *void add(nsaddr_t adresa, int hopcount, double poz_x, double poz_y)*, která má jako vstupní parametry adresu vkládaného uzlu a trojici datových polí, první typu *int* a další dva typu *double*. Nový záznam je vložen na konec seznamu. Pokud pro zadanou adresu již existuje záznam, je nahrazen.

Podobně pracuje metoda *void update(nsaddr_t adresa, int hopcount, double poz_x, double poz_y)*. Rozdíl je v tom, že pouze aktualizuje již existující záznam, ale nevytvoří nový. Metoda *void empty()* uvolní z paměti všechny uložené položky a vyprázdní tak seznam pro nové použití.

Třída dále obsahuje metody pro pohyb v seznamu *void next()* pro přechod na předcházející záznam, *void prew()* pak pro následující záznam a *void rewind()* pro přechod na začátek. Pro přístup k datům jsou implementovány jednoduché funkce *nsaddr_t adr()*, *int hop()*, *double pozX()* a *double pozY()* vracející vždy odpovídající položku.

Pro zjednodušení nebo testování jsou implementovány i metody pro výpis dat celého seznamu *void print()* případně jeho setřídění podle adresy uzlu *void sort()*. Pro procházení seznamu v cyklu je přidána metoda *bool eol(),* která testuje zda-li bylo dosaženo konce seznamu. Na jeho konci vrací hodnotu *true*. Dále ještě metody pro nalezení maxima *void maxhop()* a *void maxX()*. Tyto vyhledávací metody nastaví ukazatel na nalezený záznam a data je pak možno přečíst výše uvedenými metodami.

5.2. Začlenění protokolu AFLOCAL do NS2

Pro přidání nového protokolu do struktury simulačního prostředí NS2 je potřeba provézt několik změn [11]. Nejprve je potřeba nakopírovat zdrojové soubory protokolu do nového adresáře v adresářové struktuře NS2 (například ns-allinone-2.33/ns-2.33/aflocal/).

5.2.1. Soubor packet.h

Aby bylo možné používat nově definovanou zprávu, je potřeba přidat do souboru *common/packet.h* do definic na začátku souboru řádek s novým typem paketu.

```
static const packet_t PT_AFLOCAL = 61; // AFLOCAL
```

V tomtéž souboru pak dále ještě do třídy *p_info* a její metody *initName* přidat následující řádek.

```
name_[PT_AFLOCAL]="AFLOCAL";
```

5.2.2. Soubory cmu-trace.h a cmu-trace.cc

Trasovací objekt *Trace* slouží pro zápis trasovacích informací do výstupního souboru simulace. Pokaždé, když je nějaký paket vyslán, přijat nebo zahozen musí o tom vzniknout záznam. Pro záznam těchto informací ohledně protokolu AFLOCAL je potřeba přidat metodu *format_aflocal*. V souborech *trace/cmu-trace.h* a *trace/cmu-trace.cc* je proto potřeba provést několik změn.

V souboru cmu-trace.h přidáme metodu *format_aflocal* jako privátní metodu třídy *CMUTrace*, přidáme tedy následující řádek.

```
void format_aflocal(Packet *p, int offset);
```

V souboru trace/cmu-trace.cc na začátku připojíme zdrojový soubor pro nový paket typu aflocal.

```
#include <aflocal/aflocal_pkt.h> //AFLOCAL
```

Také přidáme definici metody *format_aflocal.*

```
void
CMUTrace::format_aflocal(Packet *p, int offset)
{
struct hdr_Aflocal* ph = HDR_AFLOCAL(p);
if (pt_->tagged()) {
sprintf(pt_->buffer() + offset,
"-AFLOCAL:o %d -AFLOCAL:s %d -AFLOCAL:ac %d ",
ph->msg_src,
ph->sn,
ph->ac);
}
else if (newtrace_) {
sprintf(pt_->buffer() + offset,
"-P AFLOCAL -Po %d -Ps %d -Pac %d ",
ph->msg_src,
ph->sn,
ph->ac);
}
else {
sprintf(pt_->buffer() + offset,
"[AFLOCAL %d %d %d] ",
ph->msg_src,
ph->sn,
ph->ac);
}
}
```

Už zbývá jen přidat do metody *CMUTrace::format* následující řádky.

```
case PT_AFLOCAL:  format_aflocal(p, offset);
                  break;
```

5.2.3.Soubor ns-packet.tcl

Pro přidání nového typu paketu pro náš protokol i do Tcl struktury je potřeba v souboru *tcl/lib/ns-packet.tcl* nalézt příkaz *foreach prot* a přidat následující řádek.

```
AFLOCAL # AFLOCAL
```

5.2.4.Makefile

Poslední nutnou úpravou je přidání cesty pro kompilaci nového protokolu. Do souboru *Makefile* v adresáři ns-2.33 přidáme do direktivy *OBJ_CC* = následující řádek.

```
aflocal/aflocal.o aflocal/aflocal_table.o \
```

5.2.5.Kompilace

Protože jsme změnili soubor *common/packet.h*, ale nebylo zasahováno do souboru *common/packet.cc*, musíme před samotnou kompilací ještě změnit čas modifikace tohoto souboru. Tím bude nově zkompilována i třída *Packet* se všemi provedenými změnami. Jedním ze způsobů je příkaz *touch*. Do konzole tedy napíšeme následující příkazy.

```
[ns-2.33]$ touch common/packet.cc
[ns-2.33]$ make
```

Pokud kompilace proběhla bez chyb, jsou v adresáři *ns-2.33/aflocal/* vytvořeny zkompilované soubory *aflocal.o* a *aflocal_table.o*.

5.2.6.Připojení k uzlu v Tcl skriptu

Po úspěšné kompilaci je protokol AFLOCAL připraven k použití v simulačním prostředí NS2. K novému uzlu je agent Aflocal připojen příkazem *attach* na začátku simulačního skriptu.

```
set node [$ns_ node]          # vytvoreni noveho uzlu
set afl [new Agent/Aflocal]   # vytvoreni agenta Aflocal
$node attach $afl 61 # pripojeni agenta afl k uzlu node na port 61
```

5.3. Popis lokalizačního procesu

Činnost lokalizačního algoritmu je do jisté míry decentralizovaná. Mnohá rozhodnutí provádí uzly nezávisle na řídícím uzlu, většinou na základě informací od okolních stanic. Přesto v síti vždy jeden uzel zastává řídící roli. Po ukončení určené fáze algoritmu v průběhu lokalizačního procesu předá řízení novému uzlu a dále už proces neovlivňuje. Důvodem pro takové rozdělení řídících funkcí je rozložení zvýšený zátěže takového uzlu. Navíc je neefektivní přeposílat potřebné informace mezi jednotlivými kotevními uzly. Tak jako je lokalizační algoritmus rozdělen do několika fází, je podobně i činnost agenta Aflocal členěna do 5 částí. Pro spuštění procesu existuje v Tcl skriptu jediný příkaz, který pověří vybraný uzel k zahájení komunikace.

```
$ns_ at $start "$afl aflocal_start"
```

Před započetím samotného popisu lokalizačního procesu bych ještě objasnil význam kódových značek začínajících označením AC_ nebo NS_. Jsou to direktivou

definované makra preprocesoru (v souboru aflocal.h), která jsou před překladem nahrazena příslušným číselným kódem. Použití textové podoby ale přispívá přehlednosti a srozumitelnosti kódu. Seznam všech použitých kódů včetně jejich číselné reprezentace a popisu je uveden v tabulkách jako příloha 1 a příloha 2.

5.3.1.Hledání kotvy 1

Nejprve je v Tcl skriptu na zvoleném uzlu (nazvěme jej uzlem N0) zavolána metoda *command*. Na základě zadaného příkazu, tedy „aflocal_start“ je následně spuštěna metoda *startN0()*. Ta nejprve uloží adresu uzlu do proměnné *index*. Je aktualizována proměnná *kotva*, uzel je nastaven jako kotevní bod 0 se vzdáleností 0. Následně je vytvořena nová instance třídy *Aflocal_table* a přiřazena k ukazateli *tabulka_hc*. Aktuální uzel je ihned přidán jakožto první záznam. Status uzlu symbolizovaný proměnnou *node_status* je nastavena na definovanou hodnotu *NS_N0*. Je nastaven časovač na krátké zpoždění, po kterém je metodou *sendHello* odeslána všesměrově zpráva s *ac* kódem *AC_HELLO_SRC_N0*. Nakonec je nastaven časovač voláním metody *setTimer* s parametrem kódu *NS_N0*.

Metoda *sendHello* inkrementuje sekvenční číslo pro zprávu, jako cílovou adresu i příjemce nastaví všesměrovou adresu *IP_BROADCAST* (konstanta definovaná v souboru *common/ip.h*), jako zdroj zprávy vlastní adresu a do datové položky pro celé číslo nastaví 0. Toto pole přenášeného paketu bude postupně inkrementováno s každým dalším skokem. Všechny potřebné informace předá metodě *sendMsg*, která sestaví potřebný paket a předá jej k odeslání.

Paket přijmou všechny uzly v dosahu. Nižší vrstvy síťového modelu nakonec předají paket metodě *recv*. Na základě použitého *AC_HELLO_SRC_N0* kódu uzel rozpozná že jde o Hello zprávu od uzlu N0 a předá jej metodě *recvHello*. Metoda nejprve porovná hodnotu celočíselného pole *hdr->i* (*hdr* je ukazatel na strukturu paketu AFLOCAL) s uloženou vzdáleností od uzlu N0 (na začátku jsou všechny vzdálenosti nastaveny na 255). Porovná se také uložená hodnota sekvenčního čísla *node_sn* s sekvenčním číslem paketu. Pokud některá z podmínek vyhoví, je aktualizována proměnná *kotva* pro daný kotevní uzel. Je vymazán seznam sousedních uzlů *sousede* a metodou *setTimer* nastaven časovač s kódem *NS_CANDIDATE_N1*. Hello zpráva je ještě přeposlána dál (s inkrementovanou hodnotou *i*).

Příjme-li uzel hello zprávu se vzdáleností horší o více než jedna, pošle znovu hello zprávu se svou vlastní vzdáleností. Informuje tak své sousedy o existenci kratší cesty, než kterou mu poslali.

Po vypršení časovače nastaveného kódem *NS_CANDIDATE_N1* je vyvolána metoda *timer_expire*. Na základě kódu uloženého v proměnné *timer_status* jsou

vykonány příslušné příkazy. Do seznamu *sousede* je přidán aktuální uzel, následně je všesměrově rozeslána zpráva s kódem *AC_HELLO_NEIGHBOR* a změřenou vzdáleností od kotevního uzlu N0. Po odeslání zprávy sousedům je nastaven časovač s hodnotou *NS_HELLO_REPLY_N0*.

Okolní uzly přijímají od svých sousedů zprávy s *ac* polem nastaveným na *AC_HELLO_NEIGHBOR*. Tyto pakety jsou předány metodě *recvHelloNeighbor*, která uloží adresu sousedního uzlu společně s datovými položkami do seznamu *sousede*. Tam se postupně shromáždí informace od všech sousedů uzlu.

Po vypršení časovače a volání metody *timer_expire()* je pro nastavený kód *NS_HELLO_REPLY_N0* prohledán seznam *sousede*. Je nalezen soused s největší vzdáleností od kotvy N0. Pokud je jím aktuální uzel, pošle svou vzdálenost přímo kotevnímu uzlu. V opačném případě neprovádí nic a čeká na další příkaz.

Poslání vzdálenosti kotevnímu uzlu N0 zajišťuje opět metoda *sendMsg*. Zpráva je opatřena *AC_DATA* kódem a jako cílová adresa přímo adresa kotevního bodu N0. Nakonec je ještě nastaven časovač s kódem *NS_WAIT_ACK_N0* pro případné opakované vysílání.

Paket s nastaveným kódem *AC_DATA* je přijat uzlem N0 a předán metodě *recvData*. Adresa odesilatele a datové položky jsou vloženy do seznamu *tabulka_hc* a je odesláno ACK potvrzení příjmu (metoda *sendMsg* s adresou odesilatele a *AC_DATA_ACK* kódem). Současně je znovu nastaven časovač uzlu N0.

Vzdálený uzel přijme potvrzovací zprávu *AC_DATA_ACK* metodou *recvDataAck*, jejíž jedinou úlohou je zastavit běh časovače a tím zamezit opakovanému vysílání. Pokud ale z nějakého důvodu není ACK potvrzení doručeno před vypršením časovače, je v metodě *timer_expire* pod kódem *NS_WAIT_ACK_N0* vyvoláno nové vyslání změřené vzdálenosti. Opakovaný přenos ale probíhá všesměrovou cestou. Paket je tak předáván po jednotlivých uzlech sítě až k cílovému. Je odeslán s kódem *AC_DATA_BROADCAST* a na místě pole *src_status* je vzdálenost k uzlu N0. Paket je přijat sousedními uzly a na základě jeho kódu je předán metodě *fwdData*. Ta nejprve zjistí, zda už tentýž paket jednou nepřeposílala. Následně porovná svou vzdálenost k uzlu N0 s hodnotou pole *src_status* sníženou o jedna. Pokud je vzdálenost nižší nebo rovna, je paket přeposlán dál. Tímto způsobem je minimalizováno zatížení sítě pouze na uzly v potřebném směru.

Po vypršení časovače u kotevního uzlu N0 (kód *NS_N0*) je možno předpokládat, že známe nejvzdálenější uzel. Nejprve je zkontrolován počet záznamů seznamu *tabulka_hc*. Je li větší než jedna, je uzel s největší vzdáleností vybrán jako nový kotevní uzel N1. Jeho adresa i vzdálenost je zaznamenána do proměnné *kotvy* a je mu poslána zpráva s řídícím kódem *AC_COMMAND_SET_N1*, která ho pověřuje dalším řízením lokalizačního procesu.

5.3.2.Hledání kotvy 2

Přijetím zprávy s kódem *AC_COMMAND_SET_N1* je uzel vyzván k pokračování lokalizačního procesu. Je volána metoda *startN1*. Ta nastaví vlastní adresu a vzdálenost 0 pro proměnnou *kotva* pro uzel N1. Následně je vytvořena nová instance třídy *Aflocal_table* přiřazená k ukazateli *tabulka_hc*. Uzel je také do seznamu ihned přidán jakožto první záznam. Status uzlu je nastaven na roli *NS_N1*, nastaven časovač je krátký časovač a po jeho vypršení je vyslána hello zpráva s kódem *AC_HELLO_SRC_N1*. Jejím účelem je aby všechny uzly sítě zjistily adresu a vzdálenost bodu N1. Nakonec je nastaven časovač příkazem *setTimer(NS_N1)*.

Odeslání i zpracování hello zprávy je srovnatelné s fází hledání kotvy 1. Jednotlivé uzly si předávají hello zprávu, paket je předán metodě *recvHello*. Ta na základě role zdrojového uzlu a kódu *AC_HELLO_SRC_N1*. Metoda *recvHello* inkrementuje přijatou vzdálenost a porovná ji s uloženou. Pokud je přijatá vzdálenost nižší (nebo pokud jde o novou hello zprávu s novým sekvenčním číslem), uloží si uzel novou vzdálenost společně s adresou uzlu a přepošle zprávu dále s novou hodnotou vzdálenosti. Je také vyprázdněn seznam *sousede*. Nakonec ještě nastaví časovač voláním *setTimer* s hodnotou *NS_CANDIDATE_N2*. Pokud je přijatá vzdálenost horší o více než jedna znamená to, že sousední uzel nezná nejkratší vzdálenost, proto je poslána nová hello zpráva se vzdáleností aktuálního uzlu.

Po vypršení časovače je vyvolána metoda *timer_expire* a pro nastavenou hodnotu *timer_status* na *NS_CANDIDATE_N2* nejprve přidán aktuální uzel do seznamu *sousede*. Následně je všesměrově odeslána sousedům zpráva s kódem *AC_HELLO_NEIGHBOR* a změřenou vzdáleností od kotvy N1. Nakonec je kódem *NS_HELLO_REPLY_N1* nastaven časovač.

Sousední uzly přijímají *AC_HELLO_NEIGHBOR* zprávu a uloží si informace od svých sousedů. Po vypršení časovače s kódem *NS_HELLO_REPLY_N1* tak každý uzel zná vzdálenosti svých sousedů od N1. Ze seznamu je vybrán uzel s nejvyšší vzdáleností, a jen je-li jím sám uzel, může poslat odpověď zpět kotevnímu uzlu N1. Je poslána zpráva adresovaná uzlu N1, jako *ac* kód je použita hodnota *AC_DATA* a celočíselné pole i nese hodnotu vzdálenosti. Po odeslání je nastaven časovač kódem *NS_WAIT_ACK_N1*, během kterého uzel čeká na potvrzení. Pokud před jeho vypršením neobdrží uzel potvrzení, pokusí se odeslat data ještě jednou, tentokrát ale všesměrově. Pole *src_status* zprávy AFLOCAL je nahrazeno vzdáleností k uzlu N1, kód je změněn na *AC_DATA_BROADCAST*. O předávání paketu mezi uzly se stará metoda *fwdData*, která vždy nejprve zkontroluje, zda již tento paket nesměrovala. Pokud ne, sníží hodnotu vzdálenosti přenášenou v poli *src_status* a následně ji porovná se svou vzdáleností. Pokud je rovna nebo nižší (vzdálenost uzlu nižší než v poli *src_status*), přepošle zprávu dále.

Kotevní uzel N1 během čekání přijímá data od vzdálených uzlů. Při přijetí dat přidá informace do seznamu *tabulka_hc* a obratem na zdrojovou adresu odešle potvrzení *AC_DATA_ACK*. Cílový uzel po obdržení zastaví svůj časovač a zamezí se tak opakovanému vysílání. Po vypršení časovače kotevního uzlu N1 je vyvolána metoda *timer_expire* a pod kódem *NS_N1* je provedena požadovaná akce. Nejprve zjistíme počet záznamů v seznamu *tabulka_hc*. Pokud je rovna jedné, znamená to, že nebyla přijata ani jedna odpověď a fáze hledání je spuštěna znova metodou *startN1*. V opačném případě, tedy když *tabulka_hc* obsahuje informace od několika vzdálených uzlů, je z nich vybrán ten nejvzdálenější. Právě ten uzel bude použit jako kotevní bod N2. Jeho adresu a vzdálenost je zaznamenána do proměnné *kotvy* a následně je mu odeslána zpráva *AC_COMMAND_SET_N2*.

5.3.3.Hledání kotvy 3

Přijetím zprávy s kódem *AC_COMMAND_SET_N2* je uzel vybrán jako kotevní bod 2, nazvěme jej N2. Je spuštěna metoda *startN2*. Nastaví proto do své proměnné kotva jako kotevní bod 2 svou adresu a nulovou vzdálenost. Je vytvořena nová instance třídy *Aflocal_table* a přiřazená ukazateli *tabulka_hc*. Uzel N2 je přidán jako první záznam. Role uzlu je nastavena do proměnné *node_status* na hodnotu *NS_N2*. Je odeslána hello zpráva s kódem *AC_HELLO_SRC_N2*. Nakonec je ještě nastaven časovač voláním metody *setTimer* s parametrem *NS_N2*.

Hello zprávu přijmou okolní uzly. Paket je předán na zpracování opět metodě *recvHello* (detailně popsané dříve), která ze zprávy určí vzdálenost ke kotevnímu uzlu (podle pole *src_status* určí roli zdrojového uzlu), nastaví časovač s kódem *NS_CANDIDATE_N3* a zprávu předá dále. V případě příjmu hello zprávy se vzdáleností horší o více než jedna vygeneruje novou hello zprávu, aby souseda informovala o kratší cestě k N2.

Po vypršení časovače se ve vyvolané metodě *timer_expire* pro kód *NS_CANDIDATE_N3* provede kontrola, zda je rozdíl vzdáleností od uzlu N1 a N2 menší nebo roven jedné. V tom případě se jedná o jednoho z kandidátů na kotevní bod 3. Je nastaven znovu časovač (*NS_HELLO_REPLY_N2*), po jehož vypršení je znovu provedena kontrola rozdílu vzdáleností k oběma kotevním bodům. Při splnění podmínky je vypočtena hodnota pomocné proměnné *vaha* typu *double*. Cílem je získat jediné číslo, na jehož základě je možné určit nejvhodnějšího kandidáta na roli uzlu N3. Výpočet této hodnoty popisuje následující část kódu.

```
// vypocet vahoveho koeficientu
vaha=2.0-abs(kotva[2][1]-kotva[1][1]); //rozdil vydalenosti idealne 0
vaha += (double)(kotva[2][1]+kotva[1][1])/100.0; //socet maximalni
vaha += (double)(kotva[0][1])/10000.0;         // ten blizsi Node0
```

Co nejmenší rozdíl vzdáleností od N1 a N2 má nejvyšší prioritu, je tedy v řádu jednotek. Protože odečítáme od čísla 2.0, bude po tomto odečtu hodnota *vaha* v rozmezí <2,0; 1,0>. Pro úplnost ještě uvedu, že funkce *abs* vrací absolutní hodnotu svého parametru. V dalším řádku je přičten součet obou vzdáleností podělený 100. To sníží jeho váhu na výsledné hodnotě. Za předpokladu, že součet vzdáleností bude menší než 100, se hodnota zvýší o méně než 1,0. Poslední uvedený řádek řeší situaci, kdy více uzlů bude mít stejný rozdíl i součet vzdáleností. Proto je jako třetí parametr ovlivňující výsledný rozhodovací koeficient přičtena vzdálenost od startovacího uzlu N0 podělená 10000. Výsledný váhový koeficient je odeslán *AC_DATA* zprávou kotevnímu uzlu N2. Nakonec je opět nastaven časovač s kódem *NS_WAIT_ACK_N2*, po jehož vypršení bude přenos opakován (opět všesměrově, tak jak bylo již popsáno dříve).

Uzel N2 postupně přijme vypočtené hodnoty od všech kandidátů (tedy uzlů s rozdílem vzdáleností maximálně jedna). Po vypršení časovače je pro *timer_status* kód *NS_N2* nejprve zjištěn počet záznamů v seznamu *tabulka_hc*. Pokud není jedna (což by znamenalo, že uzel N2 neobdržel žádná data od kandidátů na N3) tak ze seznamu vybere uzel s maximálním koeficientem (přenášen i uložen jako položka *x*) a vybere jej tak jako nový kotevní bod N3. Jeho adresa je uložena do proměnné *kotva* a je mu odeslána zpráva s kódem *AC_COMMAND_SET_N3*.

5.3.4.Hledání kotvy 4

Fáze hledání kotevního uzlu 4 (dále jen N4) je zahájena přijetím zprávy s příkazem *AC_COMMAND_SET_N3* nově nalezeným uzlem N3. Podobně jako u ostatních kotevních bodů je i zde zavolána odpovídající metoda *startN3*. Ta nastaví proměnnou *kotva* pro kotvu 3, tedy vlastní adresu a nulovou vzdálenost. Také je vytvořena nová instance seznamu *Aflocal_table* s ukazatelem *tabulka_hc*. Je přidán první záznam pro tento uzel, nastavena stavová proměnná *node_status* na *NS_N3*. Obdobně jako v předchozích případech je odeslána hello zpráva, tentokrát s kódem *AC_HELLO_SRC_N3*. Nakonec je nastaven časovač pro *NS_N3*.

Okolní uzly přijmou hello zprávu a předají ji ke zpracování metodě *recvHello*. Ta již byla detailně popsána dříve. Podstatné je že uzly v případě nižší vzdálenosti aktualizují svou proměnnou kotva pro daný uzel na nižší hodnotu a zprávu předají dále. Také nastaví časovač kódem *NS_CANDIDATE_N4*.

Po vypršení časovače nastaveného tímto kódem je porovnán rozdíl vzdáleností od uzlu N1 a N2. Požadována je hodnota menší nebo rovna jedné. Tato podmínka je stejná jako u hledání N3. Pokud uzel podmínce vyhoví, je nastaven časovač pro kód *NS_HELLO_REPLY_N3*. Ten pak provede výpočet váhového koeficientu.

```
vaha=(1.0 - abs(kotva[2][1]-kotva[1][1]))/10.0; // rozdil idealne 0
vaha +=  (double)kotva[3][1];                   // co nejdal od N3
```

Do výpočtu je zahrnut rozdíl vzdáleností mezi N1 a N2 (přesněji jejich absolutní hodnota) podělený 10. Tím se tomuto kritériu sníží důležitost. Hlavní význam na rozhodování má vzdálenost od uzlu N3. Rozdíl N1 a N2 jen rozhodne v případě dvou stejně vzdálených uzlů. Vypočtená váha je poslána uzlu N3 a je nastaven časovač pro případné opakování přenosu. Pokud by uzel neobdržel ACK potvrzení před vypršením časovače, pošle data znovu, ale tentokrát všesměrově (popsáno dříve).

Kotevní bod N3 přijímá data zaslaná kandidáty na roli N4 a ukládá je do seznamu *tabulka_hc*. Přijetí dat potvrdí odesláním *AC_ACK_DATA* zprávy. Po vypršení časovače pro kód *NS_N3* je zkontrolováno, zda je počet záznamů v seznamu *tabulka_hc* rovný jedné. Pokud tomu tak je, znamená to, že uzel nepřijal žádné data a celá fáze hledání N4 se opakuje. V opačném případě je vybrán záznam s nejvyšší hodnotou v poli *x* (zde byla uložena hodnota váhového koeficientu) a je nastaven do proměnné *kotva* na pozici kotvy 4. Uzlu je odeslána zpráva s kódem *AC_COMMAND_SET_N4*, čímž je mu fakticky předáno řízení lokalizačního procesu.

5.3.5.Hledání kotvy 5

Přijetím zprávy s kódem *AC_COMMAND_SET_N4* je volána metoda *startN4*. Ta nastaví aktuální uzel do role kotevního uzlu N4, aktualizuje svou proměnnou kotva svou adresou a nulovou vzdáleností. Je vytvořena nová instance třídy *Aflocal_table* přiřazená ukazateli *tabulka_hc*. Také je přidán první záznam pro vlastní uzel. Stejně jako v předchozích případech je rozeslána hello zpráva, tentokrát s kódem *AC_HELLO_SRC_N4* všem okolním uzlům a nastaven časovač pro *NS_N4*.

Okolní uzly přijmou hello zprávu a podle stejného postupu popsaného výše aktualizují svou vzdálenost k uzlu N4 a zprávu předají dál. Zároveň si nastaví časovač s použitím kódu *NS_CANDIDATE_N5*. Po vypršení časovače je zjištěno, zda rozdíl vzdáleností k N1 a N2 je menší nebo roven jedné. Zároveň Také musí platit, že rozdíl vzdáleností k N3 a N4 musí být menší nebo roven dvěma. Tím jsou vybrány jen uzly nacházející se ve středu. Po vypršení dalšího časovače s kódem *NS_HELLO_REPLY_N4* je tato kontrola zopakována a je vypočítán váhový koeficient reprezentující vhodnost kandidáta.

```
// vypocet vahoveho koeficientu
vaha=2.0-abs(kotva[2][1]-kotva[1][1]); // rozdil vydalenosti N1 a N2
vaha+=(1.0-abs(kotva[4][1]-kotva[3][1]))/100.0; // rozdil N3 a N4
```

Základem pro koeficient je opět rozdíl vzdáleností od uzlu N1 a N2. Ta je odečtena od čísla 2,0. V druhém řádku je požadavek na co nejmenší rozdíl vzdáleností k N3 a N4. Tento rozdíl je odečten od jedné a podělen 100. Hlavní vliv na výběr z kandidátů na uzel N5 má tedy opět vzdálenost k N1 a N2. Stejné vzdálenosti jsou pak rozlišeny rozdílem vzdáleností k N3 a N4. Výsledek je poslán zpět uzlu N4. Je také nastaven časovač pro opakovaný přenos *NS_WAIT_ACK_N4*. Pokud před jeho vypršením nebude přijato *ACK_DATA* potvrzení, budou data poslána znovu, ale všesměrovou cestou (popsáno dříve).

Kotva N4 přijímá data metodou *recvData* (také podrobně popsána dříve). Přijaté informace uloží do seznamu *tabulka_hc* a odešle zpět *AC_DATA_ACK* zprávu potvrzující příjem. Po vypršení časovače je pro kód *NS_N4* provedena kontrola počtu záznamů v seznamu *tabulka_hc*. Pokud je pouze jeden, je hledání N5 spuštěno znova. V opačném případě je ze seznamu vybrán záznam s největší hodnotou váhového koeficientu. Adresa vybraného uzlu je uložena do proměnné *kotva* na pozici kotevního uzlu 5 a je mu poslána řídící zpráva *AC_COMMAND_SET_N5*.

5.3.6. Výpočet souřadnic

Nový kotevní bod N5 příjme zprávu *AC_COMMAND_SET_N5*, čímž je spuštěna metoda startN5. Uzel si nastaví svou adresu a nulovou vzdálenost v proměnné *kotva* pro kotevní bod 5. Nastaví svou roli na *NS_N5* a odešle hello zprávu s kódem *AC_HELLO_SRC_N5*. Nakonec ještě nastaví svůj časovač, tentokrát ale kódem *NS_VYPOCET_SOURADNIC*.

Okolní uzly přijmou hello zprávu a postupem pospaným již dříve pomoci ni zjistí adresu a vzdálenost uzlu N5. Zprávu si mezi sebou předávají. Zároveň si nastaví časovač hodnotou *NS_VYPOCET_SOURADNIC*.

Po uplynutí stanoveného času si na základě známých vzdáleností od všech kotevních uzlů každý uzel spustí výpočet souřadnic. Pomoci vzorců 4.01 a 4.02 jsou vypočteny polární souřadnice (*r, fi*). Ty jsou ještě následně převedeny na souřadnice kartézské (*poz_x, poz_y*).

```
r =(double)kotva[5][1];
fi = atan2( (kotva[1][1]-kotva[2][1]), (kotva[3][1]-kotva[4][1]));
poz_x = r * cos(fi);
poz_y = r * sin(fi);
```

Po vypočtení souřadnic je nastaven časovač s kódem *NS_OPT_SEND*. Po jeho vypršení uzel informuje své sousedy o svých souřadnicích všesměrovou

zprávou *AC_HELLO_NEIGHBOR*. Na informace od svých sousedů je třeba počkat, je proto nastaven časovač pomoci *NS_OPT_WAIT*.

Po vypršení časovače s kódem *NS_OPT_WAIT* dal uzel dostatek času svým sousedům na výpočet jejich polohy a její odeslání. Pro potřeby vyhodnocení chyby určení polohy je proto vypočten ještě jeden parametr. Uzel na základě svých vypočítaných souřadnic a obdržených souřadnic spočítá svou průměrnou vzdálenost k sousedům. Pro každého ze sousedů tedy vypočítá vzdálenost, sečte ji s ostatními a podělí počtem sousedů. Výsledná hodnota průměrné vzdálenosti sousedů je uložena v proměnné *opt_e*.

5.3.7. Optimalizace souřadnic

Ačkoli třída *Aflocal* obsahuje funkci pro optimalizaci polohy, tato metoda není zcela implementována. Problémem je měření fyzické vzdálenosti mezi uzly. V prostředí NS2 není v současnosti začleněna podpora pro měření rádiových parametrů spoje mezi uzly. Není tedy možné realizovat optimalizaci polohy na základě měření fyzické vzdálenosti (metodou měření síly přijímaného signálu RSSI) a podle ní upravovat vypočtené souřadnice. Pokusy o vytvoření algoritmu optimalizace pracujícím bez vazby na fyzické prostředí nepřinesla výsledky. Ve výsledných simulacích proto optimalizace není použita.

5.4. Simulace v prostředí NS2

Vlastní simulace v simulačním prostředí NS2 je ovládána skrze Tcl skript. Jsou v něm jednotlivá nastavení pro simulované prostředí, topologie a rozmístění uzlů i časy spuštění jednotlivých akcí nebo přenosů. Výstupem takovéto simulace je pak trasovací soubor a soubor pro vizualizaci programem NAM.

V trasovacím souboru (obvykle s příponou .tr) je zaznamenán průběh simulace [8]. Je zde textový záznam o odeslání, přijetí nebo zahození každého paketu. Každý řádek souboru odpovídá jednomu časovému okamžiku, kdy byl paket přijat, odeslán, předán nebo zahozen.

Soubor pro vizualizaci průběhu simulace pro program NAM také obsahuje záznam o každém vyslání, přijetí či zahození paketu, je ale určen hlavně pro zobrazení průběhu simulace. Obsahuje také záznam o topologii sítě, barvě datových toků nebo uzlů. Program NAM je Tcl/TK vizualizační nástroj pro názorné zobrazení průběhu simulace zaznamenané do nam souboru. Funguje jako přehrávač již provedené simulace, kdy je možno ji přehrát navolenou rychlostí nebo se v ní posouvat. Každý paket je od doby svého vyslání po svůj zánik znázorněn. Znázorněn i jeho postup sítí. Program NAM je součástí balíku programu NS2.

5.4.1. Simulační skript pro protokol AFLOCAL

Simulační skript je textový soubor, obvykle s příponou tcl. Musí obsahovat několik definovaných náležitostí [9]. I když je našim zájmem simulace AFLOCAL protokolu, nevyhneme se použití jiných protokolů a funkcí. Jejich nastavení je nutno provést před zahájením samotné simulace.

Bezdrátový model sítě standardu 802.15.4 je v prostředí NS2 implementován, je ale potřeba jej správně nastavit. Budeme také studovat energetickou náročnost lokalizace, je proto potřeba nastavit i EnergyModel. Protože protokol AFLOCAL využívá i unicastové přenosy, nevyhneme se použití směrovacího protokolu. Použit je AODV protokol. Význam jednotlivých zastavení jsou popsána v [9][12][13] a také komentována v samotném skriptu.

5.4.2.Průběh simulace

Pokud máme simulační nástroj NS2 správně nainstalován, jsou do něj správně přidány soubory nového protokolu AFLOCAL a provedeny potřebné úpravy (viz kapitola 5.2), pak je připraven k použití. Hotov musí být i simulační soubor. Simulace pak lze spustit z konzolového například okna příkazem:

```
[script]$ ns aflocal_simulace.tcl
```

Spustí se program ns a zahájí simulaci podle popisu v Tcl souboru. Zdrojový kód protokolu AFLOCAL obsahuje v klíčových místech příkazy pro výpis na standardní výstup. O průběhu simulace jsme tedy informováni. Na konci simulace je cyklem v Tcl skriptu ještě vypsána pro každý uzel informace o nalezených kotevních uzlech, vzdáleností k nim, vypočtených souřadnicích i průměrné vzdálenosti sousedů.

5.4.3.Vyhodnocení simulace

O výsledku simulace se zle přesvědčit pohledem do konzolového okna. U všech uzlů jsou uvedeny adresy a vzdálenosti ke kotevním uzlům, vypočteny jsou i souřadnice a průměrná vzdálenost sousedů (slouží k vyhodnocení přesnosti). Tento závěrečný výpis, společně s trasovacím souborem a souborem nam slouží pro vyhodnocení přesnosti a energetické náročnosti lokalizace. Pro představu uvádím příklad konzolového výstupu reprezentující dva řádky.

```
100.00 [NodeA: 13, status 99] Kotvy (adr/hop): (24/05) (42/11)
(06/01) (48/05) (00/07) (24/05) ; SouradniceXY [-0.9806, 4.9029],
Prumerna vzdalenost: 1.4522
100.00 [NodeA: 14, status 99] Kotvy (adr/hop): (24/04) (42/04)
(06/08) (48/10) (00/02) (24/04) ; SouradniceXY [3.5777, -1.7889],
Prumerna vzdalenost: 1.5230
```

Jako první údaj je čas simulace. Protože je výpis spouštěn na konci simulace (dlouhé 100s), je zde čas 100. V následující hranaté závorce je identifikace uzlu, tedy jeho adresa a aktuální role. Následuje výpis kotevních uzlů a vzdáleností k nim. Na konci jsou uvedeny vypočtené souřadnice a průměrná vzdálenost k sousedům.

V uvedeném příkladu můžeme tedy o uzlu s adresou 13 (adresa odpovídá číslu uzlu v NAM) říci, že je vzdálen od startovacího bodu 5 skoků, od uzlu s adresou 42, který byl vybrán jako kotva N1 11 skoků, od N2 1 skok, 5 skoků od N3, od N4 7 skoků a středový uzel N5 je vzdálen 5 skoků. Dále je zde informace o vypočtených souřadnicích X -0,9806 a Y 4,9029. Nakonec ještě vidíme, že průměrná vzdálenost jeho sousedů je 1,4522.

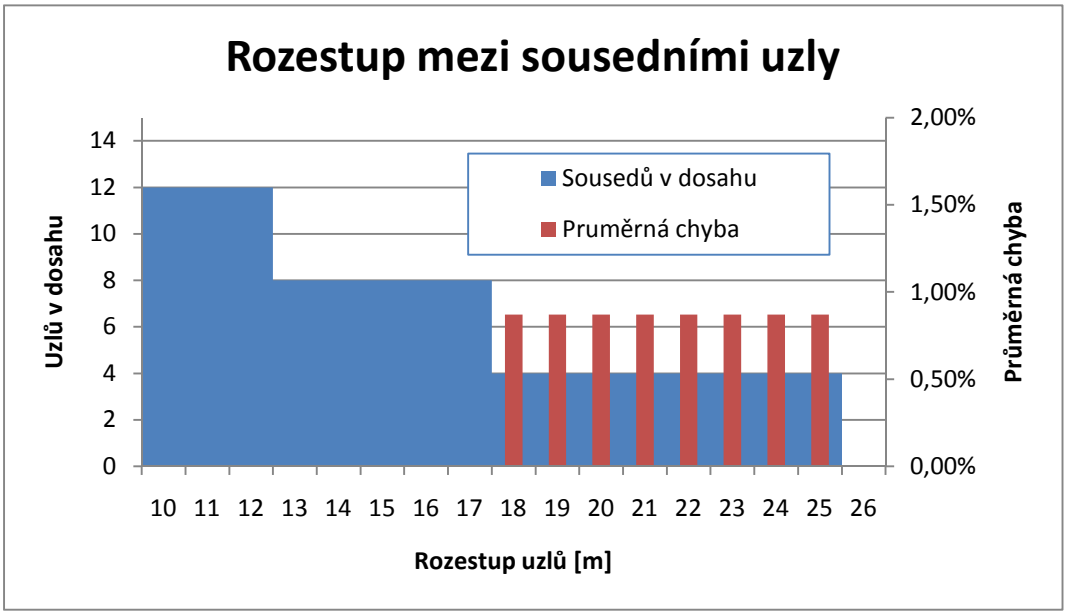
6. Výsledky simulace

Lokalizační protokol AFLOCAL byl podroben mnoha simulacím. Během nich byly simulovány různé scénáře s cílem určit vliv topologie, velikosti sítě nebo energetické nároky na síťové uzly. Také měly za cíl odhalit slabiny vytvořeného lokalizačního protokolu AFLOCAL. Všechny výsledky simulací byly posléze vhodně vyhodnoceny a vyneseny do grafů.

Při všech simulacích byla použita délka simulace 100 sekund. Start lokalizačního algoritmu byl spuštěn příkazem v čase 10 sekund. Na samotnou lokalizaci měla tedy síť vždy maximálně 90 sekund. I když některé simulace skončily bez úspěšné lokalizace, nebylo to způsobeno nedostatkem času ale zahazováním paketů (viz dále). Také hodnota rádiového dosahu byla nastavena na 25 metrů. Vliv změny rádiového dosahu je totiž totožný se změnou rozestupů uzlů.

6.1. Hustota sítě

Nejprve byla provedena řada simulací pro určení optimální topologie sítě. Pro síť se 49 uzly uspořádanými do mřížky byla zjišťována závislost výsledku lokalizace na rozestupu vedlejších uzlů, tedy vzdálenosti bodů v mřížce. Rádiový dosah byl pro všechny simulace nastaven na 25 metrů, neměnné zůstaly i ostatní parametry sítě včetně časů spuštění a ukončení simulace i startu lokalizačního algoritmu. Byly provedeny simulace pro rozestup uzlů od 10m do 26m, s krokem 1m. Výsledky jsou shrnuty v následujícím grafu (Obr.10.).



Obr.10. Vliv hustoty sítě na výsledek lokalizace

Pro nízké hodnoty rozestupů simulace nedopadla úspěšně. V zadaném čase nebyly sítě schopny dokončit určení všech kotevních bodů a zjištění vzdáleností k nim. Problémem je ve velkém překrytí rádiového dosahu uzlů a tím velký provoz na nosném kanále. Dochází k množství kolizí a vysokému zatížení jednotlivých uzlů okolním provozem. To způsobuje problémy nejenom protokolu AFLOCAL, ale především směrovacímu protokolu AODV. Ten pracuje jen s obtížemi. Lokalizace ztroskotá právě kvůli AODV protokolu, kdy směrovací protokol není schopen nalézt cestu pro paket sítí. To je v trasovacím souboru zastoupeno řádkem se zahozeným paketem a důvodem jeho zahození s chybou směrovacího protokolu RTR NRTE. Jde o chybu DROP_RTR_NO_ROUTE, kdy se směrovacímu protokolu nepodařilo nalézt cestu k cílovému uzlu.

Dalším problémem jsou zahozené pakety z důvodu přeplněná odchozí fronty, která se díky velkému provozu zaplňuje. Společně s kolizemi při vysílání to působí problémy hlavně při vyšší hustotě sítě. Vznikají tak i chyby v měření vzdáleností od kotevních bodů. Řešením při unicastovém přenosu je potvrzování přijmu. Pokud nedorazí potvrzovací paket před vypršením časovače, vysílání se opakuje. Bohužel tato metoda nejde použít při všesměrovém vysílání hello zprávy, protože vysílací uzel nezná topologii sítě ani počet všech uzlů. Nemůže proto vědět, kolik případných potvrzení příjmu má očekávat ani komu opakovat přenos. Ztrátovost pro jednotlivé simulace je znázorněna v grafu jako Příloha 5.

Od rozestupu 18 metrů a více klesne počet sousedních uzlů v rádiovém dosahu na 4 uzly, a toto se jeví jako ideální kombinace. Provoz na rádiovém kanále se sníží, odchozí fronty uzlů se stačí vyprazdňovat a také AODV protokol pracuje korektně. Výsledná průměrná chyba vzdálenosti sousedních uzlů mezi vypočtenou a skutečnou polohou je pak se zvyšujícím se rozestupem konstantní. Změna nastává až při rozestupu 26 metrů, kdy je přesažen rádiový dosah uzlů.

Problémem tedy není samotný rozestup uzlů, ale počet uzlů v rádiovém dosahu. V reálném prostředí lze předpokládat zvýšenou chybovost samotného přenosu u hranice rádiového dosahu, která může také zhoršit výsledky lokalizace. V prostředí NS2 však tyto fyzikální vlastnosti zahrnuty nejsou.

6.2. Velikost sítě

Velikost sítě, tedy počet jejich uzlů zásadním způsobem charakterizuje oblast použití protokolu. Pro simulace byla použita rovnoměrně rozložená síť uzlů. Uzly jsou rozmístěny do mřížky, vzdálenost mezi nimi je 20 metrů při rádiovém dosahu 25m. Uzel má tedy v rádiovém dosahu 4 sousedy. Byly provedeny simulace pro sítě velikosti od 9 do 121 uzlů. Pro větší sítě se protokolu nepodařilo dokončit lokalizaci. Kvůli rozlehlosti sítě narůstá počet opakovaných přenosů, navíc AODV pro každý

požadavek posílá všesměrově dotaz. Tyto dotazy dohromady se zprávami protokolu ARP a AFLOCAL zaplavující síť a jsou příčinou neúspěchu lokalizace.

6.2.1. Chyba lokalizace

Pro velikost sítě od 9 do 121 uzlů byla lokalizace úspěšná. Z dat vypsaných do konzole na konci simulace a vytvořené topologie na začátku nam souboru byly vyhodnoceny výsledky. Je třeba si uvědomit, že lokalizace neposkytuje souřadnice přímo odpovídající skutečnému rozmístění. Pokud totiž v průběhu hledání kotevních bodů měly některé stejnou rozhodovací váhu, nelze s jistotou předpovědět, který byl nakonec vybrán. Výsledné souřadnice tak mohou být oproti skutečnosti osově nebo středově přetočeny a pootočeny [7]. Také měřítko neodpovídá.

Transformací výsledných dat ale získáme skutečnou i změřenou síť ve stejných souřadnicích. Výsledné grafické srovnání je uvedeno jako Příloha 3 pro síť velikosti 25 uzlů. Znázornění skutečné a spočtené topologie do jednoho grafu je pro názornost jistě přínosná, nicméně pro porovnání výsledků u různých sítí nevhodná. Byl proto stanovena metodika výpočtu chyby lokalizace, která reprezentuje přesnost lokalizace ve vztahu k sousedním uzlům. To poskytuje lepší výsledky než prosté srovnání souřadnic [7].

Pro vyhodnocení přesnosti lokalizace každý uzel na konci simulace vypsal své souřadnice a průměrnou vzdálenost sousedních uzlu. Po transformaci souřadnic skutečné topologie do stejného měřítka a výpočtu skutečné průměrné vzdálenosti uzlu jsou tyto hodnoty porovnány. Je spočítána chyba pro každý jednotlivý uzel podle vzorce (6.01), kde $V_{AFLOCAL}$ je simulací změřená průměrná vzdálenost sousedů uzlu a $V_{skutečná}$ je vypočtená skutečná průměrná vzdálenost z transformované topologie.

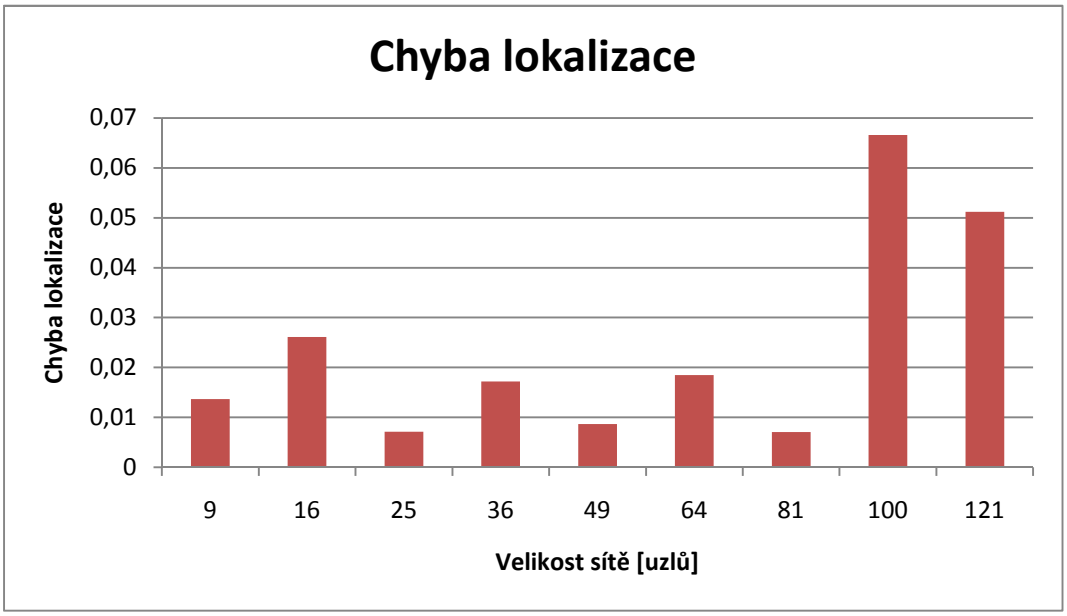
$$Err = \left(\frac{V_{AFLOCAL} - V_{skutečna}}{V_{skutečna}}\right)^2 \qquad 6.01$$

Celková chyba lokalizace je pak určena průměrem těchto chyb (6.02), kde *Err* je dílčí chyba jednotlivých uzlů a *N* je počet uzlů.

$$Chyba\ lokalizace = \frac{\sum_{i=1}^{N} \sum Err}{N} \qquad 6.02$$

Takto vypočtenou chybu lokalizace je pak možno srovnávat pro sítě různé velikosti. Algoritmus už ze své podstaty nemá vazbu na skutečné souřadnice, proto chyba lokalizace přímo nezohledňuje přesnost souřadnic, ale pracuje se vztahy mezi sousedními uzly.

Vypočtené celkové chyby lokalizace simulací pro velikost sítě od 9 do 121 uzlů jsou vyneseny do následujícího grafu (Obr.11.).



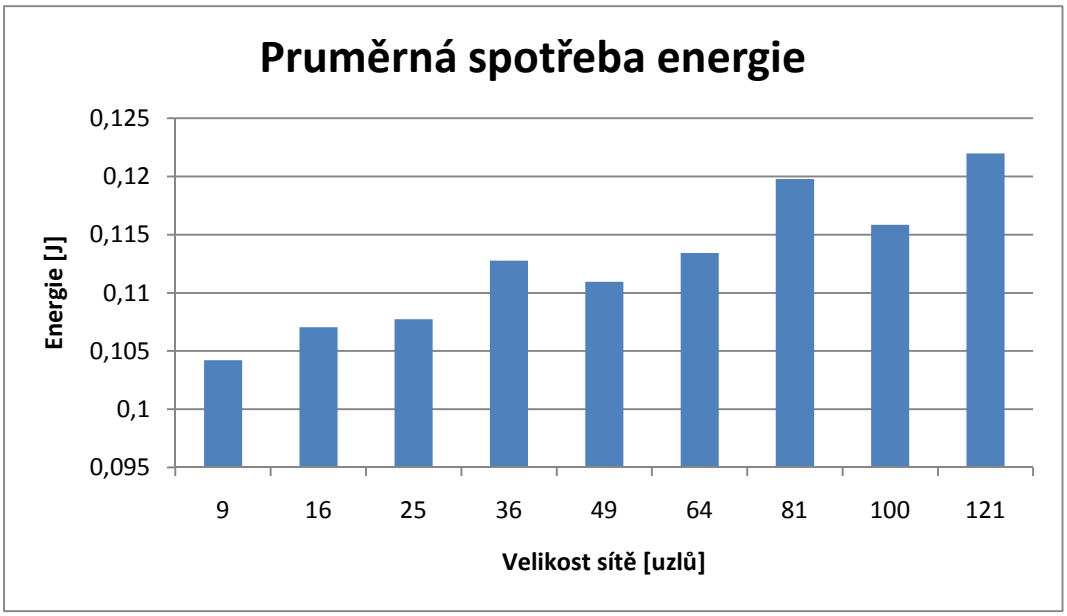
Obr.11. Graf závislosti chyby lokalizace na velikosti sítě

Z vypočtených hodnot průměrné chyby lokalizace je možno vysledovat střídající se menší a větší chyba lokalizace. To je způsobeno rozměry sítě, kdy při sudém počtu uzlů na délce mřížky (např. 36=6x6) neexistuje uzel v přesném středu. Středová kotva N5 pak neleží na pomyslné ose mezi kotvami N1, N2 a N3, N4. To způsobuje deformaci vypočtených souřadnic (Příloha 4.) a navýšení chyby lokalizace.

Dále je také patrný výrazný nárůst chyby lokalizace mezi velikostí sítě 81 a 100 uzlů. To je způsobeno větším zatížením sítě a s tím související větší pravděpodobností chybného přenosu. Z analýzy trasovacího souboru vyplývá, že počet zahozených paketů s rostoucí velikostí sítě roste (Příloha 6.).

6.2.2.Spotřeba energie

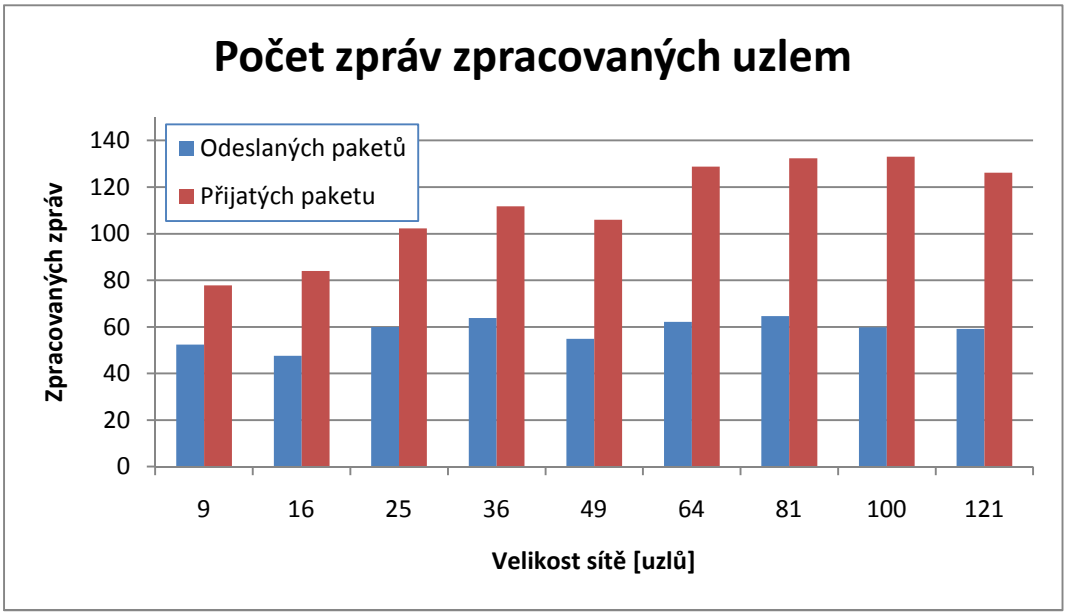
Dalším důležitým aspektem, který byl vyhodnocován v závislosti na velikosti sítě, je energetická náročnost. Jako počáteční energie každého uzlu byla nastavena hodnota 17280J. Je to energie odpovídající dvěma nabíjecím AA článkům kapacity 2000mAh (1kWh = 3600J). Na výsledek ale nastavení počáteční energie nemá vliv. Vyhodnocen je pouze úbytek, tedy spotřebovaná energie. Výsledky závislosti průměrné spotřeby uzlu na velikosti sítě pro čas ukončení lokalizačního procesu jsou zobrazeny na následujícím grafu (Obr.12.).



Obr.12. Průměrná spotřeba jednoho uzlu v závislosti na velikosti sítě

Z grafu je jasně vidět, že s rostoucí velikostí sítě stoupá i spotřeba uzlů. Hlavní příčinou ale je, že ve větší síti je větší ztrátovost paketů a tím i větší počet opakovaných přenosů. To se samozřejmě nepříznivě projeví na spotřebované energii. V souvislosti s opakováním přenosů roste i doba běhu lokalizace (Příloha 7.), což se také na spotřebě nepříznivě projeví.

Rostoucí počet opakovaných přenosů lze vysledovat z průměrného počtu zpracovaných paketů uzlem znázorněném na následujícím grafu (Obr.13.).



Obr.13. Počet zpráv zpracovaných uzlem při různé velikosti sítě

7. Závěr

S rostoucí popularitou bezdrátových senzorových sítí roste i oblast nasazení těchto sítí. Díky snadnému budování i rozsáhlých sítí, nízké ceně, vysoké spolehlivosti a energetické úspornosti umožňující dlouhodobou práci na baterie si nacházejí místo v mnoha zajímavých aplikacích. Jednou z nich je pak i lokalizace uzlu v rámci bezdrátové senzorové sítě. Umožňuje získat informaci o poloze jakéhokoliv uzlu sítě i bez použití GPS, které je nákladné a energeticky náročné. Navíc dokáže pracovat v uzavřených či zastavěných prostorách.

Lokalizace zařízení v bezdrátové senzorové síti je komplexní problém, zahrnující problematiku měření vzdálenosti uzlů, synchronizaci jednotlivých kroků a výpočet optimalizace polohy. Tento úkol nelze řešit lokálně na jednom uzlu, ale jako posloupnost dílčích operací na různých uzlech sítě. Lokalizační algoritmus Anchor-Free Localization (AFL) pracuje bez předem daných kotevních bodů, což zjednodušuje budování sítě a umožňuje snadnou expanzi.

Pro simulaci lokalizace uzlu ve WSN je do prostředí NS2 implementován nový protokol AFLOCAL. Byl vytvořen zcela nový aplikační agent AFLOCAL začleněný do struktury NS2. Byla definována struktura přenášené zprávy, metody a příkazy pro ovládání z Tcl skriptu. Po spuštění procesu příkazem „aflocal_start“ je spuštěn proces nalezení kotevních bodů. Tyto kotevní body jsou vybrány na základě předem daných kritérií. Všechny uzly sítě jsou také průběžně informovány o již vybraných kotevních bodech a ukládají si postupně jejich vzdálenosti. Po určení posledního z kotevních bodů každý uzel sítě vypočte svoje souřadnice. Ty určují jeho pozici v rámci sítě. Souřadnice ale nemají vazbu na skutečnou polohu sítě. Už z principu zvoleného algoritmu lokalizace žádný z uzlů nezná svou fyzickou pozici.

Původně zamýšlená optimalizace vypočtené polohy algoritmem Mass-Spring nebyla implementována. V prostředí NS2 není začleněna podpora pro měření parametrů rádiového spoje. Proto není možno změřit sílu přijímaného signálu RSSI a na jejím základě optimalizovat polohu.

Simulací nového protokolu v prostředí NS2 byly stanoveny jeho omezení. Protokol nepracuje správně v sítích s velkou hustotou uzlů, kdy dochází k velkému překrývání rádiových dosahů. Je to způsobeno větším provozem na rádiovém kanále, což vede k více kolizím. V takovémto prostředí nepracuje správně ani AODV směrovací protokol, jehož služby AFLOCAL protokol využívá pro unicastové přenosy. Opakování přenosů společně s AODV pakety zvyšuje celkový provoz a vede až k zahazování paketů. Problémem jsou také zprávy zahozené pro nenalezení cesty k cílovému uzlu směrovacím protokolem AODV. Vyhodnocením výsledků se jako optimální jeví rozestup uzlů 20 metrů při rádiovém dosahu 25 metrů.

Dále byla zkoumána závislost přesnosti lokalizace na velikosti sítě v rozmezí 9 až 121 uzlů. Byl definován výpočet pro chybu lokalizace, pomocí které je možno srovnávat sítě různých velikostí. Z obrázku 11 je vidět, že nejnižší chybu lokalizace vykazovala simulace pro 81 uzlů (Chyba lokalizace 0,007069334) následována sítí s 25 uzly (Chyba lokalizace 0,007132616). Mezi velikostí sítě 81 a 100 uzlů se projevil výrazný nárůst chyby lokalizace (z 0,007069334 na 0,066588748, tedy téměř 10×). Příčinu lze vysledovat i v nárůstu počtu zahozených paketů (Příloha 6.). Jedná se tedy o velikost sítě, při níž se výsledný provoz sítě (nárazově) blíží jejímu zahlcení.

Dalším důležitým vyhodnocovaným aspektem byla energetická náročnost. Průměrná spotřeba uzlu u různě velkých sítí byla od 0,1042075J u sítě s 9 uzly až po 0,1219722J u sítě s 121 uzly. Se vzrůstající velikostí sítě tedy rostla i spotřeba energie. Příčinou je množství zpracovaných zpráv (Obr.13.) a také doba trvání lokalizace (Příloha 7.). Lokalizace trvala od 62,54s u nejmenší sítě až po 71,42s u sítě největší.

Pro velké sítě nebo sítě s velkou hustotou uzlů se lokalizace potýká s problémy. Velký provoz způsobený dalšími protokoly společně s častým opakováním přenosů zahlcuje síť. Kolize jsou problémem pro všesměrové vysílání, kdy není možno zajistit zpětnou vazbu od příjemců potvrzením přenosu. Problém nastane zahozením všesměrového paketu při hello zprávě. Uzel tak nezjistí pravdivou vzdálenost ke kotevnímu bodu a následně spočtené souřadnice jsou zatížené chybou. Pro unicastové vysílání je problémem hlavně AODV směrovací protokol. Pokud se v důsledku provozu nepodaří směrovacímu protokolu nalézt cestu k cílovému uzlu, nelze zprávu odeslat.

Některé chyby by však mohla redukovat vhodně použitá metoda optimalizace. Měřením skutečných vzdáleností k sousedům na základě přijímané síly signálu RSSI a následná optimalizace popsaná v kapitole 4.3.4. si dokáže poradit s deformací sítě a přiblížit tak vypočtenou topologii té skutečné. V prostředí NS2 ale není implementována možnost měření RSSI ani obdobného parametru rádiového kanálu. Proto je tato možnost uváděná pouze jako teoretická.

8. Použité zdroje

[1] Wireless sensor network [online]. 2008 [cit. 2009-12-02]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Wireless_sensor_network>.

[2] *IEEE 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) Specification*. 2006. 322 s.

[3] *ZigBee* [online]. 2008 , 9.11.2009 [cit. 2009-12-07]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/ZigBee>.

[4] KINNEY, Patrick. *ZigBee Technology: Wireless Control that Simply Works* [online]. c2009 [cit. 2009-12-08]. Dostupný z WWW: <http://www.zigbee.org/LearnMore/WhitePapers/tabid/257/Default.aspx>.

[5] BRADÁČ, Zdeněk. Bezdrátový komunikační standard ZigBee. *Automatizace* [online]. 2005, roč. 48, č. 4 [cit. 2009-12-07]. Dostupný z WWW:<http://www.automatizace.cz/article.php?a=638>.

[6] J. Koton, P. Číka,V. Křivánek . *Standard nízkorychlostní bezdrátové komunikace ZigBee* [online]. 2006 [cit. 2009-12-07]. Dostupný z WWW: <http://access.feld.cvut.cz/view.php?cisloclanku=2006032001>.

[7] XU, Yurong. *Anchor-Free Localization in Mixed Wireless Sensor Network Systems* [online]. 2008 [cit. 2009-12-10]. 172s. Dostupný z WWW: <www.cs.dartmouth.edu/reports/TR2008-626.pdf>.

[8] *Ns (simulator)* [online]. 2009 [cit. 2009-12-12]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Ns_(simulator)>

[9] *The Network Simulator - ns-2* [online]. 2006 [cit. 2009-12-13]. Dostupný z WWW: <http://www.isi.edu/nsnam/ns/>.

[10] Nissanka B. Priyantha, Hari Balakrishnan, Erik Demaine, and Seth Teller. *Anchor-Free Distributed Localization in Sensor Networks* [online]. 2003 [cit. 2009-12-11]. Dostupný z WWW: <cricket.csail.mit.edu/papers/TechReport892.pdf>.

[11] Francisco J. Ros, Pedro M. Ruiz. *Implementing a New Manet Unicast Routing Protocol in NS2*. 2004 [cit. 2010-02-12]. Dostupný s WWW: <http://masimum.dif.um.es/nsrt-howto/pdf/nsrt-howto.pdf>

[12] Elmurod Abduvoyitovich Talipov. *Sensor Network 802.15.4 AODV Simulation* [online]. 2009 [cit. 2010-05-07]. Dostupný z <http://elmurod.net/wps/?p=72>

[13] *Ad hoc On-Demand Distance Vector Routing* [online]. [cit. 2010-05-05]. Dostupný z WWW <http://en.wikipedia.org/wiki/Ad_hoc_On-Demand_Distance_Vector_Routing>

9. Přehled použitých zkratek a symbolů

ABL	Anchor-Based Localization
ACK	acknowledgement
AFL	Anchor-Free Localization
AOA	Angle-of-Arrival
AODV	Ad hoc On-Demand Distance Vector Routing
ARP	Address Resolution Protocol
ASK	Amplitude-Shift Keying
BPSK	Binary Phase Shift Keying
CAP	Contention Access Period
CFP	Contention Free Period
CSMA-CA	Carrier sense multiple access with collision avoidance
DSSS	Direct-sequence spread spectrum
FFD	Full Functional Device
GNU GPL	GNU General Public License
GPS	Global Positioning Systém
GTS	Guaranteed Time Slot
IEEE	Institute of Electrical and Electronics Engineers
NAM	Network Animator
NS2	Network Simulator 2
O-QPSK	Offset Quadrature Phase-Shift Keying
PAN	Personal area network
PSSS	Parallel Sequence Spread Spectrum
RFD	Reduced Functionality Device
RI	Radio Interferometry
RSSI	Received signal strength indicator
TOA	Time of Arrival
WPAN	Wireless Personal Area Network
WSN	Wireless sensor network

10. Seznam příloh

Příloha 1. Tabulka stavových kódů uzlu a časovače

Příloha 2. Tabulka kódů zpráv protokolu AFLOCAL

Příloha 3. Zobrazení skutečné a změřené topologie sítě velikosti 25 uzlů

Příloha 4. Zobrazení skutečné a změřené topologie sítě velikosti 36 uzlů

Příloha 5. Závislost ztrátovosti na rozestupu uzlů

Příloha 6. Závislost počtu zahozených paketů na velikosti sítě

Příloha 7. Délka lokalizačního procesu pro různé velikosti sítě

Příloha 8. Obsah přiloženého CD

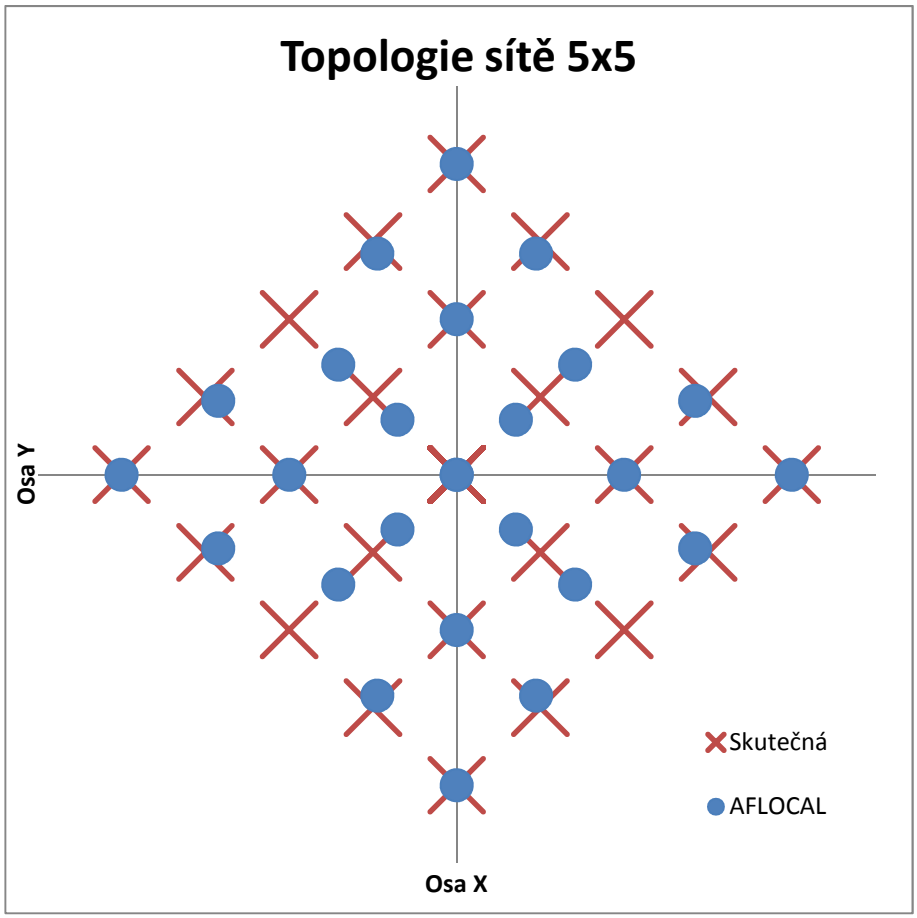
11. Přílohy

Příloha 1. Tabulka stavových kódů uzlu a časovače

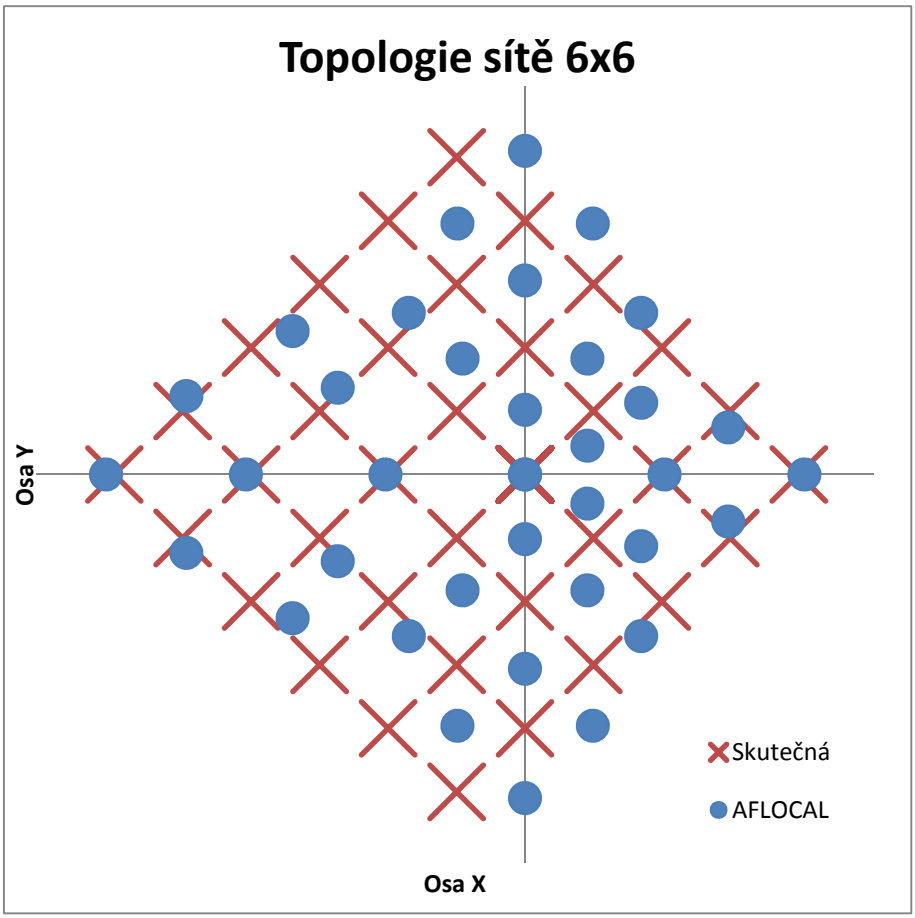
Kód	Hodnota	Popis
NS_N0	0	Uzel s rolí kotvy 0
NS_N1	1	Uzel s rolí kotvy 1
NS_N2	2	Uzel s rolí kotvy 2
NS_N3	3	Uzel s rolí kotvy 3
NS_N4	4	Uzel s rolí kotvy 4
NS_N5	5	Uzel s rolí kotvy 5
NS_CANDIDATE_N1	11	Kód pro nastavení časovače pro kandidáta na uzel N1
NS_CANDIDATE_N2	12	Kód pro nastavení časovače pro kandidáta na uzel N2
NS_CANDIDATE_N3	13	Kód pro nastavení časovače pro kandidáta na uzel N3
NS_CANDIDATE_N4	14	Kód pro nastavení časovače pro kandidáta na uzel N4
NS_CANDIDATE_N5	15	Kód pro nastavení časovače pro kandidáta na uzel N5
NS_VYPOCET_SOURADNIC	16	Kód pro nastavení časovače pro výpočet souřadnic
NS_HELLO_REPLY_N0	20	Kód pro nastavení časovače pro odpověď uzlu N0
NS_HELLO_REPLY_N1	21	Kód pro nastavení časovače pro odpověď uzlu N1
NS_HELLO_REPLY_N2	22	Kód pro nastavení časovače pro odpověď uzlu N2
NS_HELLO_REPLY_N3	23	Kód pro nastavení časovače pro odpověď uzlu N3
NS_HELLO_REPLY_N4	24	Kód pro nastavení časovače pro odpověď uzlu N4
NS_HELLO_REPLY_N5	25	Kód pro nastavení časovače pro odpověď uzlu N5
NS_WAIT_ACK_N0	30	Kód pro nastavení časovače čekání na potvrzení od uzlu N1
NS_WAIT_ACK_N1	31	Kód pro nastavení časovače čekání na potvrzení od uzlu N2
NS_WAIT_ACK_N2	32	Kód pro nastavení časovače čekání na potvrzení od uzlu N3
NS_WAIT_ACK_N3	33	Kód pro nastavení časovače čekání na potvrzení od uzlu N4
NS_WAIT_ACK_N4	34	Kód pro nastavení časovače čekání na potvrzení od uzlu N5
NS_WAIT_ACK_N5	35	Kód pro nastavení časovače čekání na potvrzení od uzlu N6
NS_OPT_SEND	60	Kód pro nastavení časovače pro odeslání souřadnic sousedním uzlům
NS_OPT_WAIT	61	Kód pro nastavení časovače čekání na souřadnice sousedů
NS_NODE	99	Uzel s rolí obecného uzlu (výchozí)

Příloha 2. Tabulka kódů zpráv protokolu AFLOCAL

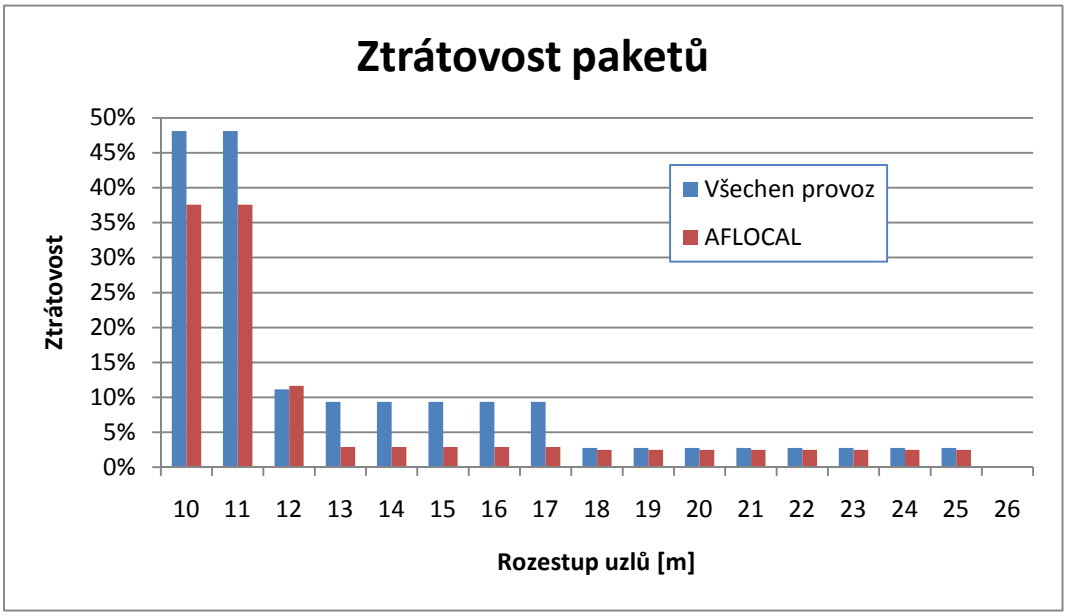
Kód	Hodnota	Popis
AC_HELLO_SRC_N0	100	Hello zpráva od uzlu N0
AC_HELLO_SRC_N1	110	Hello zpráva od uzlu N1
AC_COMMAND_SET_N1	119	Řídící zpráva pověřující uzel do role N1
AC_HELLO_SRC_N2	120	Hello zpráva od uzlu N2
AC_COMMAND_SET_N2	129	Řídící zpráva pověřující uzel do role N2
AC_HELLO_SRC_N3	130	Hello zpráva od uzlu N3
AC_COMMAND_SET_N3	139	Řídící zpráva pověřující uzel do role N3
AC_HELLO_SRC_N4	140	Hello zpráva od uzlu N4
AC_COMMAND_SET_N4	149	Řídící zpráva pověřující uzel do role N4
AC_HELLO_SRC_N5	150	Hello zpráva od uzlu N5
AC_COMMAND_SET_N5	159	Řídící zpráva pověřující uzel do role N5
AC_HELLO_NEIGHBOR	160	Zpráva pro sousední uzly, datové pole zprávy jsou příjemcem uloženy
AC_DATA	190	Datová zpráva pro kotevní uzel
AC_DATA_BROADCAST	195	Datová zpráva pro kotevní uzel přenášená všesměrově
AC_DATA_ACK	199	Potvrzovací zpráva datové zprávy



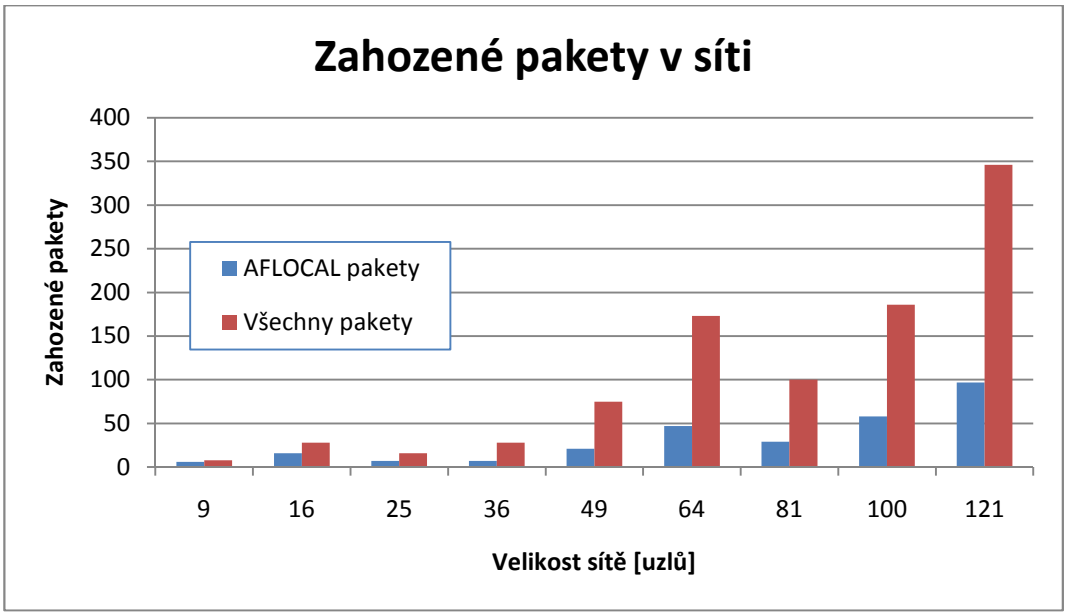
Příloha 3. Zobrazení skutečné a změřené topologie sítě velikosti 25 uzlů



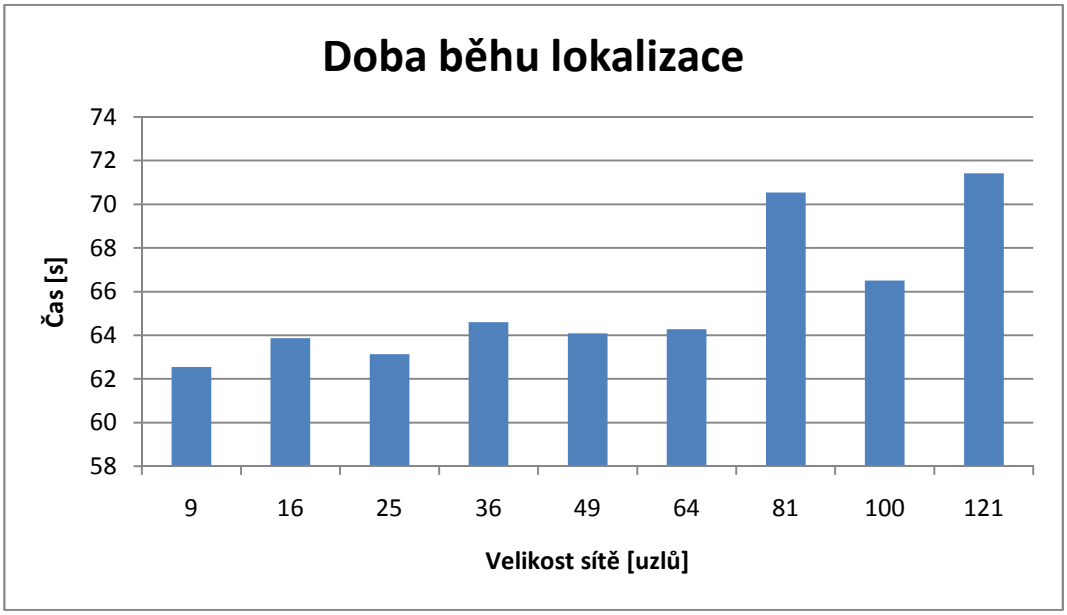
Příloha 4. Zobrazení skutečné a změřené topologie sítě velikosti 36 uzlů



Příloha 5. Ztrátovost paketů v závislosti na rozestupu uzlů



Příloha 6. Závislost počtu zahozených paketů na velikosti sítě



Příloha 7. Délka lokalizačního procesu pro různé velikosti sítě

Příloha 8. Obsah přiloženého CD

\Diplomova_prace_elektronicka_verze\DP_Martynek_Tomas.pdf

\Tcl_script\aflocal.tcl

\zdrojove_kody\aflocal\aflocal.cc

\zdrojove_kody\aflocal\aflocal.h

\zdrojove_kody\aflocal\aflocal_pkt.h

\zdrojove_kody\aflocal\aflocal_table.cc

\zdrojove_kody\aflocal\aflocal_table.h

\zdrojove_kody\common\packet.h

\zdrojove_kody\tcl\lib\ns-packet.tcl

\zdrojove_kody\trace\cmu-trace.cc

\zdrojove_kody\trace\cmu-trace.h

\zdrojove_kody\Makefile