VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

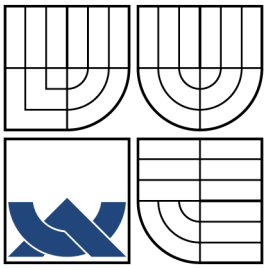
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÍ BEZDRÁTOVÝCH SENZOROVÝCH SÍTÍ

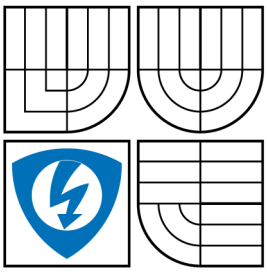
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ MAŇÁK
AUTHOR

BRNO 2009



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

ZABEZPEČENÍ BEZDRÁTOVÝCH SENZOROVÝCH SÍTÍ
SECURITY IN WIRELESS SENSOR NETWORKS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ MAŇÁK
AUTHOR

VEDOUCÍ PRÁCE — ING. MILAN ŠIMEK
SUPERVISOR

BRNO 2009

ZDE VLOŽIT LIST ZADÁNÍ

Z důvodu správného číslování stránek

ZDE VLOŽIT PRVNÍ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ZDE VLOŽIT DRUHÝ LIST LICENČNÍ SMOUVY

Z důvodu správného číslování stránek

ABSTRAKT

Tato práce pojednává o návrhu simulačního modelu bezdrátové senzorové sítě v prostředí Castalia se zaměřením na energetickou spotřebu senzorových uzlů za použití komunikačních jednotek Mica2 a MicaZ. Popisuje základní práci se simulačním programem Castalia včetně jeho instalace. Dále pojednává o zabezpečení bezdrátových senzorových sítí a o návrhu jejich zabezpečení.

KLÍČOVÁ SLOVA

bezdrátová senzorová síť, senzorový uzel, základová stanice, zabezpečení, šifrování

ABSTRACT

This bachelor thesis deals with the proposal of simulation model for wireless sensor networks in program Castalia with a focus on energy consumption of sensor nodes using the communication units Mica2 and MicaZ. It describes the basic work with simulation program Castalia, including its installation. Furthermore, deals with the security of wireless sensor networks and theirs security proposal.

KEYWORDS

wireless senzor network, senzor node, base station, security, cryptography

MAŇÁK J. *Zabezpečení bezdrátových senzorových sítí*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 61 s., Počet stran příloh 4 s. příloh. Vedoucí bakalářské práce Ing. Milan Šimek.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Zabezpečení bezdrátových senzorových sítí“ jsem vypracoval samostatně pod vedením vedoucího bakalářské 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 bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

(podpis autora)

PODĚKOVÁNÍ

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

V Brně dne ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Bezdrátové senzorové sítě jsou novou rychle se rozvíjející oblastí v monitorování a získávání informací o okolním prostředí. Potřeba jejich zabezpečení je velmi důležitá, tak jako u ostatních sítí, zejména proto, že při monitorování se jedná o citlivá a data náchylná na zneužití. Zabezpečení je o to komplikovanější, jelikož se jedná o bezdrátovou komunikaci, kde může přenos kdokoliv odposlouchávat či se do něj vložit.

V první části této práce je pojednáno o základních vlastnostech bezdrátových senzorových sítí, o jejich struktuře, využití a základních požadavcích na její jednotlivé části. Jsou zde uvedeny používané operační systémy, programovací jazyky a simulační nástroje pro bezdrátové senzorové sítě.

V další části je pojednáno o problematice zabezpečení bezdrátových senzorových sítí. Jsou zde uvedeny základní šifrovací metody a metody využívající se při zabezpečení senzorových sítí spolu se stručným přehledem možných útoků na tuto síť. Je zde popsána funkce zabezpečovacího protokolu SPINS s ukázkou jeho energetických nákladů. Ke konci této části je uveden návrh zabezpečení senzorové sítě z hlediska šifrovací metody a komunikační režie.

V poslední části práce seznamuje se simulačním programem Castalia, ve kterém je navržen model bezdrátové senzorové sítě, na které jsou provedeny tři experimenty se zaměřením na porovnání energetických nároků jednotlivých senzorových uzlů při použití komunikačních jednotek MicaZ a Mica2.

Jelikož patří program Castalia mezi méně známé simulační programy, je na konci práce v příloze A podrobně rozepsán postup instalace simulačních programů Oment++ a Castalia tak, aby ji zvládl i méně zkušený uživatel.

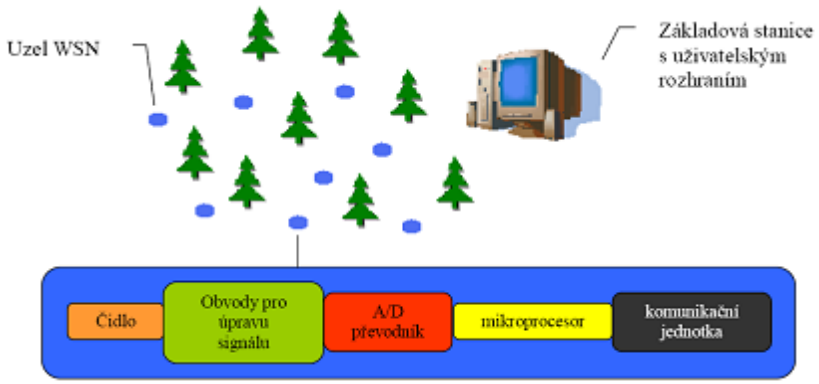
1 BEZDRÁTOVÉ SENZOROVÉ SÍTĚ

1.1 Úvod

Bezdrátové senzorové sítě se skládají z prostorově distribuovaných autonomních zařízení používajících čidla pro monitorování různých parametrů (teploty, tlaku, zvuku, vibrací či pohybu atd.) okolního prostředí, ve kterém se nacházejí. Senzorové sítě byly původně vyvíjeny pro vojenské účely k monitorování bojiště. V dnešní době jsou WSN (wireless sensor network) využívány v mnoha civilních oblastech, například k monitorování životního prostředí, ve zdravotnictví, v domácí automatizaci apod.[18]

Senzorová síť patří do tzv. mesh sítí, což jsou bezdrátové sítě skládající se z velkého počtu zařízení (stovky, tisíce). V případě senzorových sítí jsou to stovky až tisíce (podle rozsáhlosti sítě) senzorových uzlů a jedna základová stanice. Senzorové uzly WSN mezi sebou navzájem komunikují a pořizují jednotlivé informace o daném zkoumaném prostředí. Tyto informace jsou posílány do základové stanice, v které jsou vyhodnoceny a dočasně ukládány. Základová stanice také slouží jako brána mezi uzly a koncovým uživatelem.

Každý uzel se skládá z jednoho nebo více senzorů tzv. smart senzorů (senzorových čidel), bezdrátového komunikačního zařízení (komunikační jednotkou), mikroprocesoru a samozřejmě zdroje energie, kterým obvykle bývá akumulátor.[18] Obecně lze senzor charakterizovat jako zařízení, které je v přímém styku s měřenou veličinou, kterou převádí (ve většině případů) na elektrický signál. Citlivá část senzoru je označována jako čidlo. Smart senzor neboli inteligentní senzor, je potom zařízení, které se převážně skládá z čidla, obvodů pro úpravu, zpracování a analýzu signálu, A/D převodníku a obvodů pro obousměrnou komunikaci s okolním prostředím.[12]



Obr. 1.1: Schéma bezdrátové senzorové sítě

1.2 Základní požadavky na senzorový uzel

Senzorové uzly jsou malé a nízkonákladové zařízení. Proto je jejich návrh poměrně složitý. Hlavními problémy jsou výkon uzlů a kapacita akumulátoru. Vzhledem k tomu, že prioritou při vývoji je minimalizace nákladů, velikosti a spotřeby elektrické energie, existuje jen malá šance, že dojde ve vývoji k výraznému zvyšování výpočetního výkonu a paměti.

Životnost senzorového uzlu je dána kapacitou akumulátoru. Ta často nejde měnit vůbec nebo je její výměna určitým způsobem omezená, proto tedy doba životnosti a použitelnosti senzorového uzlu v síti závisí i na kapacitě akumulátoru a spotřebě senzorového uzlu. Spotřeba senzorového uzlu je spjata s funkcí, kterou senzorový uzel vykonává, tedy s jeho výpočetními algoritmy, které jsou v něm implementovány. Dalším spotřebičem energie v senzorovém uzlu je komunikační zařízení, tedy vysílací a přijímací obvody a anténa, která patří mezi největší konzumenty kapacity akumulátoru. Komunikace mezi uzly je tedy z hlediska energie velice nákladná, proto musí brát dané algoritmy tuto skutečnost v potaz. Akumulátor je jednou z největších částí samotného uzlu. Velikost uzlu je tedy přímo úměrná kapacitě akumulátoru, kterou využívá.[8]

Od senzorových sítí se požaduje, aby se rychle a snadno instalovaly a udržovaly. Mezi žádoucí účinky pro senzorové uzly patří: jednoduchá instalace, samostatná identifikace, samostatná diagnóza, spolehlivost, časová připravenost a koordinace s dalšími uzly, některé softwarové funkce a DSP (digitální signální procesor nebo také digitální signálový procesor), standardní kontrolní protokoly a síťové rozhraní.[9]

DSP je procesor optimalizovaný pro algoritmy používané při zpracování digitálně reprezentovaných signálů, kde hlavním nárokem na systém bývá průběžné vyhodnocování velkého množství dat zpracovávaných procesorem.[18]

Samostatná organizace v ad hoc sítích (sítě skládající se ze rovnocenných zařízení, které komunikují každý s každým) zahrnuje obě komunikace, a to samoorganizaci a polohovou samoorganizaci. V první řadě se musejí senzorové uzly spustit, detekovat všechny ostatní a vytvořit komunikační síť.[9]

Dalšími požadavky jsou např. odolnost vůči přírodním podmínkám, provoz na rozsáhlých plochách, bezobslužný provoz, mobilita, dynamická topologie sítí, různorodost uzlů a taky schopnost uzlu pracovat při selhání ostatních uzlů.

1.3 Využití bezdrátových senzorových sítí

WSN slouží převážně k dozoru (pozorování) a ke sledování např. ve zdravotnictví [13]. U pozorování se předpokládá s mnohonásobně propojenými senzory (akustické, seismické, video) rozloženými po celé oblasti, např. bojiště. Navrhovaná pozorovací

aplikace může být použitá na tuto senzorovou síť, která bude poskytovat informace o daném prostředí koncovému uživateli. V takovéto senzorové síti, kde je komunikační model tzv. „many-to-one“ (mnoho s jedním), se rozsah zkoumaných dat může pohybovat od hrubých senzorických dat až po velice kvalitní popis zkoumaného prostředí. Aplikace má mít určité Qos požadavky ze senzorové sítě, požaduje minimální procentuální senzorové pokrytí v oblasti, kde se předpokládá výskyt daného jevu, nebo požaduje maximální pravděpodobnost zjištěných ztracených událostí. Zároveň se od sítě očekává, že bude tyto informace o kvalitě služby poskytovat po dobu měsíců až let, bez nutnosti vnějších zásahů. Pro splnění těchto vlastností je důležitý návrh senzoru, jak z hlediska hardwaru tak i softwaru.

Ve zdravotnickém okruhu jde především o monitorování pacientů, o odstranění pacienty omezujících, velikých a starých „drátových“ sledovacích zařízení, dále o monitorování obětí hromadných dopravních nehod, monitorování při běžném životě k zjištění a poskytnutí včasných preventivních opatření proti různým nemocem. V tomto využití jsou senzory odlišné, od miniaturních a tělonosných senzorů až po vnější senzory jako jsou video kamery a polohovací zařízení. Jedná se o náročnou oblast, kde spolehlivá a flexibilní aplikace musí být navržena tak, aby data zaznamenaná senzorem byla použita jako vstupní data pro danou aplikaci.

Většina senzorů používaných v osobních zdravotních monitorovacích zařízení, např. PDA s aplikací vyhodnocující EKG, EMG, tlak, pulz, krevní oběh, atd., je napájena akumulátory a komunikuje bezdrátově, je zřejmé, že daná aplikace vyžaduje účinné, bezpečné a spolehlivé síťové protokoly.

1.4 Standardy WSN

Kvůli velkému počtu výrobců senzorů a sítí na trhu začal v roce 1993 IEEE (Institut pro elektrotechnické a elektronické inženýrství) a NIST (americký Národní úřad pro standardy a technologie) pracovat na standardu pro inteligentní senzorové sítě. Výsledkem byl standard IEEE 1451. Účelem tohoto standardu bylo zjednodušit odlišným výrobcům vývoj inteligentních senzorů a rozhraní těchto zařízení do sítě [9]. Některé standardy se v současné době buď upravily, nebo zcela vyvinuly speciálně pro senzorové sítě. Jedním takovým je standard IEEE 802.15.4., na jehož základě je vystavěna bezdrátová komunikační technologie ZigBee, která patří do Mesh sítí. Podle literatury [17] je výhodou Mesh sítí jejich schopnost samostatné organizace a opravy, což znamená, že každý komunikující uzel „hlídá“ svoje nejbližší sousední uzly, s nimiž vytváří přenosové cesty a linky, měří sílu RF (radio frequency) signálu, získává informaci o synchronizaci a přeskakování frekvencí (frequency hopping) atd. Každý uzel má poté schopnost směrovat provoz od svých nejbližších sousedů na

základě vzájemných RF spojení a výkonnosti sítě. Tento princip vytváří dynamicky rozšířitelnou, jednoduchou komunikační síť s možností snadné rozšíření. Standard IEEE 1451 v sobě zahrnuje několik skupin a to 1451.0 – 1451.6.[19]

1.5 Operační systémy

Operační systémy pro bezdrátové senzorové sítě jsou obvykle méně komplikované než obecné operační systémy. Zejména proto, že operační systém nemusí obsahovat podporu uživatelského rozhraní, také je poměrně zbytečné, aby obsahoval mechanizmy jako např. virtuální paměti, či mapování paměti už jen z hlediska hardwarového řešení uzlů a také aplikace senzorových sítí nejsou stejně interaktivní jako u aplikací na PC.

Hardware bezdrátových senzorových sítí není odlišný od tradičních vestavěných systémů, a proto je možné využít tradiční operační systémy, jako ECOS nebo uC/OS. Nicméně, tyto operační systémy jsou často navrženy s podporou real-time, kterou senzorové sítě nepodporují.[18]

Mezi první operační systém pro WSN patří TinyOS, který vytvořila společnost Intel ve spoluprácí s vědci z Kalifornské univerzity v Berkeley. Ti vytvořili tzv. open source operační systém s názvem TinyOS. Ten senzorovým uzlům a celkovým senzorovým sítím umožňuje podávat shrnující výtahy ze získaných dat či různě tříděné informace. Operační systém TinyOS je napsán v programovacím jazyce nesC, který je rozšířením programovacího jazyka C.

Existují také operační systémy, které umožňují programování v C. Jsou to například SOS či Nano-RK.[18]

1.6 Programovací jazyky

Programování senzorových uzlů je oproti běžným počítačovým systémům obtížnější. Programování pouze uzlů dává možnost vzniku novým programovým modelům, většina je v současné době vytvářena v programovacím jazyce C. Programovacími jazyky jsou tedy: c @ t (Computation at a point in space (@) Time), DCL (Distributed Compositional Language), galsC, nesC, Protothreads, SNACK, SNAPpy (Python), SQTL, Java Sun SPOT. [18]

1.7 Simulační nástroje pro WSN

Existují platformy speciálně navrženy tak, aby simulovaly komunikaci ve WSN, jako např. TOSSIM, která plně simuluje TinyOS či síťový simulační nástroj NS-2 a

J-Sim[18]. Dalším simulačním nástrojem, je simulační program Castalia pracující na základě programu Omnet++.

Omnet je objektově orientované simulační prostředí s veřejným zdrojem, modulační a otevřenou architekturou, založené na komponentech, se silnou podporou GUI a s vloženým simulačním jádrem [10, 11]. Jeho základní využití je v simulaci komunikačních sítích a díky jeho všeobecné a flexibilní architektuře je používán také v ostatních oblastech jako simulace IT systémů, modelování telekomunikačních sítí, protokolů, jednotlivých úloh sítě, internetové simulace (IP, IPv6), mobilní a ad-hoc simulace, atd..

Castalia je simulátor pro bezdrátové senzorové sítě a podobné síťové systémy, založený na platformě Oment++, sloužící převážně pro testování algoritmů a protokolů vyvíjených pro bezdrátové kanály a rádiové (přijímač – vysílač) modely s reálným chováním uzlů zaměřené zejména na rádiové spojení [3, 1]. Program Castalia má pokročilý kanálový a vysílací model založený na empiricky měřených datech, vysoce flexibilní model fyzických procesů, obsahuje simulační zařízení pro měření rušivých signálu, zkreslení a spotřeby el. energie nejen jednotlivých uzlů, ale také mikroprocesoru, obsahuje Medium Access Control protokol s laditelnými parametry, je určen pro adaptaci a rozvoj. Není typickou senzorovou platformou, poskytuje spolehlivý a reálný systém a je určen pro prvotní ověření spolehlivosti algoritmů před jejich zavedením do reálné senzorové sítě [1].

2 ZABEZPEČENÍ V WSNS

2.1 Úvod

Potřeba zabezpečit senzorovou síť, jakož i ostatní sítě, je velmi důležitá, nejenom proto, aby námi měřená data nemohl nikdo odposlouchávat a popř. zneužít, ale také, aby nikdo do naší sítě nevkládal falešné informace. Například při monitorování ve zdravotnictví – stav pacienta, jde o velmi citlivá a důvěryhodná data, kde každá nepravdivá informace by mohla způsobit mnohé následky.

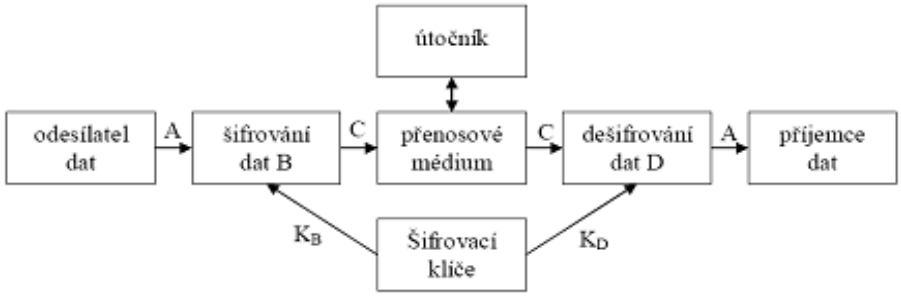
Řešení zabezpečení v bezdrátových sítích není stejné u počítačových sítí. Vyskytuje se zde spousta problému a omezení např. senzorové uzly mají omezenou šířku pásma (10–250Kb/s [14, 6]), operační pamětí (10–100 Kbytu s 8-bitovým mikrokontrolérem jako CPU [6]), ukládací pamětí (100–1000 Kbytu [6]) a v neposlední řadě jsou omezeny zdrojem energie. Vyskytují se zde napadení nejen z řad odposlouchávání a úpravy přenášených dat, ale také rizika, že při měření určité oblasti může být uzel zničen či nahrazen (např. budeme měřit koncentraci zajíců na zelném poli, může se stát, že nám nějaký senzor sežere sám zajíc nebo konkurence v pěstování zelí nám vymění část uzlů, abychom si mysleli, že nám pole zajíci neožírají). Z důvodů těchto omezení jsou v WSN některé zabezpečovací metody nepoužitelné. V senzorové bezdrátové síti jde o zabezpečení jejich částí, tedy základové stanice a jednotlivých uzlů. Také zabezpečení komunikace mezi uzly samotnými a uzly a základovou stanicí.

Základová stanice je vstupní branou pro komunikaci uzlů s okolním světem, bude-li ohrožena, může být daná senzorová síť nepoužitelná. Proto je zabezpečení základových stanic velmi důležité. Má podobné vlastnosti jako uzel, navíc má dostatečné napájení, aby svou životností předčila všechny ostatní uzly, také má dostatečnou paměť na uchovávání šifrovacích klíčů. Jednotlivé uzly jsou na tom hůře, jak je uvedeno v předchozím odstavci.

2.2 Základy šifrování

Jako základním bezpečnostním prvkem je všeobecně nazýváno šifrování nebo-li kódování. Chceme-li předat někomu zprávu tak, aby ji třetí strana nerozuměla, musíme ji určitým způsobem zašifrovat (zakódovat) a následně tomu druhému poskytnout tzv. klíč k poslané zprávě, aby ji mohl dešifrovat a porozumět. Čím složitější je pak šifrování, tím je menší pravděpodobnost zjištění obsahu zprávy třetí stranou.

Při zabezpečení sítí se používá kryptografických algoritmů, což jsou matematické funkce, kdy zprávy jsou šifrovány klíčem skládajícího se z různých symbolů. Celý



Obr. 2.1: Schéma šifrovacího systému. (A – data; B – šifrování pomocí šifrovacího klíče K_B; C – zašifrovaná data $C = B(A, K_B)$; D – dešifrování pomocí šifrovacího klíče K_D; A – dešifrovaná data $A = D(C, K_D)$)

šifrovací systém je znázorněn na obr. 2.1. Odesílatel odešle nezabezpečená data A příjemci, ta jsou zašifrována šifrovací algoritmem B pomocí šifrovacího klíče K_B. Šifrovaná data jsou označována C a mají tvar:

$$C = B(A, K_B). \tag{2.1}$$

Poté jsou data přenášena přes přenosové médium (síťový kabel, vzduch), kde by je útočník mohl odposlouchat, popřípadě modifikovat. Před přijetím zašifrovaných dat C příjemcem jsou data dešifrována dešifrovacím algoritmem D pomocí dešifrovacího klíče K_D. Dešifrovaná data mají tvar:

$$A = D(C, K_D). \tag{2.2}$$

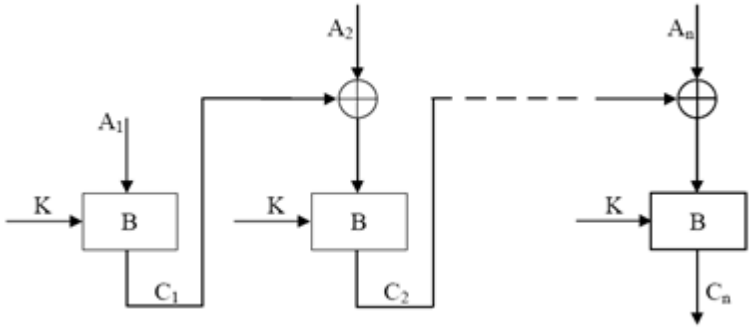
Šifrovací klíče K_B a K_D mohou být, podle druhu šifrování, stejné (K_B=K_D) nebo různé ($K_B \neq K_D$). Jsou-li oba klíče stejné, jedná se o symetrický kryptografický systém. Jsou-li klíče různé, jedná se o asymetrický kryptografický systém [2, 7, 16].

Symetrická kryptografie

Symetrický kryptografický systém používá k šifrování a dešifrování stejný klíč. Tento klíč zná jak odesílatel, tak i příjemce dat. Symetrické kryptografické algoritmy můžeme rozdělit na dva druhy – blokové a proudové [7, 16].

- V blokových symetrických systémech jsou data šifrována po blocích s konstantní délkou. Délka šifry a délka dat jsou shodné, také velikost šifrovacího klíče je rovna velikosti datového bloku. Př.: MAC (message authentication code), obr. 2.2. Informace je rozdělená do bloků, každý následující blok je před šifrováním zpracován šifrou předchozího bloku.

- V proudových symetrických systémech jsou data šifrována po jednotlivých bitech.



Obr. 2.2: Symetrický blokový algoritmus MAC. (A – datové bloky, B – šifrátor, K – klíč, C – šifry)

Symetrický kryptografický systém se používá k šifrování velkých objemů dat. Je výpočetně méně náročný (rychlejší než asymetrický). Nevýhodou je dvojí sdílení šifrovacího klíče, není vhodný pro rozsáhlejší sítě, protože je potřeba 2^n klíčů, kde n značí počet mezi sebou komunikujících stanic.

Asymetrická kryptografie

Asymetrický kryptografický systém používá k šifrování a dešifrování dat dva odlišné klíče – veřejný a tajný. Veřejný klíč je volně přístupný, tajný klíč je uchován v tajnosti. Podle toho, jestli je tajný klíč uchován v přijímači nebo vysílači, zajišťuje asymetrický kryptografický systém důvěrnost nebo autentičnost [2]. Jestliže má tajný klíč příjemce dat, znamená to, že dešifrovat data může pouze on, tím je zajištěno, že si daná data dešifruje opravdu jen příjemce. Jestli že má tajný klíč vysílač, znamená to, že data může zašifrovat pouze on, tím je zaručena autorizace (tzv. digitální podpis).

Asymetrický kryptografický systém se používá k šifrování malých objemů dat. Mají jednodušší distribuci klíčů. Nevýhodou je jejich výpočetní náročnost (pomalé šifrování a dešifrování).

2.3 Metody šifrování v WSN

V senzorových sítích se vyskytují odlišné návrhy správy klíčů, které jsou navrhovány podle bezpečnosti klasických počítačových sítí [15].

Symetrický klíč

Nejjednodušším návrhem je sdílení jednoho symetrického klíče pro celou senzorovou síť. Tento klíč zajišťuje zabezpečení komunikační linky a je znám každému uzlu,

proto je distribuce klíče v tomto návrhu téměř nepřítomná, tím je minimalizována spotřeba energie potřebná na komunikační režii. Bohužel toto řešení není nejbezpečnější, protože útočník může získat klíč zachycením komunikace na jednom uzlu.

Master key - Link key

Dalším návrhem je užívání dvou klíčů, a to Master key (univerzální klíč) a Link key (spojovací klíč). V této síti slouží univerzální klíč k nastavení zabezpečení komunikačních linek mezi jednotlivými uzly pomocí spojovacích klíčů. Každý uzel pomocí univerzálního klíče nastaví sadu spojovacích klíčů, která odpovídá komunikačním linkám mezí ostatními uzly. Po zavedení tohoto zabezpečení jsou univerzální klíče vymazány ze všech uzlů, tím se zamezí útoku získání klíče z jednotlivého uzlu. Nevýhodou těchto sítí je, že nelze přidat další uzly, jelikož je univerzální klíč vymazán, a tedy nové uzly se nezačlení do sítě.

Veřejné šifrování

Ne příliš vhodné pro senzorové sítě je veřejné šifrování (asymetrické kryptografické algoritmy). Operační paměť senzorového uzlu je příliš malá (10-100 Kbytu [6]) pro náročné výpočetní úkony a pro udržení dostatečně velkých proměnných pro zajištění bezpečnosti, které se používají těchto v šifrovacích funkcích (viz. výše).

Symetrické spojovací klíče

Návrh, kde mají uzly předem nakonfigurovánu sadu symetrických spojovacích klíčů, pomocí nichž zajišťují bezpečné spojení mezi jednotlivými uzly, nebyl také příliš vhodný, protože si uzel musel pamatovat velký počet klíčů, závisející na počtu uzlů v síti.

Postupné zavádění klíčů

Další návrh používal postupné zavádění klíčů. Chce-li uzel komunikovat s jiným uzlem nežli byla základová stanice, pošle žádost o spojovací klíč základové stanici. Ta na základě této žádosti odpovídá klíčem, který se nadále používá během další komunikace. Nevýhodou tohoto řešení, je, že základová stanice musí udržovat dostatečně velkou databázi spojovacích klíčů.

Předdistribuční klíče

V dalším vývoji klíčové správy pro senzorové sítě byl navrhnut předdistribuční protokol s párově náhodným klíčem. V tomto protokolu má systém velký blok symetrických klíčů, které náhodně po částech posílá do jednotlivých uzlů, mají-li dva

uzly stejné klíče, mohou mezi sebou komunikovat. Nebezpečí zde hrozí, že útočník může zrekonstruovat kompletní sadu symetrických klíčů.

Správa distribuce klíčů je tedy vyvíjena ze dvou hlavních hledisek. Za prvé z hlediska zajištění hardwarové podpory pro efektivní používání veřejných kryptografických algoritmů. Za druhé k vytvoření takových náhodných klíčových distribučních protokolů, aby vytvořily co nejvhodnější spojení a zamezily rekonstrukci spojovacích klíčů.

2.4 Požadavky zabezpečení WSN

Požadavky zabezpečení WSN jsou především diskrétnost dat, autorizace dat, datová integrita a datová aktuálnost [14].

- Při diskrétnosti dat jde především o to, aby uzly jedné senzorové sítě nepředávaly informace do jiné senzorové sítě. V mnoha případech si uzly vyměňují velmi citlivá data, například při distribuci klíče. Standardní utajení těchto dat je v šifrování pomocí tajného klíče, který je k dispozici pouze přijímači, čímž je dosažena potřebná diskrétnost.

- Autorizace nebo-li ověřování dat je také důležitou součástí aplikací pro senzorové sítě, zejména při budování samotné sítě, kdy provádíme administrativní úkoly, jako je kontrola činnosti uzlů či přeprogramování sítě. Také může dojít ke vložení cizích dat ze strany útočníka, proto se musí příjemce ujistit, že přijaté data pochází opravdu od požadovaného zdroje. Autorizace dat tedy umožňuje přijímači ověřit si, že data byla opravdu poslána od správného zdroje.

- Integrita dat v komunikaci senzorové sítě zajišťuje, že data přijatá přijímačem nejsou během přenosu útočníkem nijak zaměněna.

- V senzorové síti musíme mít data nejen zabezpečená a ověřená, ale také, při velkém měření, které se časem mění, musíme zajistit, aby přijatá data byla aktuální. Aktuálností dat znemožníme případnému útočníkovy opětovné čtení starých zpráv. Autoři [14] rozlišují dva typy aktuálnosti. Slabá, ta poskytuje pouze část organizační zprávy, ale neobsahuje informaci o zpoždění, a silná, která obsahuje celou dvojici žádost a odpověď a umožňuje odhad zpoždění.

Důležitá je především autorizace vysílaných dat pro celou senzorovou síť. Mnoho návrhů řešení autorizace je založeno na asymetrickém digitálním podpisu, které jsou v senzorových sítích nepoužitelné (dlouhé podpisy s vysokou komunikačními náklady 50-100 Kbytu na paket, nákladné vytvoření a ověření podpisů). V minulosti

bylo navrhováno čistě symetrické řešení, ale to bylo také nepraktické, jelikož šifrování zabíralo mnoho bytů z paketu. Protokol řešící autorizaci vysílaných dat je TESLA protokol (timed efficient stream loss – tolerant authentication) [14], který je vhodný pro internet s běžnou pracovní stanicí (např. PC), ale ne pro zdrojově závislé senzorové uzly. Vývojem protokolu TESLA a jeho implementací v senzorových sítích se zabývá Kalifornská univerzita Barkeley. Tento upravená protokol nazývají μTESLA.

2.5 Tipy útoků na senzorovou síť

Jako každá bezdrátová síť, tak i WSN obsahuje tři slabá místa, které se dají využít k napadení sítě – útočníci mohou jednoduše odposlouchávat bezdrátový přenos, můžou do přenosového kanálu vložit nežádoucí data a na konec si mohou přehrávat dříve zachycené pakety.

V senzorové síti sice neexistuje přesné rozdělení komunikace do vrstev, ale pro lepší pochopení zabezpečení je vhodné si takovéto rozdělení vymyslet. TinyOS rozděluje komunikační schéma do tří vrstev: Fyzická vrstva, MAC vrstva (datově linková vrstva, zabývající se výkonem, časováním a synchronizací uzlů) a Aplikační (síťová) vrstva [15]. Jednotlivé útoky a jejich zabezpečení, pak můžeme rozdělit do těchto vrstev:

Fyzická vrstva

Do fyzické vrstvy můžeme zahrnout komunikační zařízení. Zde dochází k útokům ze strany rušení tzv. DoS útoky (denial of service - odmítnutí služby), kde se útočník snaží narušit provoz vysíláním silnějšího signálu. Bude-li signál dostatečně silný, může dojít až k zablokování sítě. Ochranou proti tomuto útoku je v rozšíření komunikačního spektra, tak aby mohl vysílač komunikovat přes různé, bezpečně zašifrované, rozsahy spektra. Dalším podobným útokem může být rušení impulsními signály buďto náhodně, nebo rychle za sebou.

MAC (linková) vrstva

Útoky na linkové vrstvě jsou založeny na blokování a zahlceni přenosového kanálu velkým počtem žádostí a paketů. Mezi tyto útoky patří nepřetržitý kanálový přístup (vyčerpání), kolize a nedostupnost.

- Útok označovaný jako nepřetržitý kanálový přístup je běžný v každé bezdrátové síti s komunikačním standardem 802.11 a jiných. Napadený uzel přerušuje Media Access Control protokol neustálými žádostmi nebo přenosem přes kanál.

Takto nakonec dojde k vyčerpání ostatních uzlů v síti s ohledem na přístupový kanál. Tento útok je většinou způsoben zasíláním velkého počtu RTS (request to send) paketů přes dané médium. Jelikož dochází k mnoha kolizím mezi síťovými pakety, ztrácejí jednotlivé uzly svoji energii. RTS je žádost vysílací stanice o vysílání směrovaná příjemci.

- Kolize je podobná útoku vyčerpání. K dosažení tohoto útoku stačí, když útočník vyvolá kolizi v jednom oktetu během přenosu. Tím se změní datová část paketu, která způsobí změnu kontrolního součtu. Oktet je datová část paketu skládající se z osmi částí (1byt=8bitů=>1byt=1 oktet) neboli 8 bitová datová část paketu.

- Opakováním předchozích dvou útoků, může dojít až k nedostupnosti. Tento útok může být označován jako DoS útok i když jeho důsledek je menší omezení výkonu.

Síťová vrstva

U síťové vrstvy jde především o zajištění bezpečného směrování přenosů informací mezi uzly navzájem a mezi uzly a základovou stanicí. Mezi směrovací útoky patří:

Výběrové přeposílání (Selective Forwarding) V senzorové síti se předpokládá, že se data přenášejí z jednoho uzlu na druhý (multi-hop síť). Výběrové přeposílání nastává tehdy, odmítne-li uzel přesměrovat určité pakety a zahodí je. Při zahození všech paketů se pak tento útok nazývá černá díra. Výběrové přeposílání se může docílit útoky Sinkhole a Sybil (Sybil attack, Sinkhole attack)

Útok sinkhole V tomto útoku se napadený uzel snaží stáhnout veškerou komunikaci na sebe tím, že se chová jako nejvhodnější spojovací uzel z hlediska směrovacího algoritmu (nejblíže k základové stanici, největší dosah, kvalitní spojení, atd.) Tohoto útoku se dá dále využít k výběrovému přeposílání

Útok Sybily (Sybil attack) Zde se jeden uzel vzhledem k ostatním uzlům chová jako několik uzlů. To může vézt k rozpojení směrování, protože se může stát, že se více uzlů připojí ke stejnému falešnému uzlu.

Útok Červí díra (Wormhole attack) Při útoku červí dírou dochází k propojení dvou vzdálených uzlů pomocí nepoužívaného kanálu s nízkým zpožděním. Tím se docílí menší doby přenosu dat, protože data, která by byla původně odeslána přes několik uzlů, jsou poslána pouze přes jeden. Pokud jsou takto poslána i směrovací

data, může se stát, že dva vzdálené uzly se budou považovat za sousední. Takto může dojít k sinkhole útoku, je-li takto spojen uzel se základovou stanicí, tím se daný uzel jeví jako nejbližší k základové stanici.

HELLO flood útok Tento typ útoku se vyskytuje pouze při užití určitých protokolů, v kterých uzly posílají svým sousedům tzv. HELLO pakety. Příjme-li uzel takovýto paket, znamená to, že jeho odesílatel je jeho sousedem a je v jeho radiovém dosahu. Útočník může pomocí výkonnější antény (anténa, která má větší dosah než běžná anténa senzorového uzlu) poslat tyto pakety i vzdálenějším uzlům, ty se pak budou snažit data posílat k tomuto sousedovi, který takto není v jejich radiovém dosahu, do nekonečna.

Falešné potvrzování Falešné potvrzování je útok zaměřující se na algoritmy používající ověřování na linkové vrstvě. Zfalšováním těchto potvrzení přesvědčí útočník uzel, že jeho nefungující (mrtví) sousední uzel je v pořádku (naživu) a nebo jeho slabé spojení je spolehlivé a bezpečné. Pakety, které jsou pak směrovány přes tento uzel, jsou s velkou pravděpodobností ztraceny. Útok je podobný útoku selective forwarding.

DoS útoky Mezi útoky typu DoS můžeme zahrnout většinu útoků na senzorové sítě, jelikož téměř u všech může dojít k odmítnutí služby.

Podrobnější vlastnosti útoků jsou popsány v literaturách [15] a [8].

2.6 Zabezpečovací protokoly

SPINS

SPINS (security protocols for sensor network) se skládá ze dvou zabezpečovacích protokolů SNEP (sensor-network encryption protocol) a μTESLA (micro timed efficient stream loss-tolerant authentication) [14]. SPINS protokoly používá symetrické kryptografické funkce. Kvůli omezení uložení programu (100-1000Kbytu flash paměti [6]) vytváří veškeré kryptografické šifry z jednoho blokového kódu pro opětovné použití. Zabezpečovací prvky obou mechanizmů se zavádí pomocí tajného klíče, který je sdílen mezi každým uzlem a základovou stanicí.

SNEP

SNEP zajišťuje základní bezpečností prvky. Jsou to diskrétnost (utajení), integrita, autorizace dat, dvojité ověřování údajů a datová aktuálnost. Má nízkou komunikační

režii (8 bytu na zprávu), hodnotu čítače, který využívá většina kryptografických protokolů, nepřenáší, ale nechává na obou koncových bodech, zajišťuje rovnoměrné sémantické zabezpečení, které zabraňuje odposlechu zašifrovaných zpráv. Dále používá k zajištění autorizace a integrity autorizační kód zprávy MAC a spolu s výše uvedeným tvoří celkový šifrovací WSN protokol SNEP.

Autorizace dat pomocí MAC se provádí následovně: je-li MAC vyhodnocena jako správná, přijímač si může být jist, že data byla poslána od požadovaného odesílatele. Sémantická bezpečnost zvyšuje hodnotu čítače po každé správě, proto je stejná správa vždy šifrována jinak. Rozsah čítače je dostatečně veliký, proto se jeho hodnoty nikdy neopakují po celou dobu životnosti uzlu.
Šifrovaná data mají tento tvar

$$E = \{D\}_{(Ks,C)}, \text{ kde D jsou data, } K_S \text{ je šifrovací klíč a C je čítač.} \tag{2.3}$$

MAC má tvar

$$M = MAC(\mathrm{K}_m,\mathrm{C}|\mathrm{E}) \ . \tag{2.4}$$

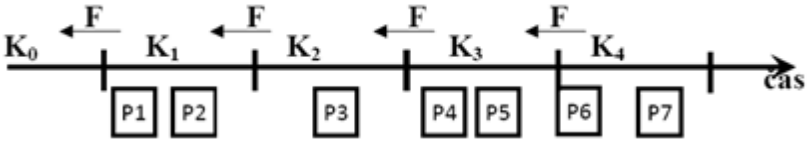
Klíče K_m a K_s jsou odvozeny od hlavního zabezpečovacího klíče K.
Zpráva poslaná z uzlu A do uzlu B má výsledný tvar:

$$\text{A->B:}\{\mathrm{D}\}_{(Ks,C)}, M = MAC(K_m, C|\{D\}_{(Ks,C)}). \tag{2.5}$$

μTESLA

μTESLA zajišťuje efektivní autorizaci vysílaných dat. Oproti původnímu protokolu TESLA, který ověřuje počáteční paket pomocí digitálního podpisu, používá μTESLA pouze symetrické mechanizmy, což je z hlediska energie méně nákladné. Místo zveřejnění klíče v každém balíčku zveřejňuje klíč jednou za určitou dobu. Neuchovává v uzlech jednocestné klíčové řetězce, ale redukuje počet autorizovaných vysílačů. μTESLA řeší problém nesouměrných kryptografických mechanizmů, které mají vysoké nároky na výpočet, komunikaci a uchovávání, v zavedení asymetrie do zpožděného zveřejnění symetrického klíče, což má za následek účinné vysílací autorizační schéma. μTESLA požaduje, aby základová stanice spolu s uzly byla volně časově synchronizována a aby každý uzel věděl o horní hranici maximální synchronizační chyby.

V případě, chce-li základová stanice poslat autorizační paket, vypočítá MAC na paket s klíčem, který je v tomto časovém okamžiku tajný. Po přijetí paketu uzel ověří, zda příslušný MAC klíč nebyl doposud odhalen základovou stanicí. Pokud přijímací uzel zjistí, že MAC klíč je znám pouze základové stanici, je si jistý že přijatý paket nebyl během přenosu pozměněn. Jednotlivé pakety ukládá uzel v bufferu



Obr. 2.3: Příklad použití autorizace pomocí klíčů protokolu μTESLA

(vyrovnávací paměť). Během odhalování klíčů vysílá základová stanice verifikační klíče všem příjemcům. Příjme-li uzel odhalovací klíč, ověří správnost MAC klíče. Pokud klíč souhlasí, použije ho k autorizaci paketu, který má uložen v bufferu. Příklad protokolu μTESLA je znázorněn na obrázku 2.3. Tento příklad byl převzat od autorů [14].

Každý klíč K klíčového řetězce odpovídá jednomu časovému úseku. Všechny pakety poslané v daném časovém úseku jsou ověřeny (autorizovány) jedním klíčem. Dva časové intervaly zde znázorňují dobu, po kterou je daný klíč v jednotlivém intervalu odhalen. Předpokládá se, že přijímací uzel je volně časově synchronizován a zná klíč K_0 (z autorizační cesty). Pakety P1 a P2, poslané v intervalu jedna, obsahují MAC s klíčem K_1. Paket P3 obsahuje MAC používající klíč K_2. Do této doby přijímač nemohl ověřit žádné pakety z toho důvodu, že se předpokládá, že pakety P4, P5 a P6 jsou ztraceny stejně jako paket, který měl odhalovat klíč K_1. Ve čtvrtém intervalu vysílá základová stanice klíč K_2, kterým uzel autorizuje pakety kontrolou $K_0 = F(F(K_2))$, díky tomu také zná $K_1 = F(K_2)$, a tak může autorizovat pakety P1 a P2 s klíčem K_1 a balíček P3 s klíčem K_2. Místo přidávání odhalovacího klíče do každého datového paketu, je tento klíč nezávislý na vysílaných paketech, je pouze svázán s časovými intervaly. V souvislosti s μTESLA odesílatel vysílá aktuální klíč periodicky ve speciálním paketu.

Tato sada protokolů řeší mnoho zabezpečovacích problému, ale některým dodatečným vlastnostem nezabrání. Např. úniku informací skrz skryté kanály; protokoly nepracují přímo s napadenými senzory, ale zajišťují, aby napadený senzor neodhalil klíč všech senzorů v síti; neřeší otázku DoS útoku, protože v bezdrátová síti může útočník kdykoliv provést DoS útok tím, že přetíží přenosový kanál silným signálem.

3 NÁVRH ZABEZPEČENÍ WSN

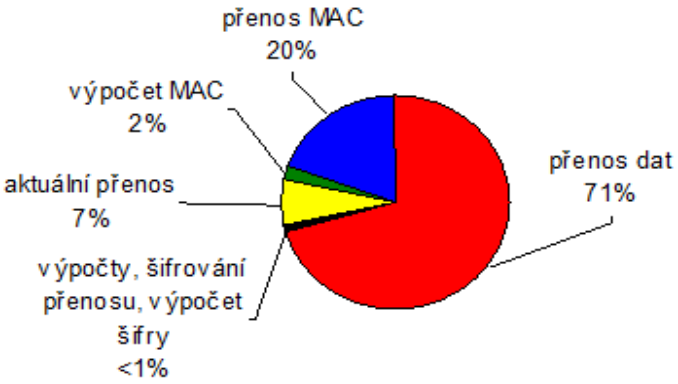
Z předchozí kapitoly vyplývá, že vhodnou metodou pro zabezpečení bezdrátových senzorových sítí je symetrická kryptografie a její různé modifikace, jelikož není tak energetický náročná jako asymetrická kryptografie. Protože každý senzorový uzel má řadu omezení, která byla vypsána již výše (omezená šířka pásma, menší operační a vyrovnávací paměť, omezení ve zdroji elektrické energie atd.), bude se návrh lišit od běžných bezdrátových sítí, které takto omezeny nejsou.

Jako první při návrhu, je vhodné vědět, jak rozsáhlou síť budeme zabezpečovat. V rámci bezdrátové senzorové sítě se bude jednat o síť složenou z několika stovek zařízení (senzorových uzlů + jedna či více základových stanic), proto při použití symetrické kryptografie budeme potřebovat dostatečné množství šifrovacích klíčů pro zabezpečení jednotlivých přenosů, což by neměl být problém, protože základová stanice zpravidla nebývá, z hlediska paměti sloužící pro uchovávání daných klíčů, tak omezena jako senzorové uzly. Výhodou symetrické kryptografie zde bude použití pouze jednoho klíče na obou stranách komunikace (vysílač a příjemce). Pokud bychom ale šifrovací klíč ponechali pro každý přenos jak ve vysílači, tak v přijímači, stačilo by útočníkovi pouze odposlouchat přenos na jedné straně. Proto se klíče určitým způsobem mění, a to třeba takovým, jaký popisují autoři protokolu μTESLA [14]. Zjednodušeně lze daný algoritmus popsat tak, že se používá dvojice šifrovacích klíčů, a to autorizačních a verifikačních, jako první je posílán s daty autorizační klíč, který je v daném okamžiku tajný, na straně příjemce dojde k ověření, zda je klíč opravdu tajný, pokud ano, je si přijímací strana jista, že data nebyla během přenosu modifikována, ty jsou následně uložena ve vyrovnávací paměti daného senzorového uzlu. Při následném odhalování klíčů vysílá základová stanice verifikační klíče, a to všem příjemcům, senzorový uzel po přijetí tohoto klíče ověří správnost autorizačního klíče a pokud souhlasí, použije ho na ověření daných dat, které má uložené ve vyrovnávací paměti. Takto dochází ke zpětnému ověření dat pomocí symetrické kryptografie.

Dále bychom se měli zaměřit na vysílací režii, jelikož právě komunikace senzorových uzlů spotřebovává nejvíc elektrické energie, a tedy závisí na životnosti jednotlivých uzlů, oproti výpočtu zabezpečovacích šifer. Jedná se zejména o vysílání a příjem nejenom šifrovacích klíčů, ale i dat, je dána časovými intervaly v kterých senzorový uzel naslouchá či spí (úspora energie) nebo vysílá a přenáší data (spotřeba energie). Daná závislost je přehledně zobrazena na obr. 4.1. Tuto režii by opět měla řídit základová stanice, jelikož z hlediska životnosti v závislosti na elektrické energii není omezená. Co tedy základová stanice bude dělat? Mějme situaci, že při zprovoznění senzorové sítě v čase t_0 budou všechny senzorové uzly ve stavu spánku, což znamená, že jejich spotřeba bude minimální. Proto, aby senzorové uzly začaly

pracovat, bude potřeba je převézt z režimu spánku do aktivního režimu (tzv. probuzení). Podle literatury [1] je probouzení senzorových uzlů prováděno nejčastěji pomocí sledu signálů (tzv. probouzecích signálů). Délka těchto signálů je závislá na počtu senzorových uzlů, které chceme uvést do aktivního režimu a také na době, jak dlouho daný senzorový uzel byl v režimu spánku. Základová stanice tedy začne vysílat (v čase t_0) sled probouzecích signálů uzlům, které bude potřeba uvést do aktivního režimu. Mějme čas t_1, kdy budou dané uzly již probuzeny a nastaveny do režimu naslouchání (režim, ve kterém nenastává ještě přenos dat, ale uzly již čekají a poslouchají přenosový kanál). Čím menší bude interval t_0 až t_1, tím bude spotřebováno méně energie, ale bude menší pravděpodobnost probuzení daných uzlů. V dalším časovém úseku může nastat přenos, po kterém se senzorový uzel opět nastaví do režimu spánku, čas přechodu senzorového uzlu zpět do režimu spánku označme jako t_2. Časový úsek mezi t_0 a t_2 bude tedy jeden pracovní cyklus senzorové sítě. Je žádoucí, aby naslouchací interval byl tak malý, aby spící interval pro daný pracovní cyklus byl také dostatečně malý, tak aby zpoždění v přenosu dat bylo co nejmenší. Pokud nastavíme naslouchací interval příliš malý, ačkoli v něm bude potřeba přenést data, nebudou mít možnost k přenosu.

V těchto situacích může dojít k útokům, kdy útočník může předstírat základovou stanici a nastavovat uzly do příliš zdlouhavých intervalů a tím zvýšit spotřebu uzlu, bude-li naslouchací interval příliš dlouhý, nebo zamezit přenosu dat, bude-li spící interval příliš dlouhý. Situace by se dala řešit použitím ověřovacího klíče ve sledu probouzecích signálů, který by byl při každém novém probouzení vypočítán, jelikož náklady na výpočet šifer jsou v porovnání s přenosem minimální.



Obr. 4.1: Energetické náklady zabezpečovacího protokolu SNEP pro senzorové sítě (překreslen z [14])

4 ENERGETICKÉ NÁKLADY

V této kapitole je pojednáno o energetické náročnosti jednotlivých procesů senzorového uzlu. Jak je uváděno ve veškerých literaturách, největším problémem ve vývoji senzorových uzlů je jejich energetická náročnost, protože nemají neomezený zdroj napájení. Jsou napájeny z akumulátoru, který svou výdrží určuje celkovou životnost celého uzlu, proto řešení spotřeby jednotlivých částí uzlu nelze brát na lehkou váhu.

Příkladem může být protokol SPINS [14]. Autoři zde uvádějí, jak je vidět z obrázku 4.1, energetické náklady jednotlivých částí senzorového uzlu s použitím zabezpečovacího protokolu SNEP, který je součástí protokolu SPINS. Nejvíce energie je spotřebováno jednotlivými přenosy, jak už datovými, tak i vyžádanými protokoly (poskytování a ověřování klíčů, jednotlivé žádosti např. o přenos, MAC, atd.). Z hlediska zabezpečení a šifrování takové energetické nároky nejsou. Použitím proudové šifry pro šifrování je velikost zašifrované zprávy stejná jako velikost samotné přenášené informace. MAC používá 8 bytu pro každou zprávu o velikosti 30 bytu, navíc MAC vykonává potřebnou integritu, potom není zapotřebí používat další integritní mechanizmy (např. 16-bit kontrolní součet CRC). Šifrování a potvrzování zpráv ukládá režii 6 bytu na zprávu, zato nezašifrované zprávy s ověřováním integrity něco o kolo 20%.

Tab. 4.1: Využití paměti RAM šifrovacími moduly

modul	velikost RAM
RC5	80bytu
TESLA	120bytu
šifrování/MAC	20bytu

Do energetických nákladů můžeme zahrnout i výkon jednotlivých kryptografických primitiv, který jsou závislé na použitelné šířce pásma daných senzorů. To je spojeno s použitím jednotlivých zabezpečovacích modelů a jejich využitím paměti RAM viz. tabulka 4.1 [14]. Například modul RC5 potřebuje na nastavení klíče 8 tisíc instrukčních cyklů (4ms potřebné k přenosu 40 bitů), k zašifrování 8 bytového bloku, 120 instrukčních cyklů, celkové využití paměti RAM pak je 80 bytu.

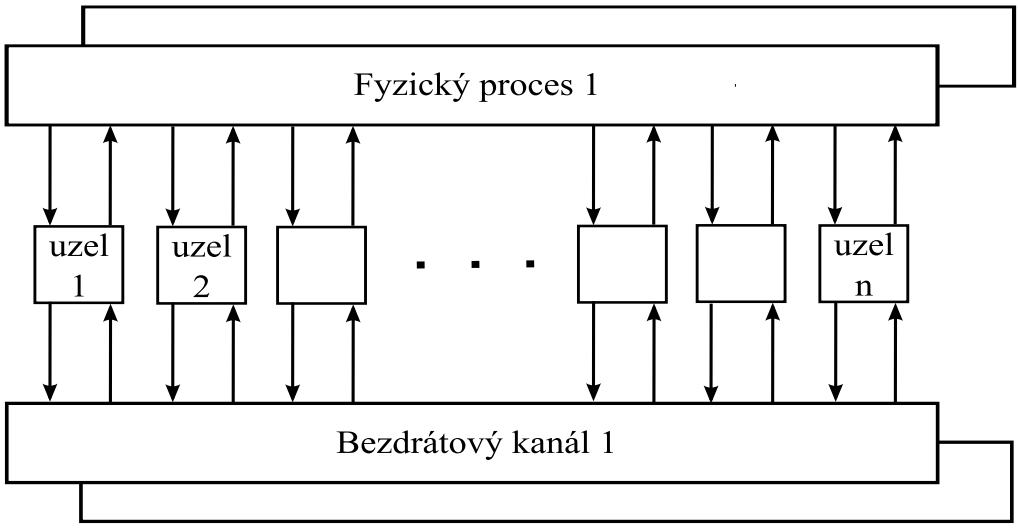
5 NÁVRH MODELU WSN

5.1 Seznámení s programem Castalia

Následující text je přebrán z uživatelské příručky pro tento program [1].

Celá práce s programem Castalia spočívá v úpravě zdrojových konfiguračních souborů. Pro každou vlastnost senzorové sítě jsou vytvořeny zdrojové soubory simulující danou vlastnost, jejichž úpravou docílíme požadované vlastnosti. Pro zjednodušenou práci je doporučeno, podle autorů návodu k instalaci [3], nainstalovat si program Eclipse, ve kterém se nám bude lépe orientovat mezi výše uvedenými konfiguračními soubory.

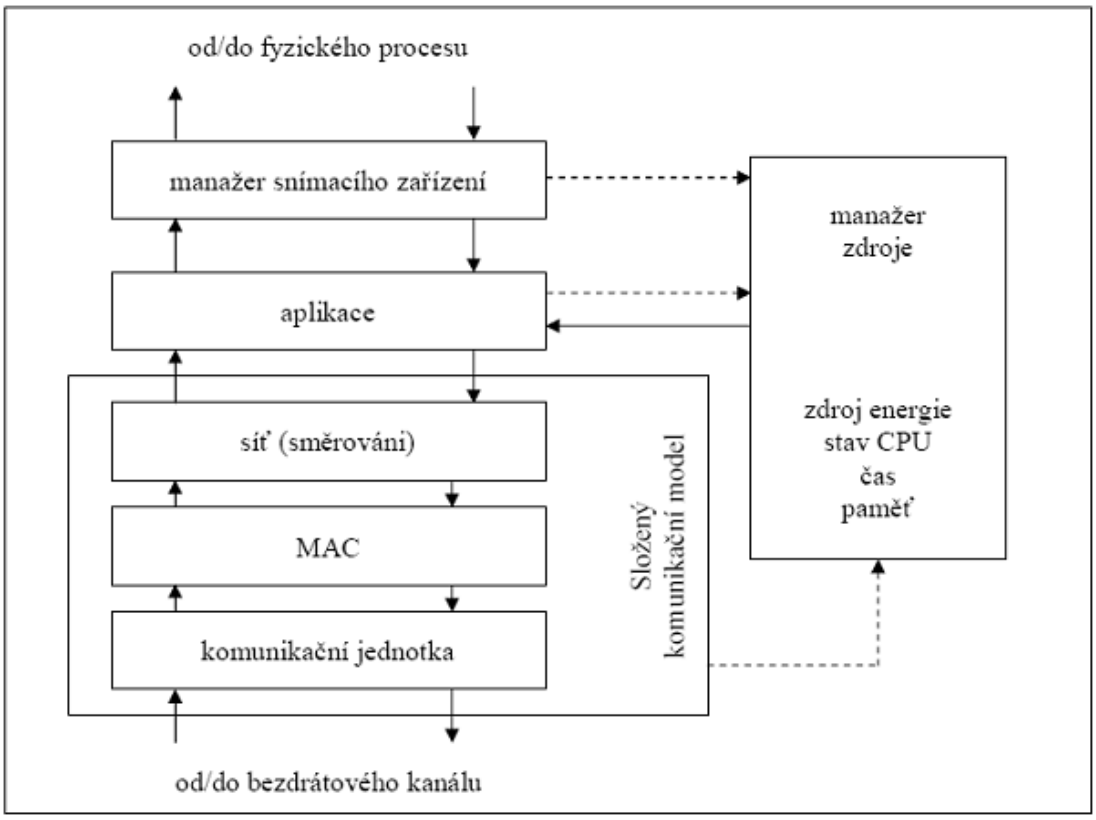
Struktura základního modulu programu je zobrazena na obrázku 5.1.



Obr. 5.1: Moduly a jejich spojení v programu Castalia

Je vidět, že uzly nejsou navzájem přímo propojeny, ale jsou spojeny přes bezdrátové kanálové moduly. Šipky představují putující zprávy od jednoho modulu k ostatním. Má-li uzel paket k poslání, předá ho bezdrátovému kanálu, který rozhodne, který uzel by měl paket přijmout. Uzly jsou též spojeny moduly fyzických procesů, které monitorují, a to tak, že pro jeden fyzický proces existuje jeden modul, který představuje uzlům získané údaje v prostoru a čase od jejich čidel. Může se zde samozřejmě vyskytnout více fyzických procesů představující multisnímací zařízení, které může senzorový uzel obsahovat, tak jako více bezdrátových kanálů znázorňujících rozmanitou bezdrátovou komunikaci (např. různé frekvence a kódy).

Vnitřní struktura modulu senzorového uzlu je zobrazena na obrázku 5.2. Modul senzorového uzlu je ve skutečnosti složený. Plné šipky znázorňují přenášené zprávy a přerušované představují rozhraní mezi moduly s jednoduchými funkcemi volání



Obr. 5.2: Složený modul senzorového uzlu

(např. většina modulů volá funkci správce zdrojů k signalizaci, že el. energie byla spotřebována). Modul Aplikace je ten, který může uživatel neomezeně měnit. Pro každý algoritmus, který chce uživatel změnit, musí být vytvořen nový modul Aplikace, podle šablony poskytnuté v Castalii. Ostatní moduly mají laditelné jen některé parametry a většinou uživatel nemůže měnit jejich funkčnost.

Tato výše popisovaná struktura je v Castalii implementována jazykem OMNeT++ NED. Pomocí tohoto jazyka můžeme jednoduše definovat jednotlivé moduly, jako jejich jméno, vlastnosti a zařízení (např. vstupní a výstupní brány) a také strukturu submodulů, jsou-li moduly složené. Soubory s příponou ".ned" obsahují kód v jazyce NED. Struktura Castalie se odráží v uspořádání adresářů ve zdrojovém kódu. Každý modul odpovídá adresáři, který vždy obsahuje soubor .ned, který definuje daný modul. Je-li modul složený z více submodelů, podadresáře definují tyto submodely. Jedná-li se o jednoduchý modul, je definován v jazyce C++ (soubory .Cc, .H). Toto kompletní uspořádání .ned souborů definuje celou strukturu programu Castalia. Tyto soubory jsou dynamicky načítány a zpracovávány (pomocí OMNeTu), proto každá změna nevyžaduje rekompilaci Castalie (s výjimkou nových modulů s novými funkcemi).

5.2 Simulační metody

Jak už bylo zmíněno, program Castalia se skládá z jednotlivých modulů, představujících části simulované senzorové sítě, proto i navrhovaná síť se bude skládat z jednotlivých modulů. Tyto moduly jsou realizovány pomocí adresářů které obsahují zdrojové kódy daných modulů. Tyto kódy poté popisují vlastnosti a chování jednotlivých modulů.

Základní simulace programu Castalia jsou umístěny v adresáři `Castalia/Simulations`. V navrhované síti budeme používat simulace valuePropagation a ConnectivityMap. V každé této složek se nacházejí min. dva soubory, a to omnetpp.ini a runAbc. První z nich je konfigurační soubor, který definuje všechny potřebné parametry pro OMNeT a Castalii, runAbc (v adresáři valuePropagation je to runValProp) je skript, který vykoná danou simulaci. Skript pustíme příkazem:

```
./runValProp
```

popřípadě (v odpovídajícím adresáři)

```
./runConMap
```

Vypíše se pár informačních řádků, které nás informují o provedení skriptu a o tom, kam se uložily výstupní hodnoty simulace. Tyto hodnoty se ukládají do souboru `Castalia-Primary-Output.txt`. Tento textový soubor obsahuje výsledky dané simulace. V úvodních řádcích jsou uvedeny standardní výstupy programu OMNeT++ a po nich informace o načtení souborů .ned. V další části jsou vypsané výsledky jednotlivých kol simulace a následuje souhrn všech nastavených parametrů jednotlivých modulů a kol, které jsou zahrnuty v souboru ometpp.ini. Část výsledků simulace valuePropagation může vypadat následovně:

```
Calling finish() at end of Run #1...
Node [4] Value: 0
Node [4] spent energy: 2.57516
Node [5] Value: 0
Node [5] spent energy: 2.57539
Node [6] Value: 40.1312
Node [6] spent energy: 2.57619
Node [7] Value: 0
Node [7] spent energy: 2.576
```

Daný výsledek nám uvádí spotřebovanou energii jednotlivých senzorových uzlů, která je udávána v Joulech a celková její hodnota je stanovena na 29160 Joulů.

K této hodnotě dojdeme zcela jednoduše. Senzorový uzel je napájen dvěma AA bateriemi a platí:

$$1\,AA\,battery : 1.5V * 2700\,mAh = 4050\,mWh,$$

$$4050\,mWh * 3600sec/h * 1/1000 = 14580W/s = 14580\,J,$$

$$2\,AA\,batterries : 2 * 14580 = 29160\,J.$$

Celá simulace kontroluje účinek kanálů a několika parametrů MAC úrovně na přenosu hodnot celou sítí. Simulace začíná se senzorovými uzly, které zjišťují vlastnosti fyzického procesu, každý získává určitou hodnotu (např. teplotu). Pokud je tato hodnota mimo zadanou mez (v tomto případě třeba teplota 15°C), musí být zaslána. Pokud senzorový uzel přijímá tuto hodnotu od nějakého jiného senzorového uzlu, pokouší se ji poslat dál a poté nastaví svůj flag, že splnil svou povinnost. Každý senzorový uzel se chová podle určité MAC úrovně založené na daných parametrech úrovně. V závislosti na tomto chování a stavu kanálů mezi senzorovými uzly je daná hodnota přenášena sítí. V tomto případě pouze senzorový uzel číslo 6 zjišťuje hodnotu větší jak 15 (40+šum), ostatní nulu. Proto je uzel 6 jediný zdroj hodnoty pro vysílání.

Výsledky simulace ConnectivityMap mohou mít takovouto podobu:

```
** Node [3] received from:
[3<--4] --> 140 out of 500
[3<--6] --> 160 out of 500
[3<--7] --> 500 out of 500
```

Daná simulace nám představuje kvalitu spojení mezi jednotlivými uzly, a to tak, že pro každý uzel n se vypíší uzly, od kterých uzel n přijal pakety. V uvedeném příkladu se uzel 3 spojil se třemi dalšími uzly: s uzlem 7 dokonale, s uzly 4 a 6 s pravděpodobností 25-30%. Celý výstup poskytuje textový typ mapy propojení. Způsob, jakým je tato mapa vytvořena, je následující: V aplikační úrovní každého uzlu je naprogramován přenos 500 paketů v jednom timeslotu, tak že nenastanou srážky s ostatními uzly. Uzel, když nepřenáší, stále poslouchá a čeká na příchozí pakety a když jeden příjme, přičte jedničku ke svému čítači paketů přijatých od daného uzlu.

Nastavení jednotlivých simulací je obsaženo v souboru omnetpp.ini. Jsou v něm obsaženy příkazy, které odkazují na jednotlivé soubory s parametry daných modulů. Ty jsou nazývány jako `include` příkazy a jsou uvozeny komentářem `#Choose a parameters file...`. Tyto soubory mají koncovku .ini a nacházejí se v v adresáři `Castalia/Simulations/Parameter_Include_Files`. Namísto velkého souboru

omnetpp.ini, který by obsahoval veškeré parametry, máme možnost volby z více předem vytvořených struktur s množstvím souborů pro nastavení jednotlivých parametrů. Proto můžeme jednoduše měnit potřebné parametry snadno a rychle přímo v souboru omnetpp.ini, než v rozsáhlém a nepřehledném jednotném souboru.

Otevřeme-li soubor omnetpp.ini zjistíme, že se skládá z několika sekcí. První, označována jako `[General]`, začíná názvem cesty k souboru, který obsahuje kompletní seznam .ned souborů a který chceme dynamicky načíst. Soubor nedfiles.lst nacházející se v nejvyšší složce programu Castalia je vytvořen automaticky při instalaci Castalie. Musíme se ujistit, že zadaná cesta k tomuto souboru je správná vzhledem k umístění aktuálního souboru omnetpp.ini. Další příkaz odkazuje na důležitý soubor programu Castalia, který definuje tzv. generátory náhodných čísel RNGs (random number generators – generátory náhodných čísel), které jsou základem pro jednotlivé procesy jako stínící efekt, náhodný start aplikace či související s nastavením časovačů. Pak následuje nastavení času simulace (sim – time – limit), který je definován uživatelem. Nakonec jsou definována jména dvou výstupních souborů.

OMNeT může pracovat jak v příkazovém řádku, tak v grafickém rozhraní. Castalia používá příkazový řádek, jelikož je jednodušší pro spuštění dávkových úkolů. Grafické prostředí může být užitečné pro výsledné odlaďování, ale pokud je spuštěn vizualizační nástroj Castalia, není potřebné. Protože používáme rozhraní příkazového řádku, musíme definovat sekci `[Cmdenv]` uvnitř souboru omnetpp.ini. K tomu slouží soubor ../Parameter_Include_Files/omnet_cmdenv_reporting.ini, ve kterém se nacházejí specifické parametry OMNeTu související s příkazovým řádkem. Např. nastavíme-li `express-mode=no`, budou všechny události/diagnostické zprávy modulů tištěny do výstupu. To je užitečné zejména v odlaďovací fázi (kdy rozšiřujeme/upravujeme zdroje Castalie). V ostatních případech nepotřebujeme takové množství informací a navíc nepotřebujeme aby simulace byla pomalá. Proto mějme `module-messages=yes`.

Poslední sekcí je `[Parameters]`, kde se nacházejí specifické parametry čistě pro Castalii. První je definováno jméno souboru, který obsahuje každou odlaďovací informaci. Pak jsou definovány určité všeobecné parametry pro celou síť. `SN.field_x` a `SN.field_y` udávají rozměry pole, kde budou rozmístěny senzorové uzly. Dále je definován počet senzorových uzlů v sítí a také důležitý parametr – rozmístění senzorových uzlů a také počet fyzických procesů. Dále jsou zahrnuty soubory z adresáře Parameter_Include_Files. Pak jsou zde parametry související s Aplikačním modulem. Důležitým je `SN.node[*].appModuleName`, který specifikuje, jaká aplikace bude vykonávána. Zde nastavíme jméno toho aplikačního modulu, který chceme použít. Jméno modulu je definováno v souboru .ned aplikačního modulu. Parametrem `printDebugInfo` povolujeme (zakazujeme) odlaďovací výstup z aplikace. Parametr `priority`, pro určení priority, pokud povolíme vykonání více aplikací na uzlu.

Poslední dva parametry se týkají datových paketů posílaných aplikací na nižší komunikační úrovni.

Na konci se nachází sekce `[Run]`, kde jsou definované vícenásobné běhy dané simulace. Změníme-li nějaký parametr, změní se nám i výstupní hodnoty. Není tedy vhodné, z praktického hlediska, měnit samostatně parametry přímo v .ini souboru, protože je používán i jinými aplikacemi. OMNeT definuje vícenásobné spouštění pokaždé, když je předefinován nějaký z používaných parametrů. Dosahuje toho vytvořením sekcí [Run 1], [Run 2]... [Run n] (v souboru omnetpp.ini), v kterých je předefinován pouze ten parametr, který chceme změnit.

5.3 Příprava simulace

Jak je vidět z obrázku 5.2, každý senzorový uzel vytvořený v programu Castalia se skládá z určitých modulů simulující vlastnosti jednotlivých částí modulu. Proto i vlastní simulace se bude skládat z několika modulů. Nyní budou popsány jednotlivé moduly a jejich parametry, které můžeme měnit a které ovlivňují danou simulaci.

5.3.1 Bezdrátový kanál

Model bezdrátového kanálu je založen na práci autorů [20]. Autoři vysvětlují povahu PRR (packet receptions rates) v experimentálním nastavení s použitím zařízení mica2 skutečných modelů bezdrátoveho kanálu a komunikační jednotky. Použitý kanálový model je logový stínovaný bezdrátový kanál, který udává výkonovou ztrátu v dB dané vzdálenosti dvou uzlů d a další parametry.

$$PL(d) = PL(d_0) + 10 \cdot \eta \cdot log(\frac{d}{d_0}) + X_\sigma. \tag{5.1}$$

PL(d) je dráha ztrát o vzdálenosti d, $PL(d_0)$ je známá dráha ztrát o referenci vzdálenosti d_o, η je exponent dráhové ztrátovosti a X_σ je náhodná proměnná s gaussovským rozložením se střední hodnotou okolo nuly se směrodatnou odchylkou σ. Parametry $PL(d_0)$, d_o, η, σ jsou definovány jako parametry modulu bezdrátového kanálu mající prefix „SN.wirelessChannel.“.

Na základě ztrátového výkonu a přenosového výkonu vysílače můžeme vypočítat výkon signálu přijatého v přijímači. Také znalostí šumu (nebo obecných interferencí) v tomto přijímači můžeme vypočítat SNR (signal to noise ratio – poměr signál šum, někdy nazývaný jako SIR – signal to interference ratio). Z této veličiny a parametrů komunikační jednotky můžeme vypočítat pravděpodobnost přijatých paketů v přijímači od určitého vysílače. Tepelný šum je jedna stála úroveň interference (tj., spodní hladina šumu, parametr komunikační jednotky). Vícenásobné současné přenosy se mohou také navzájem rušit (působit jako interference sobě navzájem). Jestliže uzly

A, B, C přenášejí, uzly B a C mohou zasahovat do přenosu uzlu A, uzel A a C do přenosu uzlu B, atd.. V Castalii se interference počítají dynamicky z různých přenášejících uzlů, a tak jsou dynamicky vypočítány SNR a výsledná pravděpodobnost příjmu paketů.

Soubory s předdefinovanými parametry modulu bezdrátového kanálu (*.ini) jsou umístěny v adresáři `Simulations/Parameter_Include_Files/WChannel`. Máme možnost nastavit tyto parametry:

- `SN.wirelessChannel.collisionModel` nastavení modelu kolizí. Nastaví-li 0, nenastanou žádné kolize (vzájemné srážky paketů). Nastaví-li 1, jedná se o základní model kolizí, kde když dva uzly přenášejí (v normálním rozsahu přijímače), je kolize standardně v přijímači. Pro daný interferenční model můžeme mít dvě možnosti: a)kolizi jako takovou nebo b) přijímač přijímá ze dvou přenosů (pokud je dostatečně silný). Nastavením kolizní proměnné na hodnotu 2, je použit aditivní interferenční model, kde je přenos z dalších uzlů vyhodnocen jako interference s účinkem lineárního přičítání v přijímači.

- `SN.wirelessChannel.allBidirectionalLinks` nastavení kvality spojení mezi senzorovými uzly. Nastaví-li se jako „true“, pak jestliže je spojení dobré (PRR>95%) v jednom směru, pak je udržována stejná kvalita spojení i v opačném směru. Nastaví-li se jako „false“, poté zde není žádný vzájemný vztah a kolísání je vybíráno nezávisle. V realitě není kvalita mezi párem senzorových uzlů úplně bez vzájemného vztahu, ale jsou zde navíc veliké změny.

- `SN.wirelessChannel.sigma` nastavení směrodatné odchylky měnících se ztrát při přenosu. Nastavíme-li ji na nulu, dostanou všechny uzly v jisté vzdálenosti od vysílače stejnou sílu signálu. Jestliže jsou též poskytnuty ostré limity pro bezdrátový příjem (tj. perfektní nebo žádný příjem balíčku), může být vytvořena jednoduchá jednotka diskového modelu (diskový proto, že vysílač má dosah tvaru disku). Chceme-li napodobit tento model (tj. přenosy v určitém rozsahu od vysílače, jsou dokonale přijaty a mimo tento rozsah nepřijato nic), měla by být sigma nastavena na 0 a použit ideální modul komunikační jednotky.

- `SN.wirelessChannel.PLd0` nastavení dosahu signálu vysílacího uzlu (rozsah diskového modulu v m).

- `rxSignal_ConnectivityMap` nastavení výkonu přijímaného signálu v dBm mezi uzly.

- `PRR_ConnectivityMap` nastavení pravděpodobnosti přijetí paketů mezi různými uzly. Oba `Connectivity` parametry jsou řetězce oddělenými čárkou ve tvaru:

  ```
  node1->node2(txlevel)=value
  ```

  Value je hodnota zapsaná v dBm pro první parametr a pravděpodobnost příjmu paketu pro druhý.

5.3.2 Komunikační jednotka

Modul komunikační jednotky se snaží zachytit co nejvíce vlastností reálné všeobecné nízkonákladové komunikační jednotky, která bývá používána v platformě senzorové sítě. Jako takový podporuje několik stavů (přenos, příjem/poslech, spánek) s různou spotřebou energie a časové ztráty pro přechod z jednoho stavu do druhého. Podporuje více úrovní přenosového výkonu. Také podporuje detekci vysílání (za pomocí modulu bezdrátového kanálu). Uživatel může měnit datovou rychlost a ostatní parametry, které ovlivňují pravděpodobnost přijetí paketů daného SIR (Signal to Interference).

Soubory obsahující parametry komunikační jednotky se nacházejí v adresáři `Simulations/Parameter_Include_Files/Radio`. V programu Castalia jsou zahrnuty dvě komunikační jednotky, a to TelosB_CC2420 a Mica2_CC1000. Pro naše účely budou použity Mica2_CC1000 a MicaZ_MPR2400CA, který byl vytvořen podle daného datasheetu [4] (popis komunikační jednotky MicaZ je uveden v následující kapitole).

Každá komunikační jednotka je definována svým .ini souborem, kde máme možnost nastavit tyto parametry:

- `dataRate` rychlosti přenosu paketu.
- `modulationType` typ modulace. 0 – ideální komunikační jednotka, 1 – FSK, 2 – BPSK, QPSK, OQPSK.
- `encodingType` typ kódování, zatím implementováno pouze NRZ.
- `noiseFloor` tepelný šum v dBm za nepřítomnosti dalších interferencí, který závisí na teplotě a použité šířce pásma.
- `receiverSensitivity` přijímací citlivost v dBm, ačkoli je parametrem komunikační jednotky, je používán modulem bezdrátového kanálu k účinnějším výpočtům.

- `rxPower`, `listenPower`, `sleepPower` udávají spotřebu energie v mW v jednotlivých stavech.
- `txPowerLevels` udává různý výstupní výkon skrz tx úroveň v dBm.
- `txPowerConsumptionPerLevel` nastavení celkové spotřeby energie komunikační jednotky.
- `txPowerLevelUsed` nastavení aktuální používanou velikost výkonu.
- `txModeUsed` určuje, jak bude proveden přenos paketu. Paket může být buď 1) poslán, 2) jednou zkontrolovat výstupní odezvu nosného kanálu a když je volno, poslat paket, jinak ho zahodit, 3) opakovaně kontrolovat volný kanál (což znamená, že by komunikační jednotka měla být opakovaně v naslouchacím režimu) a pak poslat paket.
- `bufferSize` velikost vyrovnávací paměti.
- `maxPhyFrameSize` maximální velikost snímaných vzorků v bytech.
- `phyFrameOverhead` velikost režie snímaných vzorků v bytech.
- Dalších 6 parametrů definuje časové ztráty všech možných přechodů mezi jednotlivými stavy.
- `delayCSValid` nastavení zpoždění pro přenášený nosný signál.

5.3.3 MAC

Soubory obsahující parametry MAC modulu se nacházejí ve standardním adresáři `Simulations/Parameter_Include_Files`.

- `dutyCycle` časový úsek, ve kterém zůstává senzorový uzel v režimu poslouchání kanálu. Pro $(1 - dutyCycle)$ z časového úseku, uzel spí. Tento parametr je důležitý z hlediska úspory elektrické energie, protože můžeme pevně nastavovat dobu naslouchání. Je bezrozměrným číslem, udávaným jako poměr $listening/(sleeping + listening)$.
- `listenInterval` čas, kdy uzel zůstává v režimu poslouchání. Poté, co proběhne jeden naslouchací a spící interval, začíná cyklus znovu. Je žádoucí, aby naslouchací interval byl malý tak, aby spící interval pro daný pracovní cyklus byl malý tak, aby zpoždění v přenosu dat bylo co nejmenší. Pokud nastavíme naslouchací interval příliš malý, a v něm bude potřeba přenést data, nebudou mít možnost k přenosu.

- `BeaconIntervalFraction` V pracovním cyklu potřebujeme získat způsob, jak upozorníme (probudíme) spící senzorový uzel předtím, než začne přenos dat. Tento parametr vyjadřuje část maximálního intervalu daného probouzecího signálu (= spící interval), který náš interval probouzecího signálu doopravdy je. Čím menší, tím méně spotřebované energie, ale menší šance na probuzení daných sousedních senzorových uzlů.

- `numTx` počet pokusů pro přenesení částí dat určených k přenosu. Čím větší tento parametr, tím je spotřebováno více energie, ale může být dosaženo větší účinnosti (dosažených uzlů).

- `reTxInterval` interval mezi opětovnými přenosy.

- `probTx` pravděpodobnost přenosů. Tato hodnota v kombinaci s počtem opětovných přenosů může vytvořit předpokládané množství přenosů na uzel, rovné hodnotě necelých čísel.

- `randomTxOffset` náhodný offset přenosu. Když se rozhodne o určitém přenosu, nepřenáší se ihned, ale čeká se náhodný čas rovnoměrně rozložený v [0.. Random Transmission Offset]. Tato náhodnost významně pomáhá k předejití kolizí.

- `backoffType` interval pozastavení přenosu, pokud je obsazený kanál, je přenos na určitou dobu pozastaven a MAC přepíná komunikační jednotku do spícího režimu. Parametr backoff určuje interval tohoto pozastaveni. Nastaví-li se na 0, potom je pozastaven na dobu spícího intervalu. Nastaví-li se na 1, je pozastaven na konstantní čas *backoffBaseValue*. Nastaví-li se na 2, závisí na následující hodnotě *times*, kdy nebyl kanál vyhodnocen jako čistý. Je pozastaven na dobu $(backoffBaseValue) \cdot (times)$. Nastaví-li se na 3, je opět závislý jako při hodnotě 2 a pozastavení je dáno vztahem $(backoffBaseValue)^{times}$.

5.3.4 Síťový (směrovací) modul

V programu Castalia jsou implementovány dva jednoduché směrovací protokoly:

- První `simpleTreeRouting` založen na stromové struktuře, kde základová stanice vysílá broadcast směrovací pakety s úrovní 0, každý uzel, který tento kontrolní paket po prvé příjme, si nastaví vysílací uzel jako rodič (kořen) a úroveň na úroveň+1. Takto má každý uzel jednoho rodiče (kromě základové stanice), kterého může použít ke směrování dat směrem k základové stanici.

- Druhý `multipathRingsRouting` nepoužívá uzel rodiče, ale pouze úroveň. Když přijde paket z vyšší úrovně, pokouší se ho poslat do další nižší úrovně. To znamená, že pakety jsou posílány více cestami, které jsou zhruba založeny na kruhovém uspořádání uzlů se stejnou hodnotou úrovní.

5.3.5 Modul fyzického procesu

V praxi jsou snímaná data obvykle generována jako náhodné čísla pro senzorové uzly nebo má každý uzel konstantní hodnotu a nebo v lepším případě, je uzlům dáván sled snímaných dat. Pro návrh algoritmů je zapotřebí vytvořit model fyzikálního procesu, který je dostatečně flexibilní a shoduje se skutečnými procesy (např. prostorová korelace dat, časová proměnlivost). Pro tento účel je v programu Castalia vytvořen všeobecný model fyzikálního procesu, který poskytuje snímacím zařízením senzorového uzlu potřebná data.

Základem tohoto modelu jsou zdroje hodnot, které své hodnoty rozptýlí po prostoru. Zdroje se mohou měnit v čase a pozici, tj. změnit svou pozici a hodnotu. Model určující hodnotu fyzického procesu v jistém místě a čase je vyjádřen pomocí rovnice:

$$V(p,t) = \sum_{all\ sources\ i} \frac{V_i(t)}{(K \cdot d_i(t) + 1)^a}, \tag{5.2}$$

kde: $V(p,t)$ udává hodnotu fyzikálního procesu v bodě P, v čase t,
$V_i(t)$ udává hodnotu i-tého zdroje v čase t,
$d_i(t)$ udává vzdálenost od bodu P po i-tý zdroj v čase t,
K, a jsou parametry udávající rozptýlení hodnoty daného zdroje.

Parametry které můžeme nastavovat jsou:

- `SN.physicalProcess[0].multiplicative_k` a `SN.physicalProcess[0].attenuation_exp_a` slouží k nastavení parametrů K a a a jsou vedeny přímo v souboru omnetpp.ini

- `SN.physicalProcess[0].source_i` parametr pro výpočet $V_i(t)$ a $d_i(t)$, kde i je i-tý zdroj. Toto je řetězec ve tvaru: „time pos_x pos_y value; time pos_x pos_y value; time pos_x pos_y value; . . . “ Každá část (time, pos_x, pos_y, value) se nazývá snímek zdroje.

- `SN.physicalProcess[0].max_num_snapshots` maximální možné množství snímků pro všechny zdroje.

- `SN.physicalProcess[0].numSources` počet zdrojů ve fyzickém procesu.

- `SN.physicalProcess[0].inputType` nastavení zda půjde o jednoduchý nebo složitější model. Nastavíme-li ho na 1, pak bude vybrán složitější model a výše zmíněné parametry budou interpretovány. Nastavením na 0, bude výstup fyzického procesu určen výhradně následujícím parametrem.

- `SN.physicalProcess[0].directNodeValueAssignment` přímé nastavení statických hodnot jednotlivým uzlům. Tento parametr je řetězcem a jeho syntaxe je: „(default_value) nodeID_A:value_A nodeID_B: value_B . . . “. Ta znamená, že uzly, jejichž ID je uvedeno v řetězci, nabývají hodnoty spojené s tímto ID a uzly, které nejsou v tomto řetězci, nabývají implicitní hodnoty. Například, jestliže máme řetězec „(0) 6:40“, všechny uzly přijmou hodnotu 0 kromě uzlu 6, který přijme hodnotu 40.

5.3.6 Modul snímacího zařízení

Modul snímacího zařízení souvisí s modulem fyzického procesu, jelikož je i v realitě činnost snímacího zařízení ovlivňována okolním prostředím (např. teplotní čidla mohou být ovlivněny klimatickými podmínkami + blízkým ohněm). Parametry programu Castalia, které souvisí s modulem snímacího zařízení jsou:

- `SN.node[*].nodeSensorDevMgr.numSensingDevices` počet různých typů snímacích zařízení nacházejících se na senzorovém uzlu.

- `SN.node[*].nodeSensorDevMgr.sensorTypes` řetězec jména pro každý z typů snímacího zařízení (prostorově oddělen)

- `SN.node[*].nodeSensorDevMgr.pwrConsumptionPerDevice` pole řetězců spotřebované energie na vzorek (v mJ) pro každé snímací zařízení (prostorově odděleno).

- `SN.node[*].nodeSensorDevMgr.corrPhyProcess` pole řetězců, které udává index fyzického procesu s kterým jednotlivé snímací zařízení komunikuje (prostorově oddělen).

- `SN.node[*].nodeSensorDevMgr.maxSampleRates` pole řetězců, které udává maximální rychlost vzorků za sekundu pro každý typ snímacího zařízení (prostorově oddělen).

- `SN.node[*].nodeSensorDevMgr.devicesBias` pole řetězců ovlivňující každé snímací zařízení. Obsahuje parametr sigma, který slouží k nastavení nových snímacích zařízení.

Tab. 5.1: Vybrané vlastnosti komunikační jednotky MicaZ

Processor/Radio Board	MPR2400CA	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Current Draw	8 mA	Active mode
	$< 15\mu$ A	Sleep mode
RF Transceiver		
Frequency band	2400 MHz to 2483.5 MHz	
Transmit (TX) data rate	250 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-90 dBm (min), -94 dBm (typ)	
Outdoor Range	75 m to 100 m	1/2 wave dipole antenna, LOS
Indoor Range	20 m to 30 m	1/2 wave dipole antenna
Current Draw	19.7 mA	Receive mode
	11 mA	TX, -10 dBm
	14 mA	TX,-5 dBm
	17.4 mA	TX, 0 dBm
	20μA	Idle mode, voltage regular on
	1μA	Sleep mode, voltage regulator off
Electromechanical		
Battery	2X AA	batteries Attached pack
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack

- `SN.node[*].nodeSensorDevMgr.devicesNoise` pole řetězců se sigmou šumu pro jednotlivé snímací zařízení. Slouží k nastavení hodnoty šumu jednotlivých zařízení.

5.4 Komunikační jednotka MicaZ

Modul MicaZ (obr. 5.3) je vytvořen v souladu s normou IEEE 802.15.4 a standardem ZigBee. Pracuje v bezlicenčním pásmu 2.4.až 2.48 GHz. Používá modulaci DSSS (Direct sequence spread spectrum), která je odolná proti RF rušení a poskytuje tak základní zabezpečení dat. Přenosová rychlost je až 250 kb/s. Je založen na otevřeném operačním systému TinyOS, používaným v sítích typu mash, umožňuje se tak přizpůsobit jednotlivým komunikačním protokolům a technickým prostředkům. komunikace probíhá na MAC úrovni kódován algoritmem AES-128 (128bitový algoritmus Advanced Ecryption Standard). Je určen pro vnitřní monitorování a zabezpečení budov, snímání zvuků, videa a vibrací a dalších rychle snímaných dat, k implementaci do rozsáhlých sítí (1000 a více bodů)[4]. Vybrané vlastnosti jednotky MicaZ jsou uvedeny v tabulce 5.1.

Obr. 5.3: Komunikační jednotka MicaZ

Obr. 5.4: Komunikační jednotka Mica2

5.5 Komunikační jednotka Mica2

Modul MicaZ (obr. 5.4) je třetí generací bezdrátových platforem pro smart senzory. Umožňuje vícekanálový přenos na frekvencích 868/916 MHz. Používá modulaci FSK (frequency-shift keying). Přenosová rychlost je 19.2 kb/s. Je založen na otevřeném operačním systému TinyOS, používaným v sítích typu mash, umožňuje se tak přizpůsobit jednotlivým komunikačním protokolům a technickým prostředkům. Komunikace probíhá na MAC úrovni kódován algoritmem AES-128 (128bitový algoritmus Advanced Ecryption Standard)[5]. Vybrané vlastnosti jednotky MicaZ jsou uvedeny v tabulce 5.2.

5.6 Vlastní simulace

Pro základní simulaci je navržena senzorová síť o rozměrech 50 × 50m s 9 senzorovými uzly s vlastním rozmístěním (Castalia umožňuje definovat automatické

Tab. 5.2: Vybrané vlastnosti komunikační jednotky Mica2

Processor/Radio Board	MPR400CB	Remarks
Processor Performance		
Program Flash Memory	128K bytes	
Measurement (Serial) Flash	512K bytes	> 100,000 Measurements
Configuration EEPROM	4K bytes	
Serial Communications	UART	0-3V transmission levels
Current Draw	8 mA	Active mode
	$< 15\mu$ A	Sleep mode
Multi – Channel Radio		
Center Frequency	868/916 MHz	ISM bands
data rate	19,2 kbps	
RF power	-24 dBm to 0 dBm	
Receive Sensitivity	-98 dBm (min)	
Outdoor Range	500ft	1/4 Wave dipole, line of sight
Current Draw	27 mA	Transmit with maximum power
	10 mA	TX,Receive
	14 mA	TX,-5 dBm
	$< 1\mu$A	Sleep
Electromechanical		
Battery	2X AA	Attached pack
Size (in)	2.25 x 1.25 x 0.25	Excluding battery pack
(mm)	58 x 32 x 7	Excluding battery pack

rozmístění do mřížky, náhodně do mřížky či náhodné rovnoměrné rozmístění) viz. obr. 5.5. V jednotlivých simulacích jsou porovnávány výsledky při použití radiové komunikační jednotky Mica2 a MicaZ a to pomocí jednotlivých kol simulací, kdy v prvním kole je použit rádiový modul s parametry komunikační jednotky Mica2 a v druhém kole rádiový modul s parametry komunikační jednotky MicaZ. Jelikož výsledky všech simulací jsou velice rozsáhlé, budou zde uvedeny jen nejdůležitější poznatky. Výstupní soubor obsahující všechny provedené simulace je přiložen spolu s programem na disku CD. Souřadnice jednotlivých uzlů (v metrech) jsou:

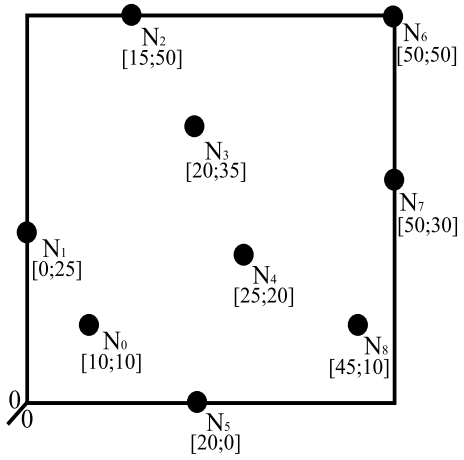
```
SN.node[0].xCoor = 10   SN.node[3].xCoor = 20   SN.node[6].xCoor = 50
SN.node[0].yCoor = 10   SN.node[3].yCoor = 35   SN.node[6].yCoor = 50

SN.node[1].xCoor = 0    SN.node[4].xCoor = 25   SN.node[7].xCoor = 50
SN.node[1].yCoor = 25   SN.node[4].yCoor = 20   SN.node[7].yCoor = 30

SN.node[2].xCoor = 15   SN.node[5].xCoor = 20   SN.node[8].xCoor = 45
SN.node[2].yCoor = 50   SN.node[5].yCoor = 0    SN.node[8].yCoor = 10
```

1. experiment se základními simulacemi

Jako první jsou spuštěny simulační metody, které byly popsány dříve, s defaultním nastavením, tedy s parametry jednotlivých modulů tak, jak jsou standardně nastaveny.



Obr. 5.5: Rozmístění uzlů navrhované senzorové sítě

valuePropagation

```
         Mica2                          MicaZ
Node [0] Value: 40.1312          Node [0] Value: 40.1312
Node [0] spent energy: 2.22099   Node [0] spent energy: 2.22019
Node [1] Value: 40.1312          Node [1] Value: 40.1312
Node [1] spent energy: 2.22099   Node [1] spent energy: 2.22019
       .                                .
       .                                .
       .                                .
Node [8] Value: 40.1312          Node [8] Value: 40.1312
Node [8] spent energy: 2.22099   Node [8] spent energy: 2.22019
```

Z výsledků aplikace valuePropagation vidíme celkovou spotřebu jednotlivých senzorových uzlů při snímání a komunikaci. Jelikož jsou všechny uzly stejné a všechny přenáší stejnou hodnotu, mají všechny stejnou spotřebu, a to při použití obou komunikačních jednotek. Rozdíl spotřebované energie za použití různých komunikačních jednotek je velmi malý, což je dáno nastavením jednotlivých parametrů modulů, kdy se jednotlivé vlastnosti obou komunikačních jednotek příliš neprojeví.

ConnectivityMap Uzly s komunikační jednotkou Mica2 jsou spojeny s pravděpodobností 40-70%. S použitím komunikační jednotky MicaZ jsou všechny uzly propojeny téměř se 100% pravděpodobností, tedy dokonale. Z čehož vyplývá, že komunikační jednotka MicaZ má při téměř stejné spotřebě elektrické energie mnohem lepší schopnost navázat spojení s ostatními uzly.

2. experiment se změnou parametrů

Ve druhém experimetnu je poukázáno na závislost spotřebované energie na nastavení jednotlivých parametrů modulů. Jelikož je u ostatních modulů vybráno reálné

chování, necháváme jejich parametry v původním nastavení a jsou měněny pouze parametry MAC modulu, který nejvíce souvisí s vysílací režií a tedy se spotřebou jednotlivých senzorových uzlů. Jedná se o tyto parametry:

```
dutyCycle, listenInterval, BeaconIntervalFraction,
numTx, reTxInterval, backoffBaseValue
```

V první simulaci byly tyto parametry nastaveny následovně, berme je jako výchozí nastavení:

```
dutyCycle=1 ;listenInterval=10; BeaconIntervalFraction=0;
numTx=1; reTxInterval=0; backoffBaseValue=16
```

Při dalších simulacích byly v jednotlivých kolech nejdříve měněny parametry samostatně, a to vždy zvlášť pro obě komunikační jednotky. Chování sítě při změně jednotlivých parametrů za použití komunikační jednotky Mica2 bylo:

- Změnou `dutyCycle` na hodnotu menší jak 1 bylo dosaženo menší spotřeby senzorových uzlů na úkor toho, že žádný z uzlů, kromě senzoru N_6, nezískal snímanou hodnotu. Při hodnotě $0,1$ byla spotřeba uzlů $0,22J$, při $0,2$ $0,44J$, atd. Od hodnoty $0,8$ začaly snímanou hodnotu získávat i ostatní uzly až po hodnotu 1 kde, jak je uvedeno v první simulaci, danou snímanou hodnotu získává každý uzel.

- Změna `listenInterval` byla patrná až při nižších hodnotách předchozího parametru (dutyCycle=0,1). Při malé hodnotě řádově jednotek ms nezískaly uzly snímanou hodnotu a jejich spotřebovaná energie mírně vzrostla (oproti $0,22J$ na $0,23J$), při větší hodnotě (100ms) získaly snímanou hodnotu již všechny uzly a spotřeba se nijak výrazně nezvýšila (řádově o jednotky mJ).

- Změnou `BeaconIntervalFraction` bylo dosaženo získání snímané hodnoty více uzly. Při nastavení na hodnotu 1, při (dutyCycle=0,1) získaly danou hodnotu uzly N_0, N_1, N_3 a N_7 a jejich spotřebovaná energie z $0,22J$ na $0,23J$. Při tomto nastavení bylo získáno nejlepších výsledků co se týče spojení a spotřeby sítě s vhodným nastavením naslouchacího intervalu (listenInterval=10-20ms).

- `numTx` měl nejlepší vliv v hodnotě 1, kdy všechny uzly získaly snímanou hodnotu a spotřebovaná energie byla $0,27J$. Při hodnotě 0, nezískal ani jeden uzel snímanou hodnotu a při vyšších hodnotách, byla pro danou síť pouze větší spotřeba eletrické energie jednotlivých senzorových uzlů.

- Další parametry na danou simulovanou síť neměly vliv, a proto o nich dále nebude pojednáváno.

Chování sítě při změně jednotlivých parametrů za použití komunikační jednotky Mica2 bylo téměř stejné jako za použití Mica2 pouze s tím rozdílem, že senzorové uzly měly menší spotřebu elektrické energie a snímanou hodnotu získaly všechny senzorové uzly.

3. experiment s heterogenní síti

Doposud byly simulace prováděny na homogenní síti, kde všechny uzly měly stejné vlastnosti. Nyní bude síť rozdělená do tří skupin po třech senzorových uzlech. Každá skupina senzorových uzlů bude mít stejné vlastnosti. V první skupině se budou nacházet uzly N_0, N_1 a N_2, ve druhé N_3, N_4 a N_5 a ve třetí N_6, N_7 a N_8. Nastavovány jsou pouze parametry z předchozí simulace, které měly radikální vliv na spotřebu senzorových uzlů a na distribuci snímané hodnoty. Jsou opět vytvořeny situace používající jednotlivě komunikační jednotky Mica2 a MicaZ. Parametry jsou voleny tak, aby došlo k nejefektivnějším výsledkům. Při následujícím nastavení

```
#***************1.skupina*********************
SN.node[0..2].networkInterface.MAC.dutyCycle = 0.5
SN.node[0..2].networkInterface.MAC.listenInterval = 5
SN.node[0..2].networkInterface.MAC.beaconIntervalFraction = 0.9
SN.node[0..2].networkInterface.MAC.numTx = 1
#***************2.skupina*********************
SN.node[3..5].networkInterface.MAC.dutyCycle = 0.8
SN.node[3..5].networkInterface.MAC.listenInterval = 5
SN.node[3..5].networkInterface.MAC.beaconIntervalFraction = 0.1
SN.node[3..5].networkInterface.MAC.numTx = 1
#***************3.skupina*********************
SN.node[6..8].networkInterface.MAC.dutyCycle = 0.2
SN.node[6..8].networkInterface.MAC.listenInterval = 2
SN.node[6..8].networkInterface.MAC.beaconIntervalFraction = 0.2
SN.node[6..8].networkInterface.MAC.numTx = 1
```

jsou výsledky simulace:

Mica2	MicaZ
Node [0] Value: 0	Node [0] Value: 40.1312
Node [0] spent energy: 1.12092	Node [0] spent energy: 1.12172
Node [1] Value: 0	Node [1] Value: 40.1312
Node [1] spent energy: 1.1208	Node [1] spent energy: 1.12159
Node [2] Value: 0	Node [2] Value: 40.1312
Node [2] spent energy: 1.12112	Node [2] spent energy: 1.12196
Node [3] Value: 0	Node [3] Value: 40.1312
Node [3] spent energy: 1.79369	Node [3] spent energy: 1.79409
Node [4] Value: 0	Node [4] Value: 40.1312
Node [4] spent energy: 1.79307	Node [4] spent energy: 1.7935
Node [5] Value: 0	Node [5] Value: 40.1312
Node [5] spent energy: 1.79314	Node [5] spent energy: 1.7936

```
Node [6] Value: 40.1312            Node [6] Value: 40.1312
Node [6] spent energy: 0.457055    Node [6] spent energy: 0.455667
Node [7] Value: 0                  Node [7] Value: 40.1312
Node [7] spent energy: 0.455161    Node [7] spent energy: 0.455835
Node [8] Value: 0                  Node [8] Value: 40.1312
Node [8] spent energy: 0.455069    Node [8] spent energy: 0.455599
```

Opět komunikační jednotka MicaZ má lepší vlastnosti. Kdybychom provedli odlišné nastavení parametrů jednotlivých skupin, nedošlo by k získání snímané hodnoty všemi uzly, a to ani pro MicaZ. Například nastavíme-li dutyCycle v první skupině na malou hodnotu, bude spotřebovaná energie minimální, ale senzorové uzly nestihnou přijmout snímanou hodnotu, jelikož po většinu času spí. Proto musí být hodnota volena, spolu s parametrem listenInterval, tak aby uzly byly dostatečnou dobu aktivní. Parametr beaconIntervalFraction poté udává pravděpodobnost vzbuzení uzlů. U druhé skupiny musíme opět zajistit, aby senzorové uzly získaly hodnotu od uzlu N_6 a dále ji poskytly první skupině. Ve třetí skupině nepotřebujeme mít uzly tak dlouho vzhůru, jelikož sousedí přímo s uzlem N_6, který je zdrojem snímané hodnoty. Nastavíme-li jednotlivé parametry vysoké, bude spotřeba senzorových uzlů zbytečně veliká.

6 VÝSLEDKY STUDENTSKÉ PRÁCE

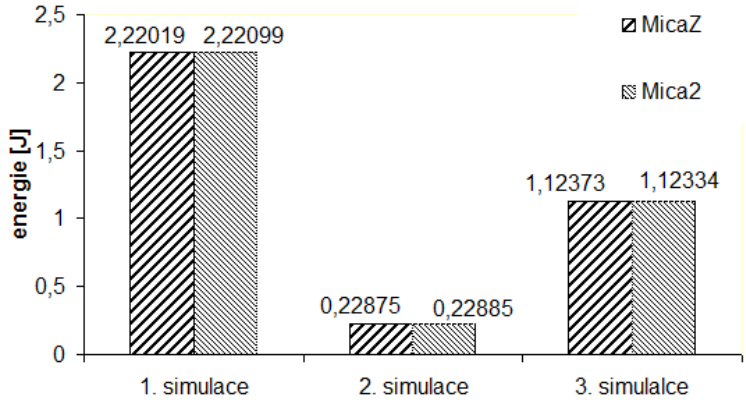
6.1 Výsledky

V jednotlivých experimentech bylo poukázáno na vliv MAC parametrů na úspěšnost spojení senzorových uzlů a jejich spotřebě elektrické energie v dané senzorové síti za použití dvou komunikačních jednotek Mica2 a MicaZ.

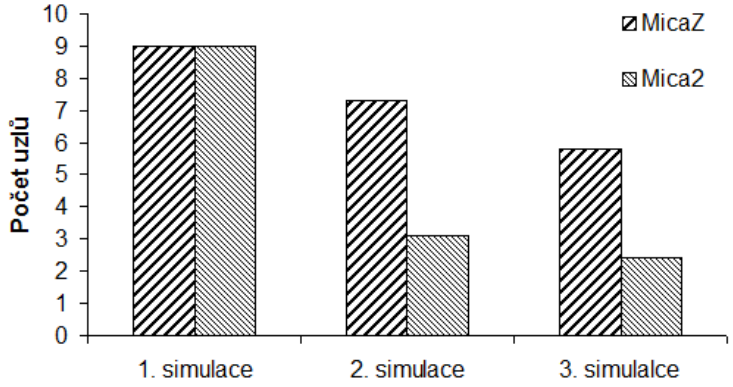
Již v prvním experimentu bylo poukázáno, že komunikační jednotka MicaZ nabízí lepší vlastnosti než Mica2 co se týče možnosti propojení senzorových uzlů a spotřeby elektrické energie. Při použití MicaZ získaly snímanou hodnotu všechny senzorové uzly se 100% pravděpodobností příjmu, zato při použití Mica2 snímanou hodnotu nezískal žádný z uzlů, kromě uzlu N_6, který byl jejím zdrojem, a to při téměř stejné spotřebě jednotlivých uzlů.

V druhém experimentu jsou zkoumány vlastnosti MAC modulu ovlivňující vysílací režii, která má největší vliv na spotřebu, a tedy i životnost senzorových uzlů (obr. 4.1). Parametry mající největší vliv jsou dutyCycle – pracovní cyklus, v programu Castalia definovaný jako poměr naslouchací interval/(naslouchací+spící interval); listenInterval – časový úsek, kdy senzorový uzel naslouchá; BeaconIntervalFraction – udává interval probouzecího signálu; numTx – počet pokusů přenosu dat. Čím menší bude parametr dutyCycle, tím méně elektrické energie spotřebují senzorové uzly, na úkor získání snímané hodnoty. Naopak čím větší bude listenInterval, bude pravděpodobnost příjmu dat větší a tím také spotřebovaná energie. Při zvyšujících hodnotách parametru BeaconIntervalFraction byla předávána snímaná hodnota více uzlům, v důsledku toho, kolik se jich probudilo a jak dlouho zůstaly v naslouchacím režimu. Parametr numTx měl pro navrženou senzorovou síť význam pouze při hodnotách 0 nebo 1, kdy stačil pouze jeden opětovný přenos, aby daná snímaná hodnota byla poskytnuta dalším uzlům.

V třetím experimentu byla simulována heterogenní síť, kde senzorové uzly byly rozděleny do tří skupin po třech senzorových uzlech se stejnými vlastnostmi jednotlivých parametrů MAC modulu. Simulace byla zaměřena na vytvoření nejefektivnějšího propojení mezi senzorovými uzly tak, aby byla snímaná hodnota přenesena celou sítí, tedy od zdrojového uzlu N_6 až po nejvzdálenější uzel N_0. K tomu byly použity poznatky získané z předchozí simulace a také komunikační jednotka MicaZ, která vykazuje lepší schopnost komunikace mezi jednotlivými uzly a také menší spotřebu elektrické energie při této komunikaci. Výsledkem byla síť s první skupinou (uzly N_0 až N_2) mající parametry nastaveny tak, aby senzorové uzly byly dostatečnou dobu v naslouchacím režimu, aby přijaly snímanou hodnotu. Druhá skupina (uzly N_3 až N_5) měly parametry nastaveny o něco vyšší, jelikož danou hodnotu musely nejenom přijmout (od třetí skupiny), ale také odeslat dál (první skupině). Ve



Obr. 6.1: Průměrná spotřebovaná energie senzorového uzlu.



Obr. 6.2: Průměrný počet aktivních senzorových uzlů.

třetí skupině (uzly N_6 až N_8) byly parametry nastaveny na co nejmenší hodnoty, protože se v této skupině nachází zdrojový uzel N_6, který poskytuje snímanou hodnotu dalším uzlům a tedy uzly nepotřebují tolik času na naslouchání.

Výsledná průměrná spotřebovaná energie jednotlivých simulací za použití komunikačních jednotek Mica2 a MicaZ je znázorněna na obr. 6.1. Průměrný počet senzorových uzlů, které během simulací získaly snímanou hodnotu, je znázorněn na obr. 6.2.

7 ZÁVĚR

Cílem této práce byl výzkum technologií bezdrátových senzorových sítí se zaměřením na zabezpečení komunikace těchto sítí a vytvoření modelu bezdrátové senzorové sítě v programu Castalia s ukázkou energetických nákladů senzorových uzlů za použití komunikačních jednotek Mica2 a MicaZ a jejich vzájemné porovnání.

Jako první práce pojednává se základními vlastnostmi WSN, s jejich složením a vlastnostmi jednotlivých částí a se širokou oblastí využití. Poukazuje na jednotlivé odlišnosti od běžných bezdrátových sítí (wifi), na jednotlivé omezení při návrhu, jak už sítí samotných, tak aplikací pro tyto sítě, kde hlavní omezující vlastností je zdroj energie senzorového uzlu.

Dále pojednává o základních metodách šifrování dat, symetrické a asymetrické kryptografií a jejich implementací na WSN sítě. Jako vhodnou šifrovací metodou u WSN je brána symetrická kryptografie a její různé modifikace, která není tak energeticky náročná jako asymetrická. Poukazuje na bezpečnostní požadavky přenosu dat a udává stručný přehled možných útoků na senzorovou síť.

V poslední části se práce zabývá návrhem modelu bezdrátové senzorové sítě v simulačním programu Castalia. Jsou zde popsány jednotlivé části programu Castalia a jejich parametry, které lze pro danou simulaci nastavovat. Byly provedeny tři experimenty na této síti pro různá nastavení parametrů jednotlivých modulů programu Castalia se zaměřením na energetické náklady. Bylo ověřeno, že největší spotřebu elektrické energie senzorového uzlu má vysílací režie, která v programu Castalia byla řízena pomocí MAC modulu a nastavením parametrů souvisejících s jednotlivými pracovními režimy senzorových uzlů. Pro jednotlivé experimenty byly použity modely komunikačních jednotek Mica2 a MicaZ. Lepších vlastností z hlediska spotřeby elektrické energie a schopnosti propojení jednotlivých senzorových uzlů dosáhla komunikační jednotka MicaZ, kde se spotřeba uzlů pohybovala řádově o desítky mJ menší, něž při použití komunikační jednotky Mica2 a z hlediska propojení uzlů, bylo za použití MicaZ propojeno až dvojnásobek senzorových uzlů nežli s Mica2.

LITERATURA

[1] BOULIS, Athanassios. *Castalia A simulator for Wireless Sensor Networks User's Manual.* [s.l.] : [s.n.], 2008. 40 s. Dostupný z WWW: <http://castalia.npc.nicta.com.au/pdfs/Castalia%20-%20User%20Manual.pdf>.

[2] BURDA, Karel. *BEZPEČNOST INFORMAČNÍCH SYSTÉMŮ.* [s.l.], 2005. 104 s. FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ. Souhrn přednášek. Dostupný z WWW: <http://adela.utko.feec.vutbr.cz/index.php?option=comv_wrapperv&Itemid=23>.

[3] *Installation directions for Castalia in Linux : How to Install Castalia* [online]. c2004 [cit. 2008-11-10]. Dostupný z WWW: <http://castalia.npc.nicta.com.au/installationLinux.php>

[4] CROSSBOW , Technology. *MICAz datasheet.* 6020-0060-04 Rev A, s. 2. Dostupný z WWW: <http://www.cmt-gmbh.de/MICAz.pdf>.

[5] CROSSBOW , Technology. *MICA2 datasheet.* 6020-0042-08 Rev A, s. 2. Dostupný z WWW: <http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf>.

[6] HEIDEMANN, John, GOVINDAN, Ramesh. *Embedded Sensor Networks.* [s.l.] : [s.n.], [200-?]. 18 s. Dostupný z WWW: <http://cs.usc.edu/~ramesh/papers/sensornet_handbook.pdf>.

[7] KUNDEROVÁ, Ludmila. *Bezpečnost IS/IT : Kryptografické systémy* [online]. 2008 , 11/04/2008 [cit. 2008-11-10]. Dostupný z WWW: <https://akela.mendelu.cz/~lidak/bis/8kryp.htm>.

[8] KůR, Jiří. *Secure Routing Protocols for Wireless Sensor Networks.* [s.l.], 2008. vii, 53 s. MASARYK UNIVERSITY FACULTY OF INFORMATICS. Vedoucí diplomové práce Mgr. Petr Švenda. Dostupný z WWW: <http://is.muni.cz/th/98692/fi_m/diplomova_prace_JK.pdf>.

[9] LEWIS, F. L. *Wireless Sensor Networks.* Associate Director for Research Head, Advanced Controls, Sensors, and MEMS Group Automation and Robotics Research Institute The University of Texas at Arlington [s.l.] : [s.n.], 2004. 18 s.

Dostupný z WWW:
<http://arri.uta.edu/acs/networks/WirelessSensorNetChap04.pdf>.

[10] *OMNeT++ Community Site* [online]. c2008 [cit. 2008-11-10]. Dostupný z WWW: <http://www.omnetpp.org>.

[11] *OMNeT++ User Manual.* [s.n.], [200-?]. Dostupný z WWW:
<http://www.omnetpp.org/doc/manual/usman.html#toc_1>.

[12] ORLÍKOVÁ, Soňa. *Inteligentní senzory.* [s.l.], 2003. 16 s. Ústav automatizace a měřicí techniky FEKT VUT Brno. Přednáška. Dostupný z WWW:
<www.roznovskastredni.cz/dwnl/pel2003/1/Orlikova.ppt>.

[13] PERILLO, Mark A., HEINZELMAN, Wendi B. *Wireless Sensor Network Protocols.* Department of Electrical and Computer Engineering University of Rochester Rochester, NY, USA. [s.l.] : [s.n.], 2005. 35 s. Dostupný z WWW:
<http://www.ece.rochester.edu/courses/ECE586/readings/perillo.pdf>.

[14] PERRIG, Adrian, et al. *SPINS: Security Protocols for Sensor Networks.* Department of Electrical Engineering and Computer Sciences University of California, Berkeley. [s.l.] : [s.n.], 2001. 11 s. Dostupný z WWW:
<http://www.ece.cmu.edu/~adrian/projects/mc2001/mc2001.pdf>.

[15] SAXENA, Mohit. *Security in Wireless Sensor Networks A Layer-based Classification.* CERIAS Tech Report 2007-04; Center for Education and Research in Information Assurance and Security, Purdue University. [s.l.] : [s.n.], 2007. 10 s. Dostupný z WWW:
<http://pages.cs.wisc.edu/~msaxena/papers/2007-04-cerias.pdf>.

[16] *Seznam encyklopedie : Symetrická kryptografie, Asymetrická kryptografie, Šifrovací klíč* [online]. c1996-2008 , 2007 [cit. 2008-11-10]. Dostupný z WWW:
<http://encyklopedie.seznam.cz>.
<http://encyklopedie.seznam.cz/heslo/444869-symetricka-kryptografie>.
<http://encyklopedie.seznam.cz/heslo/443621-asymetricka-kryptografie>.
<http://encyklopedie.seznam.cz/heslo/183704-sifrovaci-klic>.

[17] VOJÁČEK, Antonín. WirelessHART - novinka na poli průmyslové komunikace. *Průmyslové sběrnice a komunikace* [online]. 2007 [cit. 2008-11-10]. Dostupný z WWW:
<http://automatizace.hw.cz/wirelesshart-novinka-na-poli-prumyslove-komunikace>.

[18] Wikipedia, the free encyclopedia. *Wikipedia: Wireless sensor network* [online]. 2001- , last modified on 1 December 2008 [cit. 2008-11-10]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Wireless_Sensor_Networks>. <http://en.wikipedia.org>.

[19] WOJCIASZYK, Petr. *Smart Sensors and Wireless Networks: Inteligentní senzory abezdrátové sítě.* VŠB-TU Ostrava, FS, katedra ATŘ. [s.l.]: [s.n.], 2005. 9 s. Dostupný z WWW: <http://www.fs.vsb.cz/akce/2005/asr2005/Proceedings/papers/519.pdf>.

[20] ZUNIGA, Marco, KRISHNAMACHARI, Bhaskar. *Analyzing the Transitional Region in Low Power Wireless Links.* First IEEE International Conference on Sensor and Ad hoc Communications and Networks (SECON), Santa Clara, CA, October 2004.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

A	nešifrovaná data
B	šifrovací algoritmus
C	šifrovaná data
CPU	central processing unit – procesor
CRC	cyclic redundancy check – kontrolní součet
D	dešifrovací algoritmus
$d_i(t)$	vzdálenost od bodu P po i-tý zdroj v čase t
DoS	denial of service - odmítnutí služby
DSP	digitální signální procesor nebo také digitální signálový procesor
DSSS	Direct sequence spread spectrum
EKG	elektrokardiogram
EMG	elektromyografie
η	exponent dráhové ztrátovosti
FSK	frequency-shift keying
GUI	graphical user interface – grafické uživatelské rozhraní
ID	identifikační číslo
IEEE	Institut pro elektrotechnické a elektronické inženýrství
IT	information technology – informační technologie
K_B	šifrovací klíč
K_D	dešifrovací klíč
MAC	message authentication code
NIST	americký Národní úřad pro standardy a technologie
PC	personal computer – osobní počítač
PDA	personal digital assistant – osobní digitální pomocník

PL(d)	dráha ztrát o vzdálenosti d
$PL(d_0)$	známá dráha ztrát o referenci vzdálenosti d_o
Qos	quality of service – kvalita služeb
RF	radio frequency
RNGs	random number generators – generátory náhodných čísel
RTS	request to send
σ	směrodatná odchylka
SIR	signal to interference ratio
SNEP	sensor-network encryption protocol
SNR	signal to noise ratio – poměr signál šum
SPINS	security protocols for sensor network
TESLA	timed efficient stream loss – tolerant authentication
μTESLA	micro timed efficient stream loss-tolerant authentication
$V_i(t)$	hodnota i-tého zdroje v čase t
$V(p,t)$	hodnota fyzikálního procesu v bodě P a v čase t
WSN	wireless sensor network
X_σ	náhodná proměnná s gaussovským rozložením se střední hodnotou okolo nuly

SEZNAM PŘÍLOH

A PRVNÍ PŘÍLOHA

A.1 Instalace programu Omnet++

Tento návod je převzat z [10] a je doplněn o vlastní zkušenosti při instalaci na operační systém Linux Ubuntu 7.10.

Chceme-li si na našem PC zprovoznit program Castalia a začít v něm pracovat, musíme si prvně nainstalovat program Omnet++ na jehož základě Castalia běží. V literatuře [3] je doporučeno programy instalovat pod operačním systémem Linux, což doporučuji také, jelikož při instalaci pod Windows jsem měl spoustu problémů jako např. instalace všech potřebných programů, kompilátorů a podobných potřebných věcí, které jsou již v operačním systému Linux implementovány.

Jako první je samozřejmě zapotřebí daný program stáhnout. Archiv se zdrojovým kódem programu Oment stáhneme z jeho domovské stránky:

http://www.omnetpp.org/

Nejrychlejší způsobem je stáhnout tento archiv z pravého navigačního sloupce v sekci Quick download.

Po uložení tohoto archivu (pro můj případ aktuální OMNetpp-3.3p1-src.tgz), nejlépe do adresáře, kde program Oment budeme chtít mít nainstalován, rozbalíme tento archiv zadáním příkazu:

```
tar -xvzf omnetpp-x.x-src.tgz
```

kde x.x značí aktuální verzi programu Omnet.
Tím se nám vytvoří složka s názvem omnetpp-x.x, která obsahuje veškeré zdrojové kódy tohoto programu.
Poté do našeho spouštěcího souboru .bashrc nebo bash_profile, používáme-li bash; .profile používáme-li sh-like shell, přidáme tyto dva řádky:

```
export PATH=$PATH:/home/jurka/prace/omnetpp-x.x/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/jurka/prace/omnetpp-
x.x/lib
```

část příkazu „/home/jurka/prace/“ nahradíme vlastní cestou k adresáři omentpp-xx

Pokuds nevíme, kde se nachází náš spouštěcí soubor, zadáme příkaz

```
locate .bash_profile
```

popřípadě

```
locate .bashrc
```

tento příkaz nám vypíše cesty ke všem souborům typu .bashrc (.bash_profile). My si vybereme ten, který náleží našemu profilu, jelikož u našeho PC může mít vytvořený učet více uživatelů (př.: /home/jurka/.bashrc)

Jakmile přidáme tyto řádky do spouštěcího souboru, je vhodné se odhlásit a znovu přihlásit, aby se námi provedené změny aplikovaly.

Dalším krokem je upravení konfiguračního souboru configure.user. Ten se nachází v domovském adresáři (hlavní složka) omnetpp-x.x, tam tedy přejdeme. Úpravu provedeme pomocí editoru vim a to tak, že daný soubor v něm spustíme zadáním:

```
vim configure.user
```

Upravit si soubor můžeme, jak nám bude libo, ale doporučená konfigurace je následující:

řádek obsahující text `NO_TCL=true` najdeme a odkomentujeme (smažeme křížek před tímto řádkem), tím jsme zakázali Tcl/Tk, což znamená, že Omnet nebude spouštěn v GUI, které nepotřebujeme, jelikož Castalia nemá grafické prostředí

změnit řádek `WITH_PARSIM` na `no` (kdyby byl daný na `yes`, mohly by se při kompilaci vyskytnout komplikace)

nakonec se ujistíme, že je zapnuto dynamické kompilování pomocí NED, a to: `WITH_NETBUILDER = yes`

(pokud nejsme zběhlí v práci s textovým editorem, budou nám stačit následující příkazy:
písmeno „i“ slouží pro připisování do souboru – editaci vkládání (i jako insert), klávesou „Esc“ zrušíme vkládání
zadáme-li „:w“ a klávesu enter, provedené změny se uloží, zapíší (w jako write)
posledním co potřebujeme je opuštění editačního prostředí, což provedeme zadaním „:q“ a enter (q jako quit))

Konečnou fází instalace programu Omnet je zadání příkazů:

```
./configure
```

```
make
```

příkaz „./configure“ slouží pro nakonfigurování konfiguračních souborů (knihoven), příkaz „make“ pro samotnou instalaci (kompilaci) programu.

Při konfigurování (vykonávání příkazu „./configure“) mohou nastat různé chyby. Tyto chyby jsou způsobeny nepřítomností důležitých balíčků, které konfigurační soubor potřebuje, proto stačí, když tyto balíčky nainstalujeme.

Jsou zapotřebí tyto balíčky:

apt-built	g++	libxmu-dev
binutils-dev	giftrans	mpi
bison	graphviz	tcl8.4-dev
blt-dev	imagemagick	tetex-extra
doxygen	java-package	tk8.4-dev
expat	libxml2-dev	xsltproc

(Potřebné balíčky se přidávají ve správci balíčků Synaptic: Systém/Správa/Správce balíčků Synaptic)

Po nainstalování výše uvedených balíčku by se konfigurace a samotná instalace měly provést bez problémů. Pozn.: budeme-li chtít někdy v budoucnu překonfigurovat veškeré knihovny programu, provedeme to následovně:
V hlavní složce zadáme tyto příkazy:

```
./configure

  make clean

  make
```

„./configure“ pro provedení našich změn v configure.user
„make clean“ pro odinstalaci programu
„make“ pro opětovnou instalaci se zavedenými změnami

Pro překompilování pouze jedné knihovny stačí přejít do požadované složky, kde se soubor nachází a zadat příkaz „make clean“ a „make“

A.2 Instalace programu Castalia

Tento návod je převzat z [3] a je doplněn o vlastní zkušenosti při instalaci na operační systém Linux Ubuntu 7.10. Opět, chceme-li program Castalia nainstalovat na naš PC musíme v první řadě stáhnout jeho zdrojový kód. Ten je ke stažení v archivu přímo na stránkách jeho tvůrců http://castalia.npc.nicta.com.au/ v sekci „Download Castalia“ po vyplnění jednoduchého dotazníku a potvrzení licenčních podmínek.

Tento archiv (v mém případě Castalia-1.3.tar.gz) uložíme opět do požadované složky, kde ho pomocí následujícího příkazu rozbalíme

```
tar -xvzf Castalia-x.x.tar.gz
```

Tím se nám vytvoří složka se zdrojovými kódy programu Castalia s názvem Castalia-x.x

Přejedeme do adresáře Castalia/config a upravíme soubor "Castalia.config" (opět pomocí editačního programu vim) následujícím způsobem:

řádek `ROOT=$(HOME)/YOUR_PATH_TO_CASTALIA` upravíme tak, aby obsahoval pravou cestu k adresáři Castalia-x.x na našem disku př.:

```
ROOT=$(HOME)/prace/Castalia-1.3/
```

V souboru je psaná důležitá poznámka, abychom přegenerovali „omnetppconfig“ před nainstalováním samotné Castalie, to se provede zadáním následujícího příkazu:

```
opp_makemake -f --genconfig omnetppconfig
```

Přejdeme zpátky do domovského adresáře Castalia-x.x a zadáme příkaz:

```
./makemake
```

počkáme na ukončení skriptu

```
Regenerating nedfiles.lst...
Done
```

A zadáme příkaz, který dokončí instalaci programu:

```
make
```

Tímto je instalace programu Castalia dokončena.

B DRUHÁ PŘÍLOHA

B.1 Obsah přiloženého CD

CD obsahuje elektronickou verzi této práce (*bakalarska_prace.pdf*) a zdrojové soubory programu Castalia s výstupními soubory (`Castalia-Primary-Output.txt`) jednotlivých experimentů nacházejících se v příslušných adresářích:

- `Castalia/Simulations/valuePropagation` pro první simulační metodu
- `Castalia/Simulations/connectivityMap` pro druhou simulační metodu