ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

TECHNICKÁ FAKULTA

Katedra elektrotechniky a automatizace

METODY PRO ZVÝŠENÍ SPOLEHLIVOSTI A BEZPEČNOSTI KOMUNIKACE NA BEZDRÁTOVÝCH SÍTÍCH

Doktorský studijní program:	Energetika
Obor:	Energetika
Vypracoval:	Ing. Ilja Mašík
Školitel:	prof. Ing. Zdeněk Bohuslávek, CSc.

Praha 2013

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením prof. Ing. Zdeňka Bohuslávka, CSc. a použil jen pramenů citovaných v seznamu použité literatury. Tištěná a elektronická verze práce se doslovně shodují.

V Praze 17. 5. 2013 …….....

(Ilja Mašík)

Chci poděkovat všem, kteří se jakkoliv podíleli na této práci, radou či jinak. Poděkování patří zejména mému konzultantovi prof. Ing. Zdeňku Bohuslávkovi, CSc. za informační podporu a poskytnutí odborného zázemí.

Abstrakt:

Práce se zabývá spolehlivostí technologie bezdrátového přenosu dat ZigBee. Cílem práce je zvýšení spolehlivosti přenosu za předpokladu zachování kompatibility se základním standardem. V práci je popsána metoda dopředné korekce chyb a způsob její implementace do protokolu tak, aby nedošlo k narušení kompatibility. Měřením a simulací je v práci prokázán pozitivní přínos metody v případě přenosu dat v silně rušeném prostoru.

Klíčová slova: ZigBee, síť, spolehlivost, dosah, kodování, Hamming, 802.15.4

OBSAH:

I. ÚVOD 1

II. PŘEHLED SOUČASNÉHO STAVU 2

2.1 ZigBee – 802.15.4 3
2.1.1 Typy rámců 5
2.1.1.1 Signální rámec (Beacon Frame) 5
2.1.1.2 Řídící rámec (MAC Command Frame) 6
2.1.1.3 Datový rámec (Data Frame) 7
2.1.1.4 Potvrzovací rámec (ACK Frame) 8
2.1.2 CSMA-CA 9
2.1.3 Beacon-enabled / non-Beacon-enabled 11
2.1.4 Vrstvy sítě ZigBee 12
2.1.4.1 Fyzická vrstva 12
2.1.4.2 MAC vrstva 15
2.1.5 Topologie sítě 15
2.2 Používané přístupy ke zvýšení odolností vůči rušení 16
2.2.1 DSSS 17
2.2.2 OFDM 19
2.2.3 FHSS 20
2.2.4 AFH 20
2.2.5 Další metody založené na dopředné korekci chyb 22
2.3 Zdroje rušení a vznik interferencí 23
2.3.1 Soubor norem 802.11 23
2.3.1.1 WiFi - 802.11b 25
2.3.1.2 WiFi - 802.11g 26
2.3.1.3 WiFi - 802.11n 26
2.3.2 Bluetooth 802.15.1 27
2.3.1 WiMax – 802.16 27
2.3.1 Wireless USB 28
2.3.1 Interference zpožděním vlastního signálu 28

III. CÍLE DISERTAČNÍ PRÁCE 29

IV. ZVOLENÉ METODY ZPRACOVÁNÍ 30

4.1 Zákoná omezení pro provoz bezdrátových sítí 30
4.2 Chybovost přenosového kanálu 31
4.3 Kapacita přenosového kanálu 31
4.4 Úbytek signálu v přenosovém kanálu 32
4.5 Metody pro zvýšení spolehlivosti ZigBee 33

4.6 Teoretické základy bezpečnostních kódů 34
4.6.1 Kódová vzdálenost 34
4.6.1 Minimální Hammingova vzdálenost 35
4.6.2 Informační rychlost kódu 35
4.7 Kódování pro dopřednou korekci chyb 35
4.7.1 Výběr vhodného segmentu kódovacích technik 37
4.8 Kódy blokové lineární cyklické 37
4.8.1 Hammingův kód 37
4.8.1.1 Kodér Hammingova kódu (7,4) 38
4.8.1.2 Dekodér Hammingova kódu (7,4) 39
4.8.2 Rozšířený Hammingův kód 40
4.8.3 Golayův kód 40
4.8.4 BCH kód 41
4.8.4.1 Kodér BCH kódu 42
4.8.4.2 Dekodér BCH kódu 43
4.8.5 Reed-Solomonův kód 44
4.8.5.1 Kodér Reed-Solomonova kódu 45
4.8.5.2 Dekodér Reed-Solomonova kódu 45
4.9 Blokový prokladač dat 46
4.10 Srovnání vlastností různých typů kódů 47
V. VÝSLEDKY DISERTAČNÍ PRÁCE S UVEDENÍM NOVÝCH POZNATKŮ 51
5.1 ZigBee experimentální hardware 51
5.2 Metody získávání dat pro aplikaci dopředné korekce dat 52
5.3 Parametry konstrukce 53
5.3.1 Sběrné měření na pateřním spoji 53
5.3.2 Měření charakteristiky WiFi spoje v různých podmínkách 54
5.3.2.1 Síť WiFi bez zatížení 55
5.3.2.2 Síť WiFi s lehkým provozem 55
5.3.2.3 Síť WiFi se silným provozem 58
5.3.2.4 Souhrn experimentu 59
5.3.3 Zjištění reálného vlivu standardu WiFi na ZigBee 60
5.3.3.1 Experiment koexistence s WiFi – fixní vzdálenost 61
5.3.3.2 Experiment koexistence s WiFi – proměnlivá vzdálenost 66
5.3.3.3 Shrnutí experimentů s koexistencí WiFi 68
5.4 Rozbor a volba vhodné dopředné korekce chyb I. 69
5.5 Implementace Hammingova kódu a blokového prokladače 70
5.5.1 Realizace Hammingova kódu 70
5.5.2 Realizace Blokového prokladače dat 71
5.5.1 Realizace Reed-Solomonova kódu 72
5.6 Rozbor a volba vhodné dopředné korekce chyb II. 73

5.7 Výběr vhodného rozměru blokového prokladače dat 75
5.7.1 Volba blokového prokladače dat – koexistence 802.11g 76
5.7.2 Volba blokového prokladače dat – širokospektrální rušení 77
5.7.3 Volba blokového prokladače dat – shrnutí 79
5.8 Simulace ZigBee přenosu 79
5.8.1 Model I. část 79
5.8.2 Model II. Část 81
5.9 Výsledky simulace ZigBee přenosu 83
5.10 Měření účinků metody při širokospektrálním rušení 84
5.10.1 Výsledky měření 86
5.11 Měření účinků metody při koexistenci s WiFi - 802.11g 89
5.11.1 Výsledky měření 91
VI. ZÁVĚRY A DOPORUČENÍ PRO VYUŽITÍ POZNATKŮ V PRAXI 99
VII. ZÁVĚR 100
POUŽITÁ LITERATURA 102
SEZNAM POUŽITÝCH SYMBOLŮ 106
SEZNAM OBRÁZKŮ A TABULEK 108
VLASTNÍ PUBLIKACE 110
PŘÍLOHY 111

I. ÚVOD

Hlavní cíl disertační práce je zvýšení spolehlivosti datového přenosu ZigBee standardem. Toto téma je důležité z pohledu jak průmyslového využití ZigBee, tak v domácí automatizaci. V obou oborech, kde ZigBee nachází rozsáhle možnosti využití, lze nalézt silné zdroje rušení. V průmyslovém provozu se jedná například o rušení vzniklé jiskřením v komutátorech točivých strojů ale i jinde, nebo rušení vznikající při mikrovlnném ohřevu apod. V domácí automatizaci to jsou mimo již zmíněného mikrovlnného ohřevu především ostatní bezdrátové standardy, které jsou provozovány v pásmu 2,4 GHz. Konečně s rušením koexistujícími bezdrátovými standardy se lze jistě setkat i v průmyslovém provozu. Problematika rušení je v pásmu 2,4 GHz velmi frekventovaná, neboť se jedná o jedno z nejvyužívanějších nelicencovaných pásem. Do nedávna bylo téměř výhradně používáno pro datové přenosy v rámci LAN sítí v přístrojích pro domácí použití. Proto v obydlených oblastech již téměř není možné najít místa, kde pásmo není aktuálně využíváno Wi-Fi sítí. Ačkoli ZigBee technologie umožňuje při startu sítě dynamicky volit komunikační kanál na základě měření energie na dostupných kanálech, problém rušení tím nemusí být zcela eliminován. Koexistenční síť může být v okamžiku měření neaktivní, popř. může dojít ke startu sítě nové.

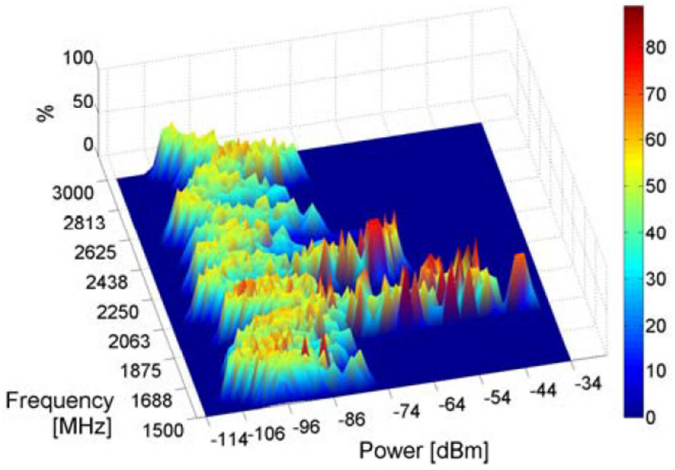
ZigBee je komerční název technologie postavené na standardu IEEE 802.15.4 [1] a lze jej provozovat v několika pásmech. Tato práce se zabývá konkrétně pásmem 2,4GHz, ačkoli obecně lze výsledky práce aplikovat v jakémkoli frekvenčním pásmu. ZigBee je standard, který cílí na nízkorychlostní přenosy dat s nízkou spotřebou energie a vysokou spolehlivostí. Vysoká spolehlivost je zajišťována nejen implementací robustní metody rozprostřeného spektra (DSSS) na fyzické vrstvě, ale také schopností vytvářet síťovou topologii typu mesh, která umožňuje dynamické směrování datových paketů v závislosti na parametrech jednotlivých spojů mezi členy sítě.

Metoda pro zvýšení spolehlivosti, popsaná v práci je vyvinuta, implementována a měřena na mikroprocesorech SoC fy. Texas Instruments, konkrétně na typu CC2530f256. Jedná se o mikrokontrolér s jádrem 8051, rozšířeným o bezdrátovou část, obsaženou přímo na čipu. V práci je popsán také přístup k fyzické vrstvě v softwaru Z-Stack, k níž není za běžných okolností přístup zcela snadný. Přínos metody by měl být především v koncových bodech sítě, kde je zvýšená pravděpodobnost rušení, případně je velmi kritické spolehlivé doručení datového paketu.

II. PŘEHLED SOUČASNÉHO STAVU

Tato část práce se věnuje aktuální problematice bezdrátových přenosů v pásmu 2,4 GHz, běžně využívaným metodám pro zvýšení spolehlivosti a v neposlední řadě ZigBee standardu samotnému. Dále se věnuje také možnostem měření spolehlivosti sítě a přehledu simulačních nástrojů. Současný stav je z hlediska naplnění prostoru radiovými signály v silném rozvoji. Bezdrátových přenosů je využíváno stále častěji ve všech odvětvích, ať již v průmyslu nebo spotřební elektronice, všude nalézají bezdrátové přenosy stále větší uplatnění. Využívání jednotlivých frekvencí a pásem je regulováno zákonem. Z tohoto hlediska je bez licence možno použít pouze pásma 13 MHz, 27 MHz, 40 MHz, 433 MHz, 868 MHz, 915 MHz, 2,4 GHz, 5 GHz, 24 GHz. Z pohledu datové propustnosti a vlastností vlnění je v době dnešních nároků nejvhodnější pásmo 2,4 GHz, 5 GHz. V prvním uvedeném pásmu je v současnosti nejhustší provoz a zároveň největší počet aplikací. V pásmu 2,4 GHz existuje mnoho standardů, které se různě frekvenčně překrývají a využívají různé typy modulací a kódování. Jako příklad lze uvést standardy WiFi, WiMax, Bluetooth nebo Zigbee. U WiFi se jedná hned o několik variant - 802.11b, 802.11g, 802.11n. Pro zvýšení odolnosti vůči rušení jsou ve zmíněných standardech implementovány některé principy, jako např.: DSSS, OFDM a FHSS. Jednotlivým postupům pro zvýšení spolehlivosti se bude práce v další části věnovat podrobněji. Již existuje řada studií o obecném bezdrátovém provozu v pásmu ISM, jako například [3][4]. V Českých podmínkách je zatím podobných měření nedostatek, nicméně hlavní technologie využívající pásmo ISM jsou celosvětově totožné. Pro zajímavost lze uvést výsledky studie provedené ve městě Aachen v Německu. Na Obr. 1 je znázorněna pravděpodobnost výskytu určitých amplitud podle energie a frekvence signálu. Graf je výsledkem časosběrného měření, probíhajícího po 8 dní v pásmu 1500 – 3000 MHz. Na Obr. 1 jsou zřetelné silné špičky, značící vysokou pravděpodobnost výskytu amplitud s energií mezi -106 a -75 dBm. Vzhledem k různorodému výskytu lze předpokládat, že se jedná o větší množství koexistujících standardů [4]. Zřetelné jsou také dva bezdrátové standardy pracující na frekvencích 1900 MHz a 2200 MHz, kde u prvního zmíněného jsou s vysokou pravděpodobností zaznamenatelné signály o energii až -30 dBm, u druhého je maximální energie s vysokou pravděpodobností cca -50 dBm. Tyto dvě nosné frekvence lze přičíst s největší pravděpodobností GSM

a HSDPA technologii. Konkrétně v pásmu 2,4 GHz lze z grafu vyvodit výskyt zvýšeného šumového pozadí s energií mezi -105 dBm a -86 dBm.



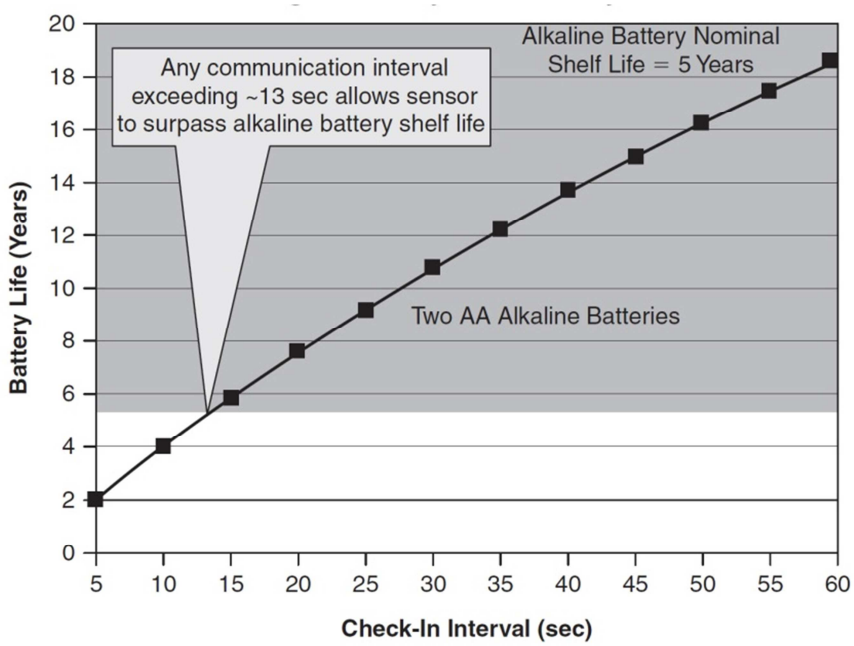
Obr. 1. Rozložení pravděpodobnosti výskytu amplitud [4]

2.1 ZigBee – 802.15.4

ZigBee je komerční bezdrátová technologie, postavená na standardu IEEE 802.15.4 [1], která podobně jako Bluetooth 802.15.1 [2] spadá do kategorie PAN sítí. Personal Area Networks jsou oblastí takzvaných osobních sítí, tedy sítí s relativně krátkým dosahem. ZigBee je možné provozovat ve třech pásmech, aby byla zajištěna použitelnost v globálním měřítku. Jedná se o pásmo 868 MHz, 915 MHz a 2,4 GHz. Na použitém pásmu pak závisí maximální přenosová rychlost, u 868 MHz je to 20 kb.s^{-1}, v pásmu 915 MHz 40 kb.s^{-1} a v pásmu 2,4 GHz je přenosová rychlost 250 kb.s^{-1} [1]. Tyto rychlosti jsou ovšem pouze teoretické schopnosti přenosu fyzické vrstvy a reálná přenosová rychlost je mnohem nižší. Síť ZigBee je tvořená dvěma typy zařízení a třemi typy rolí prvků v síti. Standard IEEE 802.15.4 definuje typ zařízení RFD a FFD. RFD (Reduced Function Device) je zařízení s omezenými vlastnostmi, schopné pouze nejzákladnějších úkonů v rámci sítě a to pro snížení nároků na hardware a energetickou spotřebu. Zpravidla je zařízení typu RFD schopno komunikovat pouze s FFD. FFD (Full Functional Device) je zařízení které podporuje všechny funkcionality sítě, je schopno komunikovat se všemi členy sítě a je schopno plnit v síti všechny role. Role v síti definuje ZigBee Aliance jako:

- Koordinátor (FFD)
- Router (FFD)
- Koncový člen (RFD)

Každý z těchto prvků má v síti jiné uplatnění. Koordinátor je základním stavebním prvkem sítě, musí být v každé síti, a to pouze jedenkrát. Router je univerzálním prvkem, jedná se v podstatě o koncový prvek s rozšířenou schopností směrovat data ostatních členů sítě a popř. přijímat nové členy sítě, které v komunikaci nedosáhnou koordinátora. Vzniká otázka, k čemu slouží koncový člen. Jedná se o zřízení bez schopnosti spoluřízení sítě, dokáže se do sítě pouze připojit a odeslat či přijmout data a několik dalších akcí (viz. Tab. 1). Výhodou tohoto prvku je energetická nenáročnost (vyplývající z možnosti dlouhodobého uspání prvku v době nečinnosti), která jeho existenci v síti „ospravedlňuje“. Energetická nenáročnost ZigBee technologie je způsobena celou řadou faktorů. Mezi ně patří propracované řízení komunikace na síti s definovanými časovými useky, po které mohou transceivery přejít do režimu spánku. Dále také zjednodušený „Stack“, který především v případě RFD zabírá velmi málo paměti (ROM i RAM). Při napájení srovnatelným bateriovým zdrojem poskytne možnost provozu roky, kdežto Bluetooth dny. Spotřeba energie je znázorněna v grafu na Obr. 2. Interval přihlášení do sítě je v grafu nejméně 5s, což pro využití v mnoha aplikacích (především v senzorice) postačuje. Již při této hodnotě je doba „výdrže“ standartních alkalických baterií 2 roky. Lze také odečíst, že při cyklickém přenášení dat v intervalu 13 s. je spotřeba ZigBee tak nízká, že baterie není vybita po dobu její životnosti cca 5 let.



Obr. 2. Výdrž alkalické AA baterie např. v bezpečnostním senzoru [10]

2.1.1 Typy rámců

Standard ZigBee definuje čtyři různé typy rámců paketů. Každý typ rámce má jiný účel a jinou datovou strukturu. Jedná se o typy:

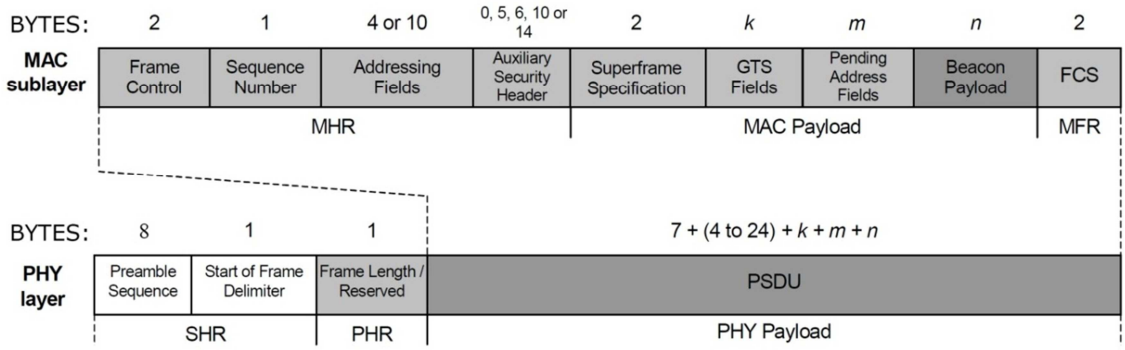
- Signální
- Řídící
- Datový
- Potvrzovací

Bez rozdílu typu, všechny pakety začínají sekvencí bitů, sloužící k synchronizaci vysílače a přijímače. Jedná se o úvodní sekvenci – Preambule, která se skládá z 32 bitů. Následuje osmibitová sekvence „11100101“ signalizující konec synchronizace a značící začátek datového přenosu [6]. Tyto dvě části jsou označovány jako Synchronizační hlavička – SHR. Následuje hlavička fyzické vrstvy o délce sedmi bitů + jeden bit nepoužitý - PHR. Jedná se o hodnotu udávající délku následujícího vysílaného datového proudu v bajtech. Její maximální hodnota je 2^7 tedy 128 B, a to je také maximální délka každého rámce (nezahrnuje SHR a PHR). Následující podrámec MAC vrstvy je již specifický podle typu paketu. Posledním společným polem pro všechny pakety je zakončení v podobě pole FCS (Frame Check Sequence), tedy kontrolní pole rámce. Jedná se o 16 bit CRC hodnotu počítanou při konstrukci paketu odesilatelem a sloužící ke kontrole správnosti přijetí dat. V následujících odstavcích jsou popsány druhy rámců v rámci dvou nejnižších vrstev sítě.

2.1.1.1 **Signální rámec (Beacon Frame)**

Signální rámec, jehož struktura je uvedena na Obr. 3, má několik úloh. Jednou z nich je distribuce garantovaných časových slotů (GTS) členům sítě v případě, že je tato funkcionalita využívána pro řízení přístupu k médiu [5]. Dále strukturu Superrámce a synchronizaci hodin všech zařízení v síti [5]. Signální rámec se skládá z hlavičky MAC vrstvy (MHR), která je, jak bylo uvedeno výše, předcházena synchronizační hlavičkou a hlavičkou fyzické vrstvy. MAC hlavička se skládá z kontrolního pole o délce dvou bajtů, které nese informace o typu rámce, adresace, anebo požadavku na potvrzení příjmu [1]. Následuje sekvenční číslo (BSN), které označuje pořadí rámce, pak následuje adresní pole, obsahující adresu odesilatele, cílové PAN ID, a dál obsahuje také parametry případného zabezpečení. Tímto parametrem končí MAC hlavička a následuje takzvaný užitečný náklad – MAC Payload. V této části paketu jsou uložena již zmíněná data o parametrech superrámce,

garantovaných slotech a v neposlední řadě adresy zařízení, které mají nedoručená data v paměti koordinátora [1]. Poslední pole - Beacon Payload, je pro případná data vyšších vrstev [1].



Obr. 3. Struktura signálního rámce [1]

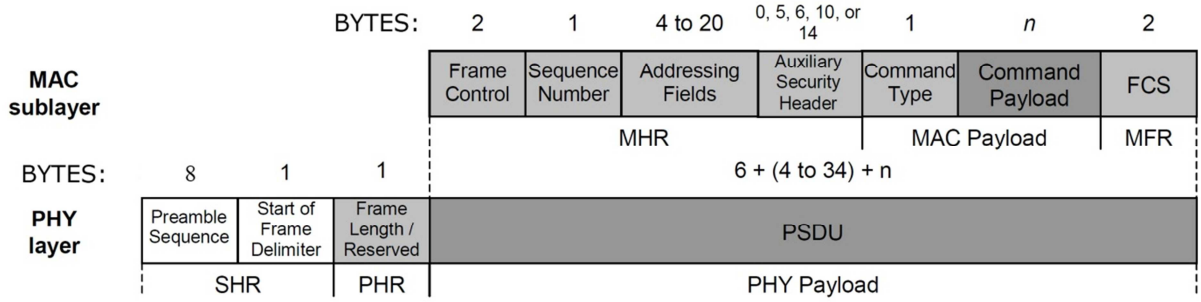
2.1.1.2 Řídící rámec (MAC Command Frame)

Řídící rámec slouží k řízení sítě, popř. zasílání žádostí. Byl-li tedy signální rámec určen především k distribuci informací o paramertrech, časování a synchronizaci sítě, pak Řídící rámec slouží k distribuci příkazů, případně žádostí mezi členy sítě. Jendá se například o příkazy spojené s registrací nových členů do sítě, žádosti o nedoručená data uložená v paměti koordinátora či vystoupení ze sítě. Seznam typů příkazů je uveden v Tab. 1 spolu s informací, jaké příkazy musí umět zpracovat každé zařízení v síti. Zkratka RFD uvedená v tabulce znamená Reduced Function Device, tedy koncové zařízení s omezenými schopnostmi pro úsporu energie. Jednotlivé sloupce pak značí, zda zařízení musí pro úspěšnou existenci v síti umět daný příkaz odeslat, přijmout, či obojí.

Tab. 1. Příkazy užívané v Řídícím rámci [1]

Command frame identifier	Command name	RFD	
		Tx	Rx
0x01	Association request	X	
0x02	Association response		X
0x03	Disassociation notification	X	X
0x04	Data request	X	
0x05	PAN ID conflict notification	X	
0x06	Orphan notification	X	
0x07	Beacon request		
0x08	Coordinator realignment		X
0x09	GTS request		
0x0a–0xff	Reserved		

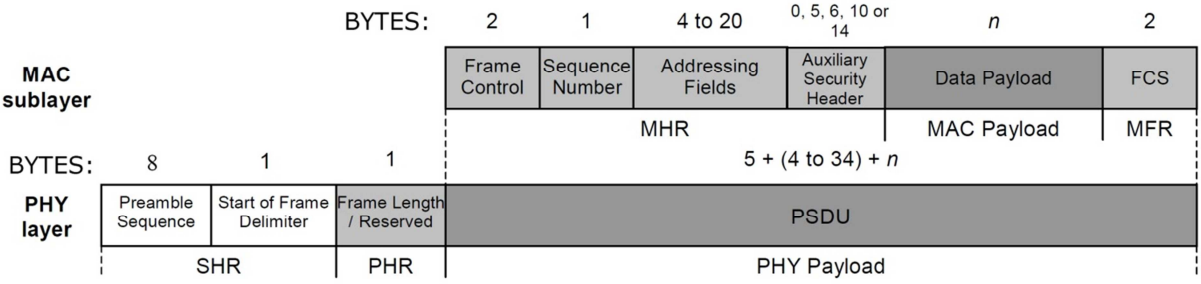
Řídící rámec a jeho struktura je uveden na Obr. 4. Jak je zřejmé, první část, tedy Synchronizační hlavička, i hlavička Fyzické vrstvy jsou stejné. Ani ve struktuře MAC hlavičky nenajdeme rozdíl, pouze konkrétní hodnota obsažená v poli Frame Control bude odlišná od předchozího případu, což definuje jinou strukturu v MAC Payload. Ten je složen z jednobytové hodnoty Command Type, která určuje typ příkazu (Tab. 1). Jako v předchozím případě, i zde je prostor pro případná další data MAC vrstvy v podobě MAC Payload.



Obr. 4. Struktura Řídícího rámce. [1]

2.1.1.3 **Datový rámec (Data Frame)**

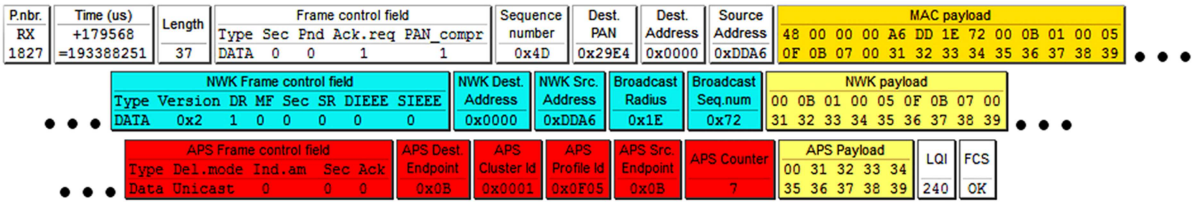
Datový rámec slouží k obecné datové komunikaci aplikací na síti. Jeho struktura je podobná ostatním rámcům, jen MAC Payload, který následuje po MAC hlavičce a již neobsahuje žádná konkrétní data Fyzické a MAC vrstvy. Data v MAC Payload jsou nicméně dále strukturována vyššími vrstvami a není možné celý datový prostor využí k přenosu dat aplikací. Základní struktura je na Obr. 5.



Obr. 5. Struktura Datového rámce [1]

Na dalším obrázku (Obr. 6) je kopie datového paketu zachycená packet snifferem (zachytávačem paketů). Paket obsahoval 10 B aplikačních dat, pro přehlednost byl rozdělen do tří řádků, podle vrstev které konkrétní část konstruují. V prvním řádku jsou data Fyzické a MAC vrstvy tak, jak byly popsány v úvodu s předponou dvou polí přiřazených zachytávacím softwarem, která udávají čas zachycení a pořadové číslo. Řádek končí polem MAC Payload, který obsahuje data

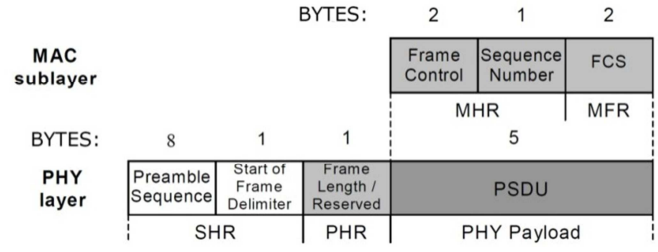
druhého řádku – Síťové vrstvy. Síťová vrstva obsahuje opět Frame control field, tedy parametrické pole určující typ datového paketu, dále adresy odesilatele, příjemce, Broadcast radius (počet přeposlání paketu před jeho zahozením) a jako poslední opět sekvenční číslo. Jako poslední je NWK Payload, který obsahuje data podpůrné vrstvy aplikace (APS). Na třetím řádku pak opět Frame control field, tentokrát ve smyslu APS vrstvy, v tomto případě se jednalo o Unicast paket. Dalšími parametry jsou Koncový bod (rozlišující více aplikací v jednom transceiveru), Koncová skupina a Koncový profil, dále parametr Koncového bodu odesílatele a počítadlo APS vrstvy. Následuje APS Payload, které již obsahuje aplikační data. Z tohoto přehledu lze přibližně spočíst, že Datový packet obsahuje v základním nastavení sítě 37 bajtů (27 + PHR + SHR) včetně synchronizační sekvence a koncového 16 bitového CRC součtu (počet se může lišit v závislosti na typu adresace, konkrétním nastavení NWK a APS vrstvy). Z uvedeného vyplývá, že na uživatelská data zbývá maximálně 98 Bytu (127-27-2). Proto při maximálním využití datového prostoru v rámci, je propustnost sítě snížena režijí o 28 %.



Obr. 6. Struktura Datového včetně vyšších vrstev.

2.1.1.4 Potvrzovací rámec (ACK Frame)

Posledním typem je Potvrzovací rámec. Tento typ rámce je nejjednodušší, má jediný účel a to potvrdit korektní příjem dat na straně přijímače. Systém potvrzování zavádí v síti zpětnou vazbu, která umožňuje odesilateli získat informaci o úspěšném přijetí příjemcem. Potvrzovací paket (Obr. 7) je složen opět ze SHR, PHR a MHR, kde v MAC Payload je umístěna kopie 16 bit CRC z potvrzovaného paketu a také sekvenční číslo paketu. Samozřejmě je i tento paket opatřen novým kontrolním součtem v MFR poli.



Obr. 7. Struktura Potvrzovacího paketu [1]

2.1.2 CSMA-CA

Zkratka CSMA-CA v překladu Carrier Sense Multiple Access with Colision Avoidance, je jednoduchá metoda, umožňující sdílení jednoho frekvenčního kanálu více zařízeními. V okamžiku, kdy jedno ze zařízení potřebuje použít přenosový kánál k přenosu, spustí kontrolní rutinu CCA (Clear Channel assessment), která vyloučí přítomnost vysílání jiného zařízení sdílející kanál. Je-li kanál „volný“, zařízení spustí vlastní vysílání. Je-li kanál vyhodnocen jako obsazený, spustí vyčkávací sekvence (backoff interval), po jejímž uplynutí je znovu spuštěna CCA rutina. Rozhodující, pro určení „volnosti“ kanálu je měření energie spektra v rozsahu měřeného kanálu po dobu osmi symbolů. K jeho vyhodnocení je možné využít třech přístupů:

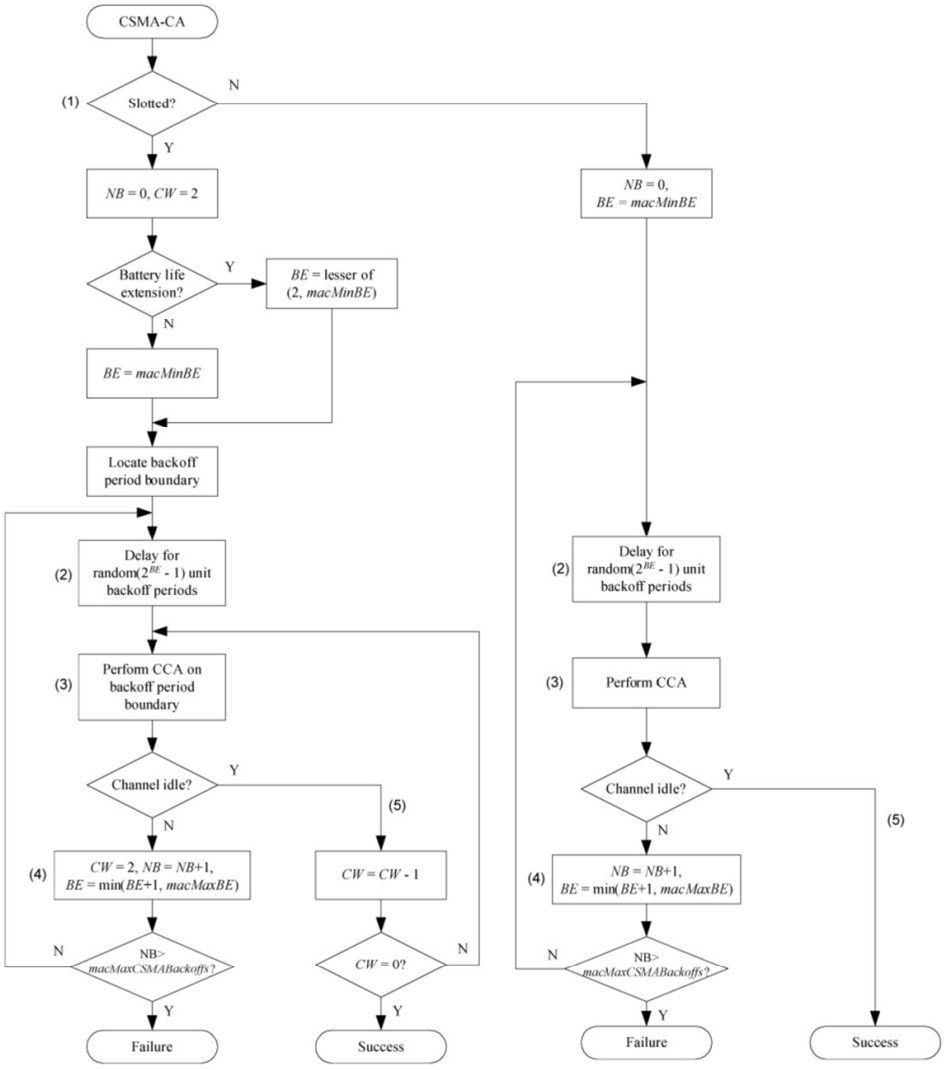
- Metoda ED (Energy Detection) je v podstatě prosté měření energie přijímané ZigBee přijímačem v daném časovém úseku. Je-li toto množství energie pod rozhodnou hranicí, je kanál vyhodnocen jako neobsazený.
- CS (Carrier Sense) je druhým přístupem, který zahrnuje vyhodnocení přijímaného signálu ve smyslu shody se standardem. Je-li přijímaný signál vyhodnocen jako signál jiného ZigBee transceiveru, kanál je obsazen, v opačném případě je vyhodnocen jako volný.
- Poslední variantou je kombinace obou předchozích způsobů. Po zjištění přítomnosti vysílání stejného standardu pomocí CS algoritmu, je dále vyhodnoceno, zda kanál vyhovuje také ED algoritmu. V každém případě, dojde-li ke zjištění, že kanál je v daném okamžiku obsazen, transceiver vysílání odloží na dobu, kterou určuje více parametrů.

V praxi CSMA-CA algoritmus pracuje se třemi proměnými NB, CW a BE [7][1].

- NB – Počet vykonaných backoff intervalů při aktuálním CSMA přístupu k médiu.
- CW – Počet po sobě jdoucích backoff intervalů vykonaných po CCA vyhodnocení kanalů jako „volný“.
- BE - Backoff exponent, parametr určující délku backoff intervalu.

Na následujícím vývojovém diagramu (viz. Obr. 8) je vyobrazen postup algoritmu v případě žádosti o CSMA-CA přístup ke kanálu. V případě spuštění CSMA-CA mechanismu, dojde (vezměme pro popis jednodušší variantu neslotového režimu) k inicializaci proměných na hodnoty BE = macMinBE a NB = 0. Funkcí s parametry

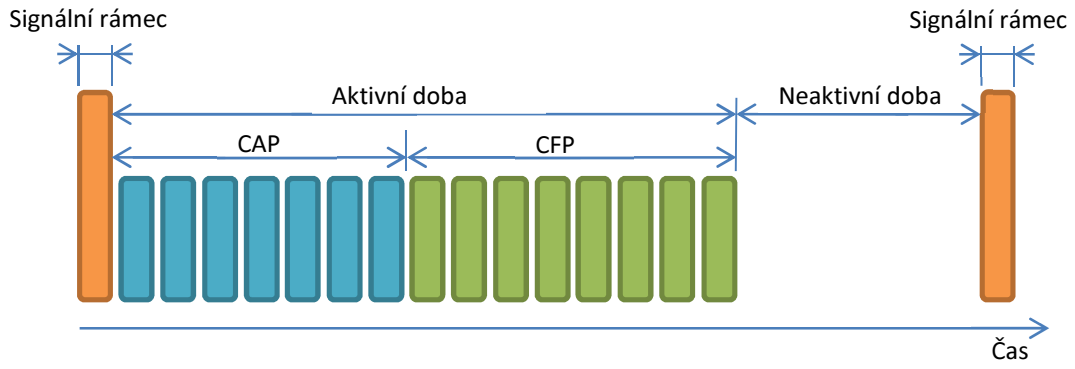
“random($2^{BE} - 1$)” je generován počet backoff intervalů po které je provedena pauza rutiny. Po jejím uplynutí dojde ke spušění CCA algoritmu v jednom ze tří výše uvedených režimů. Je-li výsledkem CCA, že kanál je volný, dojde ke spuštění vysílání. Je-li kanál vyhodnocený jako obsazený, dojde ke zvýšení proměnné NB = NB + 1, a stejně u proměnné backoff exponentu BE v případě, že BE nedosáhlo maximální hodnoty macMaxBE. Nepřesáhla-li v tomto bodě proměnná NB maximální počet backoff intervalů (macMaxCSMABackoffs), vrací se rutina zpět do bodu generování a provedení pauzy na základě nových proměnných. U slotového režimu je k tomuto postupu přidána podmínka dosažení předem určeného počtu po sobě jdoucích CCA měření s výsledkem volného kanálu. Díky tomu, že proměnná BE je začleněna ve výpočtu prodlevy mezi CCA měřeními, je s každým neúspěšným CCA vyhodnocením kanálu tato doba prodloužena a dochází k úspoře energie. Nesporným přínosem CSMA-CA metody je zvýšení spolehlivost sítě, menší počet kolizí za cenu snížení propustnosti sítě. Detailnější rozbor vlivu CSMA-CA algoritmu v [8]



Obr. 8. Vývojový diagram CSMA-CA mechanismu [1]

2.1.3 Beacon-enabled / non-Beacon-enabled

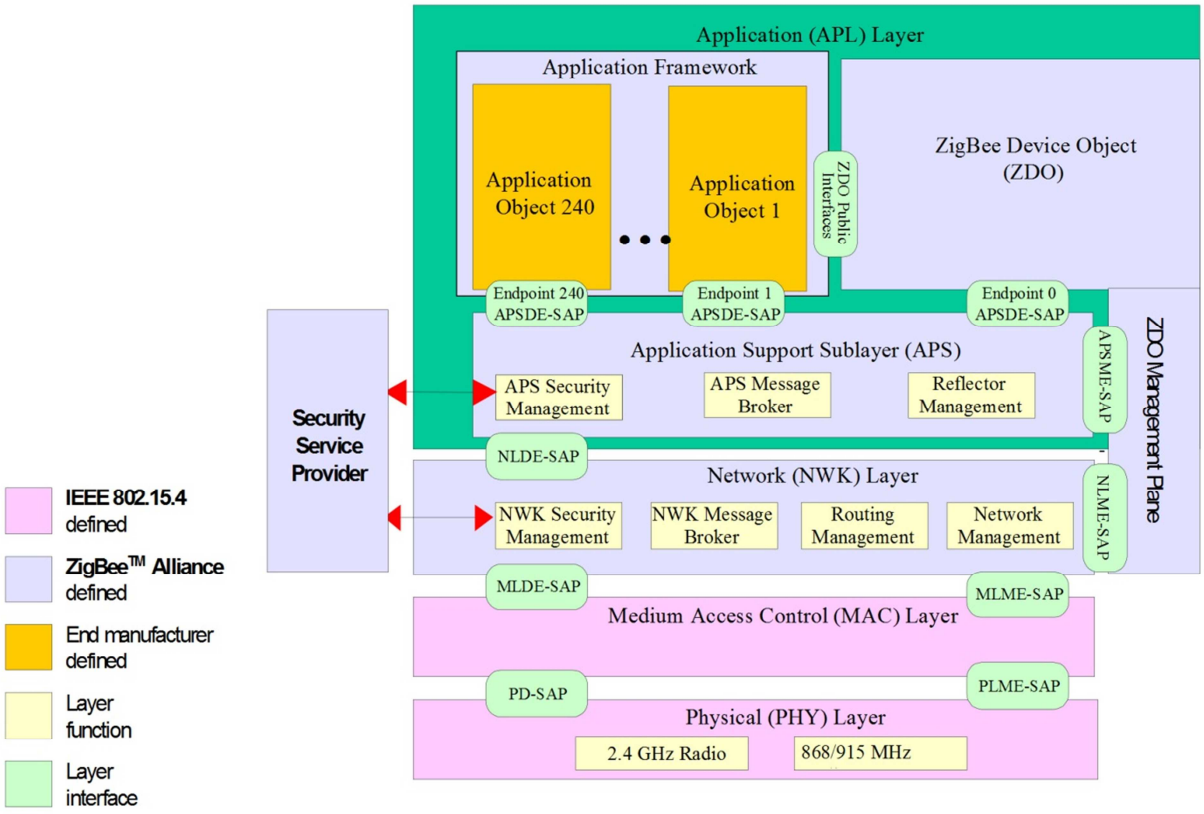
Nastavení infrastruktury řízení komunikace na síti úzce souvisí s takzvaným Superrámcem. Jeho využití/absence při řízení sítě je nazýváno podle distribučního rámce „Beacon enabled“, případně „non-Beacon-enabled“. Nejedná se v tomto případě o rámec v podobě struktury dat v paketu, ale o rámec časové organizace v síti. Cílem takového řízení provozu je především úspora energie a také zajištění volného kanálu pro prioritní zařízení (GTS sloty). Právě struktura těchto časových pravidel je distribuována a synchronizována pomocí Řídících rámců (viz. 2.1.1.2). Na Obr. 9 je vyobrazen příklad struktury Superrámce s popisem jeho částí. Každý Superrámec se muže skládat z Aktivní a Neaktivní doby a je ohraničen vysíláním Signálních rámců [1]. Aktivní část každého Superrámce je rozdělena na 16 slotů včetně slotu, ve kterém byl vysílán Signální rámec [1]. Tyto sloty lze použít dvěma způsoby. V CAP (Contention Access period) části, která začíná ihned po dokončení Signálního rámce, je prostor určen ke komunikaci kteréhokoli člena sítě pomocí CSMA-CA mechanismu (viz. 2.1.2) s výjimkou ACK paketů a datových paketů, náledujících ihned po data request [1]. CAP část by měla mít délku minimálně aMinCAPLength symbolů [1]. V následující části, CFP (Contention Free period), jsou sloty využity pro distribuci GTS (Guaranteed time slot). Tato část slouží jako vyhrazený časový úsek, ve kterém žádný z členů (mimo jednoho) nesmí kanál využít. V této části není využíváno CSMA-CA mechanismu a k médiu se přistupuje přímo. Distribuci GTS slotů zajišťuje koordinátor sítě na základě žádosti členů sítě. Slotů může být přiřazeno i více najednou pro jedno zařízení. Poslední částí je Neaktivní doba, tedy čas do dalšího Signálního rámce, po kterou mohou zařízení, především Koordinátor, přejít do spánkového režimu - čímž dojde k úspoře energie [1].



Obr. 9. Příklad struktury superrámce [1]

2.1.4 Vrstvy sítě ZigBee

Základní členění protokolu je založeno na standardním OSI (Open Systems Interconnection) modelu [6]. Ne vždy jsou všechny vrstvy v protokolu zapojeny, odvisle od typu zařízení (router, koordinátor, koncové zařízení) a jeho účelu. Nejnižší dvě vrstvy, tedy fyzickou (PHY) a vrstvu Přístupu k médiu (MAC), definuje standard IEEE 802.15.4 [1]. Uskupení ZigBee Alliance doplněním těchto dvou vrstev o Síťovou (NWK) a Aplikační (APL) vytvořilo bezdrátovou technologii ZigBee. Aplikační vrstvu lze rozdělit na samotnou aplikaci, podporu aplikace a aplikačních objektů. Pomocné podvrstvy APL mají za cíl usnadnit uživateli (aplikaci) párování, případně hromadnou komunikaci s více objekty. Rozložení a návaznost jednotlivých vrstev je možno vidět na Obr. 10.

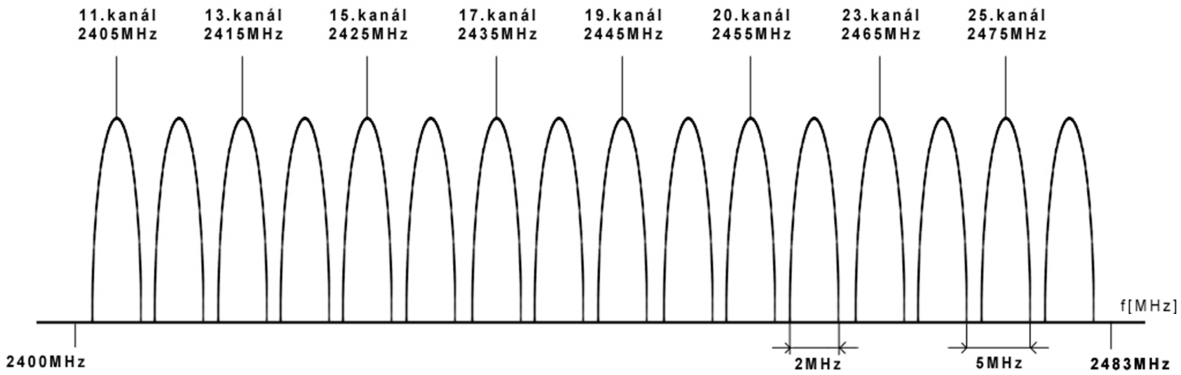


Obr. 10. Vrstvy ZigBee sítě a uzlové body [5]

2.1.4.1 Fyzická vrstva

Jak již bylo uvedeno, ZigBee je možné provozovat ve třech pásmech. Fyzická vrstva je tedy definována ve třech podobách pro pásma 868 MHz, 915 MHz a 2,4 GHz. Na použitém pásmu pak závisí maximální přenosová rychlost, u 868 MHz je to 20 kb.s^{-1}, v pásmu 915 MHz 40 kb.s^{-1} a v pásmu 2,4 GHz je přenosová rychlost 250 kb.s^{-1} [1]. V dalším textu bude rozebírána již pouze konkrétně verze pro 2,4 GHz,

kde ZigBee na fyzické vrstvě využívá ke zvýšení spolehlivosti dopřednou korekci chyb DSSS - v překladu „Metoda rozprostřeného spektra“. Pásmo je rozděleno na 16 kanálů, které mají rozestup 5 MHz a jsou široké 2 MHz. Rozložení kanálu v pásmu 2,4 GHz je uvedeno na Obr. 11.



Obr. 11. Rozložení kanálů v pásmu 2,4 GHz

DSSS

Tato metoda byla implementována již v prvním standardu 802.11 z roku 1997, kdy bylo využito Barkerova kódu. Obecně metoda využívá redundantních dat k rozprostření informace do spektra a tím zvýšila její odolnost vůči rušivým vlivům. (viz. 2.2.1) DSSS je využito i ve standardu ZigBee, kde je každá čtveřice odesílaných bitů nahrazena sekvencí 32 bitů náhradních. Protože se v této podobě již nejedná o nositele konkrétní informace, je zvykem je označovat jako chipy. Mapa přiřazení a jednotlivé chipové sekvence jsou uvedeny v Tab. 2. V této fázi je signál vysílán vysílačem s modulací O-QPSK (Offset-Quadrature Phase Shift Keying), která čtyřstavovou modulací a umožní tedy odeslat dva bity v jednom symbolu. Na straně příjemce je po demodulaci pro každou takovou chipovou sekvenci vyhodnocena Hammingova vzdálenost s každou z 2^4 variant 32 chipových sekvencí a nahrazena opět 4 informačními bity na základě nejmenší vzdálenosti. Místo 4 bitů je tedy přenášeno 32, což znamená, že propustnost kanálu je snížena na 1/8 [1]. Přínosem je schopnost statisticky odhalit a „opravit“, či spíše „nevnímat“ chyby vzniklé šumem v přenosovém kanálu. Účinnost takové dopředné korekce chyb závisí na volbě chipových sekvencí, důležitá je jejich Hammingova vzdálenost, která určí počet bitových chyb nutných k jejich vzájemné záměně a tedy chybě v přenosu.

Tab. 2. Přiřazovací tabulka pseudonáhodných chipových sekvencí [5]

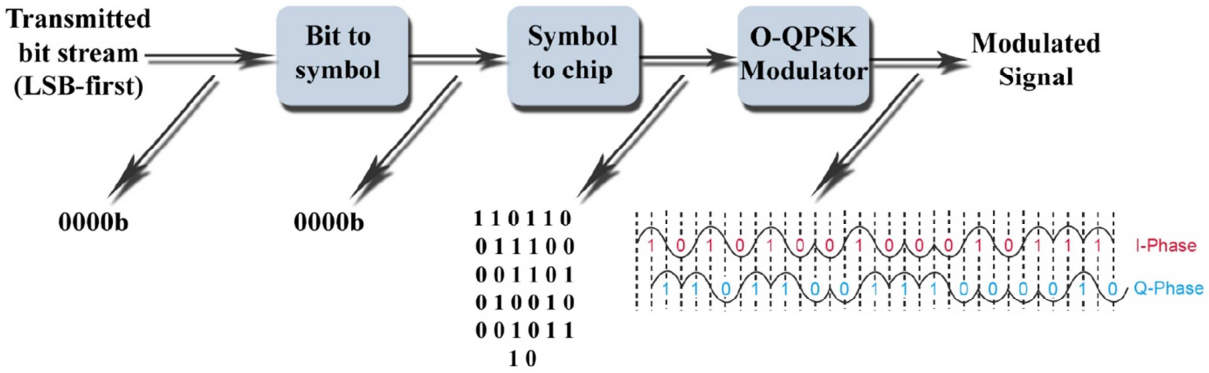
Dekadické hodnoty symbolů	Příslušné chipové sekvence
0	1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0
1	1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0
2	0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0
3	0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1
4	0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0 0 0 1 1
5	0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1 1 1 0 0
6	1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 0 1
7	1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 0 1
8	1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1
9	1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1
10	0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1
11	0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0
12	0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1 0
13	0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 1
14	1 0 0 1 0 1 1 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 1 0 0
15	1 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0

O-QPSK

Uvedená modulační metoda (Ofsetové kvadraturní fázové klíčování) má oproti QPSK nižší parazitní amplitudovou modulaci a to díky maximální možné změně fáze o -+90°. Díky zpoždění Q kanálu muže dojít ke změně stavu modulace jen v polovině bitové periody signálu druhého kanálu [31]. Nemuže tedy dojít ke změně stavu 11 <-> 00 a také 01 <-> 10 [31]. Tím dochází k maximální hloubce amplitudové modulace 30 %. Jak již bylo uvedeno, jedná se o čtyř-stavovou modulaci, je tedy možno přenést dva bity v jednom symbolu. Modulovaný signál lze vyjádřit rovnicí (1), kde s (t) je modulovaný signál, f_c je nosná frekvence kanálu, T_c pak čas změny Q o 90° [5].

$$s(t) = \frac{1}{\sqrt{2}} I(t) cos 2\pi f_c t - \frac{1}{\sqrt{2}} Q(t - T_c) sin 2\pi f_c t \qquad (1) \qquad [5]$$

Přeměna datového toku na fyzické vrstvě je vyobrazena na blokovém schématu Obr. 12. Vlevo vstupují data na fyzickou vrstvu, jedná se o bitový tok, který je následně konvertován na symboly (čtveřice bytů) a ty jsou následně v dalším bloku nahrazeny chipovými sekvencemi podle Tab. 2. Tyto dva bloky uskutečňují výše uvedené kódování DSSS. Poslední blok je O-QPSK modulátor, jehož výstup je součet složky I a Q, jejichž průběh je uveden vpravo dole.



Obr. 12. Blokové schéma zpracování datového toku na fyzické vrstvě [5]

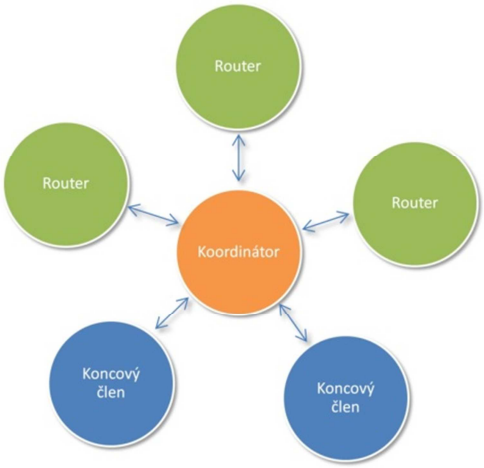
Vyjdeme-li z výše uvedeného, pak teoretická propustnost kanálu o šířce 2 MHz a O-QPSK modulaci je 2 Mb.s^{-1}, kódování je prováděno v poměru 4/32. Maximální teoretická propustnost v pásmu 2,4 GHz je tedy rovna 2000 * 4 / 32 = 250 kb.s^{-1}. Tímto výpočtem je možno ověřit, že uvedená maximální rychlost ZigBee sítě je myšlena pouze jako rychlost fyzické vrstvy a nebere v úvahu další vrstvy sítě.

2.1.4.2 MAC vrstva

Vrstva přístupu k médiu (Medium Acces Control), jak název napovídá, slouží k řízení Fyzické vrstvy (tedy hardwaru). Obecně se jedná o rozhraní mezi Fyzickou vrstvou a vrstvou Linkovou. MAC vrstva je zodpovědná za zpracování a distribuci signálních rámců, GTS management, odesílání a přijímání potvrzovacích rámců, asociace nových členů, případně jejich disasociace ze sítě [1]. V neposlední řadě je to samotný přístup k médiu CSMA-CA nebo GTS + CSMA-CA (viz. 2.1.2 a 2.1.3), adresace zařízení a zabezpečovací funkce (AES šifrování). Typy rámců a jejich struktura právě na MAC vrstvě jsou uvedeny v odstavci 2.1.1 a pododstavcích.

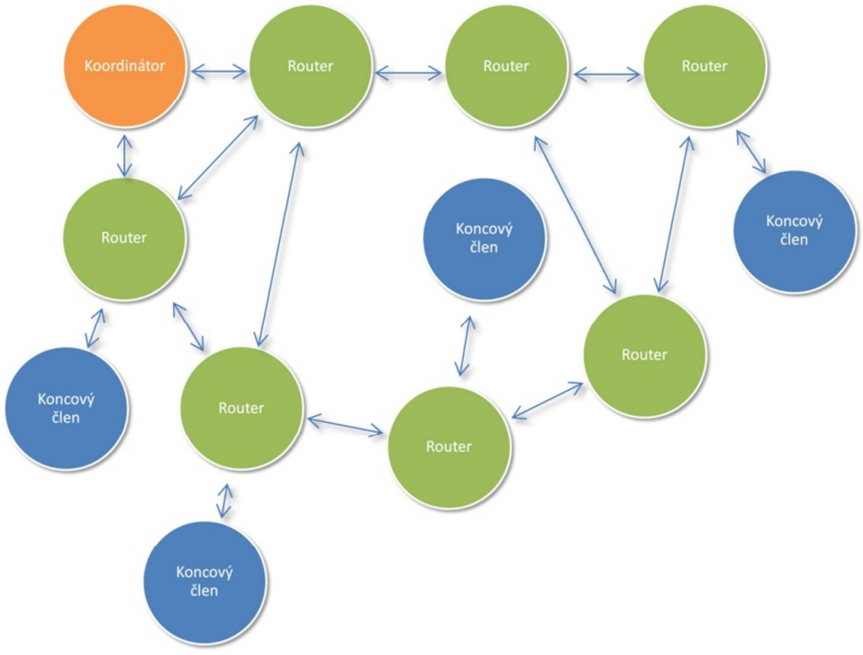
2.1.5 Topologie sítě

Standard 802.15.4 zahrnuje dvě možné topologie stavby sítě. První je typ „hvězda“ (viz. Obr. 13), který spočívá v centrálním uspořádání okolo Koordinátora sítě. Zařízení komunikují pouze s koordinátorem a jsou tedy buď iniciátory spojení, nebo koncovými body spojení. Koncová zařízení i koordinátor mohou zpracovávat specifický úkol daný aplikací. Tato topologie je vhodná například u počítačovýc periferií, hraček apod.



Obr. 13. Topologie sítě „hvězda“

Druhou možností je topologie „peer to peer“ (viz. Obr. 14), zde je umožněno komunikovat všem zařízením v síti navzájem, pokud jsou v dosahu. Tato topologie je opět řízena jedním Koordinátorem, nicméně na rozdíl od hvězdy umožňuje vytváření redundantních komunikačních spojů. Tím vzniká mnohem komplexnější uskupení, které lépe odolává případným výpadkům jednotlivých členů ať již z důvodu rušení, zatížení či poruchy.



Obr. 14. Topologie sítě „peer to peer“

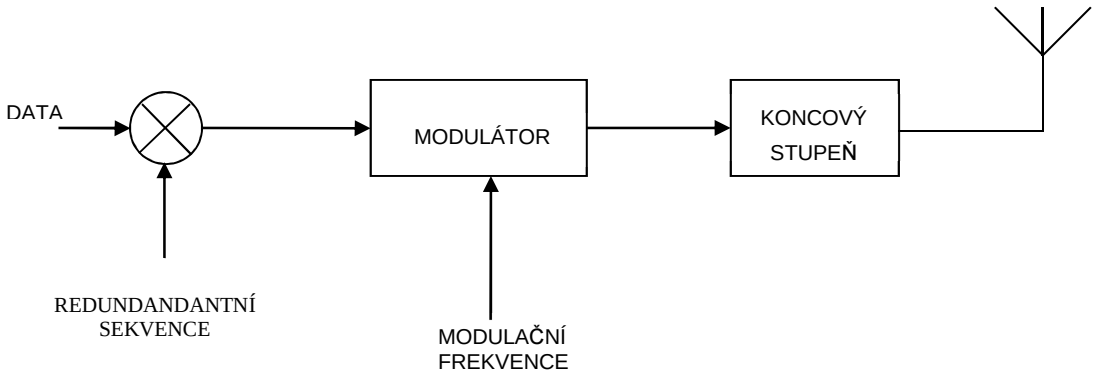
2.2 Používané přístupy ke zvýšení odolností vůči rušení

Běžné bezdrátové standardy zahrnují řadu přístupů ke zvýšení odolnosti bezdrátového přenosu k rušení. Řešení tohoto problému lze pojmout více způsoby,

některé sázejí na výpočetní metody dopředné korekce chyb, některé naopak na sofistikované nakládání s frekvenčním spektrem. Obecně lze najít společný jmenovatel pro všechny dále uvedené, tím je snaha rozprostřít signál do určeného spektra. V náledujícím textu jsou popsány nejrozšířenější techniky pro zvýšení odolnosti bezdrátového spoje.

2.2.1 DSSS

„Direct Sequence Spread Spectrum“ neboli technika přímého rozprostření do spektra, je v principu jednoduchým způsobem jak zvýšit spolehlivost bezdrátového přenosu takzvanou Dopřednou korekcí chyb. Metoda využívá rendundantních dat pro přenos informace kanálem a tím zvyšuje jejich odolnost. Blokové schéma je uvedeno na Obr. 15. Metodu lze ve stručnosti popsat na dvou příkladech. Mějme jeden bit k přenosu šumovým kanálem. Pro zvýšení pravděpodobnosti jeho úspěšného doručení jej přeneseme hned dvakrát. Tedy místo „1“ budeme přenášet „11“ a místo „0“ pak „00“. Na straně přijímače pak vzniknou hned čtyři varianty doručení, pro příklad řekněme, že bylo doručeno „01“. V tomto okamžiku ovšem nejsme schopni určit původní bit, protože nevíme, který ze dvou bitů byl poškozen. V této minimalistické variantě tedy technika rozprostřeného spektra neúčinkuje. Zvýšíme-li počet nahrazujících bitů na tři, kupříkladu nahraďme bit „1“ sekvencí „111“ a „0“ sekvencí „000“. Na straně přijímače lze pak v tomto případě jednoznačně určit správný výsledek do úrovně jedné chyby – je li sekvence přijatá s většinou bitů v úrovni 1, byl odesílaný bit „1“. Pro opravu jedné chyby je v tomto případě nutno přidat dva redundantní bity.

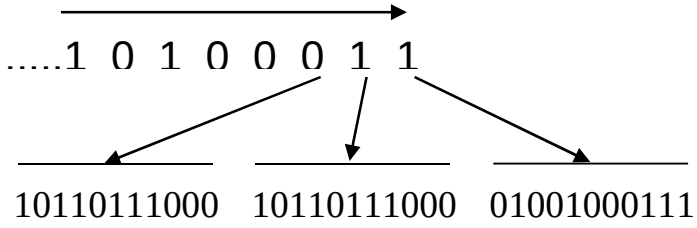


Obr. 15. DSSS kódování na straně vysílače

Z druhého příkladu je zřejmé, že právě vzájemná nepodobnost kódových slov je stěžejní pro účinnost metody. Nahrazujeme-li původní data po jednotlivých bitech,

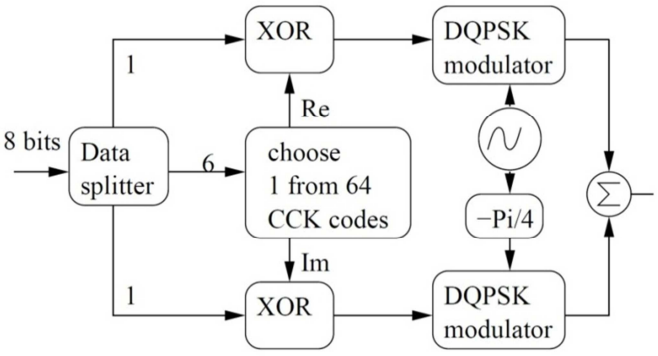
nahrazující sekvence bude logicky inverzní. V případě, že použijeme více kódových slov (nahrazujeme více bitu původní zprávy najednou), začne být vzájemná nezaměnitelnost důležitá. DSSS je využito také u ZigBee, konkrétním parametrům se věnuje část práce popisující její fyzickou vrstvu (viz. 2.1.4.1).

Do skupiny metod rozprostřeného spektra spadá například verze použitá ve standardu 802.11b, kde bylo k substituci datového toku použito Barkerova kódu. Chip Barkerova kódu byl 11 bit dlouhý a substituován byl za každý jeden bit datového toku, viz Obr. 16.



Obr. 16. Substituce Barkerovým kódem

Dalším zástupcem přístupu k substituci dat je metoda CCK, která nahradila právě Barkerův kód pro normu 802.11b v roce 1999 [17]. CCK – Complementary Code Keying, oproti Barkerovu kódu používá kratší substituční kód 8 chipů a zároveň je nahrazeno více bitů najednou. Pro rychlost 5,5 Mb.s^{-1} jsou 4 chipové sekvence, pro rychlost 11 Mb.s^{-1} je k dispozici 64 chipových sekvencí. Tato kódová slova jsou volena tak, aby je bylo možno od sebe rozlišit i v případě chyb při přenosu. Principiální schéma metody je uvedeno na Obr. 17. Data pro přenos jsou nejprve rozdělena na 6 a 2x1 bit. Šestice je substituována komplementárním kódem a zkombinována exkluzivním součtem s dvěma zbylými bity. Získáme dvě osmibitové sekvence, které se na nosnou frekvenci modulují fázově posunuty.

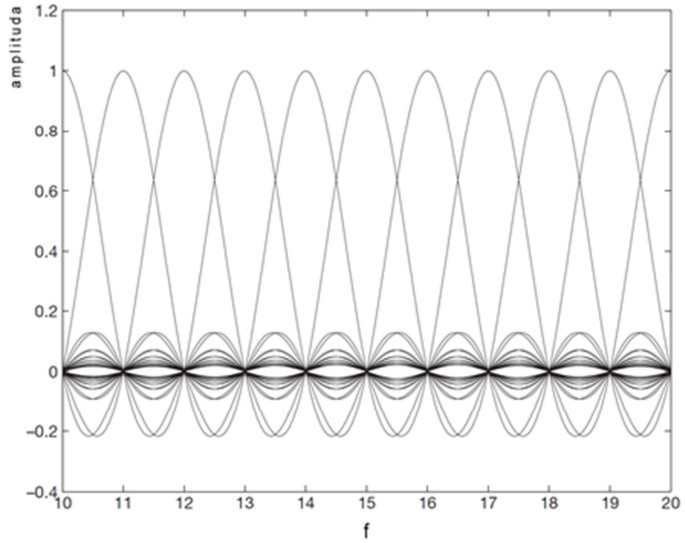


Obr. 17. CCK modulátor

2.2.2 OFDM

Jak bylo naznačeno v popisu standardu 802.11g/n, jedná se o metodu, která umožní rovnoměrnější a kvalitnější využití vymezeného spektra a která zároveň zvyšuje odolnost proti rušení (vzhledem ke klasickému datovému přenosu) [18]. Kanál, určený k přenosu, je (podobně jako například u ADSL) rozdělen na větší počet subkanálů. Tyto subkanály se chovají jako samostatné komunikační linky, samozřejmě s nižší datovou propustností. U standardu 802.11g je kanál o šířce 22 MHz rozdělen na 52 subkanálů, z nichž 48 je datových a 4 pilotní, sloužících k synchronizaci přenosu. Pro normu 802.11n jsou pak dvě varianty. Buď je využit kanál o šíři 20 nebo 40 MHz. Tomu také odpovídá počet subkanálů, užitých OFDM. Pro 20 MHz kanál je pásmo rozděleno na 56 subkanálů, z nichž jsou 4 pilotní. Kanál šířky 40 MHz je rozdělen na 114 subkanálů, kde 108 z nich přenáší data a 6 z nich je synchronizačních. U normy 802.16 (WiMax) je počet možných subkanálů až 2048.

Aby bylo frekvenčního prostoru využito co nejlépe, jsou kanály definovány s přesahem (viz Obr. 18). Tento přesah je pak eliminován ortogonalitou sousedních kmitočtů. Nedochází tedy k mezikanálovému rušení, protože v okamžiku, kdy jeden kanál přenáší konkrétní znak, sousední kanály jsou nulové [18]. Tato metoda se nechová vůči spektru agresivně, nezahlcuje určený frekvenční prostor velkým množstvým redundantních dat (jako např DSSS), ale naopak efektivně využívá přidělenou část spektra po menších dílech. Zvýšení spolehlivosti je dosaženo schopností udržet datovou komunikaci s menší datovou propustností i v případě že část kanálu je rušena. OFDM bývá velice často kombinováno právě s DSSS.



Obr. 18. Rozmístění kanálů OFDM [18]

2.2.3 FHSS

FHSS – Frequency Hopping Spread Spectrum. V překladu Metoda frekvenčního skákání je technika rozprostřeného spektra, která pro zvýšení schopnosti odolávat rušení nepoužívá statický kanál, ale cyklicky využívá všech kanálů definovaných v určeném spektru. Pro standard Bluetooth 802.15.1 [2], který FHSS definoval v dřívějších verzích, je stanoveno 79 kanálů s rozestupy 1 MHz. První kanál začíná na frekvenci 2402 MHz, poslední pak na frekvenci 2480 MHz [2]. Toto rozestavení platí shodně pro 802.11 i pro 802.15. Frekvenční skoky se konají s periodou 0,625 ms a jejich pořadí určuje pseudonáhodná sekvence (viz Tab. 3). Tato sekvence je definována normou jako pořadí náhodně generovaných čísel. Je-li tedy část pásma rušena, tato metoda při přeskocích dokáže komunikovat korektně v části volné a sníží svoji celkovou propustnost o poměrnou část spektra, kterou nelze při přeskocích použít. Tím dochází ke zvýšení robustnosti spoje bez zahlcení spektra redundantními informacemi, ačkoli jako nevýhodu pro koexistenční partnery lze chápat její mžikové používání celého spektra 2,4 GHz. I tato technika bývá kombinována s DSSS. Konkrétní vliv statického a dynamického rušení na FHSS lze vyčíst z grafu na Obr. 19.

Tab. 3. Přehled pseudonáhodné sekvence FHSS.

i	b(i)	i	b(i)	i	b(i)	i	b(i)	i	b(i)	i	b(i)	i	b(i)	i	b(i)
1	0	11	76	21	18	31	34	41	14	51	20	61	48	71	55
2	23	12	29	22	11	32	66	42	57	52	73	62	15	72	35
3	62	13	59	23	36	33	7	43	41	53	64	63	5	73	53
4	8	14	22	24	71	34	68	44	74	54	39	64	17	74	24
5	43	15	52	25	54	35	75	45	32	55	13	65	6	75	44
6	16	16	63	26	69	36	4	46	70	56	33	66	67	76	51
7	71	17	26	27	21	37	60	47	9	57	65	67	49	77	38
8	47	18	77	28	3	38	27	48	58	58	50	68	40	78	30
9	19	19	31	29	37	39	12	49	78	59	56	69	1	79	46
10	61	20	2	30	10	40	25	50	45	60	42	70	28	—	—

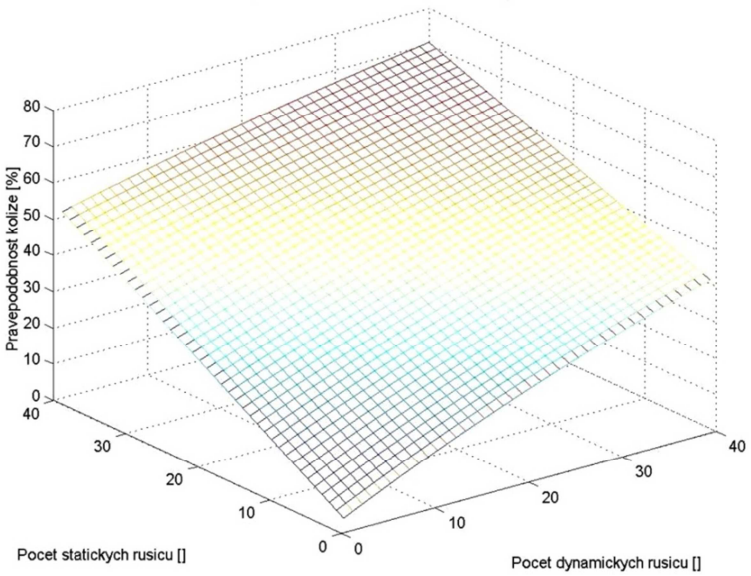
2.2.4 AFH

AFH je rozšířením metody FHSS. Přeskoky řízené pseudonáhodnou sekvencí mohou být na základě parametrů spojení konkrétních frekvencí vynechány. Metoda vychází z předpokladu, že rušení a jeho rozložení ve spektru je v každých konkrétních podmínkách různé. Proto se snaží využívat jen ty části pásma, které jsou rušením zasaženy nejméně. Vzhledem k tomu, že šířka kanálů u 802.15.1 (nejvíce využívající AFH) je 1 MHz, dokáže využít i relativně úzké části, kde rušení není omezující.

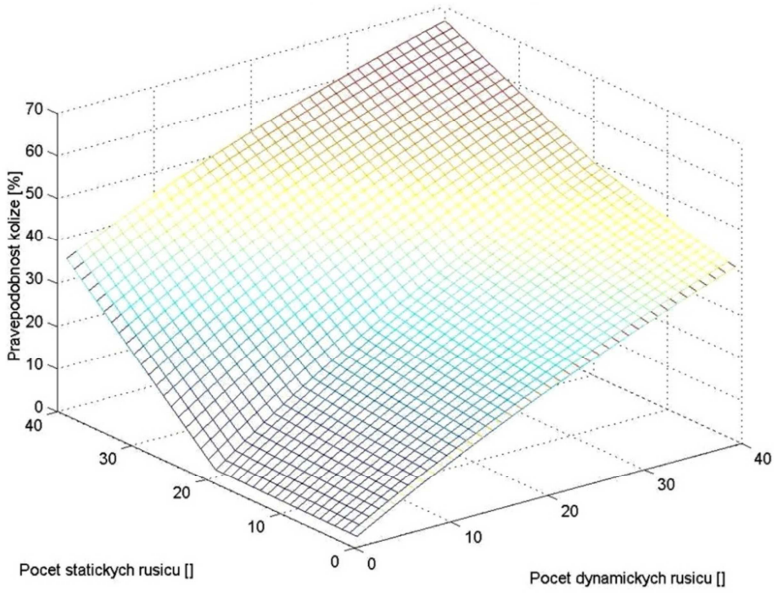
Provedené studie ukazují, že oproti pevné nosné frekvenci, dokáže AFH zlepšit přijímaný výkon o 5-20 dB [19].

Každé jedno spojení je hodnoceno z pohledu kvality (chybovost BER, odstup Signál/Šum). V případě, že je vyhodnoceno spojení jako špatné (BAD), je v pseudonáhodné sekvenci místo tohoto kanálu nahrazeno kanálem, který byl již použit úspěšně (GOOD). S určitým časovým odstupem je pak do sekvence vyřazený kanál navrácen, čímž je zajištěna opětovná kontrola rušení v již jednou vyřazených kanálech [20]. Srovnání výsledku funkce FHSS a AFH je možné z následujících grafů. Jedná se o výsledky matematického modelu, který simuloval rušivé vlivy dvou typů na metodu FHSS a AFH. Jedním druhem rušení je rušení statické, tedy na konkrétní neměnné frekvenci, druhým je rušení dynamické, tedy takové, které buď plošně, nebo pohyblivě s užším rozsahem zasahuje v přenosovém pásmu nahodile. Na grafu v Obr. 19 je zobrazena pravděpodobnost kolize s dynamickým a statickým rušením pro 1-40 jednotlivých zdrojů rušení. Na grafu v Obr. 20 je pak vyobrazena pravděpodobnost kolize AFH. Obr. 21 dále vyobrazuje přínos metody AFH oproti FHSS v podobě zisku vypočteného dle vztahu (2) [20].

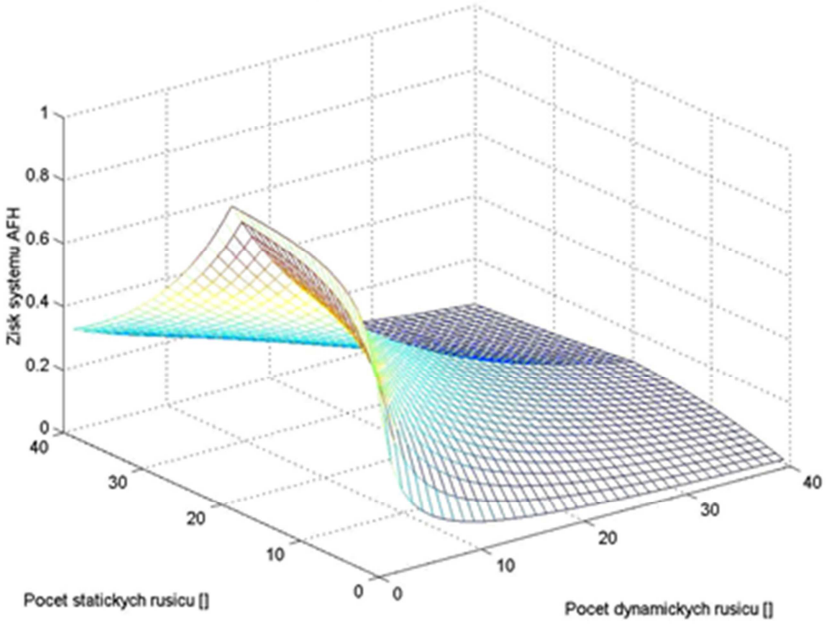
$$G = \frac{P_{FHSS} - P_{AFH}}{P_{FHSS}} (-) \qquad (2) \quad [5]$$



Obr. 19. Vliv statického a dynamického rušení na přenos FHSS [20]



Obr. 20. Vliv statického a dynamického rušení na přenos AFH [20]



Obr. 21. Zisk AFH oproti FHSS [20]

2.2.5 Další metody založené na dopředné korekci chyb

Kromě již zmíněné metody DSSS, existuje mnoho dalších. Kódování signálu a tedy jeho ochrana proti chybám, vzniklým při přenosu, je implementována v mnoha podobách snad do všech existujících standardů bezdrátového přenosu. Novějším přístupem, označovaným často za převratný, je takzvané Turbo Kódování. (Název odvozen od podobnosti s turbodmychadlem spalovacího motoru). Turbo kódy jsou novou variantou konvolučního kódování s diametrálně vyšší účinností ochrany oproti dosavadním přístupům. Při chybovosti BER 1:100000 dosahovaly dosud používané kódy maximálně na 3,5dB k Shannonovu limitu. V roce 1993 představili Berrou,

Glavieux a Titimajshim kódovací metodu schopnou při chybovosti BER 1:100000 dosáhnout až 0,7 dB k Shannonovu limitu [21].

Základem tohoto kódovacího postupu je paralelní zapojení dvou RSC kodérů na straně vysílače, kde jeden je zapojen s předřazeným překladačem. Vzniknou tak tři verze odesílaných dat, původní vstupní data, výstup z kodéru č.1 a kodéru č.2. Tento výstup je multiplexován a odeslán. Na straně přijímače není po demodulaci rozhodnuto na základě rozhodné hladiny o stavu 0 nebo 1, ale signál je vyhodnocen jako pravděpodobnost 0 nebo 1 (SISO - Soft Input Soft Output). Tato pravděpodobnost je pak iterativním způsobem opakovaně zpracována a porovnávána mezi sebou. Po průchodu dekodérem je teprve rozhodnuto o tvrdém výsledku 0/1. Tato metoda kódování je velmi účinná a aplikací je celá řada, od vesmírné komunikace po pozemní telekomunikace. Nevýhodou je náročnost dekódování a určité časové zpoždění, dané právě větším počtem nutných iterací.

2.3 Zdroje rušení a vznik interferencí

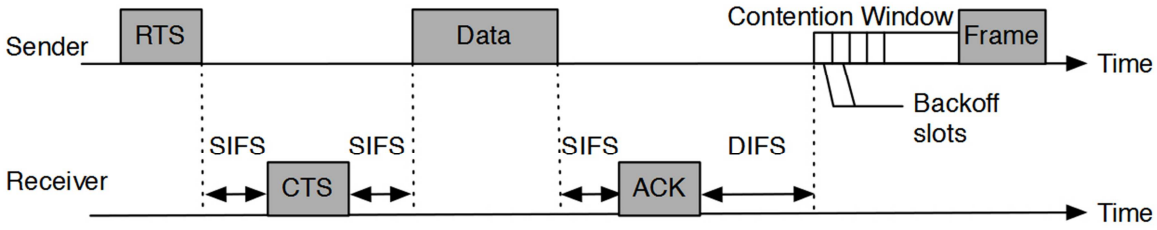
Pásmo 2,4 GHz je, jak bylo v úvodu řečeno, nelicencované, tedy volné. V současné době je jedním z nejvíce využívaných pásem a provoz v něm musí tedy odolávat vysokým nárokům z pohledu rušení. Pásmo je definováno mezi frekvencemi 2400 MHz a 2483 MHz. V následujícím textu jsou popsány běžně používané standardizované bezdrátové technologie, které mohou být koexistenčními partnery pro ZigBee. Dále jsou popsány další možné zdroje rušení, kterým je třeba při provozu sítě čelit.

2.3.1 Soubor norem 802.11

Tento soubor norem obsahuje bezdrátové standardy pro sítě WLAN (Wireless Local Area Network). Jedná se o nejčastěji se vyskytující standardy v pásmu 2,4 GHz, které v podobě nejnovějších verzí (g, n) také zabírají největší část pásma na jednu síť. Stavba rámců pod normou 802.11 je velmi podobná jako u 802.15.4, která byla dříve rozebrána s rozdílem, že typů rámcu je u 802.11 širší škála. Jejich detailním popisem se v tomto textu nebudu zabývat, nicméně protože bude v měření použita jako koexistenční partner síť 802.11g, bude zde zmíněno několik základních informací o principech komunikace v tomto standardu Datové přenosy jsou na rozdíl od ZigBee, mírně složitější například v uvodní sekvenci RTS/CTS.

- RTS – „Ready to Send“ – „Data připravena k odeslání“
- CTS – „Clear to Send“ – „Připraven k přijetí dat“

Jedná se o signální mechanismus, postavený na dvou typech rámců, jsou v úvodu datové komunikace vyměněny mezi odesilatelem a příjemcem. Na straně odesilatele (RTS) jako informace o datech pro příjemce a na straně příjemce rámec CTS jako odpověď deklarující připravenosti k příjmu dat. Následuje datový rámec a potvrzení jeho správného přijetí příjemcem ACK rámcem. Tato rutina (viz. Obr. 22) platí pro odeslání jednoho datového paketu. V případě většího množství dat odesílaných najednou (v tzv. fragmentu), není tato sekvence prováděna pro každý jednotlivý datový rámec, ale některé části sekvence jsou vynechány a nahrazeny jinými typy rámců (např. Block ACK potvrzují příjem sekvence datových paketů) [9].



Obr. 22. Přenos paketu na standardu 802.11 [50]

Rámce jsou při přenosu oddělovány krátkými časovými intervaly SIFS (Short InterFrame Space) a DIFS (Distributed InterFrame Space). Stejně jako ZigBee i soubor norem 802.11 využívá naslouchání nosnému médiu, každý přenos je synchronizován do takzvaného Contention Window. Principiálně se jedná o totožný postup, jako byl popsán u 802.15.4, kdy každý z účastníků sítě, který chce využít nosného média, v tomto okně naslouchá kanálu a v případě že je tento volný, spustí vysílání. Vyhodnotí-li kanál jako obsazený, dojde ke spuštění vyčkávací rutiny, po jejímž uplynutí je naslouchání opakováno. Viz odstavec věnovaný CSMA mechanismu 2.1.2. Tato doba se při každém vyhodnocení kanálu jako obsazeného zdvojnásobuje až do maximální hodnoty 255 [9][11]. Z pohledu koexistence těchto dvou standardů zde bude popsán konkrétní popis některých parametrů, které budou v dalším textu použity:

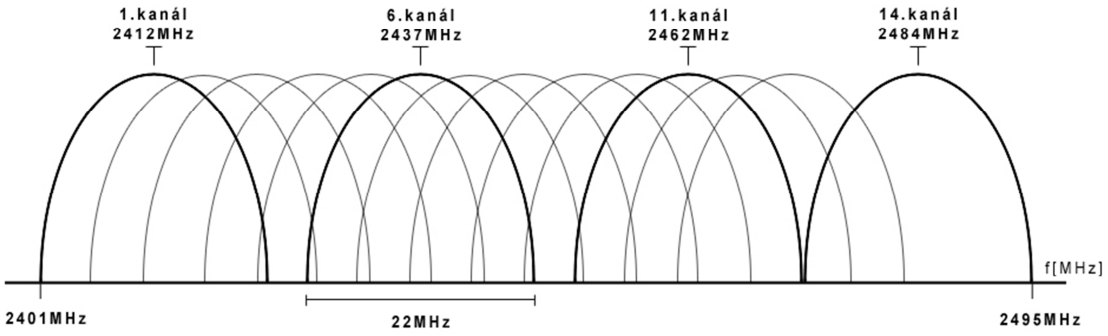
Tab. 4. Základní parametry skupiny standardů 802.11. [9][11][12]

Parametr	Hodnota	
	802.11b	802.11g
SIFS	10 µs	10 µs
DIFS	50 µs	28 µs
PHY Header	192/96 µs	20 µs
Délka RTS rámce (Byte)	20	20
Délka CTS rámce (Byte)	14	14
Délka ACK rámce (Byte)	14	14
Cont. Window Min./Max. délka (slotů)	31/1023	15/1023
Min. délka rámce	202 µs	194 µs
Max. délka rámce	1906 µs	542 µs

Detailní rozbor všech funkcionalit není cílem této práce a veškeré případné souvislosti budou rozvedeny dále v textu. V následujících několika odstavcích jsou uvedeny základní parametry fyzických vrstev jednotlivých podstandardů, které se přímo týkají možného překrytí kanálů ZigBee.

2.3.1.1 **WiFi - 802.11b**

Starší verze standardu WiFi je dnes obsažena ve většině WiFi zařízeních jako standard zpětné kompatibility. Jeho teoretická přenosová rychlost je 11 $Mb.s^{-1}$. Standard dělí pásmo na 14 kanálů o šířce 22 MHz, které mají rozestup 5 MHz [9]. Rozložení kanálů je na Obr. 23.



Obr. 23. Rozložení kanálů 802.11b/g [9]

V následující tabulce je frekvenční rozmístění kanálů v pásmu 2,4 GHz pro 902.11b/g (pro "g" v módu DSSS):

Tab. 5. Rozložení kanálů v pásmu 2,4 GHz pro 902.11b/g [9]

Kanál	Frekvence [MHz]
1	2412
2	2417
3	2422
4	2427
5	2432
6	2437
7	2442
8	2447
9	2452
10	2457
11	2462
12	2467
13	2472
14	2484

2.3.1.2 **WiFi - 802.11g**

V současné době je pomalu nahrazován standardem 802.11n, nicméně v instalovaných aplikacích WiFi převládá 802.11g valnou většinou. Jeho teoretická přenosová rychlost je 54 $Mb.s^{-1}$. Rozložení kanálů, jejich šířka i rozestup a překrytí je totožný s 802.11b (viz. Obr. 23). Zvýšení propustnosti je dosaženo použitím OFDM modulace. OFDM "orthogonal frequency-division multiplexing" je metoda modulace, která umožní provoz sousedních kanálů s větším překrytím bez vzájemného rušení. Standard 802.11g umožňuje použití jak OFDM, tak DSSS. V případě OFDM je šířka jednoho kanálu 20 MHz, který je rozdělen na 52 subkanálů, z nichž je 48 datových (sloužících k samotnému datovému přenosu) a 4 jsou pilotní pro přenos kalibračních informací a zpráv o chybách.

2.3.1.3 **WiFi - 802.11n**

Nejnovější verze souboru norem 802.11 umožňuje teoretickou přenosovou rychlost 150 $Mb.s^{-1}$. Při použití vlastnosti MIMO 4x4 lze teoretickou propustnost zvýšit na 600 $Mb.s^{-1}$. Šířka jednoho kanálu je buď 20 MHz, nebo 40 MHz při spojení dvou sousedních kanálů. Kanál o šířce 20 MHz je využit v šíři 16,25 MHz a rozdělen na 56 subkanálů, z nichž jsou 4 pilotní pro přenos korekcí chyb a kalibračních dat. Při použití kanálu o šířce 40 MHz je dělení rozšířeno na 114 subkanálů, kde 108 z nich

přenáší data a 6 z nich je vyhrazeno pro obsluhu přenosu. Pro čtyřiceti megahertzovou šíři kanálu lze zvolit 9 kanálů. Z těchto devíti se pouze dva kanály nepřekrývají, a sice kanál 3 a kanál 11. S kanálem č. 11 pak mohou nastat problémy s vyhláškami v různých zemích, protože horní hranice pásma je omezována dle potřeb jednotlivých států. Proto lze s jistotou hovořit pouze o jednom kanálu.

2.3.2 Bluetooth 802.15.1

802.15 je skupinou norem, zabývající se WPAN sítěmi (Wireless Personal Area Network). Jedná se o bezdrátové personální sítě – tedy s relativně krátkým dosahem. Jedná se o stejnou rodinu standardů, ze které pochází i ZigBee.

Bluetooth byl původně navržen jako bezdrátová náhrada za RS232 pro mobilní telefony. Standard pracuje v pásmu 2,4 GHz s omezením 2400 - 2483,5 MHz. Jako ochranné pásmo je nevyužit prostor 2 Mhz v dolní části a 3,5 Mhz na horním konci spektra. Norma definuje 79 kanálů šíře 1 MHz, jejichž střední frekvence odpovídají:

$$f = 2402+k \text{ [MHz]}, \quad k=0,\ldots,78 \qquad (3)$$

Standard definuje pro zvýšení spolehlivosti metodu frekvenčních skoků FHSS (Frequency Hopping Spread Spectrum) s cyklem až 1600 skoků.s^{-1}. Další možností je AFH (Advanced Frequency Hopping). Tato metoda přidává vlastnost mapování rušených kanálů, které vyřazuje z použití. Skoky se pak omezují jen do částí spektra, kde nebylo zaznamenáno rušení.

2.3.1 WiMax – 802.16

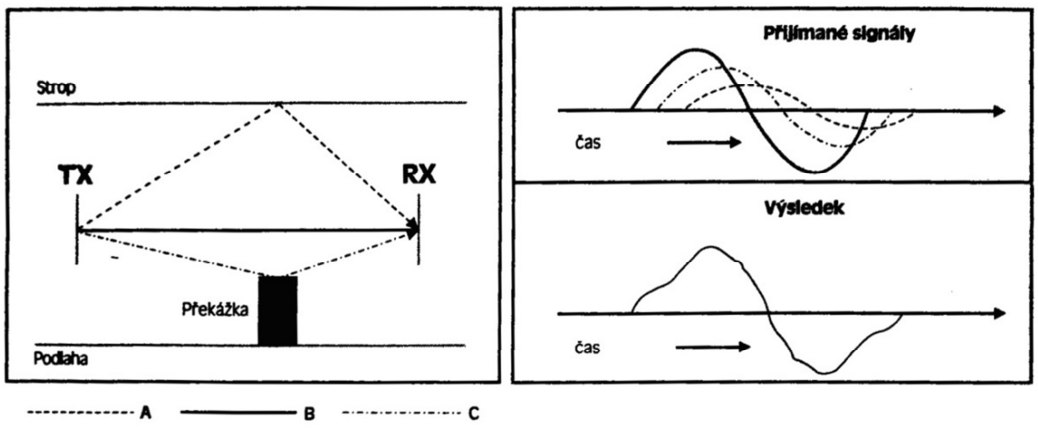
802.16 je norma zabývající se MAN (Metropolitan Area Networks) sítěmi. Wimax - Worldwide Interoperability for Microwave Access, je norma, která v současnosti také nabývá na významu především pro páteřní spojení, ale i pro koncová připojení internetu. Standard lze provozovat od 2 do 11 GHz, ať již v licencovaném, či nelicencovaném pásmu. Šířka kanálu může být použita od 1,25 až po 20 MHz. Podle šířky kanálu je pak určený frekvenční rozsah rozdělen pomocí OFDM na řadu subkanálů (až 2048). Vzhledem k zarušení pásma 2,4 GHz, je WiMax provozován téměř výhradně mimo toto pásmo, buď v alternativním nelicencovaném pásmu 5 GHz, nebo v licencovaných pásmech.

2.3.1 Wireless USB

WirelessUSB je technologie (ačkoli nezařazena do žádné standardizační struktury), vyvinuta firmou Cypress jako vlastní řešení bezdrátového připojení USB periferií k PC. Jedná se o technologii, pracující v pásmu 2,4 GHz. Dělí pásmo na 79 kanálů o šíři 1 MHz, jako je tomu u 802.15.1. Je použita metoda rozprostřeného spektra DSSS s možností alternativní změny na FHSS. Vysílací výkon zařízení je maximálně 0 dBm, tedy relativně nižší než u konkurenčních zařízení v pásmu 2,4 GHz [propagační mat. fy. Cypress]. Standard je velice podobný Bluetooth a šíře jeho využití není jasná.

2.3.1 Interference zpožděním vlastního signálu

Dalším závažným faktorem, snižujícím kvalitu spoje, jsou interference, vznikající zpožděným příjmem vlastního signálu. Interference vzniklá odrazem signálů od překážky přichází k přijímači se zpožděním, úměrným délce trasy. Na Obr. 24 ukazuje příklad odrazu signálu k přijímači alternativní trasou. Signál odražený má sníženou energii (amplitudu) a zároveň je v čase opožděn oproti signálu přijatému přímou cestou. Ukázka distorze sinusového průběhu je pak na Obr. 24 vpravo dole, kde je suma přijatých signálů odražených od dvou překážek a přímou cestou. Modelovací nástroj Matlab Simulink na tento druh vzniku rušení pamatuje a umožňuje jeho simulaci v podobě „Multipath Rayleigh Fading Channel“ modulu, který simuluje právě vicecestné šíření signálu k přijímači.



Obr. 24. Vznik interferencí zpožděným příjmem odraženého signálu [13]

III. CÍLE DISERTAČNÍ PRÁCE

V rešeršní části práce jsou shrnuty základní principy, týkající se bezdrátových přenosů. Jsou popsány nejrozšířenější běžně užívané přístupy ke zvýšení spolehlivosti v bezdrátových standardech. Jsou rozebrány důležité principy a funkce standardu ZigBee. Jako informačních zdrojů bylo použito především norem, týkajících se bezdrátových standardů, ale také vědecké literatury, spojené s měřením reálných parametrů sítí, jejich modelování a v neposlední řadě literatury, spojené s metodami dopředné korekce chyb. Cílem práce je výzkum možných metod pro zvýšení spolehlivost ZigBee spoje, tvořeného transceivery Texas Instruments CC2530EM a výzkum vlastností vybrané metody a její implementace do software transcieverů tak, aby splňovala podmínky zadání. Dalším cílem je návrh komunikačního formátu, vhodného ke zvolené metodě.

Postup při zpracování úkolu:

- Rešeršní výzkum spojený s problematikou
- Návrh a tvorba modelu ZigBee sítě
- Rozbor metod spojených se zvýšením spolehlivosti
- Návrh vhodné varianty a vyhodnocení možností její implementace
- Realizace metody na modelu ZigBee spoje
- Simulace vlivu metody na ZigBee přenos
- Realizace metody na reálném ZigBee spoji
- Měření v reálných podmínkách při koexistenci s WiFi
- Měření v reálných podmínkách s širokopásmovým rušením
- Zpracování výsledků

Hlavními podmínkami při výběru vhodných metod:

- Zachování kompatibility se stadardem ZigBee
- Zachování schopnosti obousměrné komunikace s neupravenými nody
- Zachování alespoň 50 % datové kapacity paketu
- Zvýšení odolnosti vůči bílému šumu i koexistujícím standardům
- Výběr s ohledem na nízkou energetickou náročnost

Obecně tedy lze specifikovat podmínky jako zachování všech standardních vlastností transcieveru a vytvoření pouhé nadstavby umožňující komunikaci se zvýšenou spolehlivostí. Experimentální práce bude probíhat jak v laboratorním pracovišti, kde bude zprácován upravený protokol ZigBee standardu pro mikroprocesor 8051 a také simulační model, tak na měřícím pracovišti ve volném prostředí, kde bude měřen praktický přínos metody na ZigBee spoj, zatěžovaný různými rušivými vlivy.

IV. ZVOLENÉ METODY ZPRACOVÁNÍ

Tato část práce se věnuje rozboru možných postupů při zvýšení spohlivosti ZigBee technologie. Jsou zde rozebrány možné přístupy a jejich výhody, nevýhody a zákonná omezení s důrazem na podmínky zadání práce. Dále je v této části uveden výsledek rozboru a následné zpracování metody v podobě upravených transceiverů ZigBee a také modelu.

4.1 Zákoná omezení pro provoz bezdrátových sítí

EIRP – (Equivalent isotropically radiated power) v překladu „Ekvivalentní izotropický vyzářený výkon“ je hodnota energie vyzařovaná anténou ve směru jejího maximálního zisku. Využívá se jak pro výkonové srovnání přístrojů pro bezdrátový přenos, tak pro omezení a regulaci jejich výroby. V ČR platí omezení EIRP pro pásmo 2,4 GHz na 20 dBm. Tato hodnota odpovídá 100 mW vyzářeného výkonu. Vztah mezi výstupním výkonem vysílače v mW a dBm [17]:

$$P_{out} = 10\log\left(\frac{P_{out}[mW]}{1[mW]}\right) \rightarrow (dBm) \qquad (4)$$

Vztah mezi výstupním výkonem zařízení a výslednou vyzářenou energií je dán:

$$EIRP = P_{out} + G_a - L_s \rightarrow (dBm) \qquad (5)$$

Kde: P_{out} je Výkon vysílače, G_a je Zisk antény [dBi] a L_s Ztráty anténního svodu [dB]

4.2 Chybovost přenosového kanálu

Bitové chyby jsou nedílnou součástí vlivu každého neideálního přenosového kanálu. Bitová chybovost je obvykle vyjádřena jako BER (Bit Error Rate), tedy pravděpodobnost, že daný přijatý bit je chybný. Její výpočet spočívá ve znalosti počtu odeslaných bitů a počtů chyb, které při přenosu vznikly. Vzorec pro výpočet BER je uveden v rovnících (6) a (7).

$$BER = \frac{bE}{\sum bit} \ (-) \tag{6}$$

$$BER = \frac{bE}{vbit \times t} \ (-) \tag{7}$$

Kde: B_e – počet chybných bitů [b], bit –počet přenesených bitů[b], vbit –přenosová rychlost kanálu [b.s^{-1}], t-čas

Bitová chybovost úzce souvisí s chybovostí paketovou – PER. Vznikne-li v bloku dat jednoho rámce jedna bitová chyba, je celý rámec vadný (bez použití dopředné korekce chyb). Paketová chybovost přenosu je při měření na běžných bezdrátových standardech nejdostupnějším měřítkem spolehlivosti. I v této práci budou výsledky simulací a měření vyhodnoceny ve formě paketové chybovosti PER, případně spolehlivosti přenosu (1-PER).

$$\text{PER} = \frac{\sum F_r}{\sum F_s}(-) \tag{8}$$

Kde: F_r jsou rámce přijaté (-), F_s jsou rámce odeslané (-), PER - paketová chybovost (-)

4.3 Kapacita přenosového kanálu

Kapacitu přenosového kanálu zatíženého Bílým Gaussovským šumem, lze vyjádřit na základě Shannon-Hartley teorému, který říká, že je přímo úměrná odstupu signálu od šumu a šířce kanálu [22]. Závislost kapacity lze ilustrovat rovnicí (9) [22], kde *C* je kapacita kanálu, P_S je výkon signálu na straně přijímače, P_N je výkon šumu na straně přijímače, No vyjadřuje spektrální výkonovou hustotu šumu a *BW* je šířka kanálu. Odvození rovnice lze nalézt v [23].

$$C = BW\,log_2\left(1 + \frac{P_S}{P_N}\right) = BW\,log_2\left(1 + \frac{P_S}{N_O BW}\right) \tag{9}$$

Z rovnice je zřjmé, že kapacita kanálu roste lieneárně s šířkou kanálu a logaritmicky se zvyšováním odstupu signál/šum [22]. Odvozením z uvedené rovnice (9) lze dojít k hodnotě tzv. Shannonova limitu, tedy hodnoty, za kterou nelze provozovat bezdrátovou komunikaci bez chyb, a to bez ohledu na kvalitu kódovací metody. Jak bylo uvedeno v odstavci 2.2.5, přiblížení se k Shannonovu limitu je poměrně obtížné a nejlépe se v tomto ohledu osvědčují Turbo kódy, které dosahují až 0,7 dB k Shannonovu limitu.

4.4 Úbytek signálu v přenosovém kanálu

Signál vysílaný vysílačem je spolu se zvyšující se vzdáleností od antény utlumován prostupem prostředím. Tento útlum je nejvhodněji popsán samotnou normou 802.15.4 [1], která právě pro účely modelování přenosového kanálu vztahy definuje. Matematický model útlumu signálu v závislosti na vzdálenosti od přijímače je tedy odvozen přímo pro standard 802.15.4. Útlum signálu je mírně odlišný v blízkém okolí vysílače d < 8 m, a ve vzdáleném okolí d < 8 m. Norma proto definuje dva vztahy, pro blízké a pro vzdálené prostředí.

$$d = 10^{\frac{(P_t - P_r - 40{,}2)}{20}} \quad (m) \qquad pro\ d < 8\ m \qquad (10) \quad [1]$$

$$d = 8 * 10^{\frac{(P_t - P_r - 58{,}5)}{33}} \quad (m) \qquad pro\ d > 8\ m \qquad (11) \quad [1]$$

Kde: P_r – přijímaný výkon (dBm), P_t – Vysílaný výkon (dBm), d – vzdálenost (m)

Vyjádřením z rovnic, lze získat vztah pro přijímaný výkon na straně přijímače v závislosti na vzdálenosti vysílače a jeho výkonu:

$$P_r = 20.\log_{10}\left(\frac{1}{d}\right) + P_t - 40{,}2\ \ (dBm) \qquad pro\ d < 8\ m \qquad (12) \quad [1]$$

$$P_r = 33.\log_{10}\left(\frac{1}{d}\right) + P_t + 99.\log_{10}(2) - 58{,}5\ \ (dBm) \quad pro\ d > 8\ m \quad (13) \quad [1]$$

Kde: P_r – přijímaný výkon (dBm), P_t – Vysílaný výkon (dBm), d – vzdálenost (m)

Známe-li přijímaný výkon lokální výkon šumu v místě přijímače, lze spočíst parametr odstupu signál/šum, jenž je běžným hodnotícím parametrem kvality přijímaného signálu:

$$SNR = P_r - P_n \quad (dB) \qquad (14)$$

Kde: P_r – přijímaný výkon (dBm), P_n – Výkon šumu (dBm)

4.5 Metody pro zvýšení spolehlivosti ZigBee

Předchozí odstavec nastiňuje problematiku přenosu informací šumovým kanálem, na tomto základě lze odvodit několik možných způsobů zvýšení spolehlivosti přenosu ZigBee technologie.

První, nejsnadněji realizovatelnou možností, jak na základě rovnice (9) zvýšit spolehlivost, je zvýšení vysílacího výkonu vysílače [24]. Tím by došlo ke změněně odstupu signál/šum na straně přijímače, a logaritmickou změnou ke zlepšení chybovosti BER. Toto řešení lze realizovat dvěma způsoby. První z variant je zvýšení výkonu samotného vysílače, což může nepříznivě ovlivnit spořebu vysílače. Konkrétně u ZigBee technologie, která cílí na energeticky nenáročné aplikace lze toto řešení označit za nevhodné. Alternativním způsobem je zvýšení vyzářeného výkonu pomocí zvýšení zisku antény. Toto řešení může mít pozitivní vliv, nicméně vyšší zisk antén je spojen se změnou charakteristiky vyzařovaní signálu, čímž se jejich použití stává méně univerzální. V neposlední řadě, v případě miniaturních zařízení je výměna antén nerealizovatelná. Může také být v rozporu s platnou legislativou, která omezuje maximální vyzářený výkon na 20 dBm. Této limitní hodnoty většina ZigBee transceiverů sice nedosahuje (např. transceiver použitý v této práci má maximální výkon 4,5 dBm, což spolu se ziskem antény a ztrátami na vedení (viz (5)) umožňuje dosáhnout ERIP cca 7 dBm), toto řešení však z již uvedeného důvodu energetické náročnosti a neaplikovatelnosti v plné šíři aplikací mohu pro cíl této práce vyloučit.

Druhou obecnou možností je použití zpětného kanálu pro korekci chyb [24]. Vzhledem k tomu, že ZigBee je paketovým přenosem, není třeba budování alternativního bezdrátového spoje a řešení zpětné vazby ARQ (Automatic Repeat Request) [26] [30] je již ve standardu zabudováno a provedeno pomocí potvrzování příjmu ACK (viz 2.1.1.4). Tato metoda spočívá v odeslání redundantních bitů spolu se zprávou, sloužících ke kontrole správnosti přijatých dat, jejichž vyhodnocení vede či nevede ke spuštění zpětnovazební odezvy. Redundantní data (CRC) v tomto kontrolním mechanismu nejsou obsahově významná (16 b), a do zprávy jsou přidávána na straně odesilatele. Tato korekční metoda nevede k faktické eliminaci bitových chyb, ale pouze k rozpoznání jejich vzniku a řešení opakovaného přenosu. Z tohoto pohledu není v ZigBee technologii možnost zlepšení této zpětnovazební techniky.

Třetí variantou je aplikace změny diverzity signálu. Přenos lze diverzifikovat buď z pohledu prostoru, času nebo frekvence [24]. Prostorovou diverzifikaci, můžeme chápat

jako použití více než jedné antény a přenos dat na dvou paralelních kanálech. Na straně přijímače je pak možno vyhodnocovat oba přijaté signály na základě jejich korelace. Chápeme-li ZigBee jako především miniaturizovanou a mobilní technologii, toto řešení se nejeví jako praktické. Dále by nebyla zachována podmínka kompatibility se standardními transceivery. Frekvenční diverzifikace, zahrnuje více variant. Lze ji realizovat např. jako paralelní přenos na dvou a více frekvencích, či jako metodu frekvenčního skákání, která je popsána výše v této práci. Bohužel tato varianta také nesplňuje podmínku kompatibility s běžným standardem. Poslední variantou diverzifikace přenosu je diverzifikace z pohledu času. Tento přístup spočívá v odeslání zprávy opakovaně s časovým posunem. Tato možnost se jeví jako přijatelná z pohledu kompatibility, ale standard ZigBee již její alternativu používá právě v podobě zpětné vazby ACK. V případě, že není ACK doručeno, je přenos opakován až do dovršení maximálního počtu opakování.

Čtvrtá obecná skupina možností pro zvýšení spolehlivosti, je zapojení metody dopředné korekce chyb do zpracování signálu [24]. Dopřednou korekci chyb lze opět konstruovat více způsoby (např. DSSS, viz 2.2.1). Společným jmenovatelem všech variant jsou však rendundantní data, která jsou využita pro zjištění a opravu případných chyb vzniklých při přenosu. Zavedení dopředné korekce chyb na fyzické vrstvě Zigbee nesplňuje podmínku zpětné kompatibility. Při implementaci nad fyzickou vrstvou nicméně kompatibilita může být zachována. Rámce využívající dopřednou korekci chyb, budou v případě zachování všech hlaviček vyšších vrstev pro transceivery bez schopnosti dekódování zpracovatelné beze změn. Kódované rámce bude možno standartně směrovat či uchovat v paměti. Aditivní kódovací metoda tedy spluňuje požadavky cílů práce. Vlastnosti vybraných vhodných kódovacích metod budou v následujícím textu uvedeny.

4.6 Teoretické základy bezpečnostních kódů

Problematika schopnosti samokorekce kódu byla již částečně naznačena v úvodu práce (viz 2.2.1), v této části jsou jako základ k dalšímu textu popsána některá teoretická zobecnění nastíněné problematiky.

4.6.1 Kódová vzdálenost

Kódová vzdálenost, v případě binárních kódu označovaná jako Hammingova vzdálenost, je počtem rozdílných symbolů mezi slovy A a B. Mají-li být tato dvě slova

porovnáváná z pohledu Hammingovy vzdálenosti, musí být stejně dlouhá [35]. Tato hodnota počtů odlišností je v dalším textu zajímavá především v ohledu schopnosti kódu rozlišit kódové slovo od nekódového (chybného) [24]. Uveďme příklad, mějme slova:

$$A = 11100101$$
$$B = 11010100$$

Jednoduchou operací XOR nad oběma slovy, získáme Hammingovu vzdálenost, tedy:

$$A \oplus B = 00110001$$

Z výsledku vyplývá, že Hammingova vzdálenost slov A a B je d = 3.

4.6.1 Minimální Hammingova vzdálenost

Minimální Hammingova vzdálenost je definována jako parametr daného kódu, kde d_{min} je minimální vzdálenost mezi jednotlivými kódovými slovy kódu. Z tohoto parametru lze především určit zabezpečovací schopnost kódu na základě vztahů: [27]

$$d_{min} \geq 2t_k + 1 \qquad (15)$$

Kde t_k je počet opravitelných bitových chyb kódem s d_{min}.

$$d_{min} \geq t_d + 1 \qquad (16)$$

Kde t_d je počet bitových chyb detekovatelných kódem s d_{min}

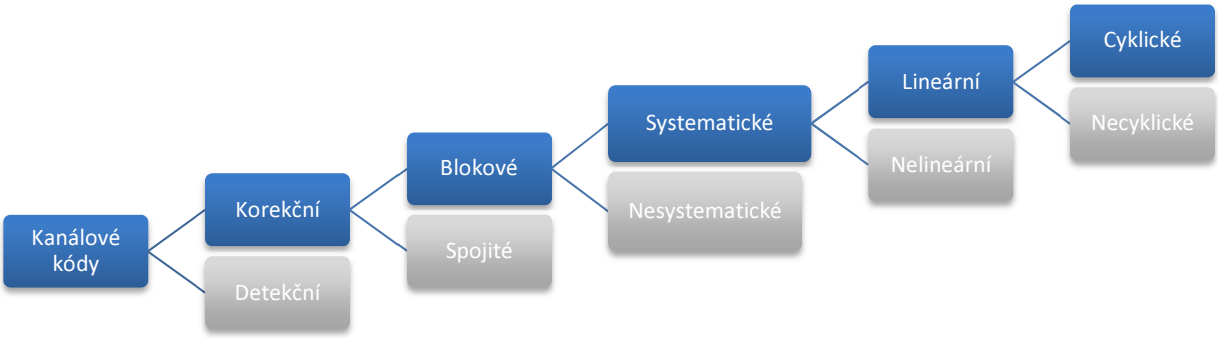
4.6.2 Informační rychlost kódu

Informační rychlostí kódu rozumíme poměr mezi počtem informačních bitů a počtem bitů kódového slova. Definujeme tedy blokový kód parametry (*n,k*), kde „*n*“ je počtem bitů kódového slova kódu a „*k*“ je počtem informačních bitů pro každé kódové slovo. Informační rychlost takového kódu je pak r = k/n. Vezmeme-li příklad v Hammingově kódu (7,4), jeho informační rychlost je cca r = 0,57. Máme-li přenosový kanál s danou propustností, použitím takového kódu bude jeho propustnost snížena na 57 % své kapacity.

4.7 Kódování pro dopřednou korekci chyb

Tato kategorie kódovacích metod, jak již bylo zmíněno, slouží ke korekci chyb vzniklých při přenosu šumovým kanálem. Metoda zahrnuje vlastnosti druhé skupiny kódovacích metod (detekčních), tedy má schopnost chyby detekovat, navíc umožňuje v omezené míře chyby detekované také opravit. Korekční kódy lze rozdělit na dvě

podskupiny, kódy Blokové a Konvoluční. Pro přehlednost je uvedeno dělení kódovacích metod na Obr. 25.



Obr. 25. Dělení kódovacích metod [28][29]

- *Blokové kódy* - zabezpečují určitou předem definovanou část vstupní informace v rámci bloku. Vstupní data, jsou-li delší, než kapacita jednoho kódového slova, jsou dělena na části, které lze samostatně dekódovat [25].
- *Spojité kódy* - jsou takové, které nejsou rozděleny do samostatných bloků a redundantní data jsou vkládána spojitě do datového toku. Na rozdíl od blokového kódu, je výsledek dékódování závislý na předchozích zprávách.
- *Systematické kódy* - využívají ke stavbě kódových slov předem definovanou a pevnou strukturu rozložení informačních bitů a bitů redundantních. Každé kódové slovo o délce „*n*“ pak lze rozdělit na „*k*“ bitů informačních a *n-k* bitů zabezpečovacích. V této podobě pak hovoříme o takzvaném (*n, k)* kódu.
- *Nesystematické kódy* - neumožňují dělení na informační a zabezpečující elementy, nýbrž pracují s polohami nul a jedniček v kódových slovech a také s Hammingovou vzdáleností kódových kombinací.
- *Lineární kódy* - jsou takové, pro které platí, že libovolná kódová kombinace může být odvozena z ostatních kódových kombinací. Tento charakteristický znak je umožněn díky základům v lineární algebře, kdy kódová slova jsou generována pomocí generující matice G, nebo v případě cyklických kódů generujícím polynomem.
- *Nelineární kódy* - za takové jsou označovány ty druhy kódů, které nespadají do charakteristiky kódů Lineárních. Liší se způsobem vzniku i zpracováním, velmi často je definují kódové tabulky obsahující všechna myslitelná kódová slova.

4.7.1 Výběr vhodného segmentu kódovacích technik

Vzhledem k parametrům zadání, ale také vzhledem k omezením, která určují hardware a forma stacku ZigBee, musí realizace metody do ZigBee stacku probíhat mezi PHY a MAC vrstvou. Tím lze zajistit, že základní funkcionality nebudou narušeny. Jedná se zároveň o nejnižší úroveň, na které lze do zpracování dat v mikrokontroléru zasáhnout z pozice aplikace. Vzhledem k těmto faktům a také s přihlédnutím k limitujícím faktorům hardwaru nebude možno získat měkká rozhodnutí dekodérů na straně přijímací. Dále vzhledem k paketovému režimu sítě jsou předmětem zájmu především kódy Lineární Blokové Cyklické.

4.8 Kódy blokové lineární cyklické

Kódy blokové lze podle [24] rozdělit na kódy BCH a Golayův kód. Kódy skupiny BCH lze dále rozdělit na specifickou podskupinu Reed Solomonových kódů a Binárních BCH. Binární pak vycházejí z principů kódů Hammingových. Rozdíly mezi kódy nespočívají pouze v počtu opravitelných chyb a informačním poměru a proto se následující text bude těmto zástupcům věnovat.

4.8.1 Hammingův kód

Byly popsány Richardem Hammingem v roce 1950 a nesou jeho jméno [36]. Hammingovy kódy jsou třídou lineárních binárních perfektních kódů pro jednonásobné chyby. Jejich popis tedy stejně jako u ostatních kódů této kategorie tvoří dvojice parametrů (n,k) kde „*n*“ je délka kódového slova a „*k*“ značí počet informačních bitů. Tvoří základ BCH kódů a délka jejich kódových slov je odvozena od počtu informačních bitů podle:

$$n = 2^m - 1$$

Kde „*m*“ je počet paritních (zabezpečujících) bitů. Dosazením celých čísel za „*m*“ dostáváme varianty Hammingových kódů:

Tab. 6. Varianty Hammingova kódu

m	2	3	4	5	6	…
(n, k)	(3,1)	(7,4)	(15,11)	(31,26)	(63,57)	…
r	0,33	0,57	0,73	0,84	0,90	…

Z Tab. 6 je zřejmé, že s každou delší variantou kódového slova se informační rychlost Hammingova kódu rychle přibližuje k *r = 1*.

Mezi výhody Hammingova kódu patří v neposlední řadě snadné dekódování. Protože každý možný výsledný syndrom po dekódování je rozvojem nějakého čísla „*k*“, pro které platí $1 \leq k \leq m$, lze pozice slova uspořádat tak, aby syndrom korespondoval s konkrétní pozicí bitu kódového slova. Oproti BCH kódům, lze tedy dekódovací proces zkrátit o generování kontrolních tabulek potřebných ke zjištění chyby na základě syndromu a ihned po vypočtení syndromu lze určit a opravit konkrétní chybu v kódovém slově. Zaměřme se v popisu dále na variantu kódu (7,4), která je pro popis nejvhodnější.

4.8.1.1 **Kodér Hammingova kódu (7,4)**

Vytváření kódových slov blokových binárních kódů probíhá nejčastěji generující maticí. Postup generování je uveden pro přehlednost krok za krokem i pro případ, že generující zařízení nebude schopno pracovat s maticemi a zjednodušit tak programování. Každou čtveřici vstupních bitů $p_1\ p_2\ p_3\ p_4$ nejprve provedeme výpočet paritních bitů operací exkluzivního součtu XOR podle rovnic (17)(18)(19)

$$p_1 = b_1 \oplus b_2 \oplus b_4 \qquad (17)$$

$$p_2 = b_1 \oplus b_3 \oplus b_4 \qquad (18)$$

$$p_3 = b_2 \oplus b_3 \oplus b_4 \qquad (19)$$

Výsledky rovnic - bity $p_1\ p_2\ p_3$, jsou paritními bity ke vstupní sekvenci. Tyto paritní bity jsou ve výsledném kódovém slově umístěny v pozicích druhé mocniny, tedy 1, 2, 4 [36]. Výsledné kódové slovo Hammingova kódu je posloupnost:

$$p_1\ p_2\ b_1\ p_3\ b_2\ b_3\ b_4$$

První, druhá a čtvrtá pozice je tedy obsazena paritním doplňkem, zbytek obsazují informační bity. Tato sedmimístná sekvence je Hammingovým kódovým slovem a je schopna korekce jedné chyby.

Druhou možností efektivněji vyjádřit a v některých případech i efektivněji zakódovat čtveřici vstupních bitů je pomocí maticových operací. Představme si vstupní čtveřici bitů jako vektor [41] a Hammingův kóder jako matici G (tzv. Generující matici). Matematickou operaci výpočtu kódového slova pak lze vyjádřit:

$$\begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} = \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{pmatrix}$$

Kde h_1 až h_7 je vektorem zabezpečené kombinace (výsledné kódové slovo).

4.8.1.2 **Dekodér Hammingova kódu (7,4)**

Odhlalení chyb v kódovém slově se děje na základě výpočtu syndromu kódového slova. Je-li vypočtený syndrom nulový, přijaté slovo je platným kódovým slovem (není v něm zřejmá chyba). Tento fakt sám o sobě nutně neznamená, že přijaté slovo bylo přeneseno bez chyby, protože mohlo dojít k tolika chybám, že původní kódové slovo bylo transformováno do jiného platného kódového slova. Jak bylo zmíněno, Hammingův kód dokáže opravit jednoduché chyby, vznikla-li tedy právě jedna chyba, vypočtený syndrom bude nenulový a jeho hodnota bude rovna poloze chybného bitu v kódovém slově. Výpočet syndromu lze opět provést jak pomocí matic, tak jej lze ilustrovat na rovnicích pro výpočet syndromu. Na základě rovnic (20)(21)(22) lze spočít výsledný syndrom (s_1 s_2 s_3).

$$s_1 = b_1 \oplus b_2 \oplus b_4 \oplus p_1 \qquad (20)$$

$$s_2 = b_1 \oplus b_3 \oplus b_4 \oplus p_2 \qquad (21)$$

$$s_3 = b_2 \oplus b_3 \oplus b_4 \oplus p_3 \qquad (22)$$

Druhou zmíněnou variantou je použití kontrolní matice H, kde výpočet syndromu probíhá následujícím způsobem:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix}$$

Na základě výpočtu syndromu přijatého kódového slova lze provést jeho případnou korekci (v případě že syndrom je nenulový). Korekce chyby proběhne prostou inverzí bitu na pozici určené syndromem. Pro názornost vyšel-li chybový syndrom S=(010) pak jeho decimální hodnota odpovídá pozici 2. Korekce bude realizována inverzí bitu na

druhé pozici přijatého kódového slova (vektoru zabezpečené komunikace). Vznikla-li při přenosu pouze jedna chyba, získáme původní zprávu vynětím původních informačních bitů ze sekvence, tedy:

$$b_1 = h_3$$
$$b_2 = h_5$$
$$b_3 = h_6$$
$$b_4 = h_7$$

4.8.2 Rozšířený Hammingův kód

Pro zajímavost je uveden rozšířený Hammingův kód, který je tvořen popsaným algoritmem klasického Hammingova kódu, ale je doplněn jedním paritním bitem. Rozšířená varianta Hammingova kódu (7,4) je tady dlouhá osm bitů - (8,4). Takové kódové slovo je stále schopno jednu chybu opravit, ale navíc případně ještě jednu chybu detekovat. Vzhledem k tomu, že jakkoli zvolená varianta dopředné korekce chyb bude muset obsahovat aditivní CRC hodnotu ke kontrole správnosti celého přenášeného bloku dat, ztráta jednoho bitu v každém kódovém slově k detekci chyb je v tomto případě kontraproduktivní.

4.8.3 Golayův kód

Stejně jako kód Hammingův a kód opakovací, je i kód Golayův perfektním. Golayův kód (23,12) je perfektní pro oblast kódu s kapacitou oprav $t = 3$. V oblasti perfektních kódů bylo také prokazáno, že neexistuje jiný perfektní kód než tři výše uvedené. [31][33][34] Kód Golayův patří jak do skupiny systematických blokových kódů a jeho generující matice je tedy v první části tvořena maticí jednotkovou a v druhé části je vytvořena vektorem (1 1 0 1 1 1 0 0 0 1 0) a jeho cyklickými posuny o jeden bit vpravo, s vyjímkou posledního řádku [27].

$$\mathbf{G}_{Gol} = \begin{bmatrix} 1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,1\,0\,1\,1\,1\,0\,0\,0\,1\,0 \\ 0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,1\,0\,1\,1\,1\,0\,0\,0\,1 \\ 0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0 & 1\,0\,1\,1\,0\,1\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0 & 0\,1\,0\,1\,1\,0\,1\,1\,1\,0\,0 \\ 0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 & 0\,0\,1\,0\,1\,1\,0\,1\,1\,1\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0 & 0\,0\,0\,1\,0\,1\,1\,0\,1\,1\,1 \\ 0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0 & 1\,0\,0\,0\,1\,0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0 & 1\,1\,0\,0\,0\,1\,0\,1\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0 & 1\,1\,1\,0\,0\,0\,1\,0\,1\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0 & 0\,1\,1\,1\,0\,0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0 & 1\,0\,1\,1\,1\,0\,0\,0\,1\,0\,1 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1 & 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1 \end{bmatrix}$$

Kontrolní matici lze snadno dopočíst z matice generující. To že kód je schopen opravy třech chyb lze odvodit z minimální Hammingovy vzdálenosti kódových slov, která nepřesáhne pro všechná kódová slova hodnotu d_{min} = 7. Z minimální Hammingovy vzdálenosti kódových slov také vyplývá opravná schopnost až tří chyb dle vztahu (15) t_k = 3.

4.8.4 BCH kód

BCH kódy jsou skupinou lineárních blokových cyklických kódů, pojmenovanou podle autorů - Boseho, Ray-Chaudhuriho a Hocquenghema. Existují v binární i nebinární podobě a všichni tři autoři popsali kódy nezávisle na sobě v letech 1959 a 1960 [37][31]. V dalším textu bude pozornost věnována již pouze binární verzi kódu. BCH kódy vycházejí z kódů Hammingových a patří mezi významné skupiny kódovacích technik. Mezi kladné vlastnosti kódu patří především schopnost opravovat shluky chyb a velmi dobrá možnost volby parametrů kódu, především Informační rychlosti kódu a s ní spojená schopnost korekce chyb [39][37]. První algoritmus pro dekódování BCH kódu byl navržen a popsán W. Petersonem roku 1960. Další modifikace a optimalizace tohoto postupu byly popsány pány Berklampem, Chienem, Forneyem a dalšími [39]. Jelikož jsou BCH kódy postaveny na základech Hammingových kódů, můžeme je popsat analogicky. Kódy jsou též označovány parametry (n, k) kde n značí délku kódového slova a „*k*“ značí počet informačních bitů v kódovém slově. Délka kódového slova odpovídá rovnici (23). Pro kód platí následující parametry:

Počet znaků kódového slova:

$$\mathrm{n} = 2^m - 1 \qquad (23)$$

Počet informačních znaků:

$$\mathrm{k} \geq \mathrm{n} - \mathrm{m} \,.\, \mathrm{t} \qquad (24)$$

Z rovnice (23) je zřejmé, že délky kódového slova postupují v krocích: 7, 15, 31, 63, 127, 255, 511 a tak dále. BCH kódy jsou generovány pomocí takzvaných vytvářecích mnohočlenů g(x), jejichž řád definuje právě počet paritních bitů r = n - k.

Tedy například pro kód BCH(15, 5) je vytvářecí mnohočlen:

$$G(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 \qquad (25) \quad [37]$$

Pro příklad uveďme také vznik mnohočlenu z kódového slova v binárním tvaru. Slovo [00010000001000] lze vyjádřit jako polynom: [38]

$$x^{10} + x^3 \tag{26}$$

4.8.4.1 **Kodér BCH kódu**

Protože teoretický základ BCH kódu je relativně robustní a také proto, že není cílem tohoto textu detailní rozbor matematických principů, ale spíše jejich praktická aplikace, nebude v této části rozebírán matematicky detailně celý postup zakódování i dekódování. Vezměme příklad kódu BCH(15,5), jehož generující polynom je uveden v (25). Mějme vstupní binární kombinaci znaků „01001“. Tuto kombinaci v polynomickém tvaru (viz (26)) lze vyjádřit jako:

$$x + x^4 \tag{27}$$

Jak uvádí [37] pro získání nesystematické výsledné zabezpečené kombinace stačí provédst násobení obou polynomů takto:

$$G(x)(x + x^4) = x^{14} + x^{12} + x^{11} + x^8 + x^4 + x^3 + x^2 + x \tag{28}$$

Výsledek násobení z rovnice (28) transformujeme zpět do binárního tvaru a získáme nesystematické kódové slovo „011110001001101“, ve kterém ovšem nelze rozlišit bity paritní a informační.

Při systematickém kódování není zabezpečená kombinace generována násobením polynomů, nýbrž jejich dělením. Pro tuto operaci nejprve dojde k rozšíření vstupní kombinace *u(x)* o *n-k* nulových pozic. Toho lze snadno docílit vynásobením vstupního polynomu členem x^{n-k}. Tato operace vytvoří prostor pro doplnění zabezpečujících bitů. Matematické vyjádření kódovacího procesu je tedy:

$$\frac{u(x)\, x^{n-k}}{g(x)} = q(x)\frac{r(x)}{g(x)} \tag{29}$$

Po vydělení není důležitý samotný výsledek q(x), ale zbytek po dělení r(x), který se stává zabezpečující sekvencí v polynomickém tvaru.

Další možností je generování kódových slov pomocí generující matice G(x), analogicky ke kodéru Hammingova kódu (4.8.1.1) nebo Golayovu kódu (4.8.3). Vstupní binární sekvenci si tedy představíme jako vektor délky 5 a tento vektor budeme násobit generujicí maticí: [42]

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

4.8.4.2 **Dekodér BCH kódu**

Postup dekódování (detekce a oprav chyb vzniklých přenosem) je oproti Hammingovu dekódování (4.8.1.2) složitější, neboť u Hammingova kódu došlo k odhalení pozice chyb již při výpočtu syndromu přijatého slova. Zde je, vzhledem k vlastnostem kódu, který umožňuje odhalení a opravu více než jedné chyby, nutno z tvaru syndromu dále výpočetně určit polohu chyb.

V případě výpočetního dekódování dosazujeme do polynomu přijaté zprávy, kořeny generující matice kódu. Kupříkladu budeme-li konkrétně hovořit o zmíněném kódu BCH(15, 5), bude patnácti místné binární slovo převedeno na polynom f(x) a postupně do něj bude dosazeno všech šest kořenů generujícího polynomu G(x) (25) [38]. Výsledek v podobě souboru šesti syndromů $s_1 \ldots s_6$ je základem pro případnou opravu či detekci chyb. Vyšlo-li všech šest syndromů nulových, lze předpokládat, že při přenosu nedošlo k žádné chybě (nebo k tolika chybám, že vzniklo nové platné kódové slovo). Pokud je některý ze syndromů nenulový, dekódovací proces pokračuje s cílem odhalit pozice a počet vzniklých chyb. Tato část je obecně nejsložitější při dekódování BCH kódu [38]. Nejprve je třeba soustavu syndromů převést na jeden polynom, takzvaný Lokátor chyb e(x) dosazením do vazebních rovnic. Ty jsou pro tento případ 3 (kód opravuje 3 chyby). Na základě výpočtu vzniklé soustavy rovnic získáme výsledný lokátor chyb. Výpočtem jeho kořenů je pak možno odhalit polohu chyb [39]. Součtem lokátoru chyb a vstupního kódového slova získáváme opravené kódové slovo, jehož informační část je původní informační sekvencí. Proces dekódování lze provést taktéž pomocí maticového počtu, jehož princip je v podstatě analogický k matematickému dekódování. Proces dekódování je tím složitější, čím delší je kódové slovo kódu.

4.8.5 Reed-Solomonův kód

Tento typ kódování je zobecněním kódů BCH, uvedených v předchozí kapitole. Kód byl popsán v roce 1960 Irvingem Reedem a Gustavem Solomonem v článku [43]. Tento typ kódu, pracující v nebinární formě (symbolové), spadá také do kategorie blokových cyklických systematických lineárních kódů [44]. RS kód je dnes používán, díky své mnohostranosti v široké škále aplikací. Jeho nejznámějším použitím je například zabezpečení informací na diskových nosičích dat CD, později i DVD a BluRay, dále také v televizní pozemní DVB-T i satelitním DVB-S vysílání. V neposlední řadě tohoto kódu využívají i datové přípojky ADSL či komunikační standardy pro kosmický výzkum, mezi prvními např. sonda Voyager.

Oproti BCH kódu, jehož generující polynom má kořeny z Galoisova tělesa GF (2), a tedy pracující s dvouznakovou abecedou, Reed-Solomonovy kódy mají koeficienty z tělesa GF(2^m). Počet znaků abecedy lze definovat dle potřeb aplikace. S počtem prvků Galoisova pole pak souvisí nejen délka kódového slova, ale také aritmetika v poli pracující tak, aby každé dvě kombinace prvků pole byly opět prvkem pole [46]. Proces zakódování vstupní informace i proces dekódování je analogický k BCH kódům s jistými odchylkami. Kde u BCH kódu lze dekódování ukončit nalezením polohy chybného bitu a jeho inverzí (důsledek dvojkové abecedy), u RS kódu je vzhledem k symbolovému zaměření třeba dále určit správnou hodnotu symbolu. Vzdálenost kódových slov je u RS kódu definována jako počet symbolů, ve kterých se daná dvě slova liší. Obecně jsou parametry kódu udávány opět parametry (n, k), kde „*n*“ je počtem symbolů kódového slova (bloku) a „*k*“ je počtem informačních symbolů v zabezpečeném bloku dat [44]. Každý z variant RS kódů obsahuje „*2t*“ redundantních symbolů, kde „*t*“ je počet opravitelných symbolových chyb. Generující polynom má pak vždy stupeň „*2t*“ a primitivní mnohočlen stupeň „*m*“. Z uvedeného vyplývá, že lze RS kód definovat i pro abecedu délky 256 znaků (GF 2^8), což umožní zpracování kódu v rámci osmibitových proměnných a usnadní tak proces implementace v softwarové podobě.

- Délka kódového slova[44][47][48]:

$$n = 2^m - 1 \tag{30}$$

- Počet informačních symbolů v kódovém slově[44][47][48]:

$$k = 2^m - 1 - 2t \tag{31}$$

- Minimální vzdálenost kódových slov[44][47][48]:

$$d_{min} = n - k + 1 \tag{32}$$

- Počet opravitelných symbolových chyb: [44][47][48]:

$$t = \frac{d_{min} - 1}{2} = \frac{n - k}{2} \tag{33}$$

Vzhledem k symbolové orientaci kódu nelze jednoznačně určit jeho schopnost opravy binárních chyb. Kterýkoli symbol se stává chybným v případě porušení jednoho či více bitů, jimiž je tvořen. Vezměme příklad v imaginárním RS kódu, definovaném s 8 b symboly a schopností opravit jeden chybný symbol. Je-li při přenosu kanálem porušen jeden bit v symbolu nebo všech osm najednou není směrodatné, neboť se stále jedná o jeden porušený symbol, který lze kódem opravit. Teoreticky je tedy kód schopen opravit 8 bitových chyb, ovšem pouze jsou-li vhodně distribuovány do jednoho symbolu. Protiváhou této situaci je porušení dvou symbolů jednou bitovou chybou. Tento případ bude nad korekční schopností kódu, protože porušeny budou dva symboly [27]. V dalším textu budou tedy uváděna rozpětí možných bitových korekcí pro RS kódy.

4.8.5.1 **Kodér Reed-Solomonova kódu**

Jak bylo již zmíněno, postup zpracování RS kódu je vzhledem k jeho příbuznosti s BCH kódy analogický. Pro získání systematického kódu lze vyjít z rovnice (29), uvedené v části textu, tykající se systematického kódování BCH kódu. Zpracování se opět řídí pravidly pro matematické operace nad daným Galoisovým tělesem [47]. Detailní rozbor operací, spojených s kódováním lze nalézt například v literatuře [48].

4.8.5.2 **Dekodér Reed-Solomonova kódu**

Dekódovací proces je již složitější, lze jej členit na několik oddělených kroků, z nichž každý lze realizovat několika alternativními postupy. Tyto postupy se liší v náročnosti a vhodnosti implementace pro konkrétní platformu. Proces dekódování lze rozdělit na:

- Výpočet syndromů kódového slova
- Výpočet lokátoru chyb a evaluačního polynomu
- Výpočet pozice chyb
- Výpočet velikosti chyb
- Oprava chyb v kódovém slově

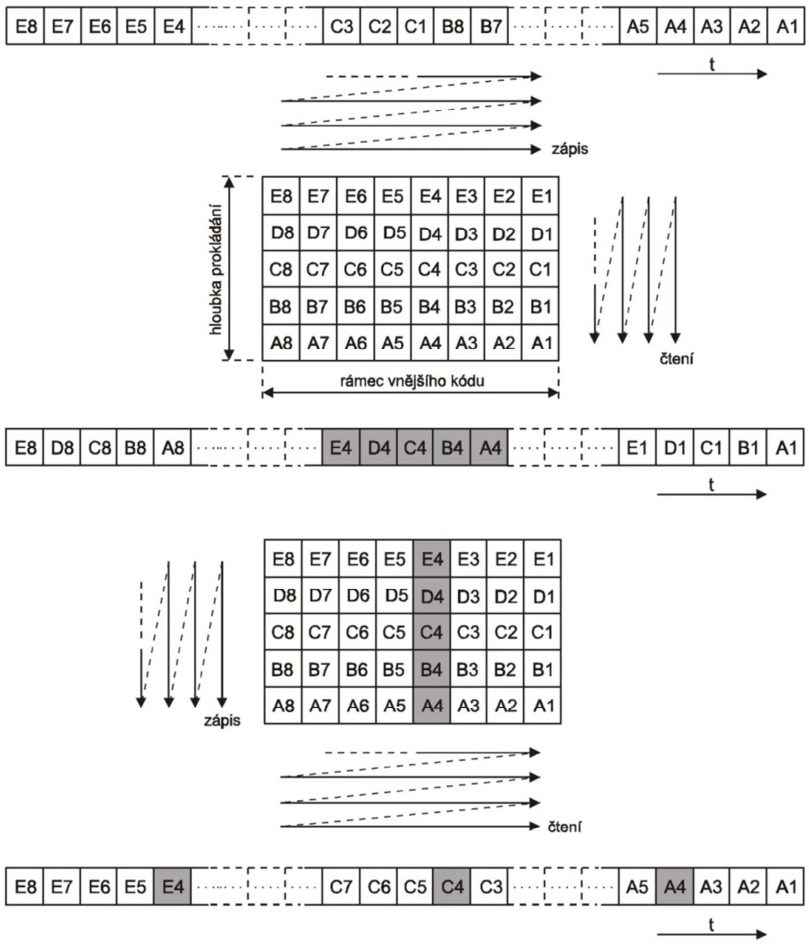
Protože teorie a konkrétní postupy jednotlivých kroků by vyžadovaly rozsáhlý rozbor, který není cílem této práce, omezí se výklad pouze na tyto základní kroky. Popsané vlastnosti kódu budou dále užity při rozboru jejich možností užití v této práci. Detailní rozbor operací spojených s dekódováním lze nalézt například v literatuře [48].

4.9 Blokový prokladač dat

Z uvedené problematiky vyplývá, že korekční schopností kódů, jsou závislé na distribuci chyb. Prokládání dat je technika, umožňující zvýšení odolnosti kódovacích metod vůči shlukům chyb. Základní vlastností všech typů prokládání je změna distribuce chyb, která následně umožní snadnější opravu vzniklých chyb [49]. Jak bylo dříve v textu popsáno, schopnost kódu opravovat shluky chyb roste s délkou kódového slova. Ne vždy je však optimální používání dlouhých kódových slov při opravách shluků chyb. Jedním z hledisek je například stoupající výpočetní náročnost kódovacích technik se stoupající délkou kódových slov.

Blokový prokladač dat, jak název napovídá, mění pořadí bitů v předem definovaném bloku dat. Algoritmus prokládání je definován rozměry prokládací matice, která je určující pro schopnosti rozprostření dat. Jak ilustruje Obr. 26, vstupní datový tok je do matice zapisován po řádcích, když je matice naplněna, data jsou do výstupního kanálu čtena z matice po sloupcích. Tím dojde ke změně pozic bitů, která je závislá právě na rozměru prokládací matice. Na straně dekódování je proveden obdobný proces, který uvede sekvenci se změněnými pozicemi zpět do původnho pořadí. Důležitým parametrem, definujícím vlastnosti prokladače (prokládací matice) je „hloubka prokladu“. Tento parametr určuje, jak dlouhá souvislá část vstupní sekvence je na výstupu prokladače na nesousedních pozicích. Tato vlastnost tedy určuje maximální shluk chyb, které prokladač dokaže redistribuovat na chyby samostatné. Druhým, neméně důležitým parametrem, je Rámec vnějšího kódu. Tento parametr určuje vzdálenost, v jaké budou dříve sousední bity po prokladu. Oba parametry je třeba vhodně přizpůsobit nejen vzhledem k použitému kódu a jeho korekčním schopnostem, ale také vzhledem k rozměrům vstupních dat [27][49]. K zařazení prokladače chyb dochází při použití dopředné korekce vždy, když pravděpodobné shluky chyb přesahují korekční schopnosti jednotlivých kódových slov kódu, nikoli však korekční schopnosti kódu v přenášeném bloku dat. Vhodnou distribucí chyb do bloku tak dojde k lepšímu využití krekčních schopností kódu.

Prokladač dat lze zároveň chápat, podobně jako kódování pro dopřednou korekci chyb, jako metody ke zvýšení zabezpečení přenášených dat. V případě, že se třetí strana snaží odposlechnout data přenášená na bezdrátové síti, jeví se jí pouze jako šum bez zjevného smyslu. Tento vliv funguje, samozřejmě, pouze ve chvíli, kdy odposlouchávající strana nezná přesné parametry kódovacích technik na straně vysílací. Jsou-li tedy podobné metody konstruovány jako unikátní pro konkrétní použití, lze je chápat jako prostředky ke zvýšení zabezpečení přenášených dat.



Obr. 26. Princip blokového prokládání dat [27]

4.10 Srovnání vlastností různých typů kódů

Jak bylo v textu již naznačeno, blokové lineární kódy lze posuzovat na základě:

- Informační rychlosti
- Počtu opravitelných chyb
- Počtu detekovatelných chyb
- Složitosti kódování / dekódování

V tomto odstavci jsou uvedeny vybrané typy kódů s různými parametry, díky čemuž bude snažší volba vhodné kódovací techniky pro zvýšení spolehlivosti ZigBee.

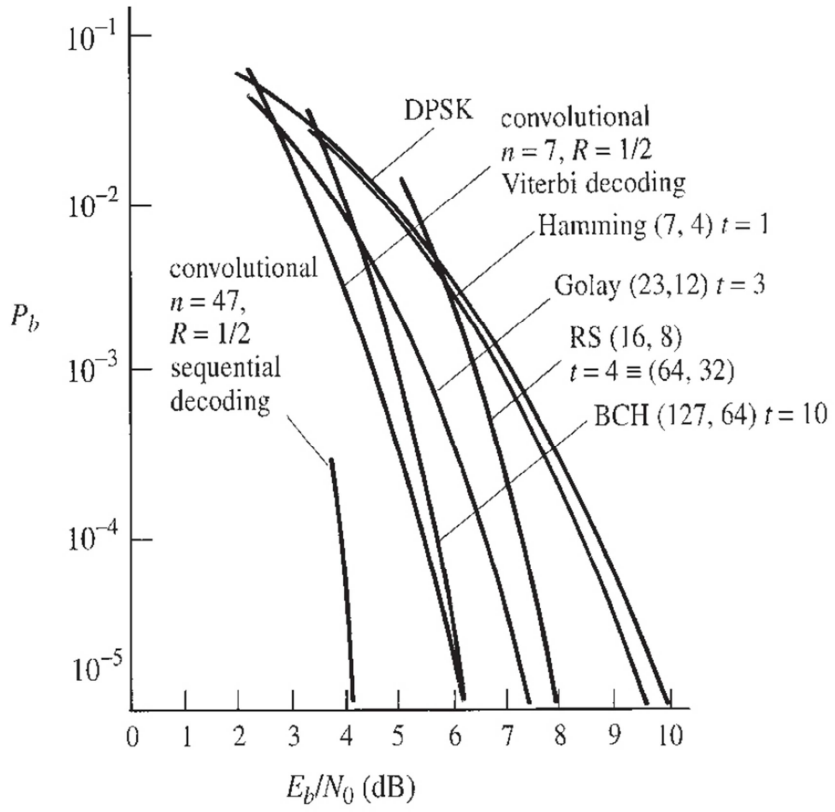
Tab. 7. Přehled BCH kódů do délky slova 511 b

n	*k*	*t*	*n*	*k*	*t*	*n*	*k*	*t*
7	4	1	255	199	7	511	358	18
15	11	1	255	191	8	511	349	19
15	7	2	255	187	9	511	340	20
15	5	3	255	179	10	511	331	21
31	26	1	255	171	11	511	322	22
31	21	2	255	163	12	511	313	23
31	16	3	255	155	13	511	304	25
31	11	5	255	147	14	511	295	26
31	6	7	255	139	15	511	286	27
63	57	1	255	131	18	511	277	28
63	51	2	255	123	19	511	268	29
63	45	3	255	115	21	511	259	30
63	39	4	255	107	22	511	250	31
63	36	5	255	99	23	511	241	36
63	30	6	255	91	25	511	238	37
63	24	7	255	87	26	511	229	38
63	18	10	255	79	27	511	220	39
63	16	11	255	71	29	511	211	41
63	10	13	255	63	30	511	202	42
63	7	15	255	55	31	511	193	43
127	120	1	255	47	42	511	184	45
127	113	2	255	45	43	511	175	46
127	106	3	255	37	45	511	166	47
127	99	4	255	29	47	511	157	51
127	92	5	255	21	55	511	148	53
127	85	6	255	13	59	511	139	54
127	78	7	255	9	63	511	130	55
127	71	9	511	502	1	511	121	58
127	64	10	511	493	2	511	112	59
127	57	11	511	484	3	511	103	61
127	50	13	511	475	4	511	94	62
127	43	14	511	466	5	511	85	63
127	36	15	511	457	6	511	76	85
127	29	21	511	448	7	511	67	87
127	22	23	511	439	8	511	58	91
127	15	27	511	430	9	511	49	93
127	8	31	511	421	10	511	40	95
255	247	1	511	412	11	511	31	109
255	239	2	511	403	12	511	28	111
255	231	3	511	394	13	511	19	119
255	223	4	511	385	14	511	10	121
255	215	5	511	376	15			
255	207	6	511	367	16			

Účinnosti kódu lze hodnotit z více hledisek, kód vhodný pro jednu aplikaci nemusí obstát v aplikacích jiných. Obecně lze usoudit, že čím delší kódová slova s větší Hammingovou vzdáleností, tím větší schopnost detekce nebo korekce chyb v kódových slovech [24]. Jako ukázku závislosti korekčních schopností kódu na jeho informační rychlosti vezměme příklad kódu BCH (63,57). Tento kód má 6 redundantních bitů, tedy informační rychlost cca 0,9. Jeho korekční schopnost je t = 1 bitová chyba. Snížíme-li informační poměr parametrem „*k*“ na hodnotu 45 informačních bitů (BCH (63,45)), sníží se kódový poměr na r = 0,7 a jeho korekční schopnost stoupne na t = 3. Dalším snížením

informační rychlosti na r = 0,4 (BCH(63,24)) zvýšíme korekční schopnost na t = 7 bitových chyb.

Dále pro přehled uveďme (Obr. 27) srovnání praktické účinnosti několika vybraných kódovacích metod s informačním poměrem cca 0,5 při užití s modulací DPSK. Varianta BCH kódu s parametry (127,64), který umožňuje opravu až 10 chyb v kódovém slově délky 127 b. Dále jsou uvedeny vlastnosti Golayova kódu (23,12) s kapacitou tří chyb v kódovém slově o délce 23 b. V neposlední řadě je na grafu uveden také Hammingův kód (7,4) pro opravu jednoduchých chyb. Vlastnosti jsou uvedeny při přenosu s BPSK modulací [24].



Obr. 27. Srovnání účinnosti kódovacích metod s kódovým poměrem cca ½ [24]

Zajímavou vlastností uvedených kódovacích technik je, že za určitých okolností bitovou chybovost zvyšují. V uvedeném grafu lze tuto vlastnost sledovat jako překročení křivky kódovací techniky přes spolehlivost prostého DPSK přenosu. Tento fakt je způsoben tím, že za určitých okolností může specifické množství bitových chyb způsobit při dekódování záměnu kódového slova za nesprávné, čímž může vzniknout větší počet chyb, než byl na vstupní straně. Při datovém přenosu v podobě rámců, je tato vlastnost irelevantní, neboť každá jedna chyba vzniklá při přenosu, vyústí v neúspěch a je proto nedůležité, že neopravitelná chyba na vstupní straně vyústí ve větší počet, neboť taková data nelze

uspěšně přijmout. V Tab. 8 je uveden výběr kódů zmíněných skupin kódovacích technik s informační rychlostí alespoň 0,5 (viz zadání práce). Tabulka kombinuje jak kódovací metody nebinární tak binární. Pro srovnání jsou tedy parametry kódů uvedeny jak v bitovém vyjádření tak symbolovém. Parametr „*n*" uvádí délku kódového slova, „*k*" je počet informačních bitů v každém kódovém slově, „*t*" je korekční schopnost daného kódu a „*r*" vyjadřuje informační rychlost neboli kódový poměr kódu. Je zřejmé, že nejsilnější samoopravné vlastnosti má kód Reed-Solomonův, nicméně výběr vhodného kódu bude proveden v následujícím textu.

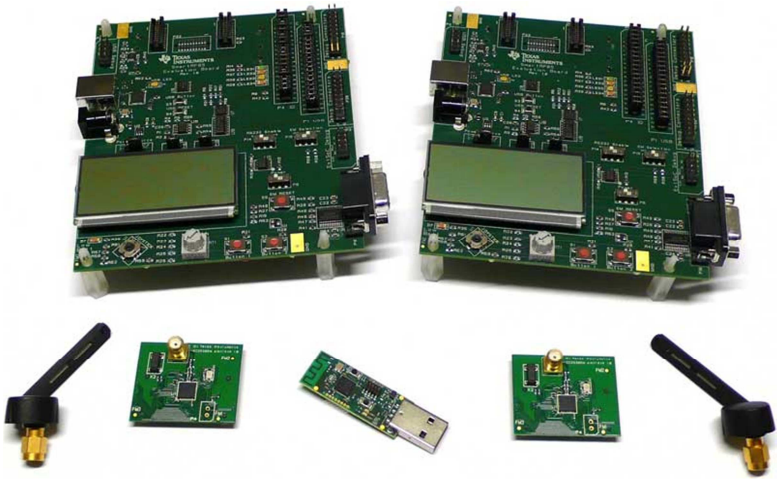
Tab. 8. Přehled parametrů kódů pro dopřednou korekci chyb s *r > 0,5*

	Parametry kódů (n,k)						
	n (symb.)	**k** (symb.)	**t** (symb.)	**n** (bit)	**k** (bit)	**t** (bit)	**r (-)**
BCH	x	x	x	7	4	1	0,571
	x	x	x	31	16	3	0,516
	x	x	x	63	36	5	0,571
	x	x	x	127	64	10	0,504
	x	x	x	255	131	18	0,514
	x	x	x	511	259	31	0,507
Golay	x	x	x	23	12	3	0,522
Reed Solomon GF(2^m)	7	4	1	21	12	1-3	0,571
	15	9	3	60	36	3-12	0,600
	31	17	7	155	85	7-35	0,548
	63	33	15	378	198	15-90	0,524
	127	65	31	889	455	31-217	0,512
	255	129	63	2040	1032	63-504	0,506

V. VÝSLEDKY DISERTAČNÍ PRÁCE S UVEDENÍM NOVÝCH POZNATKŮ

5.1 ZigBee experimentální hardware

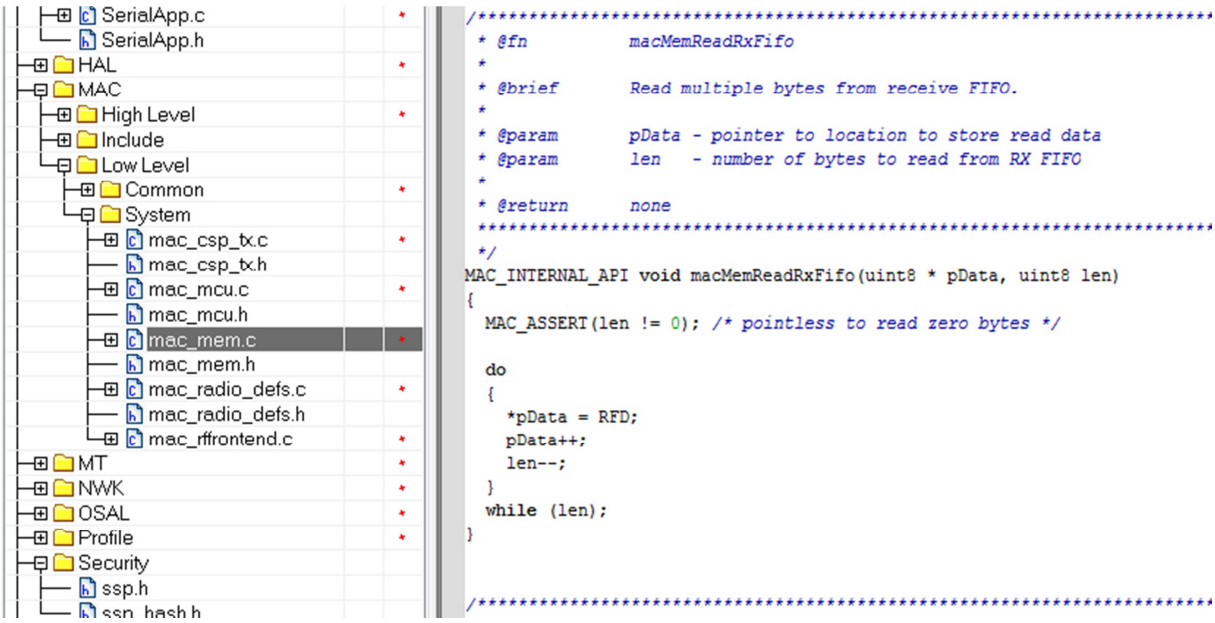
Konkrétní informace, týkající se parametrů ZigBee, jsou vztaženy k experimentálnímu hardwaru použitému v této praci. Jedná se o Development Kit CC2530 fy. Texas Instruments, která jako jedna z mnoha sériově vyrábí mikroprocesory se zabudovanou RF částí pro ZigBee. Experimentální sestava obsahuje dva transceivery, osazené mikroprocesory SoC CC2530, které disponují výstupním výkonem 4,5 dBm a citlivostí příjmu -97 dBm [14]. Základní protokol obsluhy RF části, tedy „ZigBee Stack“ je poskytován výrobcem k jednotlivým mikroprocesorům pod hlavičkou „Z-Stack“ viz [16]. Tento software je programován v jazyce C a je z větší části uživatelem upravitelný. Projekt Z-Stacku je poskytován pouze pro programovací prostředí fy. IAR Systems, tedy „IAR Embedded Workbench for 8051“, viz [15]. Toto profesionální prostředí umožňuje velice precizně optimalizovanou kompilaci C kódu do jazyka mikroprocesoru, navíc umožňuje využít konkretních optimalizací pro příslušný hardware [15]. Sestava dále obsahuje USB Paket Sniffer, tedy ZigBee zařízení, které pouze naslouchá definovanému kanálu a předává přes USB do PC veškerá data přijatá s platnou synchronizační hlavičkou.



Obr. 28. Development Kit CC2530 fy Texas Instruments [14]

5.2 Metody získávání dat pro aplikaci dopředné korekce dat

Z-Stack [16] je tedy jakýsi síťový protokol spojující i funkce „operačního systému“ obsažený v mikrokontroléru, vytvářející a udržující síť ZigBee, jak již bylo dříve zmíněno, není editovatelný v celé šíři kódu. Především MAC vrstva a některé přidružené rutiny jsou předkompilovány a v IAR projektu jsou připojeny jako knihovny funkcí. Tyto knihovny nenesou žádnou běžně čitelnou datovou strukturu, nelze je tedy ani číst ani editovat. Získání přístupu ke kritickým funkcím systému je tedy velice komplikované, v mnoha případech nemožné. Z-Stack umožňuje nastavení takzvaného PROMISCUOUS_MODE, ve kterém jsou na APS vrstvu doručovány i rámce s neplatným ověřením CRC. Tento mód je Z-Stackem nicméně nativně podporován jen z části a k jeho plnému spuštění je opět třeba zasáhnout do předkompilovaných rutin. To jeho plné spůštění znemožňuje. Alternativou se ukázal přístup k datům při předávání dat z fyzické na MAC vrstvu. Všechny tyto rutiny jsou sice také skryty v předkompilovaných knihovnách a při debugerrování kódu nelze tyto kroky vidět. Na malý okamžik nicméně dojde ke spuštění dostupného algoritmu čtení FIFO registru, viz Obr. 29.



Obr. 29. Funkce „macMemReadRxFifo“ umožňující přístup k datům na nízké úrovni

Tato funkce je jedinou uživatelsky dostupnou pro přístup ke všem přijímaným datům, je např. vhodná i ke konstrukci PacketSifferu, apod. V tomto textu bude hrát důležitou roli nejen při stanovení výchozích podmínek aplikace, kde umožní zaznamenávání chyb vzniklých přenosem, ale i pro finální konstrukci metody pro zvšení spolehlivosti. Z rozboru v počáteční části textu, především týkající se struktury rámců a principů

synchronizace je zřejmé, že tato funkce je volána jen a pouze tehdy, došlo-li k platnému rozpoznání SHR hlavičky v radiovém kanálu.

5.3 Parametry konstrukce

Připomeňme v tomto bodě hlavní podmínky konstrukce. Hlavním přínosem metody má být zvýšení odolnosti vůči šumu, rušení vznikajícímu například v točivých strojích či indukčním ohřevu, a také zlepšení koexistenčních vlastností sítě vůči konkurenčním standardům. Rušení vznikající v komutátorech točivých strojů nelze z pohledu časové distribuce matematicky charakterizovat (není rovnoměrně rozložené z hlediska energie v čase). Podobně je tomu například u indukčního ohřevu a podobných zdrojů rušení, především proto, že parametry těchto zařízení jsou různorodé a případná měření charakeru rušení by měla smysl pouze pro konkrétní případ. Lze předpovědět, že distribuce rušivých signálů z těchto zařízení bude z pohledu frekvence v rámci pásma 2,4 GHz přibližně rovnoměrná. Z pohledu času bude distribuce energie signálu velmi různorodá a nelze ji predikovat. Pravděpodobné množství vzniklých chyb lze ovšem charakterizovat pro koexistující standardy. Existujících a v současnosti používaných standardů v pásmu 2,4 GHz je větší množství, z nichž každý standard zahrnuje větší množství komunikačních rychlostí, kódovacích technik, apod. Vzhledem k tomu, že pro návrh kódovací metody je třeba vycházet alespoň z hrubých parametrů možného vzniku chyb, dovolím si podmínky stanovit na základě nejběžněji používané a nejvíce pravděpodobné koexistenční varianty a to se standardem 802.11g.

5.3.1 Sběrné měření na pateřním spoji

Protože zatížení WiFi sítě je primárně určováno provozem na nadřazené síti, jejíž komunikaci WiFi zprostředkovává, nastává otázka, jak charakterizovat právě tyto nadřazené sítě. Provoz na nich může být velice odlišný podle služeb, které jsou na takových spojích realizovány. Za průměr, lze díky měřenému oběmu, považovat data získaná od fy. Interoute (www.interoute.com), která pochází ze směrovače dat na páteřním spoji mezi Madridem a Paříží. Tento spoj zprostředkovává spojení pro tísíce uživatelů, a podle zdroje z firmy na spoji probíhá většina běžně realizovaných služeb. Směrovač dat s rychlostí 140 $Gb.s^{-1}$, konkrétně typ: “Junpier mx960“ umožňuje získání informací o celkovém počtu přenesených rámců a příslušném počtu přenesených dat na vstupním a výstupním portu. Z těchto dvou hodnot byla vypočtena průměrná velikost

přenášeného rámce pro vstupní port a výstupní port i celkový průměr. Následující tabulka uvádí výsledné hodnoty, bohužel lze získat pouze prostý aritmetický průměr, nikoli např. medián. Průměr je vypočten ze zhruba 20 G přenesených rámců, a odpovídající velikosti 14 PB datového oběmu.

Tab. 9. Statistika přenosu na páteřní síti mezi Madridem a Paříží

Směr	Rámce	Data (B)	Průměr (B)
In:	9E+09	3,5E+12	385
Out:	1,1E+10	1E+13	897

5.3.2 Měření charakteristiky WiFi spoje v různých podmínkách

Protože prostý průměr není zcela vypovídající hodnotou, vhodným nástrojem pro zajištění vstupní charakteristiky se jeví měření na reálném spoji 802.11g za různých podmínek. Měření jsem provedl s následujícím vybavením a za následujících podmínek. Za použití programu „Commview for WiFi“ (viz [51]), který umožňuje odposlech paketů přenášených v rámci WiFi kanálu, byla zachycována datová komunikace. Na pracovní stanici bylo jako přijímacího modulu využito USB WiFi modulu TP-Link TL-WN722N, který slouží jako hardwarová základna pro software „CommView for WiFi“. Měření proběhlo na kanálu 1, na kterém bylo odposlechem zjištěno, že není v dosahu příjmu provozována žádná další WiFi síť. Na kanálu 1 byla spuštěna síť WiFi směrovačem TP-Link TL-WR740N, který byl přednastaven do módu b/g. Všechny parametry byly ponechány ve výchozích hodnotách. Dalším prvkem v síti bylo mobilní zařízení (Tablet Samsung P3110), které bylo využito k tvorbě zatížení sítě. Výsledky měření, které obsahují mimo jiné informace o:

- Pořadí rámce
- Typ rámce
- MAC adresa odesilatele paketu
- Cílová stanice
- Velikost rámce
- Komunikační rychlost v $Mb.s^{-1}$

byly exportovány z programu „CommView for WiFi“ v textové podobě do tabulkového editoru. Zde byly dále zpracovány ve třídění do velikostních kategorií a také byla z parametru komunikační rychlosti s_{wifi}, délky l_{wifi} a délky příslušné hlavičky $t_{phy-hdr}$ (viz. Tab. 4) vypočtena doba, po kterou byly přenášeny podle (34).

$$t_{wifi} = \frac{l_{wifi} * 8}{s_{wifi}} + t_{phy-hdr}\ (s) \qquad (34)$$

Z hodnoty trvání přenosu byla pak vypočtena teoretická možnost kolize („kolizní délka“) se ZigBee standardem a odpovídající délka shluku potencionálně ovlivněných bitů podle (35):

$$kd = \frac{t_{wifi} * 4}{16.10^{-6}}\ \text{(b)} \qquad (35)$$

Kde 16.10^{-6} je doba trvání jednoho DSSS symbolu násobená bitovou délkou jeho významu – 4 b (viz. 2.2.1). Výsledek rovnice udává maximální možný počet bitových chyb, který může vzniknout kolizí s paketem délky t_{wifi}.

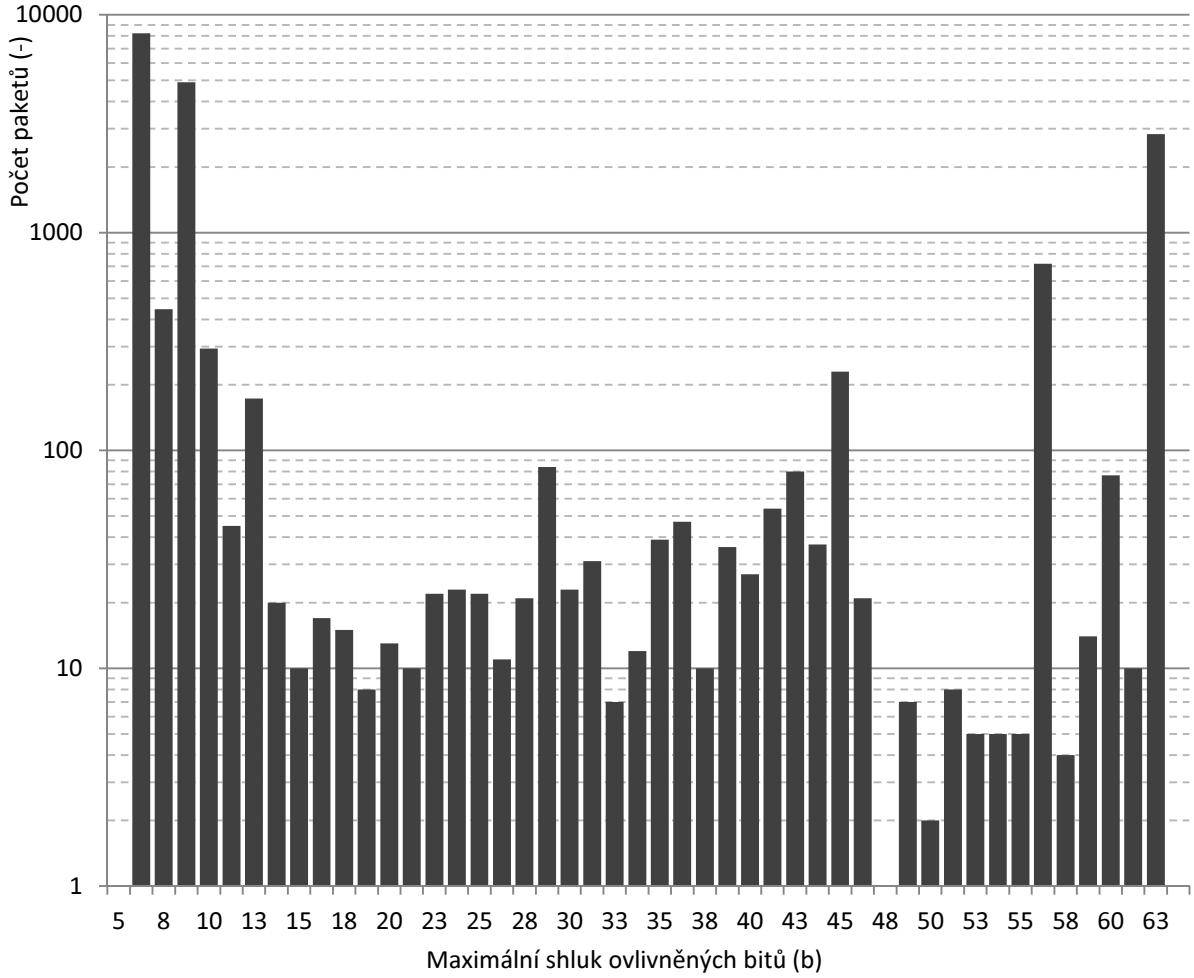
5.3.2.1 Síť WiFi bez zatížení

Při tomto měření, byla bezdrátová síť nevyužívána k přenosům dat mezi zařízeními, žadné z mobilních zařízení nebylo k síti připojeno a tak jsou zachycené pakety výhradně typu Beacon, které vysílá směrovač jakožto správce sítě. Tyto pakety mají podle paketsnifferu délku 116 B a jsou vysílány každých 100 ms rychlostí 1 $Mb.s^{-1}$. Velikost Beacon rámce je závislá na délce SSID názvu sítě, nicméně jeho velikost se nemění razantně. Lze předpokládat, že kolizí vznikne shluk chyb v interferovaném ZigBee paketu o délce 16 až 18 b v závislosti na hlavičce (192/96 μs).

5.3.2.2 Síť WiFi s lehkým provozem

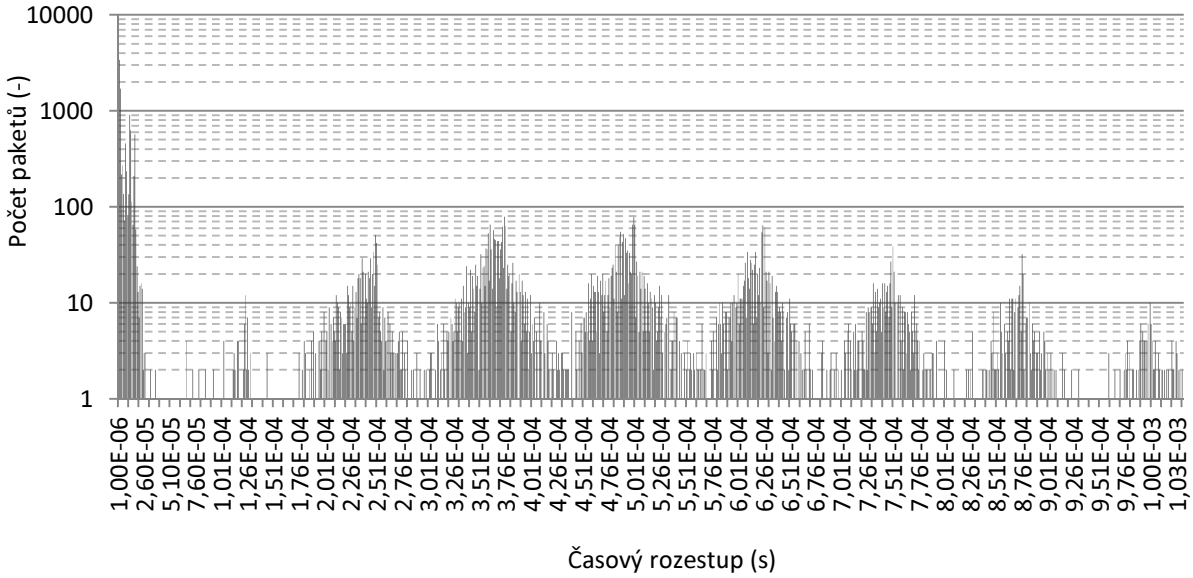
Simulace provozu na síti je záležitostí nesnadnou, neboť je závislá na nepřeberném počtu faktorů, kde hlavní roli nehraje WiFi standard. Pro získání alespoň hrubých obrysů možného vzniku chyb jsem provedl měření, kde síť WiFi byla zatěžována mobilním zařízením připojeným k internetu prostřednictvím WiFi routeru TL-WR740N. Právě provoz na TCP/IP vrstvě budu považovat za jakýsi etalon zátěže případných koexistujících WiFi sítí. Z mobilního zařízení byl pomocí internetového prohlížeče kontinuálně procházen internetový server (cdr.cz) po dobu, než bylo zaznámenáno 20000 paketů. Tyto pakety byly následně zpracovány jak z pohledu třídění jejich velikosti, tak jejich časových rozestupů. Zaznamenávány byly všechny typy paketů (datové rámce, ACK pakety i RTS/CTS pakety). Na Obr. 30 je uveden výsledek měření,

kde na ose x je vyčíslena délka zachycených paketů v podobě maximální ovlivněné délky ZigBee přenosu v bitech.



Obr. 30. Sběrné měření paketového provozu na WiFi (procházení cdr.cz)

Z grafu lze jednoznačně vyčíst naprostou převahu krátkých paketů ACK, kterých bylo zaznamenáno 8647 a také RTS/CTS paketů, kterých bylo zaznamenáno 1334. Ostatní pakety spadají pod protokol TCP/IP a jedná se buď o pakety datové, nebo ACK (rozumějme ACK pakety TCP/IP protokolu, které jsou na WiFi síti přenášeny v podobě datových paketů), kterých bylo zaznamenáno 4640. Jak je vidět na Obr. 30, největší četnosti dosahují pakety o kolizní délce 7 a 9 b. Další velkou skupinou jsou pakety s délkou možného ovlivnění až 63 b. Střední hodnota délky velkých paketů v intervalu 10 až 63 b je rovna 52 b. Na základě rozboru, uvedeném v odstavci 2.3.1 lze očekávat, že interference bude vždy tvořena jedním větším paketem o průměrné délce 52 b ovlivněných bitů, následovaná jedním ACK paketem o délce 7 b a druhým ACK rámcem vyšších vrstev TCP/IP s kolizní délkou 9 ovlivněných bitů, který je ovšem následován dalším paketem ACK WiFi s kolizní délkou 7 b. Jejich časové rozestupy jsou zobrazeny na Obr. 31.

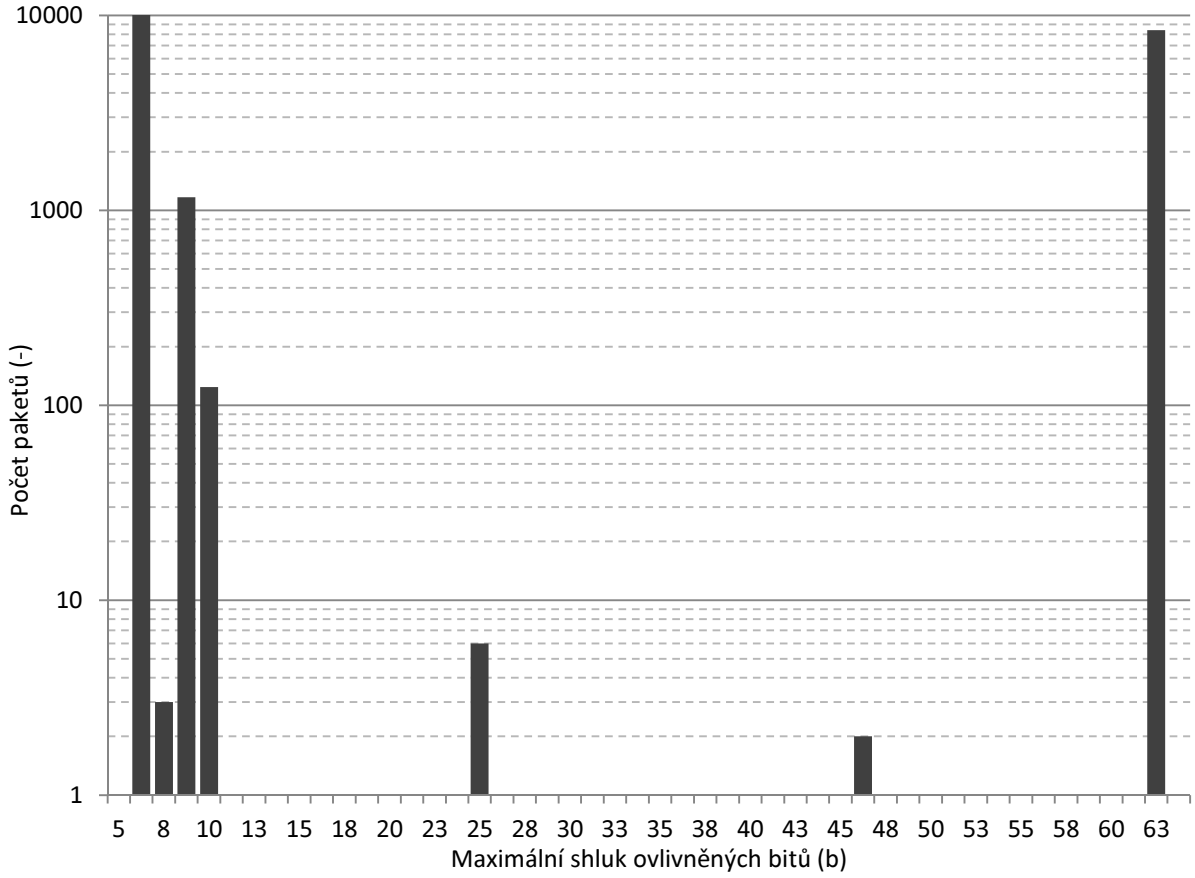


Obr. 31. Rozložení zaznamenaných paketů v čase (procházení cdr.cz)

Z grafu lze číst, že více než polovina (11900) přenášených paketů je zpožděna maximálně 25 μs. Jedná se převážně o ACK pakety standardu WiFi, které jsou vysílany neprodleně po přijetí datového rámce. Zbylé pakety jsou pak zpožděny v závislosti na odezvě internetového připojení, zpracování dat mobilním zařízením a také potřebami samotného procházení internetu. Výpočtem váženého průměru zpoždění paketů se zpožděním větším než 25 μs, lze dojít k průměrné hodnotě rozestupu paketů 540 μs. Měření víceméně odpovídá teoretickému rozboru uvedenému na začátku práce, kde jsou popsány základní principy provozu na WiFi. Lze tedy očekávat, že interferovat bude vždy sekvence s délkou 52 b (205 μs), následovaná pauzou cca 20 μs, a dále bude vysílán buď jeden ACK paket o délce 7 b (25 μs), případně druhý ACK paket TCP/IP protokolu o kolizní délce 9 b (35 μs) a následně další ACK paket 7 b. Odstup ACK paketu TCP/IP protokolu od Datového ovšem nelze přesně vyčíslit, z pořadí zachycených paketů lze pouze usoudit, že ACK pakety TCP/IP protokolu nejsou zasílány neprodleně. Budu proto uvažovat právě variantu 1:1 a prodlevu tedy 540 μs. Z těhto čísel lze vyvodit, že na každý jeden přenášený rámec ZigBee o maximální delce (1056 b – cca 4,2 ms) budou připadat 3 kolize a celkem může v jednom rámci vzniknout cca 225 ovlivněných bitů. S pravděpodobností chyby v ovlivněném bitu 0,5 lze pak předpokládat 113 chybných bitů. Vzhledem k tomu, že vážená průměrná délka „velkých“ paketů s kolizním potenciálem nad 10 b byla 52 b, což je hodnota velmi blízká maximální délce ovlivnění, lze tedy i výsledek 113 vadných bitů v jednom paketu chápat jako horní hranici vzniku chyb.

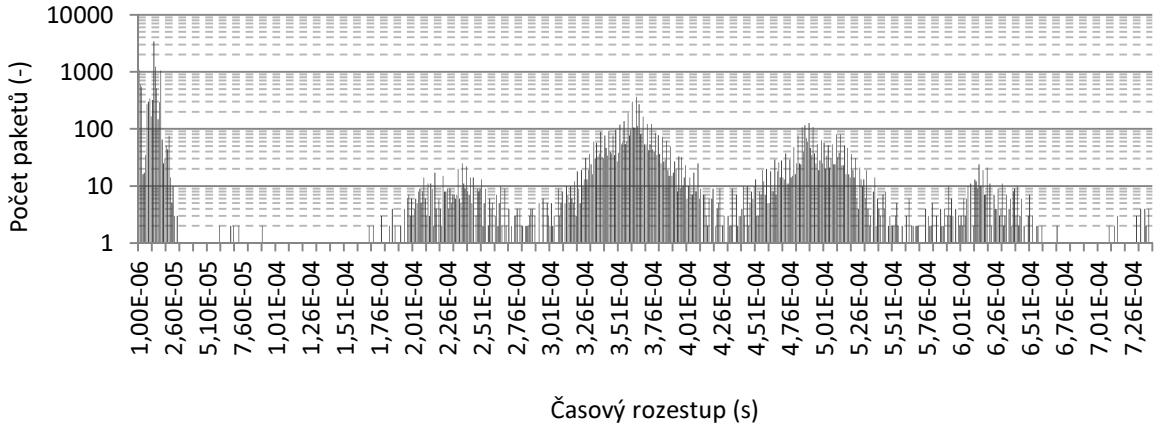
5.3.2.3 **Síť WiFi se silným provozem**

V tomto případě je experimentální sestava použita stejným způsobem, s tím rozdílem, že provoz WiFi sítě není vytvářen procházením internetu, ale stahováním souboru z hostingového serveru pomocí http protokolu. Opět bylo zaznamenáno 20000 paketů, které byly zpracovány stejně jako v předchozím případě.



Obr. 32. Sběrné měření paketového provozu na WiFi (Stahování souboru)

Z grafu na Obr. 32 je zřejmé, že v tomto případě jsou pakety téměř výhradně omezeny na datové rámce s maximální délkou (série 63 ovlivněných bitů), a rámce krátké (ACK rámce s relativně nízkým datovým obsahem lze opět rozdělit na ACK rámce WiFi standardu (10200x) a ACK pakety TCP/IP protokolu (1295)). Na vážený průměr v tomto případě nelze brát zřetel, protože existují v podstatě pouze dvě varianty kolize, buď v délce 63 b a 7 b, nebo 9 a 7 b. Časové rozestupy jsou uvedeny na Obr. 33.



Obr. 33. Rozložení zaznamenaných paketů v čase (Stahování souboru)

Vážený průměr časových rozestupů (nad 25 μs) je pro silné zatížení sítě 435 μs. Zde lze opět očekávat, že vyšší časové rozestupy odpovídají velkým rámcům, přenášeným do mobilního zařízení a také ACK TCP/IP rámcům. V krátkých časových odstupech pak ACK rámce WiFi standardu. Ovlivnění tedy může nastat zaprvé velkým datovým rámcem s kolizní délkou 63 b (250 μs) a následně s prodlevou 20 μs ACK rámcem 7 b. Jako druhou variantu pak lze pak očekávat ACK TCP/IP rámec, následovaný ACK rámcem WiFi s odstupem opět 20 μs. Počítejme i pro tento cyklus opakování se středním rozestupem 435 μs. V jednom přenosu ZigBee o 1056 b (4,2 ms) tedy mohlo dojít buď ke 210, nebo 64 chybám. Protože ACK TCP/IP rámců bylo zachyceno 1295 a „velkých" datových TCP/IP rámců 8420, lze očekávat větší kolizi s pravděpodobností 86 %.

5.3.2.4 **Souhrn experimentu**

Z dvou provedených měření a jejich rozboru nelze predikovat výskyt chyb přesně pro konkrétní podmínky. Problematika provozu na TCP/IP, který především využívá WiFi spoj, je relativně rozsáhlá a v uvedeném rozboru byla určitá jeho část zanedbána. Rozborem však byl stanoven přibližný počet chyb, kterému je třeba se při koexistenci ZigBee a WiFi bránit. Při prvním rozboru pro lehký provoz bylo odvozeno, že maximální délka chyb by měla být přibližně 113 vadných bitů v jednom ZigBee rámci. V druhém případě, kde byla WiFi síť zatížena blízko k maximu, bylo odvozeno, že mohou vzniknout dva druhy interferencí. První s vyšší pravděpodobností o délce 210 b, druhá s menší pravděpodobností o délce 64 vadných bitů. Více o problematice provozu na TCP/IP síti a jejich modelování lze nalézt v [52][53].

5.3.3 Zjištění reálného vlivu standardu WiFi na ZigBee

Protože oba popsané standardy využívají při přístupu k médiu CSMA metodu, lze očekávat, že teoreticky zjištěné parametry nemusí být dostatečně vypovídající. Pro tento experiment bylo využito softwarové úpravy ZigBee transceiverů, kde bylo umožněno aplikaci získávat všechny Fyzickou vrstvou přijímané rámce, bez ohledu na platnost kontrolního součtu CRC CCITT. Přístup mezi Fyzickou a MAC vrstvu byl popsán v bodě 5.2. Následující měření a popsané výsledky zároveň determinují maximální možnosti navazující dopředné korekce chyb.

Měření bylo provedeno na Development boardech fy. Texas Instruments SmartRF 05 osazených mikro-kontroléry SoC CC2530F256. Tyto moduly disponují výstupním výkonem 4,5 dBm a citlivostí příjmu -97 dBm. Software v ZigBee vysílači byl doplněn o aplikaci, odesílající broadcastový paket s aplikačním payloadem délky 95 B tvořeným hodnotami 0xFF. Tento paket byl odesílán s cyklem 105 ms. Hodnota byla zvolena s ohledem na zamezení vzniku trvalé interference s broadcastovými pakety na WiFi, který je odesílán cyklicky po 100 ms.

Jak bylo uvedeno, aplikace v mikrokontroléru na přijímací straně získává veškerá data předávaná z Fyzické na MAC vrstvu. Aplikace byla naprogramována tak, aby nejprve u každého takového rámce ověřila jeho délku. V případě, že délka rámce souhlasí s délkou rámce odesílaného, pak všechna takto přijatá data odešle pomocí linky RS232 do PC. Firmware v přijímacím ZigBee modulu zároveň počítá všechny přijaté rámce s délkou 123 B standartní cestou, tedy doručené APS vrstvou. V další proměnné pak byl zaznamenáván počet všech rámců získaných na MAC vrstvě s kladně ověřenou délkou (tedy takových, které mohou, ale nemusí být bezchybné).

V PC byla data přijímána aplikací vytvořenou pro toto měření v programu VisualBasic, každý takto přijatý paket z RS232 linky byl ukládán do souboru. Následně program scanoval uložené sady dat z pohledu výskytu nul v jednotlivých paketech. Byla-li v paketu nalezena alespoň jedna chyba (nula), byl tento počet načten do proměnné. Zároveň byl pro každou jednotlivou pozici v paketu sčítán počet výskytu chyb. Po dokončení kontroly logovacího souboru byl vypočten průměrný počet chyb v poškozených paketech. Do tohoto souboru tedy nejsou započteny rámce doručené bezchybně a bude možné je následně srovnat s teoretickým rozborem.

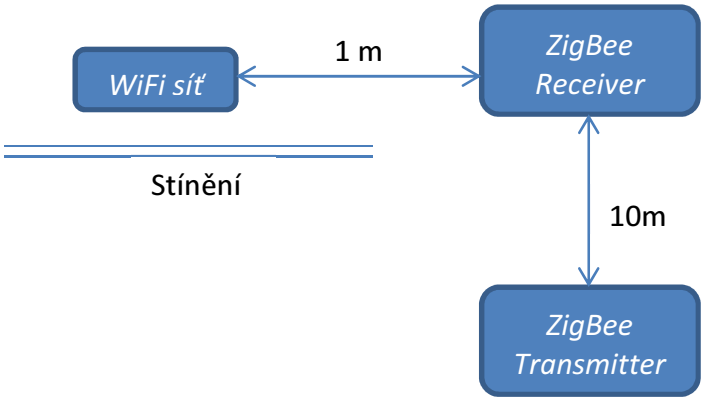
5.3.3.1 Experiment koexistence s WiFi – fixní vzdálenost

Při tomto experimentu byl hardware a software popsaný v nadřazeném odstavci doplněn o sestavu WiFi 802.11g, složenou z routeru USRobotics USR8054 a Tabletu Samsung P3110. Routerem USR8054 byla spuštěna síť na prvním kanálu v módu b/g. Tablet Samsung byl k této síti následně připojen. Datové pakety byly generovány programem Nping v PC, odkud byly přenášeny do routeru USR8054 metalickým síťovým spojem 100 Mb.s-1. Dále pak byla data směrována Routerem přes nešifrované WiFi spojení do Tabletu Samsung P3110. Paketový generátor Nping odesílal ping požadavky s různou délkou náhodných dat pro každé měření. Tyto pakety byly generovány bez prodlevy s cílovou IP adresou náležící Tabletu P3110. Cílové zařízení na ping paket odpovídá paketem stejné délky.

Program Nping byl spouštěn s následujícími parametry:

- *192.168.123.101*
- *--data-length X*
- *--delay Y*
- *--c* 2^{32}
- *--send-ip*

kde „*X*“ je délka uživatelských dat, která byla pro každé měření různá, stejně jako prodleva mezi odesílanými pakety „*Y*“. Paketový generátor „Nping“ (jako ostatní podobný software) naráží na dolní mez rozestupu paketů 1 ms, pod kterou již nelze prodlevu zmenšovat. Je-li nastavena hodnota rozestupu menší, jsou pakety odesílány s nulovým rozestupem, tedy jsou generovány „jak rychle to jde“.



Obr. 34. Rozmístění experimentální aparatury

Na Obr. 34 je uvedeno rozmístění měřící aparatury v místě experimentu. WiFi Router a mobilní zařízení bylo v bezprostřední vzdálenosti a proto jej lze považovat za bod. Ve vzdálenosti 1 m od WiFi spoje byl umístěn ZigBee přijímač. Ve vzdálenosti 10 m od

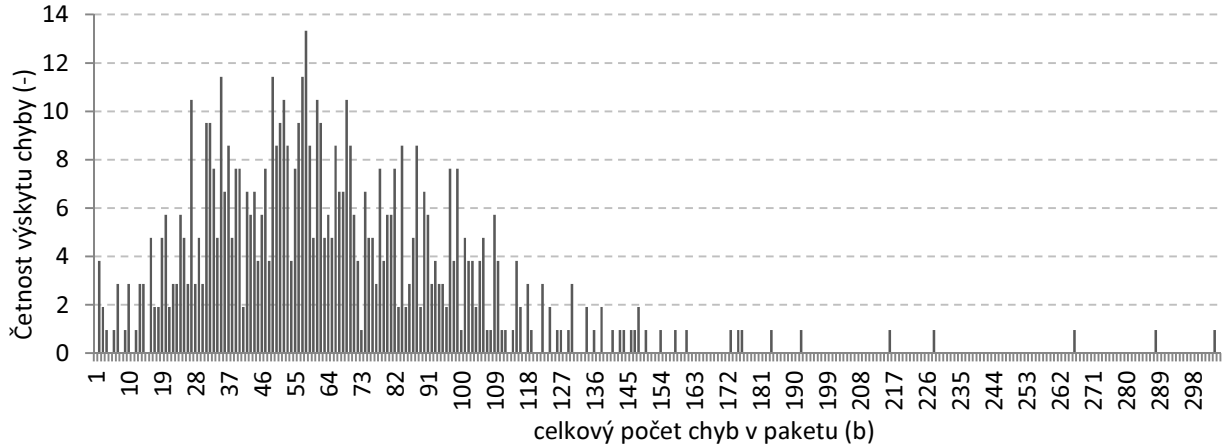
ZigBee přijímače byl pak umístěn ZigBee vysílač. Mezi WiFi sítí a ZigBee vysílačem byla umístěna clona tak, aby byly co nejvěrněji simulovány reálné podmínky, kdy CSMA metody WiFi i ZigBee transceiverů (viz. 2.1.2) mají sníženu účínost, díky nízké hladině signálu koexistující sítě na straně ZigBee vysílače, kdežto na straně přijímače je tento interferující signál dostatečný pro vznik rušení.

Tab. 10. Výsledky měření

WiFi 802.11g			***ZigBee 802.15.4***			
Velikost paketu (B)	Kolizní síla (b)	rozestup (ms)	Standard (paketů)	PER (-)	Pr. Err (b)	MAC kanál (paketů)
63	7	0	76	0,848	4,2	914
63	7	1	504	0,496	21,2	938
63	7	5	510	0,490	12,4	938
63	7	10	507	0,493	14,2	950
63	7	25	616	0,384	11,7	968
63	7	50	686	0,314	17,7	979
96	9	0	4	0,996	15,2	752
96	9	1	522	0,478	27,3	951
96	9	10	494	0,506	10,6	930
96	9	25	622	0,378	13,3	974
96	9	50	682	0,318	16,4	978
1300	53	0	3	0,997	57,4	653
1300	53	1	416	0,584	30	900
1300	53	10	424	0,576	32,3	925
1300	53	50	661	0,339	27,8	968

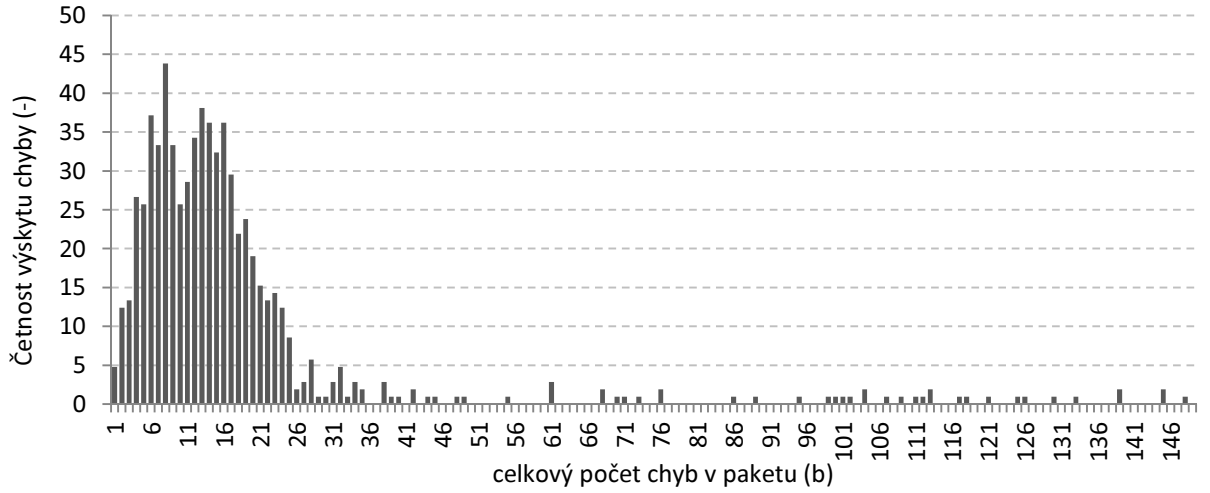
V Tab. 10 je uveden přehled výsledků měření při třech různých velikostech rámců generovaných *Nping* generátorem – 63, 96 a 1300 B. Tyto velikosti byly zvoleny dle výsledků teoretického rozboru. Pro každou velikost paketu byla provedena měření s rozestupy od 0 až do 50 ms. Pro každé jedno měření bylo 5x odesláno ZigBee odesilatelem 1000 paketů s aplikačním payloadem tvořeným hodnotami 0xFF. Z pěti měření byl následně vypočten průměr. V druhé (pravé) části Tab. 10 jsou pak uvedeny měřené reakce přenosu ZigBee. První sloupec uvádí počet standardně doručených paketů, druhý sloupec pak odpovídající hodnotu PacketErrorRate. Sloupec „Pr. Err“ obsahuje průměr počtu bitových chyb v rámcích, v nichž se vyskytla alespoň jedna chyba. Sloupec MAC kanál, obsahuje počty rámců získané na MAC vrstvě s korektní hodnotou délky rámce. Tyto počty nezohledňují správnost přijatých rámců, jsou uvedeny pouze jako informace o maximálních možnostech jakékoli dopředné korekce chyb. Uvádí tedy počet rámců, který je možno v daných koexistenčních parametrech identifikovat a předat k dalšímu zpracování. Důležitým výstupem je počet chyb

v jednotlivých přijímaných ZigBee paketech, na jehož základech budou nastaveny požadavky na protichybové kódování. Na Obr. 35 je uveden histogram četností chyb pro jednotlivé bitové pozice v aplikačním payloadu rámce ZigBee.



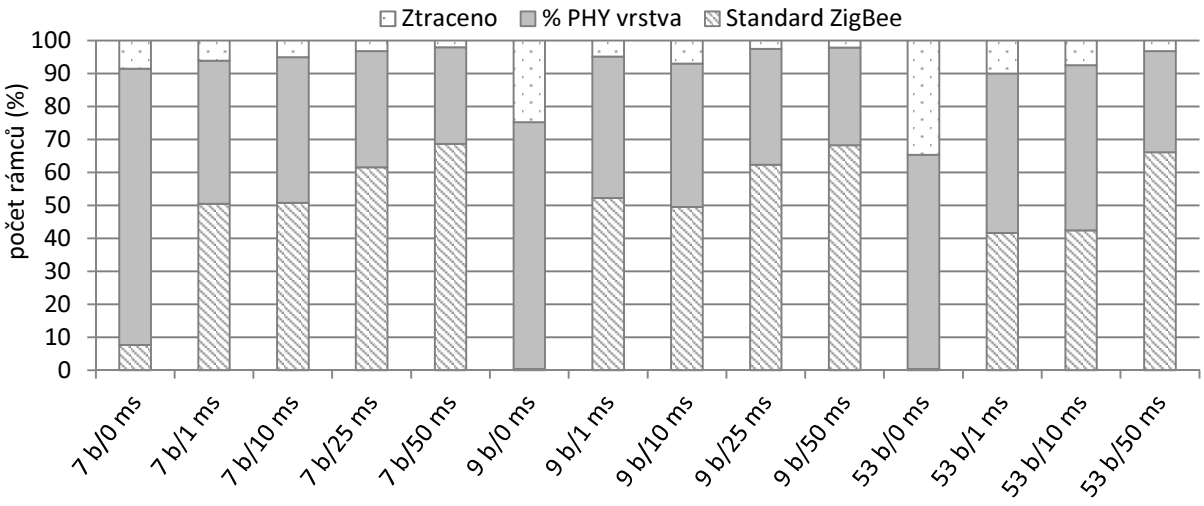
Obr. 35. Chybový histogram pro kolizní sílu interferujícího rámce 53 b s rozestupem 0 ms

Jedná se o nejsilnější variantu rušení z provedeného měření, tedy o koexistenci s přenosem „velkých“ ping rámců s kolizním potenciálem 53 b a rozestupem 0 ms. Z grafu (Obr. 35) je patrné, že těžiště histogramu je blízko průměrné hodnotě chyb (viz. Tab. 10). Maximální počet chyb v jednom rámci byl zachycen s hodnotou 304 b. Při pohledu na histogram (Obr. 35) je nicméně zřejmé, že se jedná o extrémní hodnotu. Z distribuční funkce lze vyčíst, že 90 % všech vyskytujících se chyb se vyskytuje v intervalu mezi 2 a 128 bitovými chybami v paketu. Na dalším grafu (Obr. 36) je uveden chybový histogram pro provoz na WiFi síti s ping pakety o kolizní délce 9 b, s rozestupem 0 ms.



Obr. 36. Chybový histogram pro kolizní sílu interferujícího rámce 9 b s rozestupem 0 ms

Maximální počet chyb v jednom paketu byl při tomto měření zaznamenán v počtu 150 b. Opět je zřejmé, že se jedná o extrémní hodnotu, 90 % procent chybných paketů je v intervalu 4 až 62 bitových chyb na paket. Průměrná chybovost v paketech obsahujících alespoň jednu chybu byla v tomto konkrétním měření 15,2 b. Z Tab. 10 lze dále vyčíst, že pro každou velikost generovaného ping paketu na wifi, se se zvyšujícím se rozestupem snižuje průměrný počet chyb obsažených v paketech ZigBee, a také se výrazně mění chybovost přenosu – PER. Grafické vyjádření všech variant měření je uvedeno v příloze I. této práce.

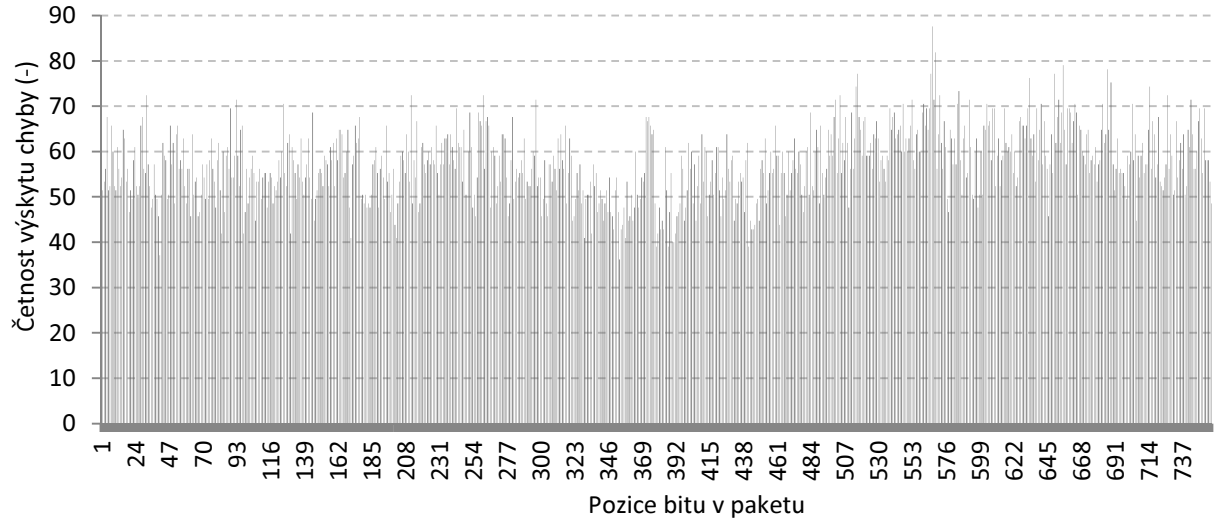


Obr. 37. Grafické vyjádření Tab. 10 v procentuálním formátu

Hodnoty uvedené v tomto měření lze vyjádřit graficky také v podobě grafu na Obr. 37. Zde je jako maximum uvedena hodnota 100 %, která odpovídá tisíci odeslaným paketům na ZigBee standardu. Pro každou měřenou koexistenční variantu graf vyjadřuje percentuální podíl mezi rámci doručenými standardně, rámci s korektní délkou, získané na MAC vrstvě a rámci ztracenými. Pro žádnou z variant měření ZigBee přenos nepřesáhl spolehlivost přenosu 69 %. Nejnižších hodnot dosahuje u těch koexistenčních variant, kde nebyl nastaven časový rozestup ping paketů na koexistenčním standardu. V těchto případech nepřesáhla spolehlivost ZigBee přenosu 8 %. I v těchto případech, bylo stále možno získávat na MAC vrstvě rámce s k případné aplikaci korekce chyb. Pro variantu měření s nejsilnějším rušením (53 b / 0 ms) bylo stále možno získat 65 % přenášených rámců, a to za okolností, kdy spolehlivost standardního přenosu nedosahuje ani 1 %. Varianty koexistence s WiFi, zatěžovanou ping pakety nižších velikostí, vykazují ještě lepší výsledky. Například pro ping pakety s kolizní délkou 7 b a nastaveným rozestupem 0 ms, je možno na MAC vrstvě získat k aplikaci dopředné korekce dat 91 % přenášených rámců. V této situaci standardní ZigBee přenos dosahuje

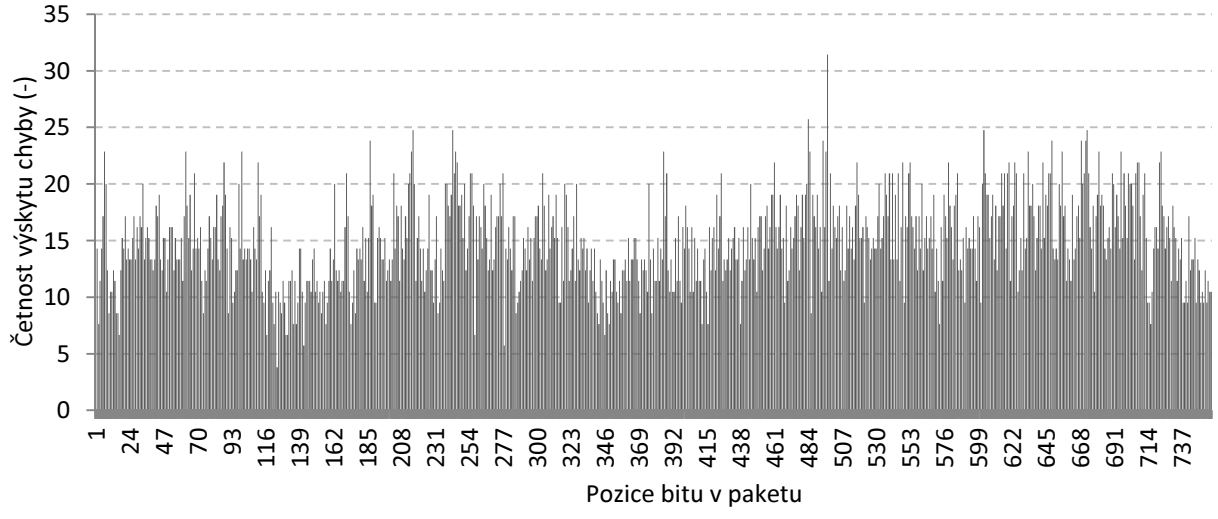
spolehlivosti cca 7 %. Z tohoto rozboru lze jednoznačně usoudit, že potenciální možnosti korekce dat jsou nezanedbatelné a bude záležet pouze na korekční síle použitého kódu, kolik z takto získaných rámců bude možno opravit.

Dalším výstupem experimentu, je ověření charakteru distribuce chyb v rozsahu datového obsahu rámce. Jak bylo popsáno v úvodu tohoto experimentu, pro každý datový rámec získaný na MAC vrstvě ZigBee přijímače, byl zaznamenáván výskyt chyb na konkrétních bitových pozicích v paketu. Tyto hodnoty byly akumulovány pro každé jedno měření (viz. Tab. 10) a výsledek je uveden v histogramu na Obr. 38.



Obr. 38. Distribuce bitových chyb pro sílu koexistujících rámců 53 b s rozestupem 0 ms

Graf ukazuje, že distribuce výskytu chyb vzhledem k jejich poloze je v tomto případě rovnoměrná. Tento fakt je důležitým východiskem pro konstrukci metody dopředné korekce chyb. Dále je uveden v tomto ohledu ještě jeden histogram, tentokrát pro koexistenční provoz s ping pakety o délce 9 b a rozestupem 0 ms (Obr. 39)



Obr. 39. Distribuce bitových chyb pro sílu koexistujících rámců 9 b s rozestupem 0 ms

Distribuce všech měřených variant lze rovněž nalézt v příloze I. této práce. Dále je uveden jeden z přijatých datových rámců, který je „prototypem“ charakteru vznikajících chyb při koexistenci se ZigBee. Na Obr. 43 je datový Payload tohoto rámce zobrazen pro přehlednost v binárním vyjádření. Odesílány byly pouze hodnoty „1“, obsažené nuly jsou tedy chybami vzniklými při přenosu. Datový rámec byl získán při měření se zátěží koexistenční sítě ping pakety kolizní délky 53 b a rozestupy 0 ms. Z rozprostření chyb je zřejmé, že se jedná o dva výrazné shluky chyb. První shluk délky 53 b (započítány i správné hodnoty mezy chybnými) a druhý o délce 19 b. Obr. 40 je uveden také pro potvrzení faktu, že ačkoli při dostatečném počtu opakování se jedná o přibližně rovnoměrné rozložení výskytu chyb, v rámci jednoho paketu se ve většině případů jedná o shluky chyb.

```
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111111
11111111111111111111111111111111111111111111111111111111111111101
11111010111010111010010011011001001100110011110011011111111111111
11111111001111111111111111111111111111111111111111111111111111111
11111111111111111111111111101110011000001000001111
```

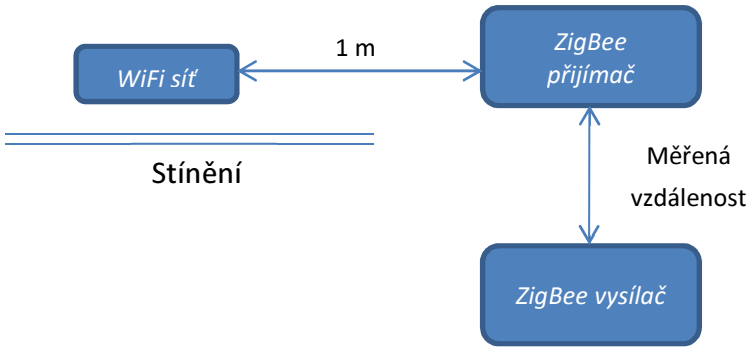
Obr. 40. Datový payload v binárním vyjádření, odesilatelem byly odesílány hodnoty 0xff

Cílem tohoto experimentu bylo ověřit praktický vliv koexistující WiFi sítě na ZigBee přenos, získání základního přehledu o chybovosti jakou je třeba eliminovat. Dále bylo experimentem ověřeno, že lze získávat datové pakety v rámci operačního systému Zstack fy. Texas Instruments na mikrokontroléru SoC CC2530 v nezanedbatelném počtu, a implementace dalšího zpracování dat má potenciál signifikantně zvýšit spolehlivost sítě. V neposlední řadě byla experimentem ověřena rovnoměrná distribuce chyb v paketech (při dostatečném počtu opakování). Tedy, ačkoli se chyby vyskytují pouze ve shlucích, tyto shluky jsou distribuovány v paketu náhodně.

5.3.3.2 **Experiment koexistence s WiFi – proměnlivá vzdálenost**

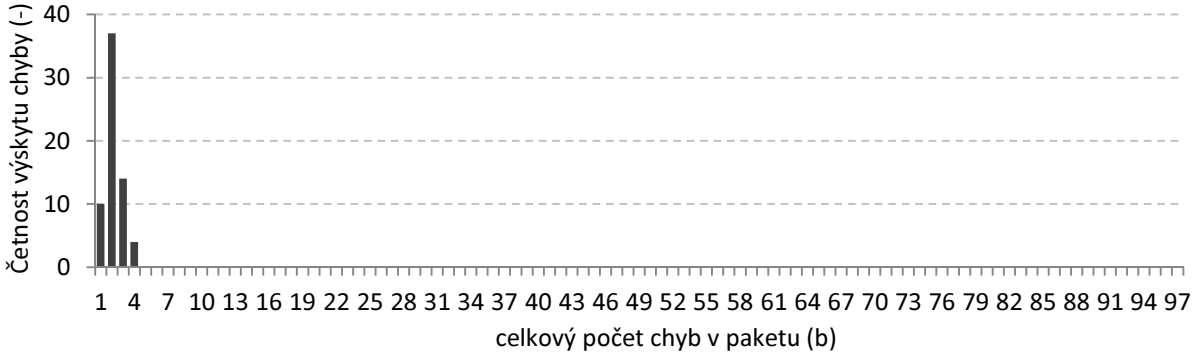
V tomto experimentu bude využito základů, popsaných v předchozím odstavci. Měření proběhlo pouze v jednom koexistenčním režimu a to s provozem ping paketů o kolizní délce 9 b a rozestupem 0 ms, přičemž byla měněna vzdálenost mezi ZigBee

přijímačem a vysílačem. ZigBee vysílač byl postupně vzdalován od přijímače, na kterém byly zaznamenávány stejné parametry příjmu jako v předchozím experimentu.



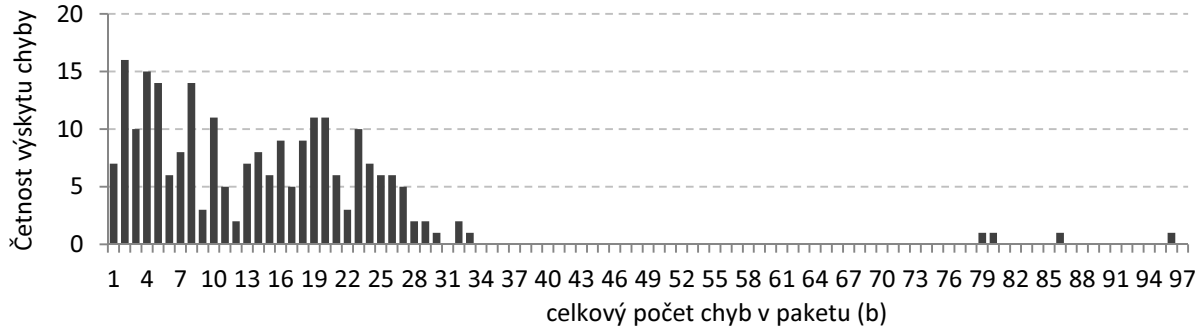
Obr. 41. Rozmístění experimentální aparatury

Výsledky měření ukazují, že ačkoli kolizní potenciál interferující WiFi sítě zůstává nezměněn, dochází ke změnám v chybovosti přijímaných rámců. Tyto vlastnosti byly v teoretickém rozboru (bod. 5.3.2) zahrnuty do parametru „pravděpodobnost chyby“ pro každý ovlivněný úsek datového přenosu. Na následujících obrázcích (Obr. 42, Obr. 43, Obr. 44) jsou uvedeny četnosti chyb v přijímaných paketech, kde se ukazuje vliv vzdálenosti jako nezanedbatelný. Vliv WiFi sítě na ZigBee přenos je na základě předešlých měření závislý nejen na provozu zatížení WiFi spoje, ale také na vzdálenosti jeho transceiverů.



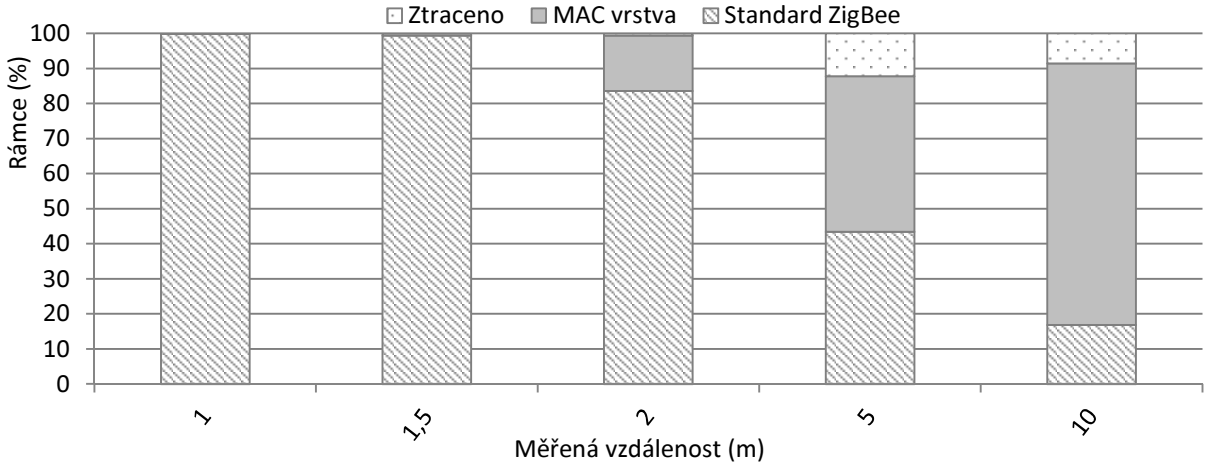
Obr. 42. Počty chyb v paketech pro kolizní sílu 7 b / rozestup 0 ms / vzdálenost 2 m

Obr. 43. Počty chyb v paketech pro kolizní sílu 7 b / rozestup 0 ms / vzdálenost 5 m



Obr. 44. Počty chyb v paketech pro kolizní sílu 7 b / rozestup 0 ms / vzdálenost 10 m

Dalším výstupem tohoto experimentu je potvrzení rovnoměrné distribuce chyb v paketech, která nemění v rámci rozsahu měřených vzdáleností své rozložení. Histogramy zaznamenané v tomto měření lze nalézt v příloze II. této práce. Dalším výstupem tohoto experimentu je vliv vzdálenosti od rušeného přijímače na spolehlivost standartního ZigBee přenosu a také na počet rámců získaných na MAC vrstvě k aplikaci dopředné korekce chyb. Tyto výsledky jsou uvedeny na následujícím obrázku (Obr. 45).



Obr. 45. Grafické vyjádření Tab. 10 měření v procentuálním formátu

Výsledky potvrzují hodnoty měřené v předchozím odstavci (viz měřená vzdálenost 10 m) a dále ilustrují vliv změn vzdálenosti na měřené parametry.

5.3.3.3 **Shrnutí experimentů s koexistencí WiFi**

Výše uvedená měření a jejich rozbor měly za cíl definovat výchozí podmínky pro hodnocení jednotlivých typů kódu a jejich vhodnosti pro účely této práce. Byly získány důležité údaje o distribuci chyb v paketech a také poznatky o množství chyb obsažených v paketech při koexistenci WiFi se ZigBee. Dále byla ověřena možnost získávání datových rámců s poškozením k následné aplikaci dopředné korekce chyb.

5.4 Rozbor a volba vhodné dopředné korekce chyb I.

Na základě provedených expermimentů bude provedeno vyhodnocení dříve popsaných kódovacích technik (viz Tab. 8). Hodnocení jednotlivých technik bude probíhat ve dvou částech. V prvním kole bude zohledněna percentuální schopnost kódů opravit soubor chybných rámců, zaznamenaných v bodě 5.3.2.3, a to v podobě vynesení opravné schopnosti kódu (v délce 756 b) na distribuční křivku zmíněného měření. Tím bude získáno procento opravitelných rámců pro konkrétní kód. Dále bude při výběru zohledněna schopnost kódu opravovat shluky chyb. Tato vlastnost bude hodnocena na základě počtu kódových slov v jednom datovém rámci. V kole druhém, budou vybrané kódy implementovány do ZigBee transceiverů CC2530 a bude hodnocena jejich efektivita z pohledu zatížení procesoru.

Část první

V následující tabulce jsou uvedeny možné varianty dříve popsaných kódovacích technik. Počet kódových slov byl volen tak, aby se co nejvíce přiblížil hodnotě 95 B, která byla přenášena při vyhodnocení výchozích podmínek, což je zároveň maximální délka payloadu pro standard 802.15.4 [1]

Tab. 11. Přehled variant implementace metod kódování

Typ Kódu	Počet kódových slov	Výsledná délka bloku (b)	Schopnost korekce chyb (b)	Schopnost korekce chyb (%)	Nutnost prokladu
Hamming (7,4)	108	756	108	92,9	Ano
BCH (31,16)	24	744	72	68,4	Ano
BCH (63,36)	12	756	60	50,2	Ano
BCH (127,64)	6	762	60	50,2	Ano
BCH(255,131)	3	765	54	41,3	Ano
BCH(1023,523)	1	1023	50	35,2	Ne
Golay (23,12)	33	759	99	88,5	Ano
RS(7,4)	36	756	36-108	23,1 - 92,2	Ano
RS(15,9)	13	780	39-156	23,9 - 97,8	Ano
RS(31,17)	5	775	35-175	18,7 - 98,3	Ano
RS(63,33)	2	756	30-180	13,1 - 98,6	Ano
RS(115,59)	1	805	28-196	11,8 - 98,9	Ne

Z Tab. 11 je zřejmé, že nejvyšší účinnost (za předpokladu shlukových chyb) mají Reed-Solomonovy kódy. Tabulka začína Hammingovým kódem (7,4), který jakkoli je v principu velmi jednoduchý, v počtu 108 kódových slov dokáže opravit 92,9 % všech délek chyb změřených ve výchozím měření. BCH kódy mají nižší účinost i v případě

dlouhých kódových slov, jako je tomu v případě BCH(1023,523), který, i když má kódové slovo delší než je určený blok dat, je schopen opravit pouze 35 % chyb z výchozího měření. Tento kód by případným zkrácením pouze ztratil na svých korekčních schopnostech [54]. BCH kódy lze tedy z výběru vyřadit. V Tab. 11 je dále uveden Golayův kód. Tento perfektní kód pro trojnásobné chyby také nedosahuje účinnosti Hammingova kódu a Reed-Solomonových kódů. Poslední varianta RS(115,59) je zkrácenou variantou RS kódu, a je upravena přesně pro použití na specifikovaném bloku. Tento kód, vzhledem k tomu, že blok zabezpečených dat může být obsažen v jednom kódovém slově, nevyžaduje zapojení prokladu. Z výsledků prvního kola lze k implementaci doporučit pouze kód Hammingův spolu s blokovým prokladačem dat a kód Reed-Solomonův s prokladačem i bez něj.

5.5 Implementace Hammingova kódu a blokového prokladače

Tato kapitola se zabývá stručným popisem a zhodnocením implementace Hammingova kódování spolu s Blokovým prokladačem dat. Blok dat pro přenos v datovém rámci byl zvolen na 95 B. Tento rozměr je odvozen od maximální dovolené délky payloadu definovaném standardem 802.15.4 [1].

5.5.1 Realizace Hammingova kódu

Algoritmus zakódování definuje vstupní a výstupní vyrovnávací paměť v délce zvoleného bloku dat. Hlavní cyklus programu *„code_buf ();“* postupně z vyrovnávací paměti čte binární hodnoty po čtveřicích bitů, které v podobě argumentů předává funkci *„hamming_bin_buf (b1,b2,b3,b4);“*. Výstupem této funkce je zpět do hlavního cyklu vrácena sedmice bitů Hammingova kódu. Tato sedmice je následně hlavním cyklem ukládána do výstupní vyrovnávací paměti. Pozice bitu, od které je uložena každá další zakódovaná sedmice, je generována funkcí *„Generate_sequential_position ();“.* Tato funkce plní úlohu duálního kurzoru, kdy po každých osmi posunech bitového kurzoru dochází k inkrementaci byte kurzoru. Tento jednoduchý postup umožňuje vynechat načtení bitových proměných do bytových registrů mikrokontroléru. Po načtení celé vstupní vyrovnávací paměti je algoritmus kódování bloku dat ukončen. Rutina samotného Hammingova kódu je uvedena na Obr. 46. Celá rutina programu je umístěna v příloze III. této práce.

```
//*********************************************************************
void hamming_buf (uint8 bit1, uint8 bit2, uint8 bit3, uint8 bit4){
uint8 i, j, coded[7], data[4];

data[0]=bit1; data[1]=bit2; data[2]=bit3; data[3]=bit4;

    for(i=0;i<7;i++){
       coded[i]=0;
       for(j=0;j<4;j++){coded[i]+= (data[j] * gmatrix[j][i]);}
       if(1==coded[i]%2){Set_Bit_buf(byte_cursor, bit_cursor);}
       Generate_sequential_position ();
    }
}
//*********************************************************************
```

Obr. 46. Rutina „hamming_buf();“

Algoritmus dekódování *„decode_buf ();“* je analogický, pomocí dříve definované funkce *„Generate_sequential_position ();“*, jsou ze vstupní vyrovnávací paměti načítány binární hodnoty po sedmicích bitů, které jsou jako argumenty předávány funkci *„deHamming_bin_buf (b1,b2,b3,b4,b5,b6,b7);“.* Výstupem této funkce jsou původní čtyři informační bity kódového slova, které hlavní dekódovací cyklus ukládá do výstupní vyrovnávací paměti. Po dosažení horní hranice vstupní vyrovnávací paměti je proces dekódování bloku dat ukončen. Rutina samotného Hammingova dekódování je uvedena na Obr. 47, celá rutina je umístěna v příloze III.

```
//*********************************************************************
void dehamming_buf (void){
uint8 i, j, syndrom[3];

    for(i=0;i<3;i++){
      syndrom[i]=0;
        for(j=0;j<7;j++) {syndrom[i]+=(H_data[j]*hmatrix[i][j]);}
      syndrom[i]=syndrom[i]%2;
    }
    for(j=0;j<7;j++){
      if((syndrom[0]==hmatrix[0][j]) && (syndrom[1]==hmatrix[1][j]) &&
    }
    if(j != 7){H_data[j]=!H_data[j];
    }
}
//*********************************************************************
```

Obr. 47. Rutina „dehamming_buf();“

5.5.2 Realizace Blokového prokladače dat

Další součástí kódovací metody s Hammingovým kódem je blokový prokladač dat. Tento algoritmus je realizován základní funkcí *„interleave_buf();”*, která prochází vstupní vyrovnávací paměť po jednotlivých bitech a zapisuje je do výstupní paměti. Sekvence nových proložených pozic je generována funkcí *„Generate_interleaved_position ();“*, která podobně jako v předchozích případech umožňuje šetřit operační paměť mikroprocesoru. Tím, že prokládácí sekvence je generována průběžně, není nutné zabrat

paměť pro prokládací matici. Algoritmus umožňuje definovat hloubku programu z vytvořeného menu programu.

```
//*********************************************************************
uint8 Generate_interleaved_position (){
uint8 tmp=0;

  byte_cursor = bit_cursor >> 3;
  tmp = bit_cursor %8;

  if(bit_cursor==755){bit_cursor = 0; byte_cursor = 0; line_cnt = 1;}

  bit_cursor = bit_cursor + intlvr_size;
  if(bit_cursor>755){bit_cursor=line_cnt++;}

  return(tmp);
}
//*********************************************************************
```

Obr. 48. Rutina „generate_interleaved_position();“

Na Obr. 48 je uvedena stěžejní rutina, generující prokládané pozice, celou rutinu lze opět nalézt v příloze č. III. Algoritmus zpětného prokládání je prakticky totožný a rutinu lze najít v příloze této práce.

5.5.1 Realizace Reed-Solomonova kódu

Algoritmus kódování a dekódování Reed-Solomonova kódu vyžaduje komplexní a rozsáhlou část kódu. Tento kód byl převzat od autora S. Rockliffa [55], jedná se o elegantně napsanou rutinu, pracující pro definovatelné délky kódových slov. Kódování probíhá v systematické formě. Při dekódování je využit iterativní Berlekamp-Massey algoritmus [45]. Implementace zahrnula iniciaci nutných proměných v paměti mikroprocesoru při jeho spuštění. To znamená spuštění funkce *„generate_gf();“*, která vygeneruje do paměti procesoru konkrétní Galoisovo pole, definované parametry *m, n, k, t* a příslušným polynomem *p*. Následně je iniciace RS kodéru dokončena spuštěním funkce *„gen_poly();“*, tvořící příslušný generující polynom. Vytvoření kódového slova probíhá zapsáním vstupních dat do paměťového pole *„data[i]”* a následným spuštěním funkce *„encode_rs();“*, jejíž výstup je v podobě sekvence paritních bitů uložen v paměťovém poli *„bb[i]”*. Dekódování je realizováno opět naplněním paměťového pole „recd[i]” a spuštění funkce *„decode_rs();“*, jejímž výstupem je opravené kódové slovo v tomtéž paměťovém poli.

Problematika implementace Reed-Solomonova kódováni v softwarové podobě je rozebírána velmi často (např. [56]) především v souvislosti s jeho výpočetní a paměťovou náročností. Výpočetní výkon počítačových procesorů a s nimi i mikrokontrolérů každý rok stoupá, nicméně pro SoC CC2530 v kombinaci se Z-Stack

systémem se zpracování Reed-Solomonova kódu stává velmi obtížně realizovatelný. Maximální paměťově únosná varianta je dekódování po blocích *n=15*. Každá další varianta RS kódu s delšími kódovými slovy je mimo paměťové možnosti mikrokontroléru CC2530 se Z-Stack.

5.6 Rozbor a volba vhodné dopředné korekce chyb II.

Tato kapitola navazuje na první část výběru vhodné kódovací techniky. V tomto bodě bude vyhodnocena náročnost zpracování obou kódovacích technik implementovaných na experimentálním hardware.

Metodika měření spočívá v reálném spuštění zmíněných funkcí v sestavách, uvedených v následující výsledkové tabulce Tab. 12. Měření proběhlo na popsaném experimentálním ZigBee hardwaru, který byl doplněn o digitální ruční osciloskop DSO DS203. Aplikace v mikrokontroléru byla tvořena vždy danou kódovací technikou a testovací funkcí. Testovací funkce, jejíž spouštění bylo inicováno tlačítkem, nejprve připravila vstupní náhodná data do vstupních vyrovnávacích pamětí a následně před spuštěním konkrétní kódovací techniky změnila hodnotu na výstupu LED1. Po dokončení měřených částí kódování opět změnila hodnotu na hardwarovém portu miktrokontroléru (LED1). K tomuto portu byl také připojen ruční osciloskop DSO DS203, kterým byla změřena časová potřeba dané techniky. Vzhledem k faktu, že kontrolu nad porty mikrokontroléru přebírá v plném rozsahu Z-Stack, nejprve bylo změřeno, jaká prodleva předchází od spuštění funkce ovládání portu k jeho změně stavu. Vzhledem k rozsahu softwaru Z-Stack by bylo obtížné dojít k prodlevě spočtením kroků mikrokontroléru. Prodleva od spuštění funkce po změnu stavu je jak pro funkci *„HalLedSet (HAL_LED_1, HAL_LED_MODE_ON);“* tak *„HalLedSet (HAL_LED_1, HAL_LED_MODE_OFF);“* rovna 33 µs. Tuto prodlevu lze z pohledu délky měřených úseků chápat jako zanedbatelnou, neboť zpracování kódovacích rutin bude přibližně o jeden řád delší.

Část druhá

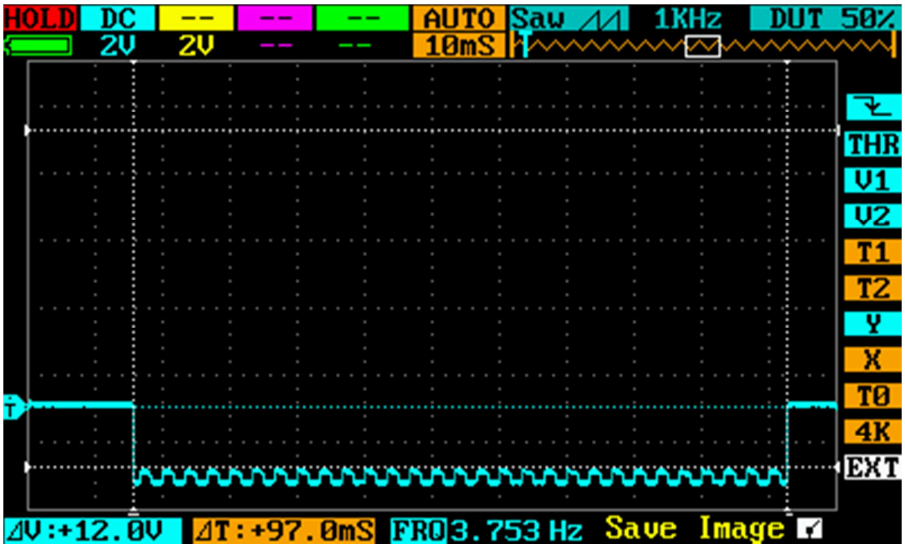
V tabulce Tab. 12 je uveden přehled výsledků měření jednotlivých sekvencí. Větší část časového úseku ve všech čtyřech případech zabrá binární zpracování vstupní vyrovnávací paměti. Operace v jazyce C na binární úrovni spočívají v maskování osmibitových registrů, což značně množí kód potřebný k relativně snadným operacím. Celkově lze říci, že složitost přípravy vstupního paměťového pole je hlavním problémem

implementace bez ohledu na kódovací techniku. V případě RS kódování bylo měřeno zpracování dekódovací části na datech s chybami. Uvedený čas je tedy maximálním zpožděním. Na Obr. 49 je pro ilustraci uveden printscreen obrazovky ručního osciloskopu DS203 při měření zpoždění dekódováním bloku dat kódem RS(15,9).

Tab. 12. Přehled výsledků měření doby zpracování v SoC CC2530

Typ Kódu	Doba zakódování bloku (ms)	Doba dekódování bloku (ms)	kódování s prokladem (ms)	dekódování s prokladem (ms)
Hamming (7,4)	31	38	49	56
RS(7,4)	37	78	55	96
RS(15,9)	42	97	60	115
Prokladač	18	18		

Z tabulky lze vyčíst, že nejmenší zpoždění má Hammingovo kódování jak při zakódování bloku dat, tak při dekódování. Vzhledem k tomu, že kódová slova implementovatelných RS kódů jsou příliš krátká, než aby bylo možno je užít bez prokladu (na základě rozboru v bodě 5.3.3.3), a zároveň s ohledem na parametry zadání, které jasně určují jako prioritu zachování nízké energetické náročnosti, lze s jistotou zvolit jako nejvhodnější právě kódování Hammingovo s použitím Blokového prokladače dat. Z pohledu korekčních schopností Hammingův kód v tomto bloku dat dokáže opravit 108 bitových chyb, což je stejný počet jako je maximální účinnost druhé nejrychlejší varianty RS(7,4). Použitím této kombinace dojde k maximálnímu zpoždění při odesílání dat 49ms respektive 56 ms při jejich přijímání na každý datový blok.



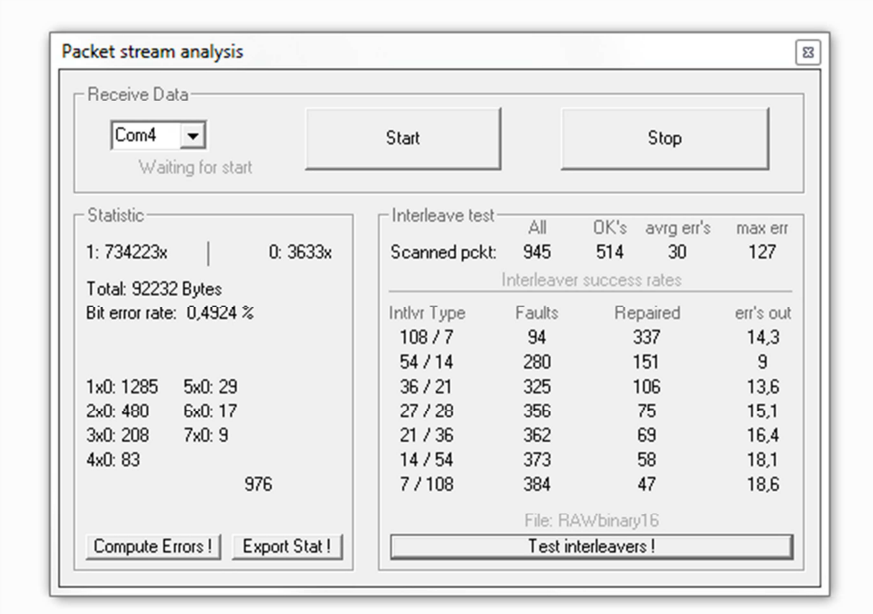
Obr. 49. Printscreen obrazovky ručního multimetru DS203 při měření doby dekódování RS(15,9)

5.7 Výběr vhodného rozměru blokového prokladače dat

Vhodná kódovací technika byla již zvolena, nyní je třeba určit, jaká hloubka prokladu bude nejvhodnější. Jak bylo uvedeno v teoretickém rozboru (viz 4.9), hloubka prokladu determinuje maximální délku shluku chyb, který lze distribuovat do paketu jako jednoduché chyby s rozestupem hloubky prokladu [49]. Jednoduchou dedukcí lze dojít i k závěru, že při dosažení maximální délky chyby takový prokládací mechanismus již jakkoli umístěnou další chybu uloží do prostoru, kde vzdálenost k dalšímu bitu bude menší než hloubka prokladu. Není tedy schopen, vyjma jednoho maximálního shluku chyb, distribuovat žádnou další chybu. V případě, že snížíme hloubku prokladu, bude sice prokladač schopen distribuovat pouze kratší maximální shluky chyb, ale zvýší se jejich počet, který je možno distribuovat. Jaký prokladač je nejvhodnější pro účely distribuce chyb, vzniklých při koexistenci s WiFi, respektive při rušení bílým šumem, bude určeno v dalším textu.

Pro následující experiment byl rozšířen testovací software, zaznamenávající data při měření v předchozích bodech. Software vytvořený v prostředí VisualBasic 6.0 umožňuje použití všech variant prokládací matice. To znamená, že pro každý zaznamenaný datový rámec provede nejprve proložení s určeným rozměrem matice a následně dekódování Hammingovým algorimem. Vzhledem k tomu, že při měření byly odesílány stejné hodnoty, lze takovou operaci provést bez ohledu na zpracování dat na straně odesilatele. Kódové slovo Hammingova kódu odpovídající informačním bitům „1111“ je „1111111“. Tedy výsledkem aplikace konkrétního prokladu a dekodéru musí pro úspěšnou opravu být celá sekvence rovna „1“. Takto vyhodnocený rámec je započten jako „Repaired“. V opačném případě je započten do hodnoty „Faults“. Tyto hodnoty jsou započítávány zvlášť pro každou variantu rozměru prokládací matice. Na Obr. 50 je uvedeno uživatelské prostředí programu, jeho zdrojový kód je umístěn v příloze IV práce. K testování jsou vybrány rozměry matice:

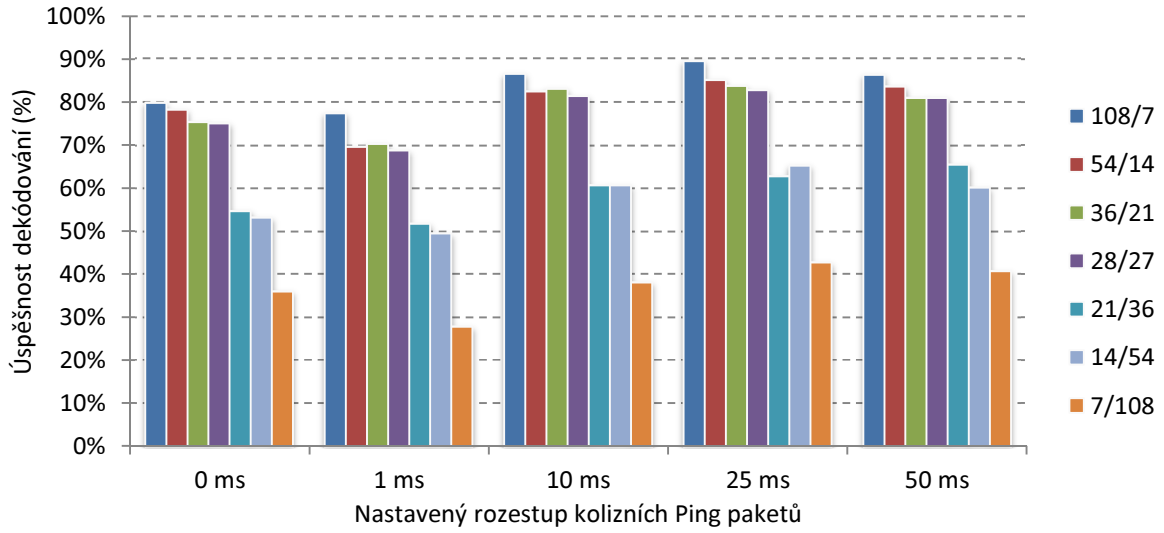
- *108 * 7*
- *54 * 14*
- *36 * 21*
- *27 * 28*
- *21 * 36*
- *14 * 54*
- *7 * 108*



Obr. 50. Uživatelské prostředí programu pro vyhodnocení účinnosti prokládacích matic

5.7.1 Volba blokového prokladače dat – koexistence 802.11g

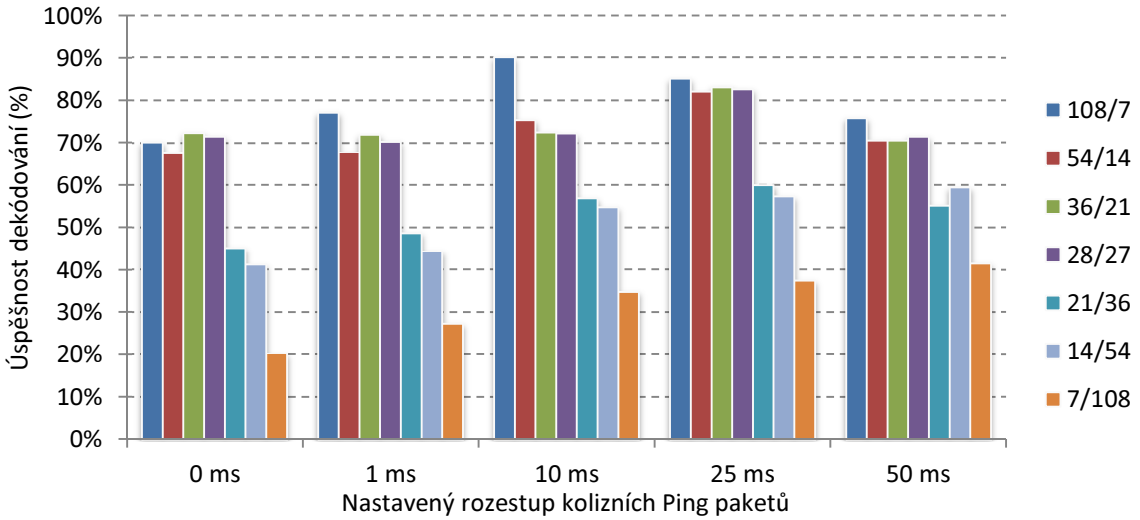
Následující výsledky vznikly uvedeným zpracováním zaznamenaných datových rámců v bodě 5.3.3. Na Obr. 51 je graf první varianty koexistence s WiFi 802.11g, kdy síť WiFi byla zatěžována „ping pakety“ o kolizní délce 7 b a souvisejícím provozem. Z výsledků vyplývá, že účinnost prvních čtyř variant rozměru prokládací matice je přibližně srovnatelný. S hloubkou prokladu 21 a níže pak účinnost klesá.



Obr. 51. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 7 b

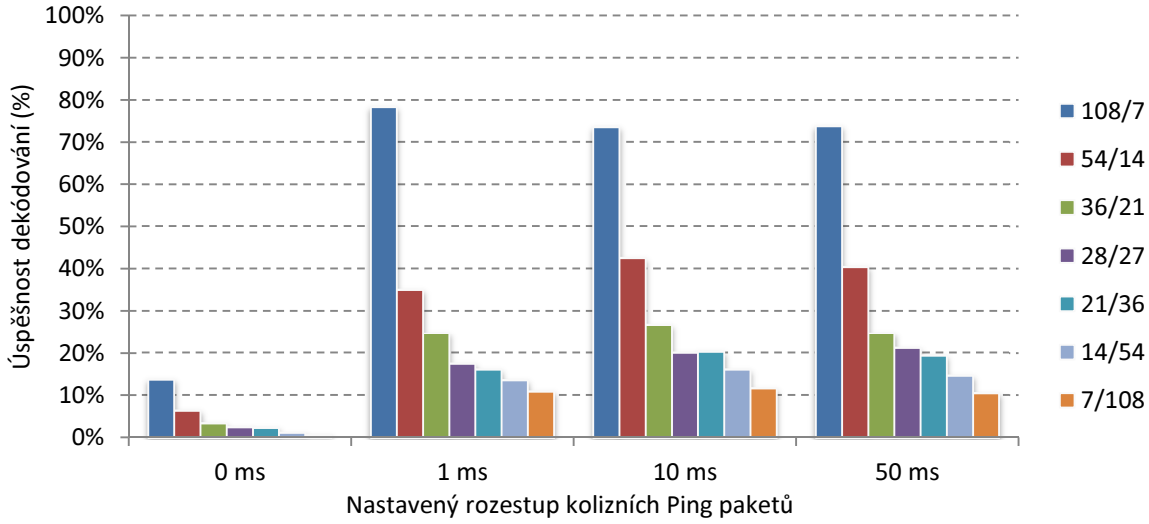
Z grafu lze zároveň vyčíst také první praktické výsledky kódovací metody, kdy při daných rozestupech ping paketů projevuje dekódovací proces při variantě prokladu 108/7 téměř 90 % úspěšnost oprav chyb. Na dalším obrázku (Obr. 52) je uveden graf

pro kolizní sílu ping paketů na WiFi síti 9 b. Rozdíl oproti předchozímu stavu není významný, stejně rozdíl v kolizní délce. Nicméně, lze pozorovat zvýšený odstup v účinnosti prokládací matice s hloubkou prokladu 108.



Obr. 52. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 9 b

Na dalším grafu (Obr. 53) již lze zaznamenat nejen celkový pokles účinnosti všech rozměrů prokladu, ale také největší rozdíl v odstupu prokládací matice 108/7. V souladu s teoretickými předpoklady lze tedy pro podmínky koexistence s 802.11g zvolit prokládací matici rozměru 108/7.

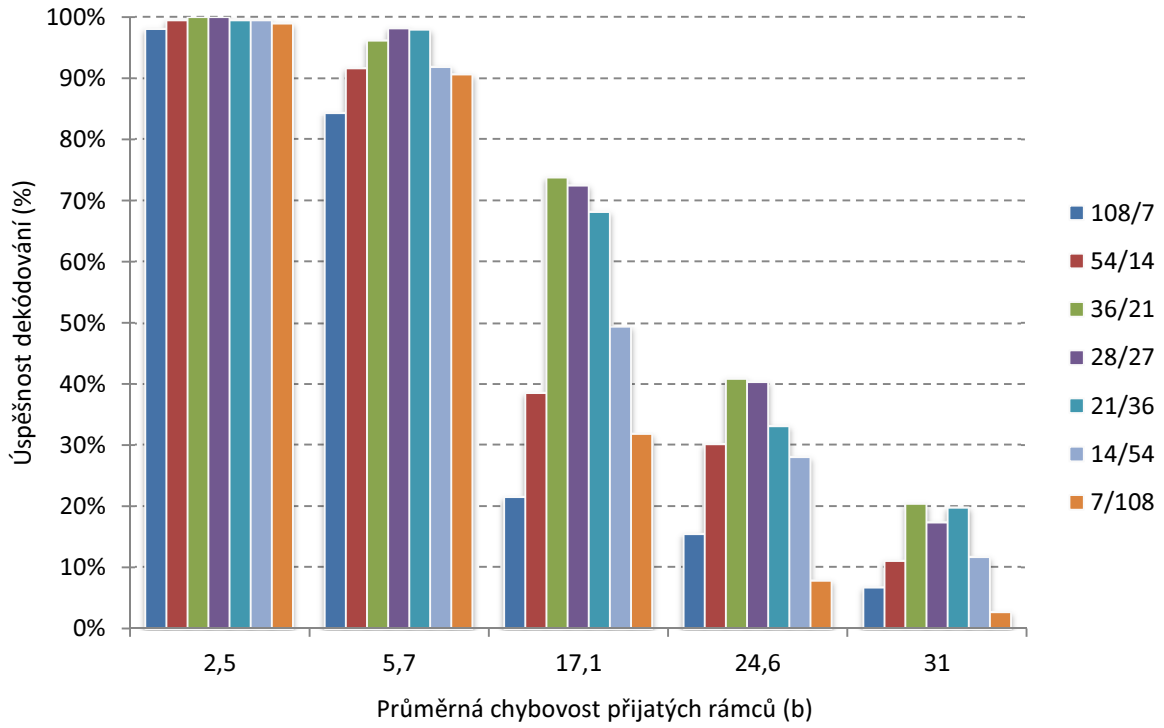


Obr. 53. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 53 b

5.7.2 Volba blokového prokladače dat – širokospektrální rušení

Při tomto experimentu byla data pro analyzu získána novým měřením. Jako interferující element byl použit komutátorový točivý stroj 12V. Tento točivý stroj byl

použit bez odrušujících kondenzátorů a umístěn byl do vzdálenosti 10 cm od antény přijímače. Aplikační software v obou ZigBee transceiverech CC2530 byl upraven totožně s měřením v bodě 5.3.3. Přijímané rámce byly zaznamenávány pro pět různých vzdáleností mezi vysílačem a přijímačem. Každá takto měřená vzdálenost byla vyhodnocena z pohledu průměru počtu chyb v rámcích s alespoň jednou chybou. Výsledky analýzy takto získaných dat jsou uvedeny v následujícím grafu na Obr. 54.



Obr. 54. Vliv prokladače na dekódování Hammingova kódu při rušení komutátorovým točivým strojem

Oproti předchozím výsledkům koexistence s WiFi, které byly ovlivněny výskytem převážně shluků chyb, je v tomto případě rozložení účinnosti jiné. Zatímco při nízkých průměrných chybovostech je účinnost všech rozměrů přibližně stejná, s přibývajícím počtem chyb v rámcích je zřejmý rapidní pokles účinnosti matic prokladu s největší a nejnižší hloubkou – 108 a 7. Nejlepších výsledků dosahují prokládací matice blízko souměrného rozměru – 28/27. V tomto ohledu jsou výsledky experimentu v rozporu s výsledky předchozího, neboť prokládací matice 108/7 jeví při širokospektrálním rušení značné nedostatky.

5.7.3 Volba blokového prokladače dat – shrnutí

V předchozích dvou odstavcích byla ve stručnosti rozebrána problematika rozměru prokládací matice při opravách chyb, vzniklých při přenosu. Dva rozdílné požadavky, na opravu dlouhých shluků chyb a zároveň většího počtu menších shluků, či jednoduchých chyb, nejsou technicky slučitelné. Řešení lze hledat pouze v použití jedné z variant a její volbu dle konkrétních podmínek. Tuto volbu by bylo možno realizovat jak staticky, tedy pevným přednastavením dle podmínek zadavatele, tak také dynamicky, kdy by aplikace v transceiveru rozhodla na základě okamžité analýzy chyb jaký proklad zvolt při dalším přenosu.

V dalším textu tedy budou uvažovány dvě varianty prokladu 108/7 pro koexistenční měření a 36/21, pro měření bez koexistence dalších sítí. Tabulky s vyčíslením výsledků měření jsou uvedeny v příloze V.

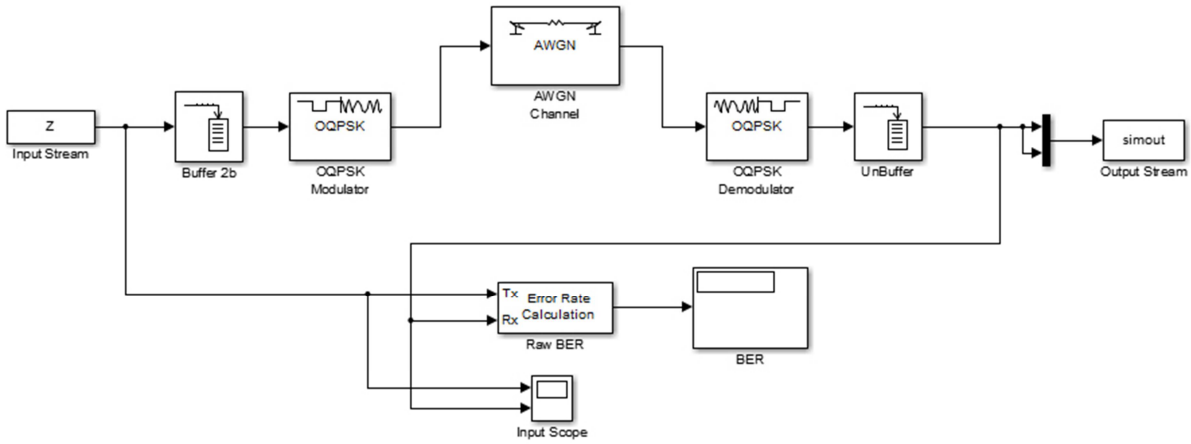
5.8 Simulace ZigBee přenosu

Tento odstavec se týká problematiky modelování bezdrátových spojů. Na základě rozboru, uvedeném v tezích této práce, byl zvolen k vytvoření modelu software MATLAB. Tento software v nadstavbě Simulink umožňuje velmi jednoduché sestavení modelu sítě z knihovný bloků. Tyto bloky lze spojovat podle funkčních požadavků. Simulace vytvořeného modelu následně probíhá spojitě nebo diskrétně. Problematika bezdrátových sítí z pohledu bitové chybovosti je v Simulinku velmi dobře realizovatelná. Zpracování této bitové chybovsti v rámci bloků a jejich částečně odlišné zpracování již v simulinku naráží na překážky a stává se těžko realizovatelným. Protože tato část modelu je již čistým zpracováním bitové chybovosti, byla tato část modelu vytvořena v programovacím jazyce VisualBasic. Tento programovací jazyk umožní zrychlit tvorbu modelu a zvýšit jeho přesnost díky schopnosti přesně napodobit procesy, probíhající v ZigBee stanscieverech. V dalších odstavcích budou popsány dvě části modelu, jejich vzájemné propojení a také výsledky simulací.

5.8.1 Model I. část

Jak bylo uvedeno, tato část modelu bude zpracována v simulačním prostředí MATLAB Simulink, využito bude především jeho knihoven RF Blockset pro sestavení bezdrátové části. Tato část modelu bude zpracovávat bitový datový tok generovaný druhou částí

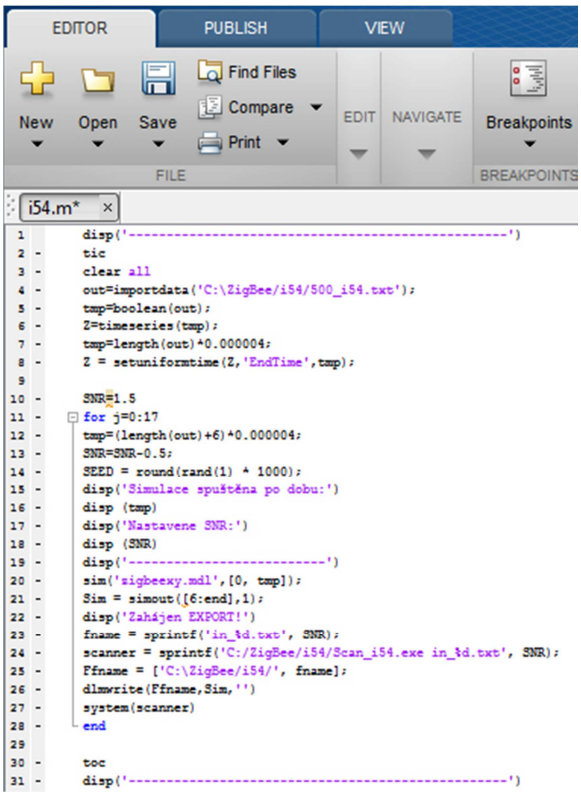
modelu. Tento model je zpracován analogicky k bezdrátové části 802.15.4 [1] standardu. Na následujícím obrázku (Obr. 55) je uvedeno schéma simulačního modelu RF části modelu. Vlevo vchází bitový datový tok do modelu, odkud je veden do bloku „Buffer 2b“, kde jsou data sestavena pro vstup do modulačního bloku OQPSK po dvojicích bitů. Tyto bitové dvojice jsou následně zavedeny do bloku „OQPSK modulátor“, kde je signál modulován čtyřstavovou modulací. Tyto dvě části představují RF část odesílatele signálu. Tento signál je dále zaveden do bloku AWGN, kde je přidán bílý Gaussovský šum. Parametr SNR (odstup signál šum) je v bloku řízen proměnnou, která je pro každou simulaci měněna skriptem (Obr. 56).



Obr. 55. Model RF části v MATLAB Simulink

Po průchodu šumovým kanálem jsou data zavedena do bloků simulující RF část přijímače, která je tvořena blokem „OQPSK demodulator“ a „UnBuffer“. Demodulátor zpětně demoduluje RF signál na dvojice bitů, které jsou následně opět převedeny na bitový tok blokem „UnBuffer“. Tento výstupní signál je opět zaveden zpět do WorkSpace odkud je exportován k dalšímu zpracování.

Data vstupující do MATLAB Workspace jsou přejímána z druhé části modelu vyšších vrstev pomocí skriptu, který data konvertuje a upravuje jejich výstupní část. Skript nejprve načte datový tok definované délky, následně jej upraví do datové podoby „TimeSeries“ a nastaví časový krok tak, aby odpovídal době symbolu na RF části. Následně je vypočtena adekvátní délka simulace, odpovídající délce bitového toku. V této fázi je již cyklicky spouštěna simulace s různým nastavením parametru SNR, kdy pro každou proběhlou simulaci jsou výstupní data analogickým způsobem transformována do podoby na vstupu. Následně je skriptem spouštěn program druhé části modelu, který automaticky data zpracuje a výsledky uloží do výsledkového souboru. Skript, spojující obě části modelu, je uveden na dalším obrázku (Obr. 56).



Obr. 56. Skript pro zpracování dat mezi částmi modelu

5.8.2 Model II. Část

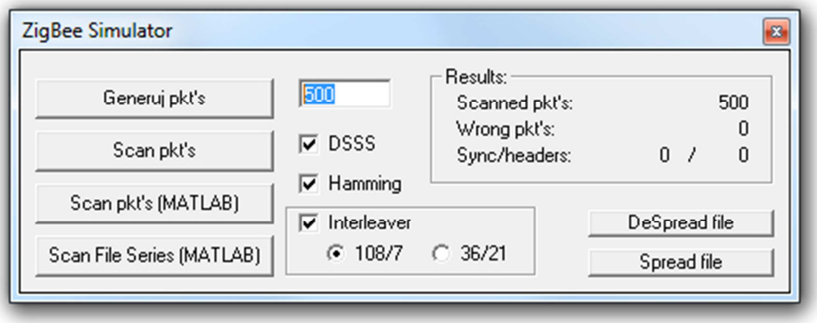
Tato část modelu generuje a zpracovává datový tok pro MATLAB Simulink. Je vytvořena v programovacím prostředí VisualBasic Studio 6.0 a je poměrně složitější než předchozí část. Jak bylo naznačeno, program v první části simulace generuje data pro zpracování RF částí modelu. Tato data jsou přesnými replikami datových rámců používaných standardem 802.15.4. Ná základě popisu v úvodu této práce (2.1.1.3), program nejprve vytvoří základní hlavičky paketů, tedy SHR, MHR a následně také hlavičky vyšších vrstev NWK a APS. Hlavičky MHR, NWK a APS jsou tvořeny hodnotami 0xAA, které byly zvoleny pro vyvážený poměr nul a jedniček. Hlavička SHR je generována přesně v souladu se standardem 802.15.4 (viz 2.1.1.3), kde je synchronizační sekvence tvořena 8 hodnotami 0x00 a následně frame delimiterem (0xE5) a délkou paketu (0x66). Tato sekvence hlaviček je doplněna o payload, kde existuje několik variant jejich generování. V závislosti na volbách z uživatelského rozhraní je payload datového rámce tvořen buď pouze náhodnými čísly doplněnými o CRC CCITT součet. Druhá varianta generuje kratší payload, který je zakódován Hammingovým kódem (7,4) výpočtem paritních bitů podle rovnic (17)(18)(19) a následně proložen blokovým prokladačem s prokládací maticí 36x21 nebo 108x7. Takto vytvořená struktura datového rámce je v souladu s normou modulována DSSS

modulací v souladu s Tab. 2. Data jsou tedy dělena na čtveřice, které jsou substiuovány za sekvence, popsané v normě 802.15.4 [1]. Takto vytvořený bitový tok je uložen do souboru, který je importován do prostředí MATLABu skriptem na Obr. 56.

Po ukončení sumace v MATLAB Simulink a exportu vzniklých dat jsou tyto opět načteny touto částí modelu. V této části je postupně načítán datový tok ze souboru, kdy je nejprve provedena demodulace DSSS a to výpočtem Hammingovy vzdálenosti podle (15) pro všechna možná kódová slova z Tab. 2, na základě které je zvolen nejpravděpodobnější původní čtyřbitový symbol. Po dokončení této demodulační sekvence je započata kontrola hlaviček v rámcích. Jsou zaznámenávány následující hodnoty:

- Poškozená Synchronizační hlavička
- Poškozená hlavička MAC vrstvy
- Poškození některé z hlaviček vyšších vrstev
- Poškození v payload v několika variantách dekódování

Poslední krok rozpoznávání chyb je opět definován z uživatelského rozhraní a spočívá v analogickém zpětném proložení dat blokovým prokladačem zvoleného rozměru matice a následně dekódování Hammingova kódu výpočtem syndromů podle rovnic (20)(21)(22) a jejich vyhodnocením. Následně je pro každý takový dekódovaný rámec spočten CRC výpočet a porovnán s uloženou hodnotou. Jsou-li tyto hodnoty rovny, není do počtu chybných rámců započten, v opačném případě je inkrementována suma chybně doručených payloadů. Nebyla-li zaznamenána v žádné z kontrolovaných oblastí chyba, je rámec započten do bezchybně doručených. Takto vzniklý soubor počtů vzniklých chyb je průběžně ukládán do „.csv“ souboru pro zpracování v tabulkovém editoru. Na následujícím obrázku je uživatelské prostředí popsaného programu. Samotný zdrojový kód je v příloze práce.

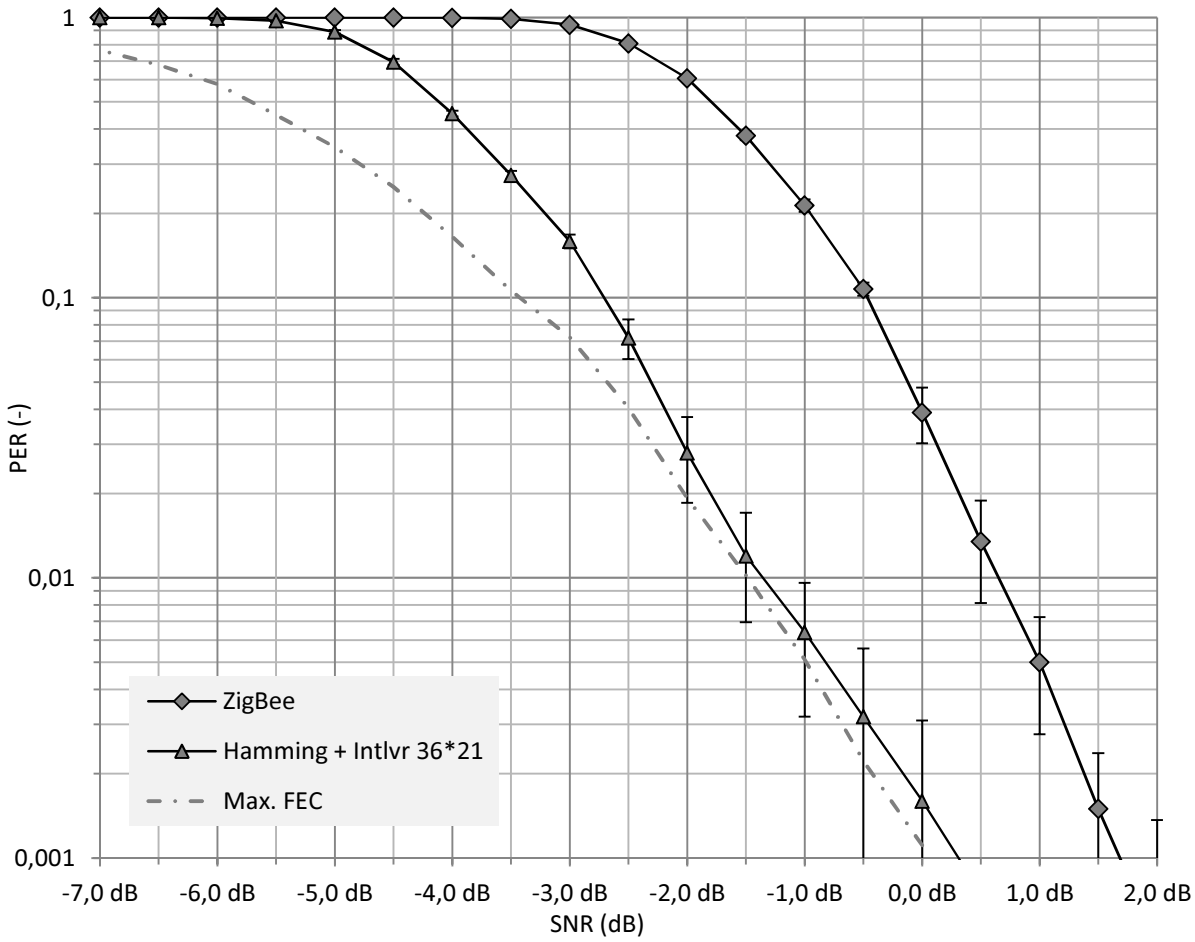


Obr. 57. Uživatelské prostředí programu pro generování ZigBee rámců

5.9 Výsledky simulace ZigBee přenosu

Tato část textu se věnuje výslekům simulací. Jak bylo popsáno v předchozím textu, simulace probíhala cyklicky pro různé podmínky. Pro zajímavost lze zmínit, že jedna simulace přenosu s 15 simulovanými hodnotami SNR v pěti opakováních zabrala více než 1 GB prostoru na pevném disku a doba zpracování je cca 4-5 h na výkonném PC. Simulace následujících výsledků zabrala zhruba 48 h simulačního času. Tyto hodnoty uvádím nikoli jako důkaz složitosti programu, ale jako potvrzení faktu, zmíněného v části, věnující se implementaci kódovací metody do procesoru 8051, a to že zpracování dat v binární podobě je ve vyšších jazycích poměrně neobratné a těžkopádné.

Simulace proběhla pro tři různé varianty nastavení. Pro nekódovaný přenos, kódovaný přenos s prokladem hloubky 36 a prokladem hloubky 108. Každý ze simulovaných bodů byl spouštěn s jinou hodnotou „initial seed“ v AWGN kanálu, aby bylo zabráněno opakujícím se výsledkům. Každý jeden bod byl simulován 5x, z těchto hodnot byl vypočten průměr a směrodatná odchylka. Výstupem modelu jsou počty chyb v jednotlivých sekcích rámců, které byly dále vyhodnoceny do formy chybovosti rámců PER. Pro výpočet PER standardního ZigBee přenosu byl za „F_r“ do rovnice (8) dosazen počet rámců doručených bez všech chyb. Za „F_s“ byl vždy dosazen počet rámcu použitý při simulaci 500. Dále byla v simulaci zjištěna maximální hranice, ke které se lze dopřednou korekcí chyb přiblížit. Tyto hodnoty byly vypočteny dosazením počtu všech rámců s korektní synchronizační hlavičkou a délkou rámce za „F_r“. Na všechny tyto rámce lze v přijímači následně aplikovat metodu dopředné korekce chyb. Třetím výstupem simulace je průběh chybovosti rámců s aplikací popsané dopředné korekce chyb, v tomto případě (pro AWGN) s prokládací maticí o rozměru zvoleném v předcházejícím textu (36x21). Do kladně dekódovaných rámců jsou započítávány pouze rámce, které mají zároveň nepoškozenu synchronizační hlavičku. Výsledek je uveden na následujícím grafu (Obr. 58).



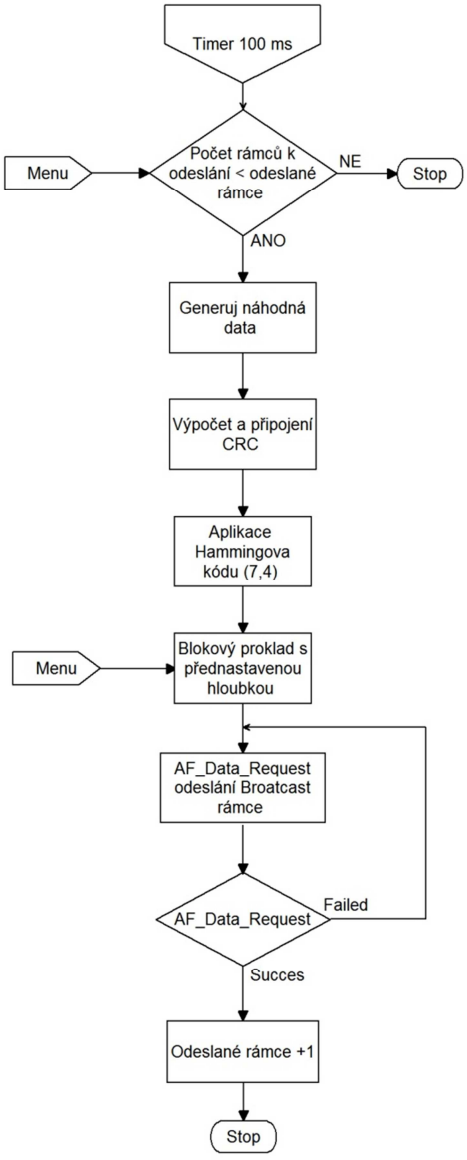
Obr. 58. Výsledek simulace ZigBee přenosu

V grafu je vyznačen maximální zisk metody (Max. FEC) čerchovanou čarou. Dále je v grafu uvedena chybovost navržené metody (trojúhelníky). Z grafu lze vyčíst, že účinnost metody je nejvyšší při nízkých bitových chybovostech odpovídajících cca PER 10 %. S klesajícím odstupem signal-šum klesá dále také počet opravitelných rámců a účinnost navržené metody se od hranice maxima vzdaluje. Při chybovosti PER 0,01 je zisk navržené metody v simulaci 1,92 dB. Při chybovosti PER 0,1 pak zisk odpovídá 2,21 dB.

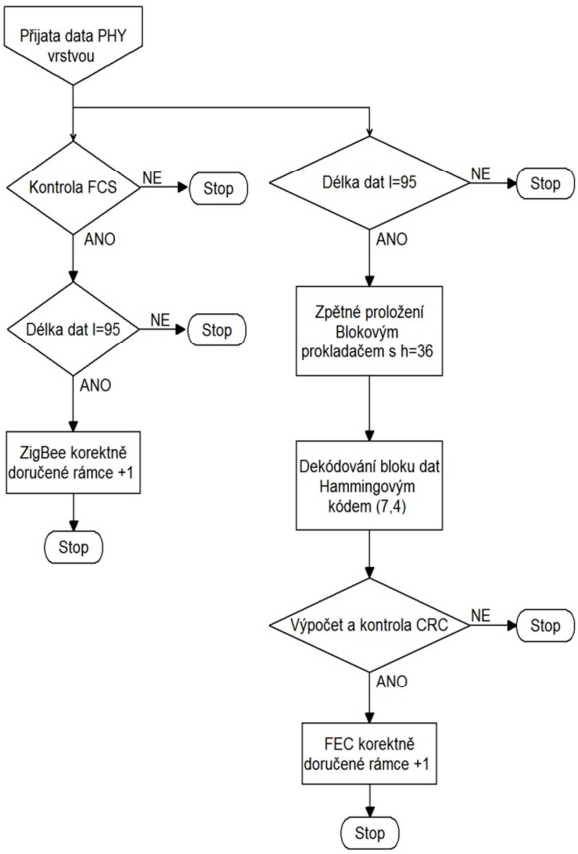
5.10 Měření účinků metody při širokospektrálním rušení

Tento odstavec se zabývá zavěrečným zhodnocením účinků metody pro dopřednou korekci chyb pro ZigBee, navrženou v této práci. Obsahuje měření, jež bylo provedeno analogicky k měření při výběru blokového prokladače. Měření proběhlo ve volném rovinatém prostoru, kde se v okruhu 3 km nevyskytují obydlené oblasti. Při měření byly použity dva transceivery ZigBee CC2530 s implementovanou metodou

dopředné korekce chyb v podobě Hammingova kódu a Blokového prokladače s hloubkou prokladu h = 36. Software v modulech pracoval podle následujících vývojových diagramů.



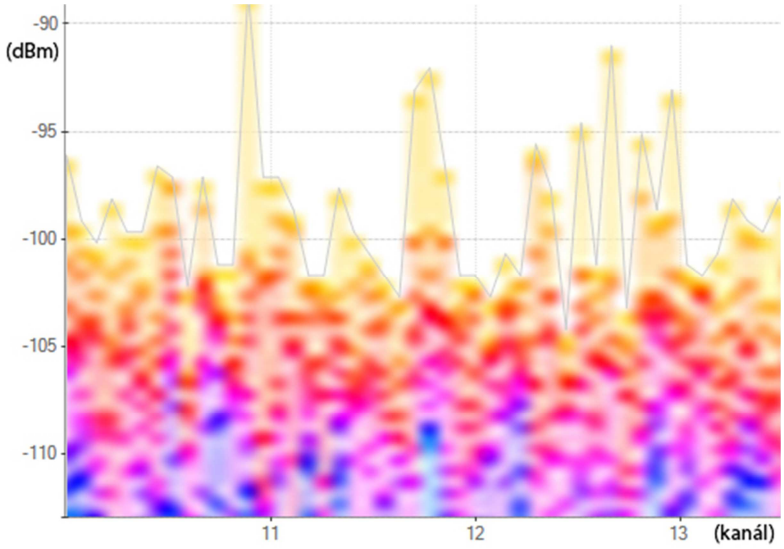
Obr. 59. Vývojový diagram aplikace odesílající rámce



Obr. 60. Vývojový diagram aplikace přijímací rámce

Rušení napodobující bílý šum bylo opět generováno malým točivým strojem s komutátorem, který byl umístěn v bezprostřední blízkosti antény přijímače (10 cm). Vzdálenost mezi přijímačem a vysílačem byla postupně zvětšována, kdy pro každý měřený bod bylo změřeno pět přenosů po 500 rámcích. Pro každé měření byl zaznamenán počet rámců korektně přijatých ZigBee standardem a počet rámců korektně přijatých s použitím metody pro dopřednou korekci chyb (FEC), popsanou v této práci. V tomto případě tedy Hammingovým kódováním v bloku dat 756 b spolu

s blokovým prokladačem s hloubkou prokladu h = 36. ZigBee spoj byl nastaven na 2 kanál 2,4 GHz spektra, tedy 12 kanál. Vysílač byl softwarově nastaven na nejvyšší vysílací výkon. Před měřením bylo také provedeno měření šumových podmínek v místě měření. Toto měření proběhlo spektrálním analyzátorem WiSpy, kterým bylo zjištěno, že hladina šumu na anténě ZigBee přijímače (se spuštěným točivým strojem) je přibližně -97dBm. Viz následující obrázek, kde je uveden otisk obrazovky.

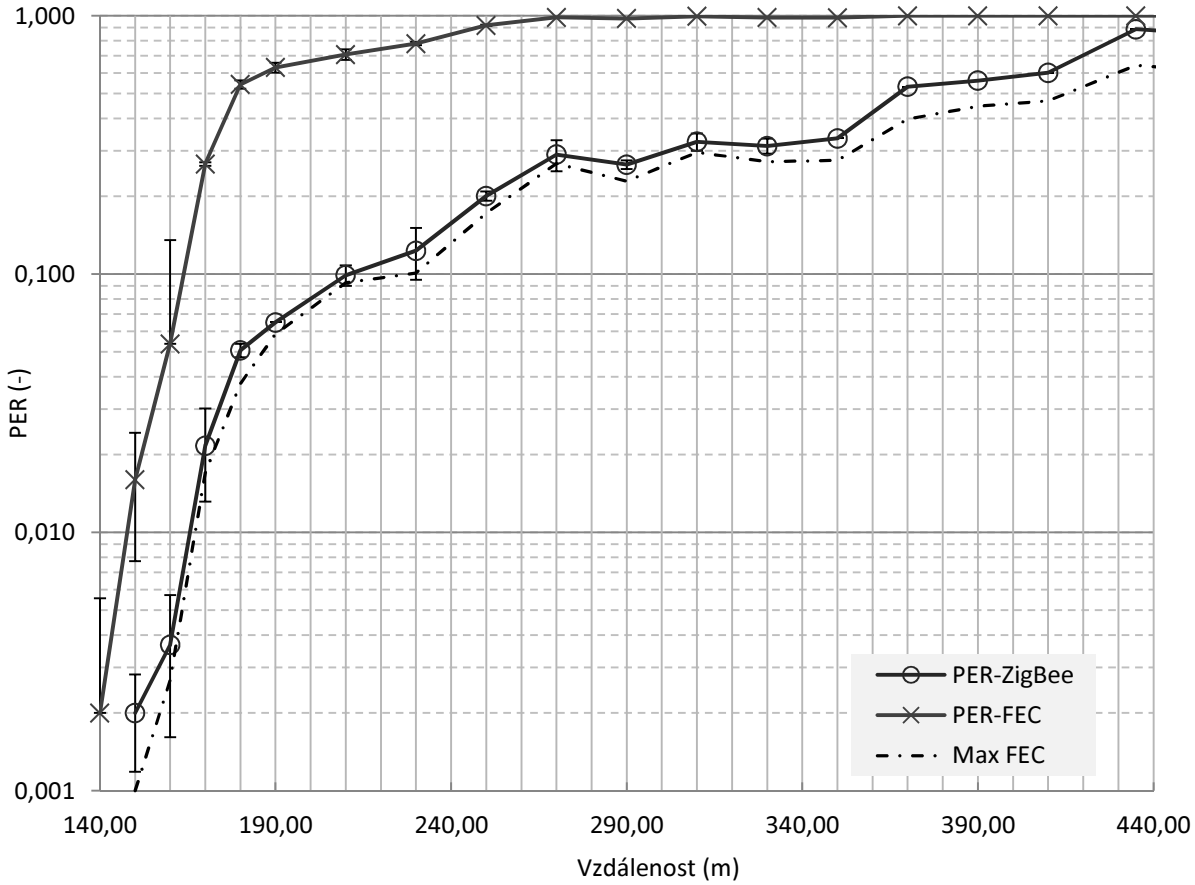


Obr. 61. Záznam měření šumových podmínek při měření

5.10.1 Výsledky měření

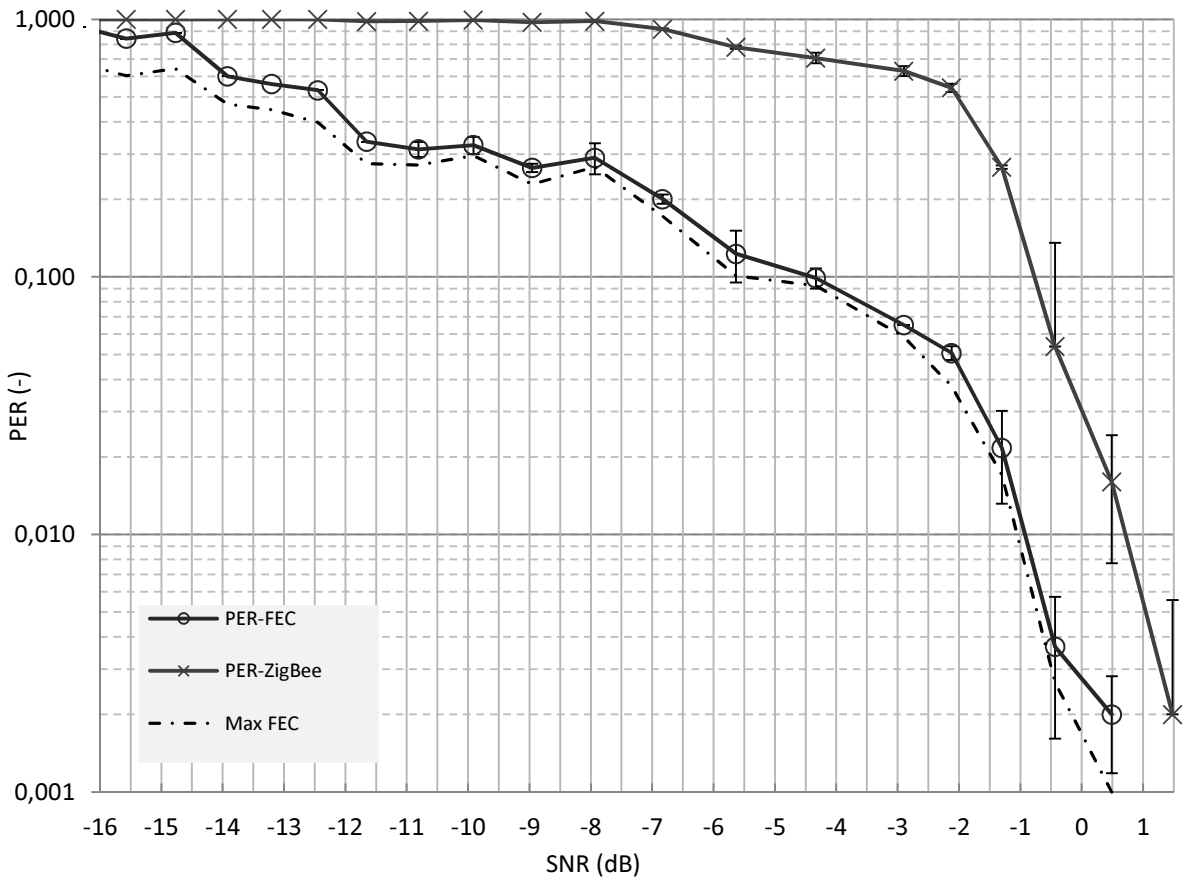
Na dalším grafu (Obr. 62) je uveden výsledek popsaného měření v závislosti na vzdálenosti. Standardu ZigBee, tedy neupravenému standardnímu přenosu, odpovídají měřené body vyznačené křížky. Body vyznačené kroužky odpovídají měřeným hodnotám metody dopředné korekce chyb (FEC). Čerchovaná čára odpovídá počtům rámců přijatých fyzickou vrstvou s korektní délkou. (Všechny tyto hodnoty jsou vyvedeny v podobě PER). Z Grafu lze číst, že kódovací metoda se až k hodnotě PER 0,3 drží velmi blízko maximálních dosažitelných hodnot. Tedy jsou úspěšně opraveny téměř všechny chybně doručené rámce. S narůstající chybovostí (PER > 0,3) již metoda ztrácí své schopnosti díky překročení samoopravných schopností kódu. Z výsledků lze také odhadnout, že při chybovosti PER 1 % lze spoj provozovat v rádiu o 18 m (+12 %) delším než klasický přenos. Se vzrůstající vzdáleností je při chybovosti PER 10 % akční rádius spoje prodloužen o 46 m (28 %). Spolu s dalším navyšováním vzdálenosti dochází k téměř úplnému zastavení přenosu ZigBee a zde se projevují největší klady zvolené metody dopředné korekce chyb. Tedy v situaci (270 m) kdy standardní ZigBee přenos již

vykazuje chybovost > 99 %, navrženou metodou lze přenášet rámce s PER méně než 30 %. Metoda se tedy jeví jako vhodná pro situace, kde muže dojít k náhlému a prudkému snížení kvality přenosového kanálu.



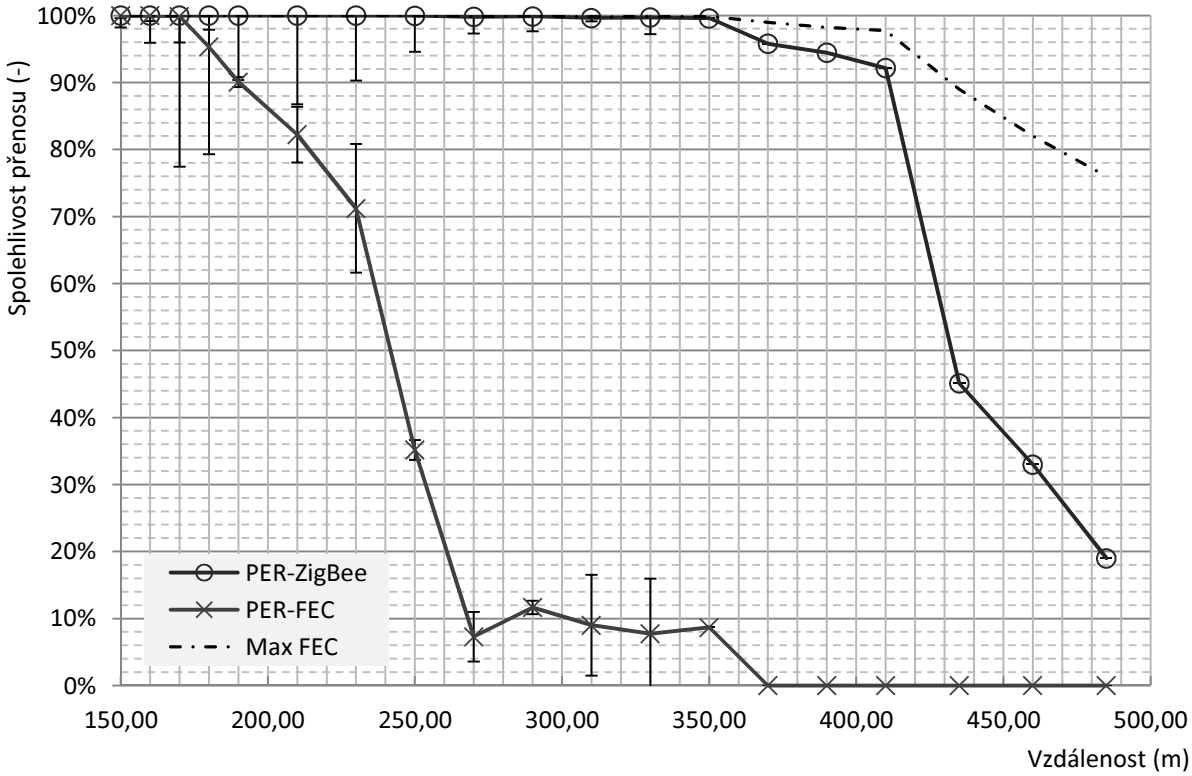
Obr. 62. Graf PER v závislosti na vzdálenosti

Na dalším obrázku (Obr. 63), je uveden graf pro stejné měření, ale vyjádřené v závislosti na vypočteném odstupu signál/šum. Hodnoty SNR byly vypočteny z parametru vzdálenosti vysílaného výkonu a také měření hladiny šumu na straně přijímače podle vztahů (12)(13)(14). Tento výstup měření lze snadněji porovnat s modelovou situací popsanou v předchozím textu. Při chybovosti PER 0,01 odpovídá zisk dopředné korekce 1,72 dB. Při chybovosti PER 0,1 je pak zisk 4,1 dB, tyto hodnoty se mírně liší od hodnot získaných simulací, což lze připsat v největší míře vlivu prostředí na měření ve volném prostoru. V absolutním měřítku SNR pak lze odchylky přičítat především nepřesnostem měření hladiny šumu. V neposlední řadě pak odchylky simulace a skutečného přenosu způsobuje odlišný charakter rušení, kdy v modelu byl použit bílý Gaussovský šum, kdežto při reálném měření se jedná o celou řadu kombinujících se zdrojů rušení, které měřením nelze rozpoznat.



Obr. 63. Graf PER v závislosti na SNR

Z průběhu lze usoudit, že největší přínos popsané metody FEC spočívá ve schopnosti přenášet rámce za okolností, kdy běžný přenos jeví nulovou spolehlivost. Technologie ZigBee používá zpětné potvrzení korektního příjmu rámce, ve výchozím nastavení je každý rámec opakován pětkrát, než je označen za nedoručitelný. Vezměme kupříkladu situaci z měření na Obr. 62, vzdálenost 340 m. Zde ZigBee projevuje PER 98,5 %, pravděpodobnost úspěšného přenosu při pěti opakováních je tedy cca 7 %. Bude-li totožný rámec přenášen s popsanou kódovací metodou, bude ve stejné vzdálenosti pravděpodobnost úspěšného přenosu při pěti opakováních 99,7 %. Pro ilustraci vlivu opakování na spolehlivost přenosu je uveden následující graf (Obr. 64), kde je předchozí měření zpracováno v podobě spolehlivosti přenosu pro 5 opakování. Uvedené hodnoty lze v grafu přehledně nalézt. Spolehlivost „1“ pro malé vzdálenosti mezi přijímačem a vysílačem je výsledkem nedostatečného počtu přenášených rámců pro jednotlivá měření. Díky maximální rychlosti 10 rámců za sekundu nelze měření provést dostatečně přesně.

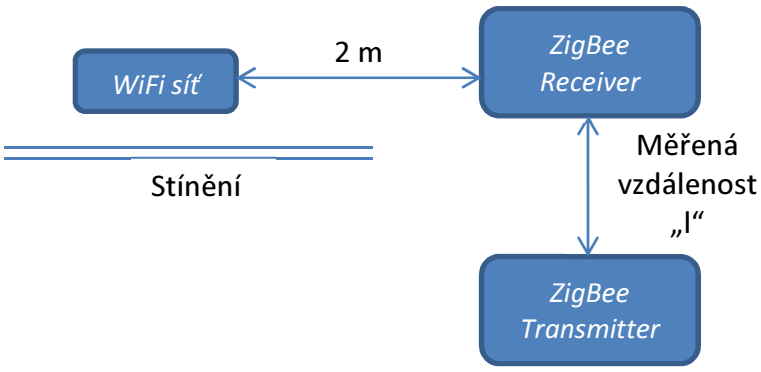


Obr. 64. Teoretická spolehlivost sítě při pěti opakováních přenosu

Z grafu vyplývá, že při spolehlivosti přenosu 99 % lze s kódováním přenášet data v okruhu více než dvakrát větším, než s běžným přenosem (170 vs. 350 m).

5.11 Měření účinků metody při koexistenci s WiFi - 802.11g

Posledním měřením jsou schopnosti popsané dopředné korekce chyb při koexistenci s WiFi 802.11g. Experiment proběhl analogicky k měření v bodu 5.3.3. Měření opět proběhlo ve volném prostranství, bez obydlených oblastí a vedení vysokého napětí v okruhu 3 km. ZigBee hardware (viz 5.1) byl opět doplněn o aplikaci obsahující popsanou dopřednou korekci chyb. Její základní princip je ilustrován vývojovými diagramy Obr. 59 a Obr. 60, jedná se tedy o totožnou aplikaci jako v předchozím měření s rozdílem hloubky prokladu, kdy v tomto měření byla v menu programu nastavena hloubka prokladu h = 108. Měření bylo provedeno v několika variantách provozu na WiFi síti a vzdálenostech mezi ZigBee transceivery „l“. Rozmístění aparatury je uvedeno na následujícím obrázku. K vytvoření WiFi spoje byla opět použita sestava WiFi směrovače USR8054 a tabletu Samsung P3110. Tablet byl připojen k WiFi směrovači s IP adresou 192.168.123.101. Tablet a WiFi směrovač byly umístěny v bezprostřední blízkosti a lze je považovat za jeden vyzařující bod.

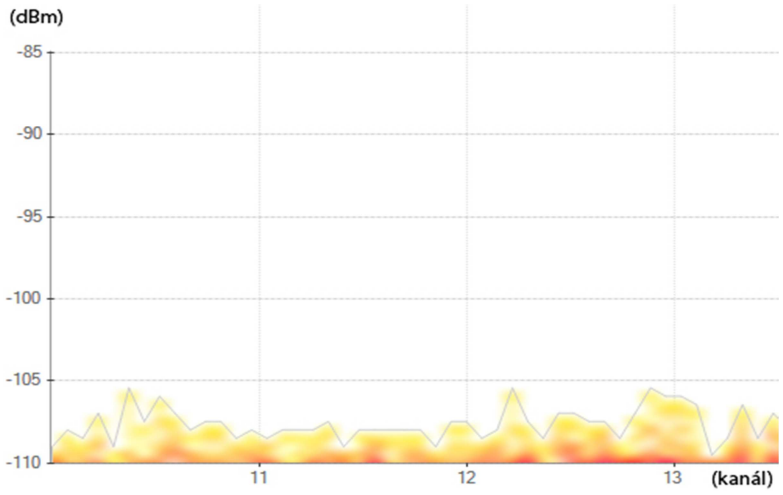


Obr. 65. Nákres rozmístění aparatury při měření koexistenčních vlastností

WiFi směrovač byl metalickým spojem 100 Mb.s^{-1} připojen k PC, kde byla spuštěna aplikace generující provoz na síti „Nping“. Tato aplikace byla spouštěna s následujícími parametry:

- *192.168.123.101*
- *--data-length X*
- *--delay Y*
- *--c 2^{32}*
- *--send-ip*

Kde X je délka uživatelských dat, která byla pro každé měření různá, stejně jako prodleva mezi odesílanými pakety Y. Prodleva byla nastavována pro každou velikost přenášených paketů ve dvou variantách -1 ms a -0 ms. První hodnota odpovídá nastavenému rozestupu, druhá varianta „0 ms“ pak pro program „Nping“ znamená generování „ping“ rámců tak rychle, jak systém umožňuje. Před měřením byla změřena hladina šumového pozadí v místě přijímače viz Obr. 66. Z měření vyplynulo, že výkon šumu nepřesáhne hladinu -105 dBm.



Obr. 66. Záznam měření šumových podmínek při měření

5.11.1 Výsledky měření

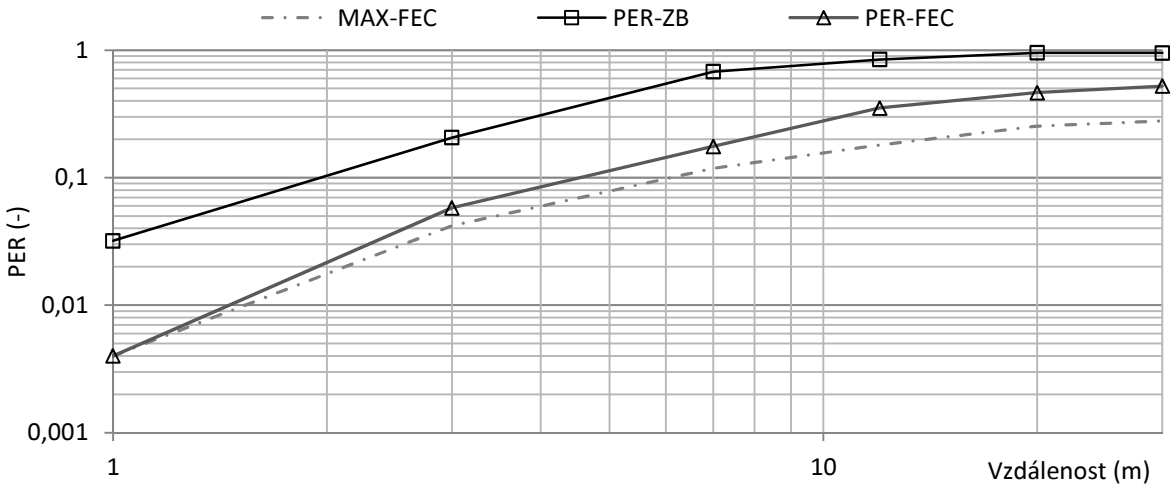
Jak bylo uvedeno, měření proběhlo v několika variantách provozu na WiFi síti. Charakter podmínek provozu na WiFi síti je uveden v podobě délky „ping“ rámce při přenosu na WiFi síti. Vzhledem k tomu, že délky rámců nejsou během přenosu konstantní, uvádím pro srovnání následující přehledovou tabulku (Tab. 13). Každý odeslaný „ping rámec" je odpovídán cílovým zařízením (Tablet Samsung) uvedenou velikostí rámce. Každý z těchto rámců je zároveň na WiFi následován ACK rámcem o délce 10 B.

Tab. 13. Přehledová tabulka rozměrů rámců pro všechná měření

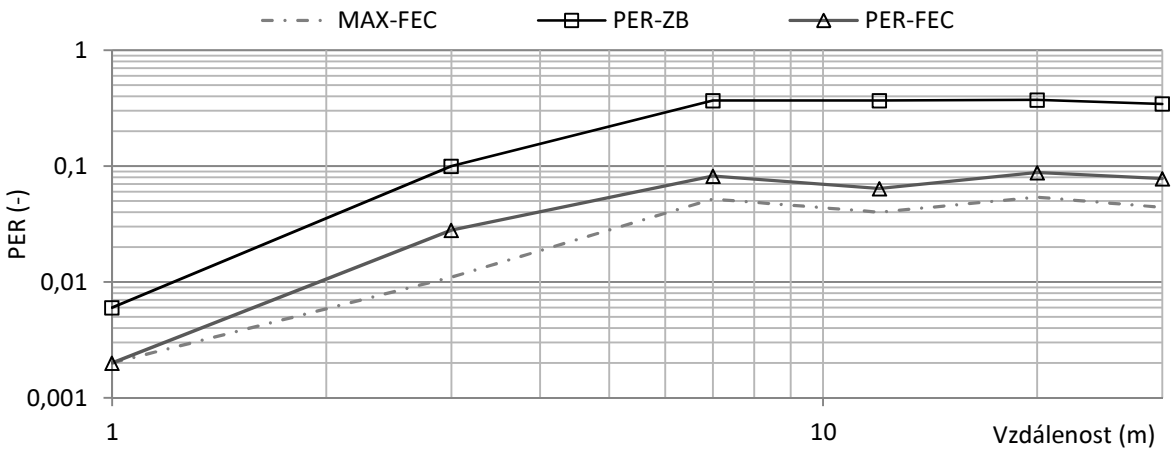
Payload	Rámec (TCP/IP)	Rámec (WiFi)	Odpověď (WiFi)	ACK (WiFi)
1 B	29 B	63 B	80 B	10 B
10 B	38 B	72 B	80 B	10 B
100 B	128 B	162 B	162 B	10 B
500 B	528 B	562 B	562 B	10 B
1450 B	1478 B	1512 B	1512 B	10 B

Měření při přenosu rámce s payloadem 1 B.

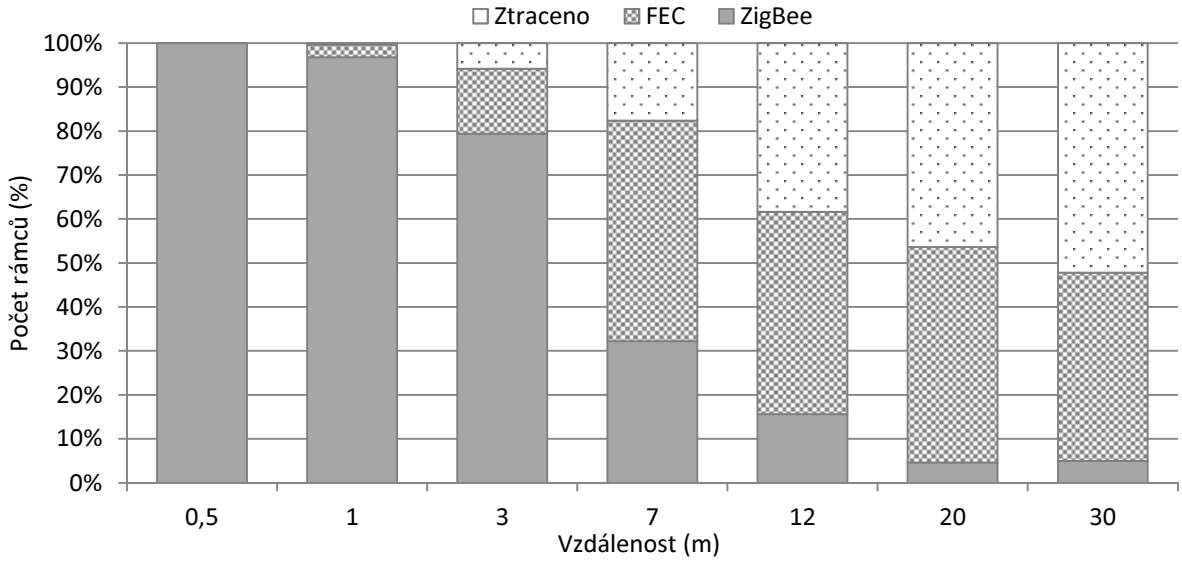
Na následujících grafech jsou uvedeny výsledky měření při koexistenční zátěži rámci s payloadem 1 B, tedy na WiFi síti byly přenášeny v definovaných rozestupech rámce 63 B + 10 B a 80 B a 10 B. Měření byla provedena pro rozestup 0 a 1 ms.



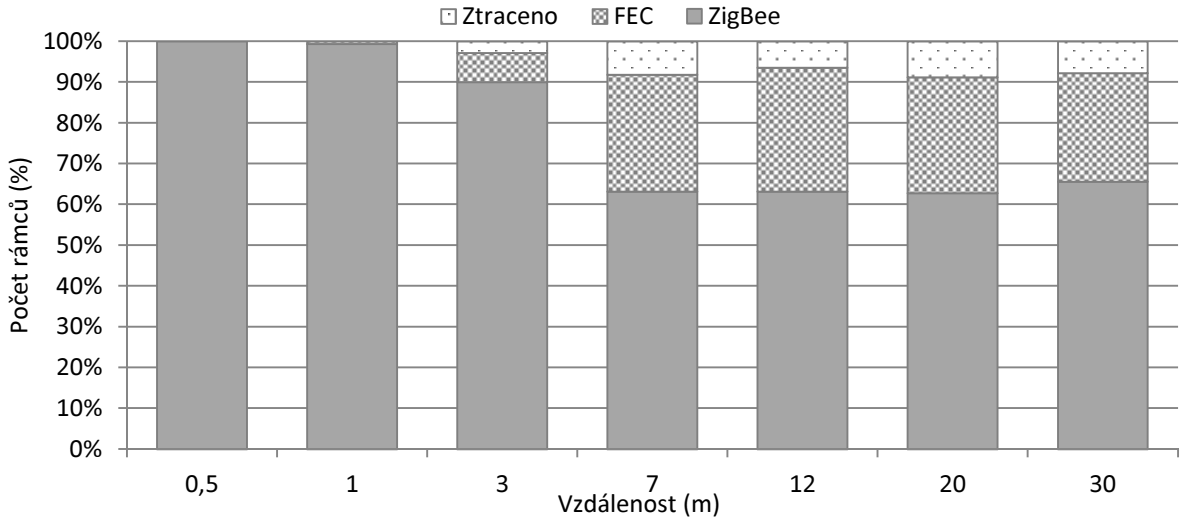
Obr. 67. Graf PER pro koexistenci s „1 B a 0 ms“



Obr. 68. Graf PER pro koexistenci s „1 B a 1 ms“



Obr. 69. Sloupcové vyjádření výsledků pro koexistenci s „1 B a 0 ms“



Obr. 70. Sloupcové vyjádření výsledků pro koexistenci s „1 B a 1 ms“

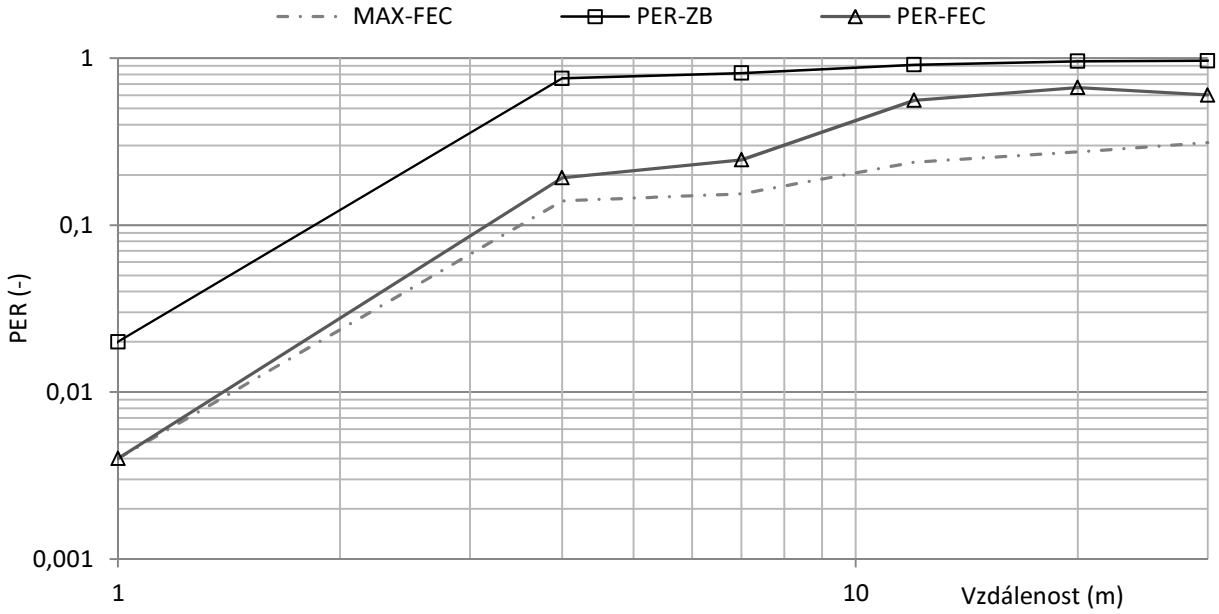
Z výsledků je zřejmé, že pro tento typ rušení má metoda vliv především při rozestupech alespoň 1 ms (prakticky se jedná o 4 rámce za 1 ms). Při tomto rozestupu dokáže spolehlivost sítě udržet nad 90 % za situace, kdy standartní přenos klesá ke spolehlivosti 60 %. Výsledky jsou uvedeny jak ve formě PER, kde je zřetelnější účinnost metody v ohledu k maximální účinnosti, tak ve sloupcovém provedení, ze kterého je lepší přehled o celkové ztrátovosti paketů.

Měření při přenosu rámce s payloadem 10 B.

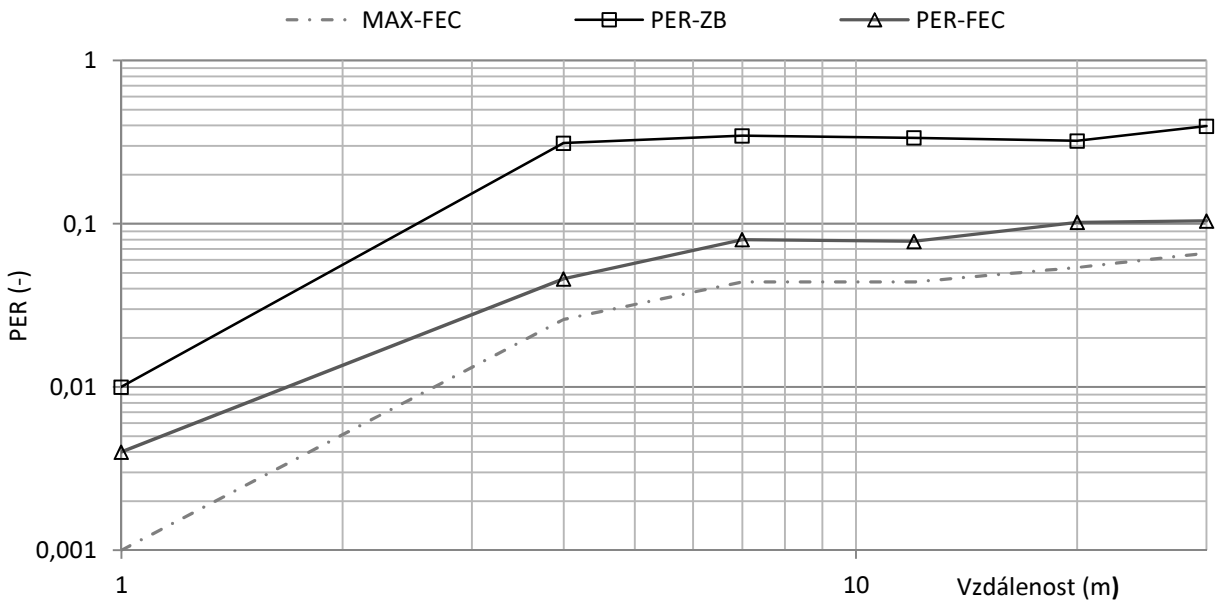
Výsledky tohoto měření se velmi podobají výsledkům z předchozího měření s 1 B, proto jsou výsledky umístěny v příloze VII.

Měření při přenosu rámce s payloadem 100 B.

V tomto měření je již znatelný pokles účinnosti metody vůči jejím maximům a to především při rozestupu 0 ms. Účinnost metody je významná pouze při vzdálenosti do cca 6 m.

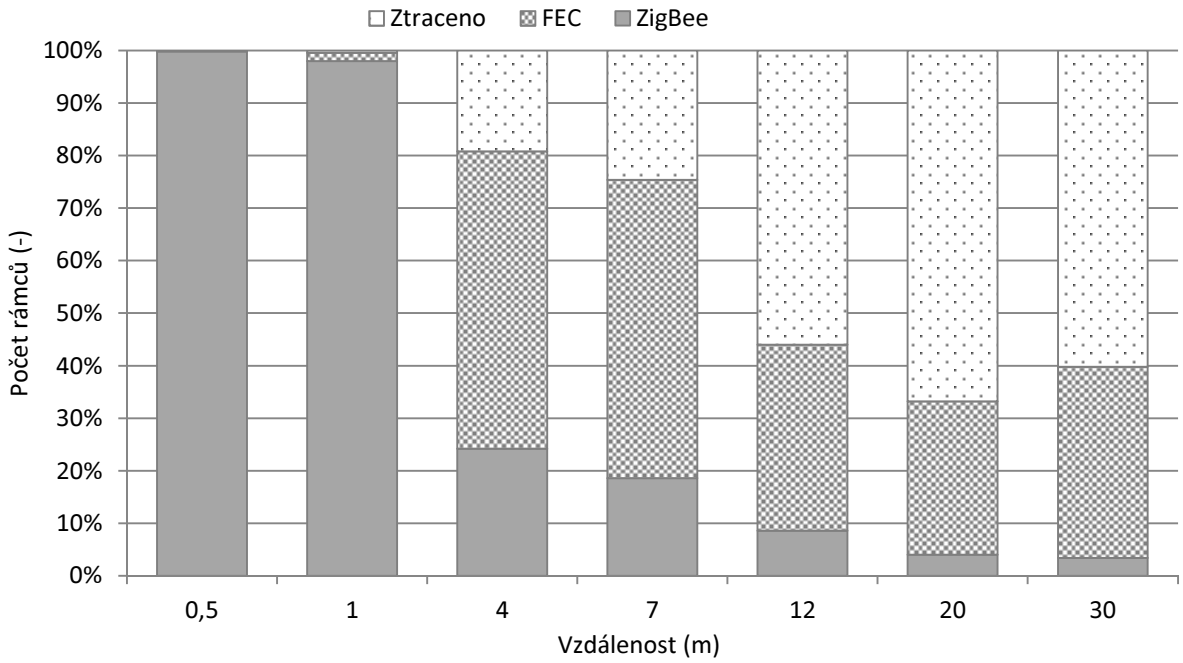


Obr. 71. Graf PER pro koexistenci s „100 B a 0 ms“

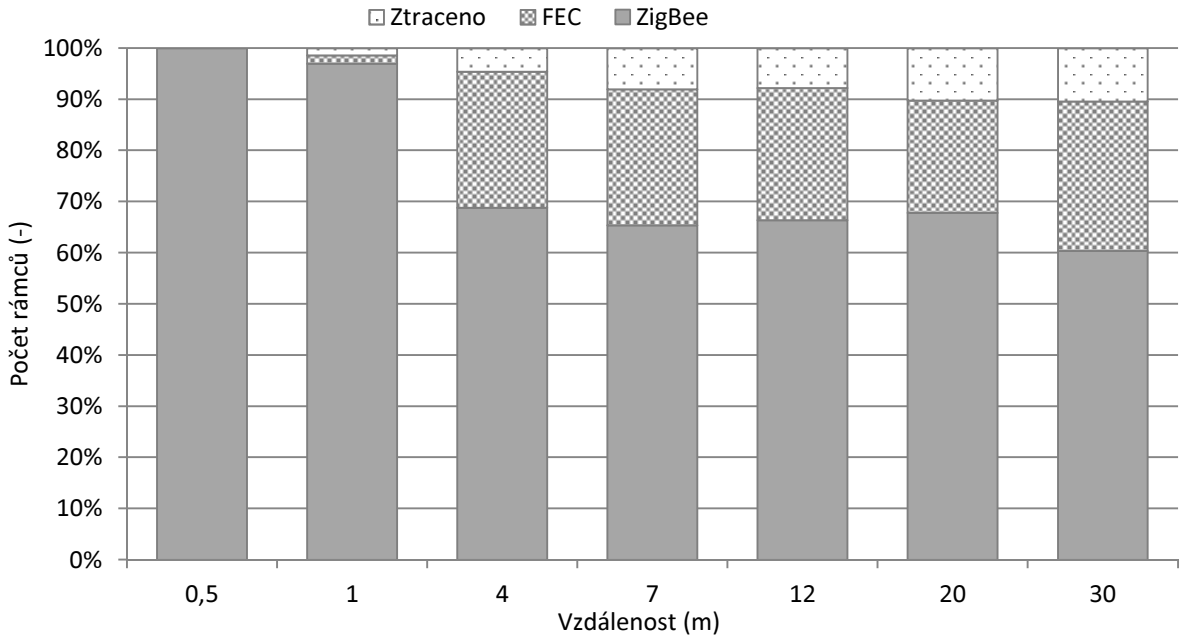


Obr. 72. Graf PER pro koexistenci s „100 B a 1 ms“

Na následujících grafech (Obr. 73 a Obr. 74) je znovu vyjádřeno měření v podobě percentuální, kde je opět zřetelný rozdíl v mezi rozestupy 0 a 1 ms. Pokles účinnosti je při rozestupu 0 ms oproti předchozímu měření zřetelný. Naopak metoda udržuje zhruba stejnou schopnost korekce chyb při rozestupu 1 ms, kdy dokáže udržet spolehlivost na cca 90 %, zatímco standardní přenos se pohybuje mezi 60 – 70 %.



Obr. 73. Sloupcové vyjádření výsledků pro koexistenci s „100 B a 0 ms“

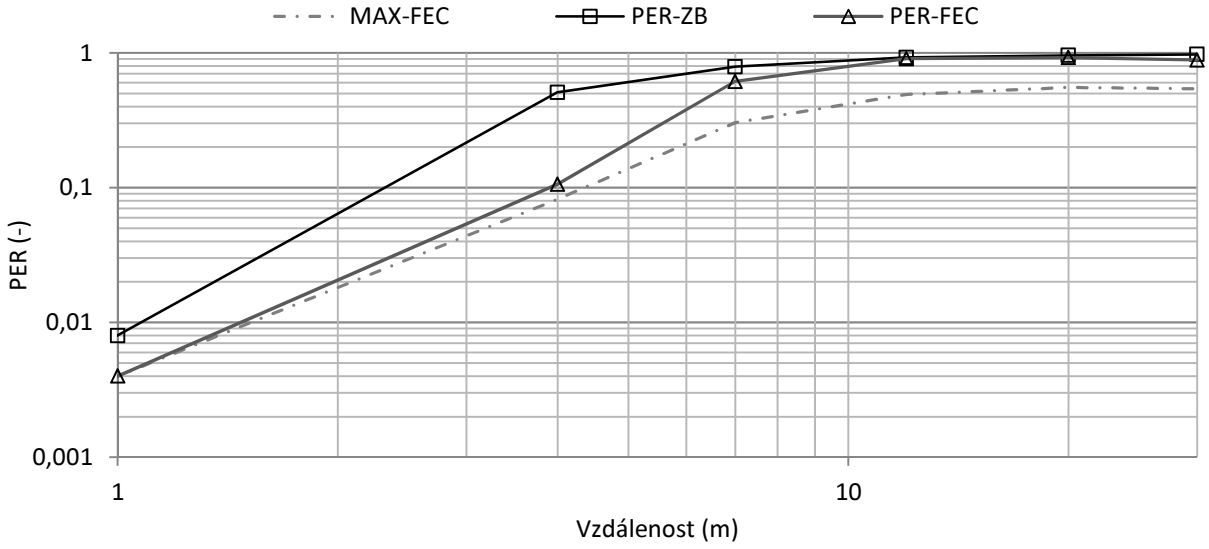


Obr. 74. Sloupcové vyjádření výsledků pro koexistenci s „100 B a 1 ms“

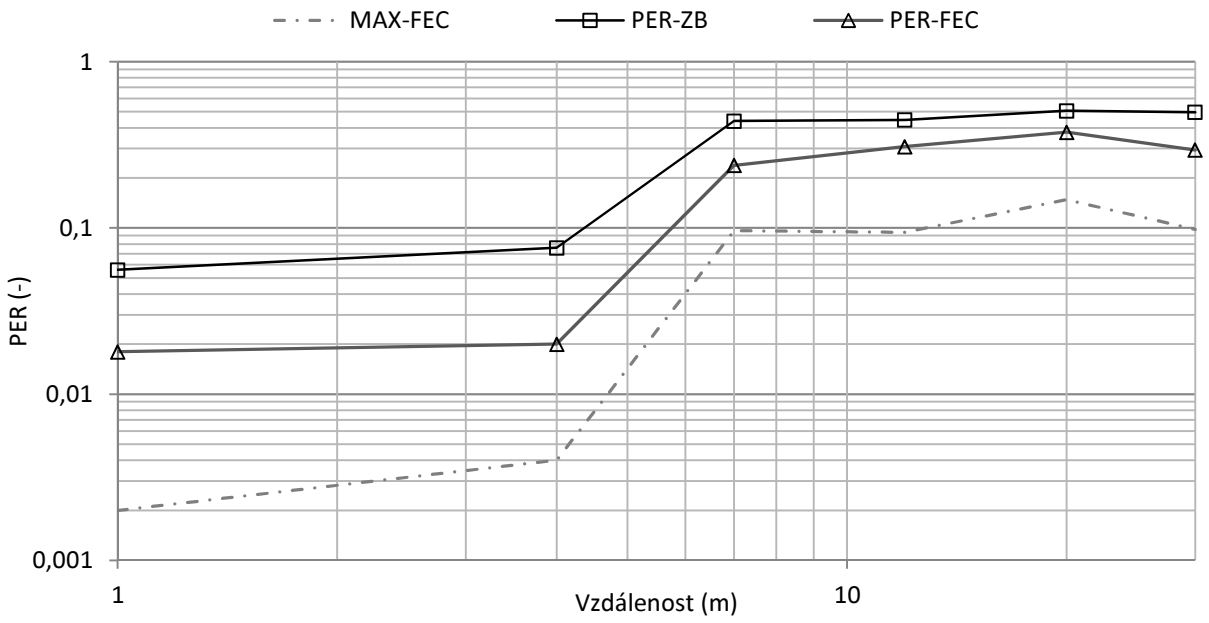
Výsledky měření s variantou koexistenčního provozu s 500 B jsou umístěny v příloze VII.

Měření při přenosu rámce s payloadem 1450 B.

Následující koexistenční varianta, dle dříve popsaných předpokladů, již silně překonává schopnosti dopředné korekce chyb. Při rozestupu 0 ms (první graf Obr. 75) je i maximální možnost dopředné korekce chyb (čerchovaná čára) velice nízká. Samotná korekční metoda pak tohoto maxima, především při větších vzdálenostech „l“, zdaleka nedosahuje.

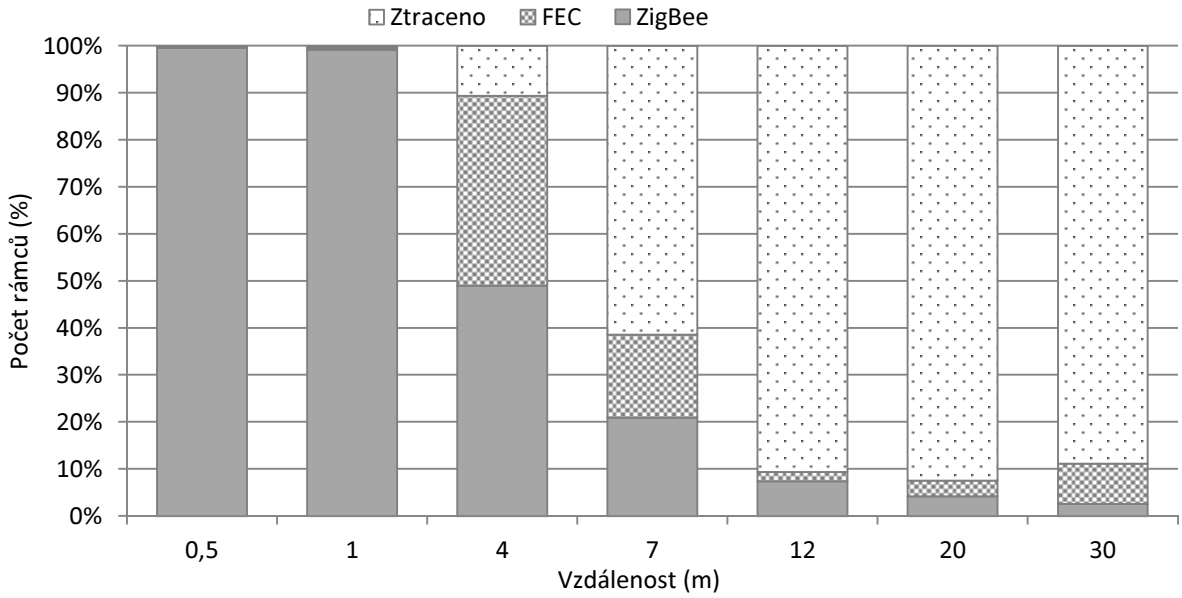


Obr. 75. Graf PER pro koexistenci s „1450 B a 0 ms“

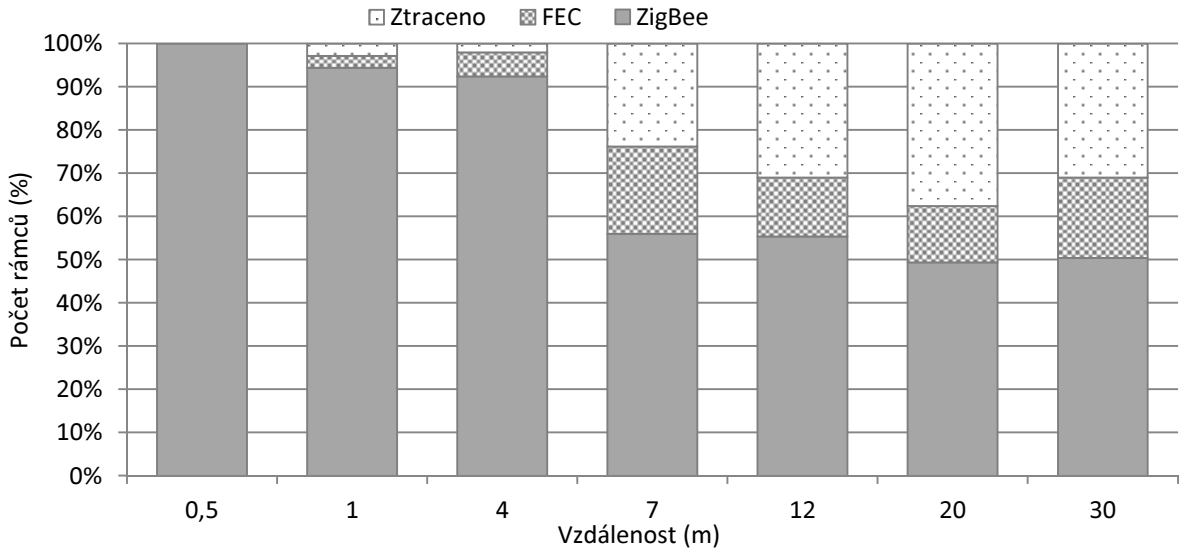


Obr. 76. Graf PER pro koexistenci s „1450 B a 1 ms“

Přehledněji se situace může jevit na následujících dvou grafech Obr. 77 a Obr. 78. Lze z nich číst, že při rozestupu 0 ms je učinnost nezanedbatelná pouze v okamžiku, kdy díky rostoucí vzdálenosti stoupne PER samotného ZigBee standardu, tento okamžik odpovídá měřené hodnotě 3 m.



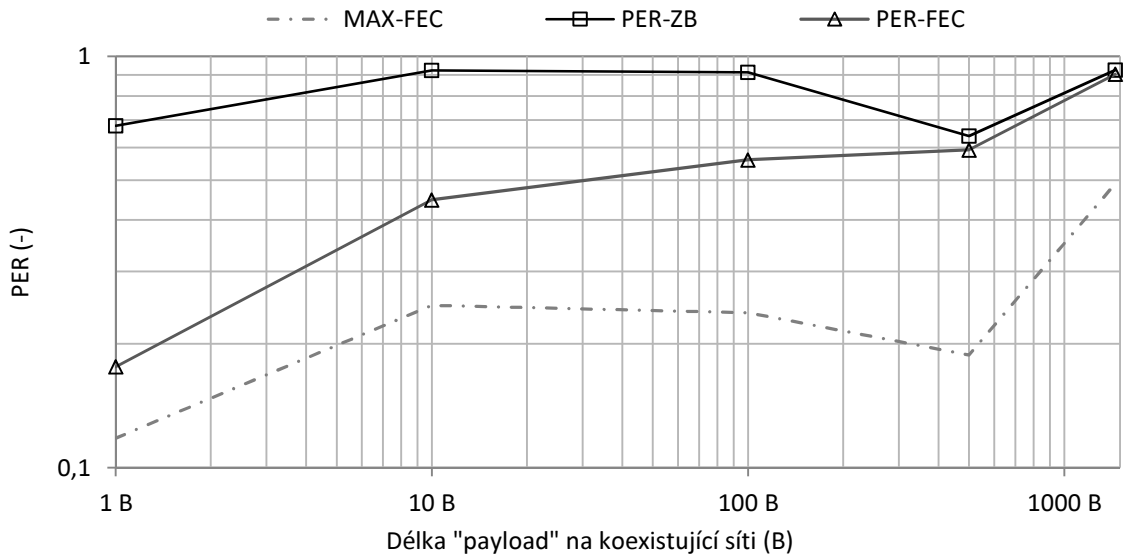
Obr. 77. Sloupcové vyjádření výsledků pro koexistenci s „1450 B a 0 ms“



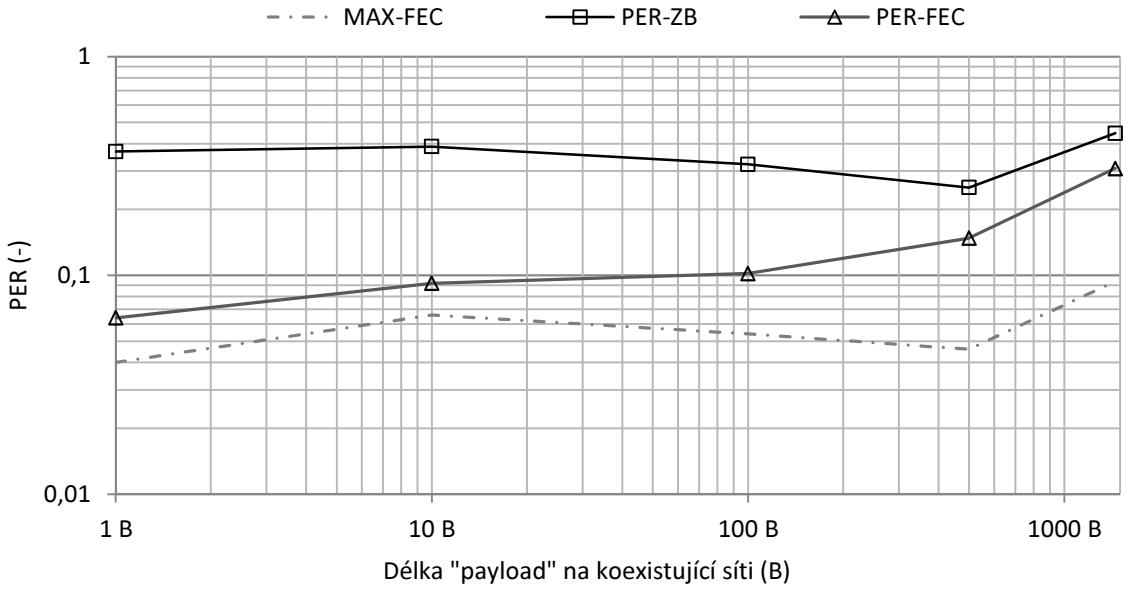
Obr. 78. Sloupcové vyjádření výsledků pro koexistenci s „1450 B a 1 ms“

Při rozestupu 1 ms je účinnost oproti předchozímu uvedenému měření snížena také velice razantně a její přínos se pohybuje při vzdálenostech nad 4 m mezi 10 a 20 %. Vezmeme-li v úvahu rozbor provedený v bodě 5.3.2, lze tuto variantu, tedy s délkou přenášených rámců na WiFi 1512 B oběma směry, považovat za absolutní extrém, ke kterému v běžných podmínkách může dojít jen výjimečně.

Na dalším grafu je výsledná závislost účinnosti metody na velikosti přenášených rámců koexistujícím standardem. Z grafu lze číst postupný pokles účinosti metody spolu s rostoucí velikostí přenášených rámců na koexistující síti, kdy ovšem maximální hranice FEC metody stoupá mírněji. Lze tedy usoudit, že rámce je možno do měřené velikosti 500 B sice získát se zhruba stejnou spolehlivostí, obsah chyb v datech již ale stoupá nad schopnosti FEC metody. Budeme-li jako interferující prvek přednostně považovat spíše mobilní zařízení využívající připojení k síti, lze očekávat spíše přenos menších rámců viz Tab. 9 uvedená v bodě 5.3.1.



Obr. 79. Graf PER pro koexistenci ve vzdálenosti 12 m a rozestupy koexistujících rámců 0 ms.



Obr. 80. Graf PER pro koexistenci ve vzdálenosti 12 m a rozestupy koexistujících rámců 1 ms.

Na grafu Obr. 80 je pak uveden stejný výsledek měření, ovšem s rozestupy 1 ms. Ukazuje se, že vliv velikosti rámců na koexistující síti je obdobný, ačkoli rušení je celkově slabší. Oproti předchozímu případu je i samotný standard ZigBee schopen přenášet rámce s PER cca 0,3.

VI. ZÁVĚRY A DOPORUČENÍ PRO VYUŽITÍ POZNATKŮ V PRAXI

Postup při řešení zadání práce s sebou přinesl i několik poznatků, které souvisí s řešenou problematikou, ač nebyly cílem práce. V následujících bodech jsou uvedeny hlavní přínosy práce:

- Analýza vzniku chyb při koexistenci ZigBee a WiFi, a také rozbor důsledků vznikajících z koexistenčních podmínek.
- Rozbor a zhodnocení jednotlivých variant kódovacích metod při opravách chyb zaznamenaných v předchozím bodě.
- Vytvoření univerzálního modelu ZigBee sítě, především jeho programové části s detailním zpracováním signálu.
- Rozbor a nalezení přístupového bodu k fyzické vrstvě ve firmwaru Z-Stack, který umožnil vznik celého řešení.
- Návrh a realizace metody dopředné korekce chyb, především ve spojení s vytvořením teoretického simulačního modelu.
- Finální měření, potvrzující výsledky získané teoretickým rozborem a detailně popisující vlastnosti realizované metody.

Doporučení pro využití výsledků v praxi souvisí především s problematikou běžného provozu na síti, který byl popsán v úvodu práce. Další vývoj by se tedy měl týkat zařazení funkcí metody do provozu na síti, kdy je třeba především vyřešit nutnost dekódování všech příchozích rámců s definovanou délkou. Tato eventualita může způsobit problémy v nízkoenergetických řešeních. Částečně lze problém řešit například zakázáním definované velikosti v protokolu či využitím 4 nevyužitých bitů v kódovaném bloku (760 vs. 756 b) jako signálního znaku, který bude další podmínkou pro dekódování.

Za další prostor k rozvoji práce pak lze označit možnosti dalšího zvýšení korekčních schopností kódu pro případy, kdy množství odesílaných dat je menší, než umožňuje blok dat. V takovém případě by bylo možno datový rámec využít například k dalšímu navýšení redundance.

VII. ZÁVĚR

Cílem práce je zvýšení spolehlivosti ZigBee standardu při zachování kompatibility s běžnými transceivery. Řešení práce lze rozdělit na několik základních oblastí.

Nejprve byly popsány a zhodnoceny možné směry řešení cílů práce. Výsledkem tohoto kroku bylo určení směru softwarového řešení. V následujícím textu byly popsány možnosti softwarových řešení, tedy jednotlivé vhodné metody pro dopřednou korekci chyb a jejich základní principy. Pro jejich precizní zhodnocení a následný výběr bylo třeba zjistit charakter chyb, vznikajících v zarušeném prostředí.

Dále byly v práci popsány detailní charakteristiky vznikajících chyb při koexistenci s WiFi 802.11g jako modelový případ rušení ZigBee sítě. Z výsledků tohoto rozboru bylo následně možno vyhodnotit jednotlivá navržená řešení z pohledu úspěšnosti korekce chyb. Na základě tohoto vyhodnocení, byly zvoleny dvě řešení, jedno na principech Hammingova kódu a druhé s korekčním kódem Reed-Solomon. Oba s použitím prokladače.

V další části byla zvolená řešení, hodnocena z pohledu zatížení mikrokontroléru. Tato vlastnost byla posuzována na základě doby zpracování algoritmu pro daný blok dat. Toto měření ukázalo, že Reed-Solomonův kód, ačkoli jeho korekční schopnosti jeví mírně lepší vlastnosti, než Hammingův kód, je pro řešení v mikrokontroléru příliš výpočetně náročný. Zvoleno bylo řešení s Hammingovým kódem a blokovým prokladem.

Následně je v práci vyhodnocen vliv rozměru matice blokového prokladače. Experimentálně získané datové vzorky byly dekódovány zvolenými rozměry prokladače pro interference vzniklé jak v důsledku koexistenčního přenosu, tak širokospektrálním rušením. Výstupy tohoto rozboru byly protichůdné, proto jsou v této práci dále měřeny dvě varianty blokového prokladače a tedy dvě výsledná řešení.

Pro obě získaná řešení byl následně vytvořen model, který byl zpracován ve dvou propojených částech. Binární část zpracování dat byla vytvořena v podobě programu v prostředí VisualBasic 6.0. Druhá část, simulující samotný bezdrátový přenos, byla zpracována v programu MATLAB Simulink. Přenos dat mezi částmi modelu byl zajištěn pomocí datových souborů. Z takto zpracovaného modelu byla získána data o úspěšnosti metody a také o maximálních možnostech tohoto řešení.

V poslední části práce byly oba typy řešení implementovány do ZigBee transceiverů CC2530. Takto upravené ZigBee mikrokontroléry byly použity

k výslednému měření účinnosti metody. Výsledky ukazují, že oproti modelové situaci má metoda mírně lepší výsledky. Tento fakt lze přisoudit odlišné povaze rušení použitého v modelu, kde kanál AWGN generuje matematicky přesný bílý šum. V praxi je ovšem charakter rušení výsledkem mnoha faktorů, které nelze přesně popsat.

Výstupní grafické závislosti jsou uvedeny na konci práce z několika pohledů spolu s vyhodnocením jejich nepřesností. Především pro měřené body s nízkou chybovostí nelze získat dostatečný počet rámců k dosažení přesných výsledků. Tato překážka spočívá v problematice zpracování protokolu ZigBee (Z-Stack), která umožňuje zasílání rámců maximálně s odstupem 100 ms.

Celkově lze navrženou korekční nadstavbu pro ZigBee standard hodnotit velice pozitivně, např. v závěrečném měření umožnila metoda datovou komunikaci se spolehlivostí 99 % prodloužit ze 170 m na 350 m za předpokladu pěti opakování nedoručených rámců. Cíle práce lze v plné šíři považovat za splněné, neboť všechny vstupní parametry byly zachovány. Snížení obsahu aplikačních dat v datových rámcích byl snížen na 57 %, a byla výrazně zvýšena schopnost odolávat vůči rušení.

POUŽITÁ LITERATURA

[1] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc*., Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Network (WPANs).* NewYork: Institute of Electrical and Electronics Engineers, Inc., 2006. 305s. ISBN 0-7381-4997-7.

[2] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, Inc., *Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs).* New York: Institute of Electrical and Electronics Engineers, Inc., 2005, 580s. ISBN 0-7381-4708-7.

[3] ROBERSON, D. A. *et al. Spectral occupancy and interference studies in support of cognitive radio technology deployment.* In: Pro-ceedings of the 1st IEEE Workshop on Networking Technolo-gies for Software Defined Radio Networks (SDR'06), 2006, s. 26–35.

[4] WELLENS, M., WU, J., MÄHÖNEN, P*. Evaluation of spec-trum occupancy in indoor and outdoor scenario in the context of cognitive radio.* In: Proc. Second International Conference on Cognitive Radio Oriented Wireless Networks and Commu-nications (CrowCom 2007), 2007, s. 1–8.

[5] ALLIANCE, ZIGBEE. *Zigbee specification.* ZigBee document 053474r06, version 1.0, 2005, 378s.

[6] FARAHANI, S. *ZigBee wireless networks and transceivers.* Newnes, 2008, 339s. ISBN:978-0-7506-8393

[7] CHALHOUB, G., HADDID, N., GUITTON, A., MISSON, M. *Deference mech-anisms significantly increase the MAC delay of slotted CSMA/CA* In: ICC, International Conference on Communications, June 2009.

[8] KOUBAA, A., ALVES, M., TOVAR, E. A *comprehensive simulation study of slotted CSMA/CA for IEEE 802.15. 4 wireless sensor networks.* In: IEEE WFCS. 2006, 6: 63-70.

[9] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS*, Inc., Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band.* NewYork: Institute of Electrical and Electronics Engineers, Inc., 2000. 89s. ISBN 0-7381-1812-5.

[10] GISLASON D. *Zigbee Wireless Networking*, Oxford: Newnes, 2008, s. 9-30, ISBN 0750685972

[11] GAST, M. *802.11 wireless networks: the definitive guide.* O'Reilly Media, Inc., 2005. ISBN: 0-596-00183-5

[12] LI, T. *et al. Performance analysis of the IEEE 802.11 e block ACK scheme in a noisy channel. In: Broadband Networks, 2005. BroadNets 2005.* In: 2nd International Conference on. IEEE, 2005. s. 511-517.

[13] ZANDL, P. *Bezdrátové sítě: WiFi praktický průvodce. 1. vydání.* Brno: Computer press, 2003. 190 s. ISBN 80-7226-632-2

[14] TEXAS INSTRUMENTS, ti.com. *CC2530 Development Kit.* [cit. 2013-04-4]. Dostupné z http://www.ti.com/tool/cc2530dk

[15] IAR SYSTEMS*, iar.com. IAR Embedded Workbench for 8051. [cit. 2013-04-4]. Dostupné z http://www.iar.com/en/Products/IAR-Embedded-Workbench/8051/*

[16] TEXAS INSTRUMENTS, ti.com. *A fully compliant ZigBee 2012 protocol stack: Z-Stack*. [cit. 2013-04-4]. Dostupné z http://www.ti.com/tool/z-stack

[17] FARSARIS, N., XENOS, T., STAVROULAKIS, P. *Airborne Video Transmission for Naval and Coast Guard Applications,* In: Proceedings of the 6th WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, World Scientific and Engineering Academy and Society (WSEAS), Corfu Island 2007. 168-174 s. ISBN: 123-456-7890-12-3.

[18] SCHULZE, H., LÜDERS, CH. *Theory and Applications of OFDM and CDMA,* Chichester : John Wiley & Sons Ltd, 2005. 403 s. ISBN-13 978-0-470-85069-5

[19] NICULESCU, C., BÎZDOACĂ, N., PANĂ C. *Multiuser transceiver what uses OFDM,* 3rd WSEAS Int. Conf. on Electronics, Hardware, Wireless and Optical Communications, World Scientific and Engineering Academy and Society (WSEAS), Austria 2004. ISBN: 960-8052-95-5

[20] PUST, R. *Analýza chování radiového systému s adaptivním frekvenčním skákáním v podmínkách intenzivního rušení [online]. 2008. Dostupný z WWW: < http://www.elektrorevue.cz/cz/clanky/komunikacni-technologie/70/analyza-chovani-radioveho-systemu-s-adaptivnim-frekvencnim-skakanim-v-podminkach-intenzivniho-ruseni/ >. Použit 2013. ISSN 1213-1539.*

[21] BERROU, C., GLAVIEUX, A., THITIMAJSHIMA*, P. Near Shannon limit error-correcting coding and decoding Turbo-codes,* In Proc. ICC'93, Geneva 1993. s. 1064-1070. ISBN: 0-7803-0950-2

[22] HARJANI, R., HARVEY, J., SAINATI, R. *Analog/RF physical layer issues for UWB systems.* In: VLSI Design, 2004. Proceedings. 17th International Conference on. IEEE, 2004. s. 941-948.

[23] SOLEYMANI, M., *et al. Turbo coding for satellite and wireless communications.* Kluwer Academic Pub, 2002.

[24] GLOVER, I. A., GRANT, P. M. *"Digital Communications",* 3rd ed., ISBN 978-0-273-71830-7, Pearson, 2010.

[25] NĚMEC, K. *Datová komunikace.* Skriptum VUT, VUTIUM 2000.

[26] NĚMEC, K. *Protichybové kódové systémy.* Elektrorevue, 2001/29, URL http://www.elektrorevue.cz/clanky/01029/index.htm

[27] HANUS, S. *Rádiové a mobilní komunikace.* Brno, FEKT VUT, 2002.

[28] HRDINA, Z., VEJRAŽKA, F. *Digitální radiová komunikace.* Skripta ČVUT v Praze. Praha, vydavatelství ČVUT, 1995. ISBN 80-01-01059-7

[29] COSTELLO, D. *et al. Applications of error-control coding.* Information Theory, IEEE Transactions on, 1998, 44.6: 2531-2560.

[30] COSTELLO, D. *Error Control Coding: Fundamentals and Applications*. Prentice Hall 1983, ISBN 978-0130426727

[31] JIROUŠEK, R. *Principy digitální komunikace*. Leda, 2006, ISBN 978-80-7335-084-0

[32] FARRELL, P. G. *Coding as a cure for communications calamities'.* Electron. & Commun. Eng. J, 1990, 6.2.

[33] VAN LINT, J. H. *On the nonexistence of perfect 2-and 3-hamming-error-correcting codes over GF (q*). Information and Control, 1970, 16.4: 396-401.

[34] VAN LINT, J. H. *Nonexistence theorems for perfect error-correcting codes.* American Mathematical Society, 1971.

[35] VLČEK, K. *Komprese a kódová zabezpečení v multimediálních komunikacích.* Vyd. 2. Praha : BEN Technická literatura, 2004. 258 s. ISBN 8073001349.

[36] HAMMING, R. W. *Error detecting and error correcting codes.* The Bell System Technical Journal 26,2, (1950), s. 147-160.

[37] ADÁMEK, J. *Kódování,* skriptum ČVUT, SNTL Praha 1986

[38] NĚMEC, K. *Protichybové kódové zabezpečení s Bose-Chaudhury-Hocquenhemovými kódy. Dostupné na WWW: http://www. elektrorevue. cz/clanky/05015/index.htm.*

[39] ČÍKA, P. *Protichybové zabezpečení BCH kódem. Dostupné na WWW: http://www.elektrorevue.cz/clanky/06015/index.htm.*

[40] MASSEY, J. *Threshold Decoding.* MIT, Press Research Monograph 20,The MIT Press, Cambridge, 1961

[41] NĚMEC, K. *Princip zabezpečení zpráv proti chybám pomocí kódů*. Elektrorevue, 2001/29, URL http://www.elektrorevue.cz/clanky/00029/index.htm.

[42] YATAO, D., *et al. Improved Decoding Algorithm for BCH Code*. Information Technology Journal, 2010, 9.6: 1251-1254.

[43] REED, I. S., SOLOMON, G. Polynomial Codes Over Certain Finite Fields, SIAM Journal of Applied Math., vol. 8, 1960, s. 300-304.

[44] WICKER, S. B., BHARGAVA, V. K. *Reed Solomon codes and their applications*, IEEE Press, 1994, ISBN: 0-78035391-9

[45] BERLEKAMP, E. R. *Section on the Berlekamp-Massey algorithm*. In Algebraic Coding Theory, s. 145 - 148. McGraw Hill, New York, NY, 1968.

[46] SLANINA, M., ŘÍČNÝ, V. *Reed-Solomonovy kódy v časové a ve frekvenční oblasti*. Dostupné na WWW: http://www. elektrorevue. cz/clanky/06011, 2006.

[47] SKLAR, B. *Reed-Solomon Codes* [online]. [cit. 2008-05-10]. Dostupný z WWW: http://www.facweb.iitkgp.ernet.in/~pallab/mob_com/art_sklar7_reed-solomon.pdf

[48] MOREIRA, J., FARRELL, P. *Essentials of error-control coding*. Wiley, 2006.

[49] SHI, Y. Q. et al. *Interleaving for combating bursts of errors.* Circuits and Systems Magazine, IEEE, 2004, 4.1: 29-42.

[50] LIANG, C. M., et al. *Surviving wi-fi interference in low power zigbee networks*. In: Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems. ACM, 2010. s. 309-322.

[51] TAMOSOFT. *Wireless Network Analyzer and Monitor – CommView for WiFi*. [cit. 2013-2-09]. Dostupné z: http://www.tamos.com/products/commwifi/

[52] LAKSHMAN, T. V., MADHOW, U. *The performance of TCP/IP for networks with high bandwidth-delay products and random loss*. Networking, IEEE/ACM Transactions on, 1997, 5.3: 336-350.

[53] CÁCERES, R. et al. *Characteristics of wide-area TCP/IP conversations*. In: ACM SIGCOMM Computer Communication Review. ACM, 1991. s. 101-112.

[54] KÁRNÁ, L., KLAPKA, Š. *Co jsou to vlastně nezávislé kódy?* MIS 2011, Matfyzpress Praha, 8 s.

[55] ROCKLIF, S. Reed-Solomon (RS) codes [online]. [cit. 2013-02-3]. Dostupné z: http://www.eccpage.com/rs.c

[56] KOTON, J., ČÍKA, P., KŘIVÁNEK V. *Samoopravné Reed-Solomonovy kódy* [online]. [cit. 2013-02-4]. Dostupné z: http://access.feld.cvut.cz/view.php?cisloclanku=2006080002

SEZNAM POUŽITÝCH SYMBOLŮ

ACK - Acknowledgement;

ADSL - Asymmetric Digital Subscriber Line;

AFH - Adaptive Frequency Hopping;

APS - Application Support Layer;

BER - Bit Error Rate;

BCH - Bose-Chaudhuri-Hocquenghem;

CCA - Clear Channel assessment;

CCK – Complementary Code Keying;

CRC - Cyclic Redundancy Check;

CSMA-CA - Carrier Sense Multiple Access with Collision Avoidance;

CTS - Clear To Send;

DIFS - Distributed InterFrame Space;

EIRP – Equivalent isotropically radiated power;

FCS - Frame Control Sequence;

GSM – Global Systém for Mobile Communications;

GTS - Guaranteed Time Slot;

HSDPA - High-Speed Downlink Packet Access;

ISM - Industrial Scientific and Medical;

KD - kolizní délka;

LQI - Link Quality Indicator;

MAC - Media Access Control;

MAN - Metropolitan Area Network;

MFR - MAC Footer;

MHR - MAC Header;

MIMO - Multiple Input Multiple Output;

NWK - Network Layer;

OQPSK Ofsetové kvadraturní fázové klíčování;

PER - Packet Error Rate;

PHR - Physical Layer Header;

ping - Packet InterNet Groper;

RAM - Random Acces Memory;

RF - Radio Frequenc;

ROM - Read Only Memory;

RS - Reed-Solomon;

RTS - Ready To Send;

SHR - Synchronization Layer;

SIFS - Short InterFrame Space;

TCP/IP - Transmission Control Protocol / Internet Protocol;

WiFi - Wireless Fidelity;

WiMax - Worldwide Interoperability for Microwave Access;

WPAN - Wireless Personal Area Network;

XOR - Exkluzivní OR;

SEZNAM OBRÁZKŮ A TABULEK

Obr. 1. Rozložení pravděpodobnosti výskytu amplitud [4] 3
Obr. 2. Výdrž alkalické baterie např. v bezpečnostním senzoru [10] 4
Obr. 3. Struktura signálního rámce. [1] 6
Obr. 4. Struktura Řídícího rámce. [1] 7
Obr. 5. Struktura Datového rámce. [1] 7
Obr. 6. Struktura Datového včetně vyšších vrstev. 8
Obr. 7. Struktura Potvrzovacího paketu. [1] 8
Obr. 8. Vývojový diagram CSMA-CA mechanismu. [1] 10
Obr. 9. Příklad struktury superrámce. Zdroj: Vlastní zpracování[1] 11
Obr. 10. Vrstvy ZigBee sítě a uzlové body [5] 12
Obr. 11. Rozložení kanálů v pásmu 2,4 GHz 13
Obr. 12. Blokové schéma zpracování datového toku na fyzické vrstvě. [5] 15
Obr. 13. Topologie sítě „hvězda“ 16
Obr. 14. Topologie sítě „peer to peer“ 16
Obr. 15. DSSS kódování na straně vysílače 17
Obr. 16. Substituce Barkerovým kódem 18
Obr. 17. CCK modulátor 18
Obr. 18. Rozmístění kanálů OFDM [18] 19
Obr. 19. Vliv statického a dynamického rušení na přenos FHSS [20] 21
Obr. 20. Vliv statického a dynamického rušení na přenos AFH [20] 22
Obr. 21. Zisk AFH oproti FHSS [20] 22
Obr. 22. Přenos paketu na standardu 802.11 [50] 24
Obr. 23. Rozložení kanálů 802.11b/g [9] 25
Obr. 24. Vznik interferencí zpožděným příjmem odraženého signálu [13] 28
Obr. 25. Dělení kódovacích metod [28][29]. 36
Obr. 26. Princip blokového prokládání dat [27] 47
Obr. 27. Srovnání účinnosti kódovacích metod s kódovým poměrem cca ½ [24] 49
Obr. 28. Development Kit CC2530 fy Texas Instruments [14] 51
Obr. 29. Funkce „macMemReadRxFifo“ umožňující přístup k datům na nízké úrovni 52
Obr. 30. Sběrné měření paketového provozu na WiFi (procházení cdr.cz) 56
Obr. 31. Rozložení zaznamenaných paketů v čase (procházení cdr.cz) 57
Obr. 32. Sběrné měření paketového provozu na WiFi (Stahování souboru) 58
Obr. 33. Rozložení zaznamenaných paketů v čase (Stahování souboru) 59
Obr. 34. Rozmístění experimentální aparatury 61
Obr. 35. Chybový histogram pro kolizní sílu interferujícího rámce 53 B s rozestupem 0 ms 63
Obr. 36. Chybový histogram pro kolizní sílu interferujícího rámce 9 B s rozestupem 0 ms 63
Obr. 37. Grafické vyjádření Tab. 10 v procentuálním formátu 64
Obr. 38. Distribuce bitových chyb pro sílu koexistujících rámců 53 B s rozestupem 0 ms 65
Obr. 39. Distribuce bitových chyb pro sílu koexistujících rámců 9 B s rozestupem 0 ms 65
Obr. 40. Datový payload v binárním vyjádření, odesilatelem byly odesílány hodnoty 0xff 66
Obr. 41. Rozmístění experimentální aparatury 67
Obr. 42. Počty chyb v paketech pro kolizní sílu 7 B / rozestup 0 ms / vzdálenost 2 m 67
Obr. 43. Počty chyb v paketech pro kolizní sílu 7 B / rozestup 0 ms / vzdálenost 5 m 67
Obr. 44. Počty chyb v paketech pro kolizní sílu 7 B / rozestup 0 ms / vzdálenost 10 m 68
Obr. 45. Grafické vyjádření Tab. 10 měření v procentuálním formátu 68
Obr. 46. Rutina „hamming_buf();“ 71
Obr. 47. Rutina „dehamming_buf();“ 71
Obr. 48. Rutina „generate_interleaved_position();“ 72
Obr. 49. Printscreen obrazovky ručního multimetru DS203 při měření doby dekódování RS(15,9) 74
Obr. 50. Uživatelské prostředí programu pro vyhodnocení účinnosti prokládacích matic 76
Obr. 51. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 7 B .. 76
Obr. 52. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 9 B .. 77
Obr. 53. Vliv prokladače na dekódování Hammingova kódu při koex. s 802.11g a kolizní délkou ping paketů 53 B 77
Obr. 54. Vliv prokladače na dekódování Hammingova kódu při rušení komutátorovým točivým strojem ... 78
Obr. 55. Model RF části v MATLAB Simulink 80
Obr. 56. Skript pro zpracování dat mezi částmi modelu 81

Obr. 57. Uživatelské prostředí programu pro generování ZigBee rámců 82
Obr. 58. Výsledek simulace ZigBee přenosu 84
Obr. 59. Vývojový diagram aplikace odesílající rámce 85
Obr. 60. Vývojový diagram aplikace přijímací rámce 85
Obr. 61. Záznam měření šumových podmínek při měření 86
Obr. 62. Graf PER v závislosti na vzdálenosti 87
Obr. 63. Graf PER v závislosti na SNR 88
Obr. 64. Teoretická spolehlivost sítě při pěti opakováních přenosu 89
Obr. 65. Nákres rozmístění aparatury při měření koexistenčních vlastností 90
Obr. 66. Záznam měření šumových podmínek při měření 90
Obr. 67. Graf PER pro koexistenci s „1 B a 0 ms“ 91
Obr. 68. Graf PER pro koexistenci s „1 B a 1 ms“ 92
Obr. 69. Sloupcové vyjádření výsledků pro koexistenci s „1 B a 0 ms“ 92
Obr. 70. Sloupcové vyjádření výsledků pro koexistenci s „1 B a 1 ms“ 92
Obr. 71. Graf PER pro koexistenci s „100 B a 0 ms“ 93
Obr. 72. Graf PER pro koexistenci s „100 B a 1 ms“ 94
Obr. 73. Sloupcové vyjádření výsledků pro koexistenci s „100 B a 0 ms“ 94
Obr. 74. Sloupcové vyjádření výsledků pro koexistenci s „100 B a 1 ms“ 95
Obr. 75. Graf PER pro koexistenci s „1450 B a 0 ms“ 95
Obr. 76. Graf PER pro koexistenci s „1450 B a 1 ms“ 96
Obr. 77. Sloupcové vyjádření výsledků pro koexistenci s „1450 B a 0 ms“ 96
Obr. 78. Sloupcové vyjádření výsledků pro koexistenci s „1450 B a 1 ms“ 97
Obr. 79. Graf PER pro koexistenci ve vzdálenosti 12 m a rozestupy koexistujících rámců 0 ms. 98
Obr. 80. Graf PER pro koexistenci ve vzdálenosti 12 m a rozestupy koexistujících rámců 1 ms. 98

Tab. 1. Příkazy užívané v Řídícím rámci [1] 6
Tab. 2. Přiřazovací tabulka pseudonáhodných chipových sekvencí. [5] 14
Tab. 3. Přehled pseudonáhodné sekvence FHSS. 20
Tab. 4. Základní parametry skupiny standardů 802.11. [9][11][12] 25
Tab. 5. Rozložení kanálů v pásmu 2,4 GHz pro 902.11b/g [9] 26
Tab. 6. Varianty Hammingova kódu 37
Tab. 7. Přehled BCH kódů do délky slova 511 b 48
Tab. 8. Přehled parametrů kódů pro dopřednou korekci chyb s $R > 0,5$ 50
Tab. 9. Statistika přenosu na páteřní síti mezi Madridem a Paříží 54
Tab. 10. Výsledky měření 62
Tab. 11. Přehled variant implementace metod kódování 69
Tab. 12. Přehled výsledků měření doby zpracování v SoC CC2530 74
Tab. 13. Přehledová tabulka rozměrů rámců pro všechná měření 91

VLASTNÍ PUBLIKACE

[1] MAŠÍK, I. Spolehlivost ZigBee v zemědělském provozu. Agritech Science. 2013, roč. 7, č. 2. ISSN: 1802-8942

[2] MAŠÍK, I. Metody pro zlepšení koexistenčních vlastností ZigBee standardu. Agritech Science. 2013, roč. 7, č. 3. ISSN: 1802-8942

[3] MAŠÍK, I. Dopředná korekce chyb nad MAC vrstvou pro standard ZigBee. ElektroRevue. 2013, roč. 15, č. 2. ISSN: 1213-1539

[4] MAŠÍK, I. Reliability of ZigBee transmission in agriculture production. Research in Agricultural Engeneering. 2013, Vol. 60, ISSN: 1805-9376

[5] MAŠÍK, I., BOHUSLÁVEK, Z. Metody pro zvýšení spolehlivosti a bezpečnosti komunikace na bezdrátových sítích. 2009. SPU v Nitre 2009, In: Informačné a automatizačné technológie v riadení.

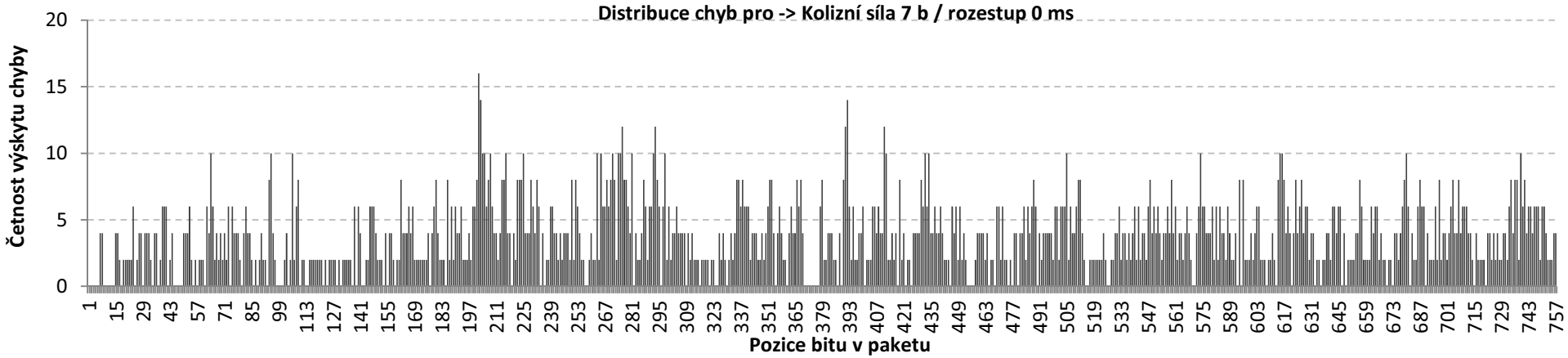
[6] LINDA, M., KÜNZEL, G., MAŠÍK, I. Měření rychlých průběhů teploty termočlánky. 2010. ISBN: 1335-2555; Místo vydání: Nitra, SK; Počet stran: 4; Název nakladatele: Slovenská polnohospodarska univerzita v Nitre; Konference: Acta technologica agriculturae; Datum zahájení: 20.9.2010; Strana 45 - 48.

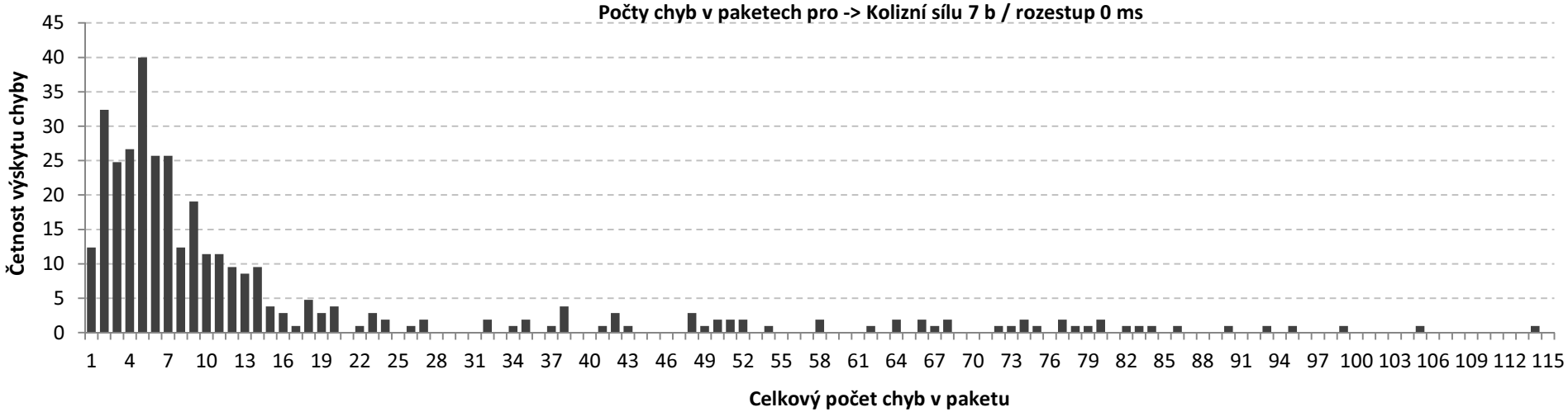
[7] LINDA, M., HROMASOVÁ, M., MAŠÍK, I. Dynamic temperature changes at resistor measured by thermocouples. 2010. UCOLIS 2010; ISBN 978-80-213-2141-0; strana 382 – 387.

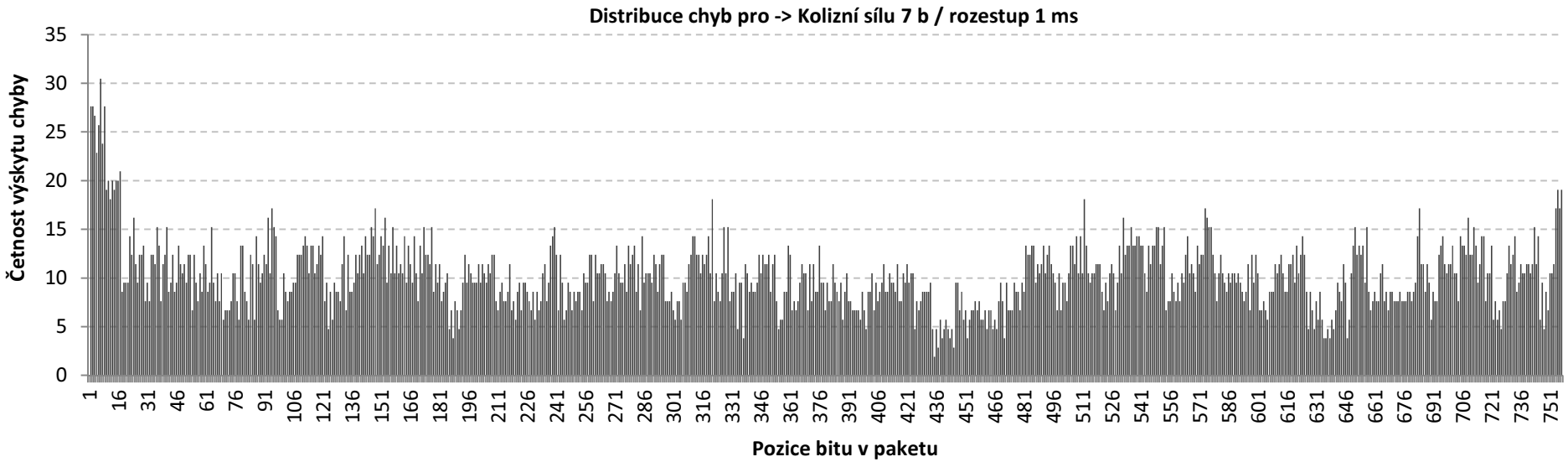
PŘÍLOHY

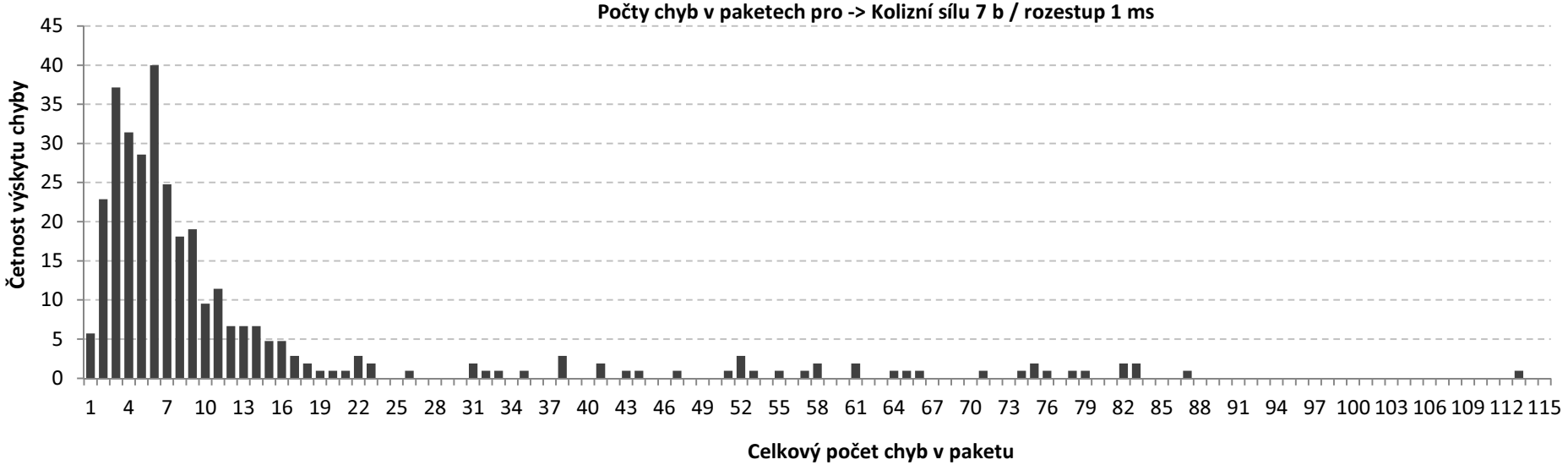
I. PŘÍLOHA – GRAFICKÉ VÝSTUPY K MĚŘENÍ V BODĚ 5.3.3.1 1
II. PŘÍLOHA – GRAFICKÉ VÝSTUPY K MĚŘENÍ V BODĚ 5.3.3.2 15
III. RUTINY SPOJENÉ S HAMMINGOVÝM KÓDOVÁNÍM A BLOKOVÝM PROKLADÁNÍM 17
IV. ZDROJOVÝ KÓD PROGRAMU PRO ANALÝZU DAT ZE ZIGBEE TRANSCEIVERŮ. 21
V. VÝSLEDKY ANALÝZY DAT PŘI VOLBĚ ROZMĚRU PROKLÁDACÍ MATICE 28
VI. SOFTWARE PROGRAMU PRO SIMULACI ZIGBEE PROTOKOLU 29
VII. VÝSLEDKY MĚŘENÍ PŘI KOEXISTENCI S WIFI 802.11G 39

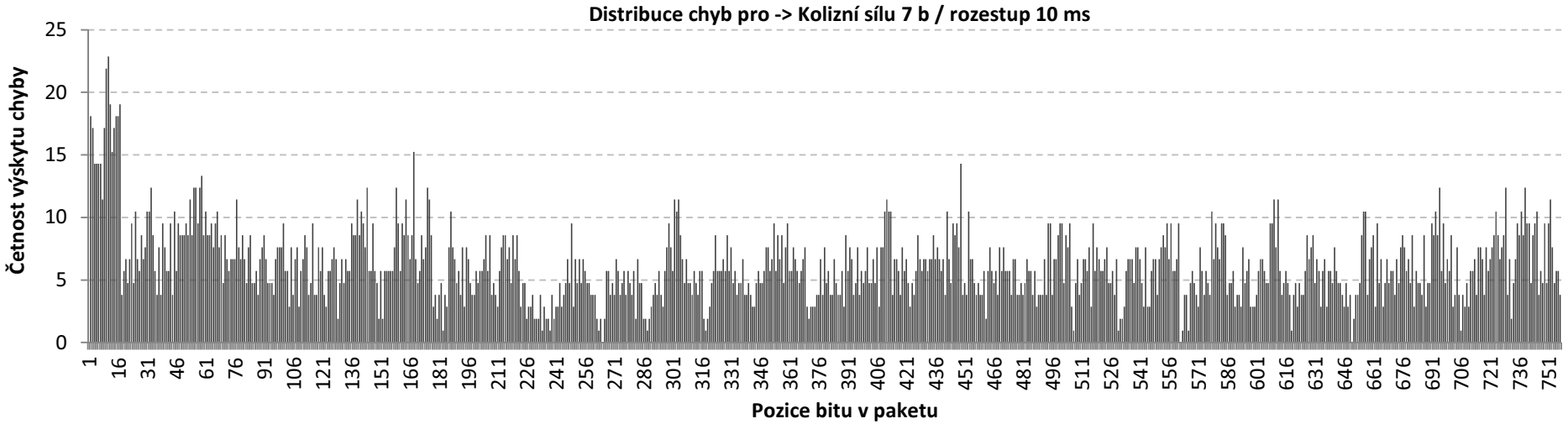
I. Příloha – Grafické výstupy k měření v bodě 5.3.3.1

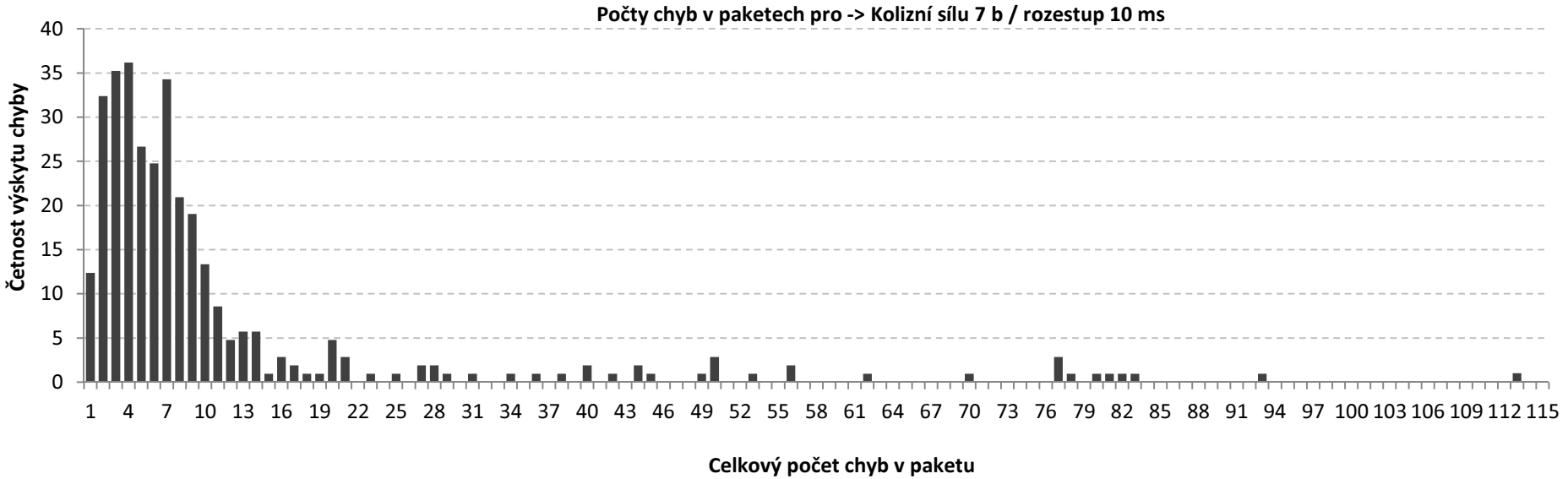


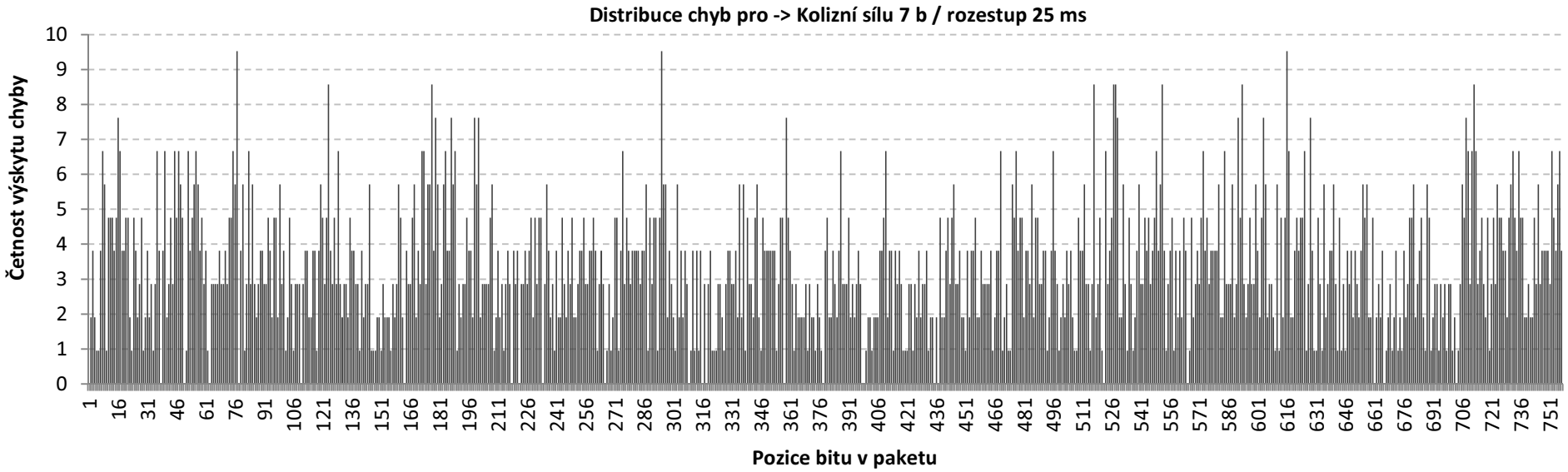


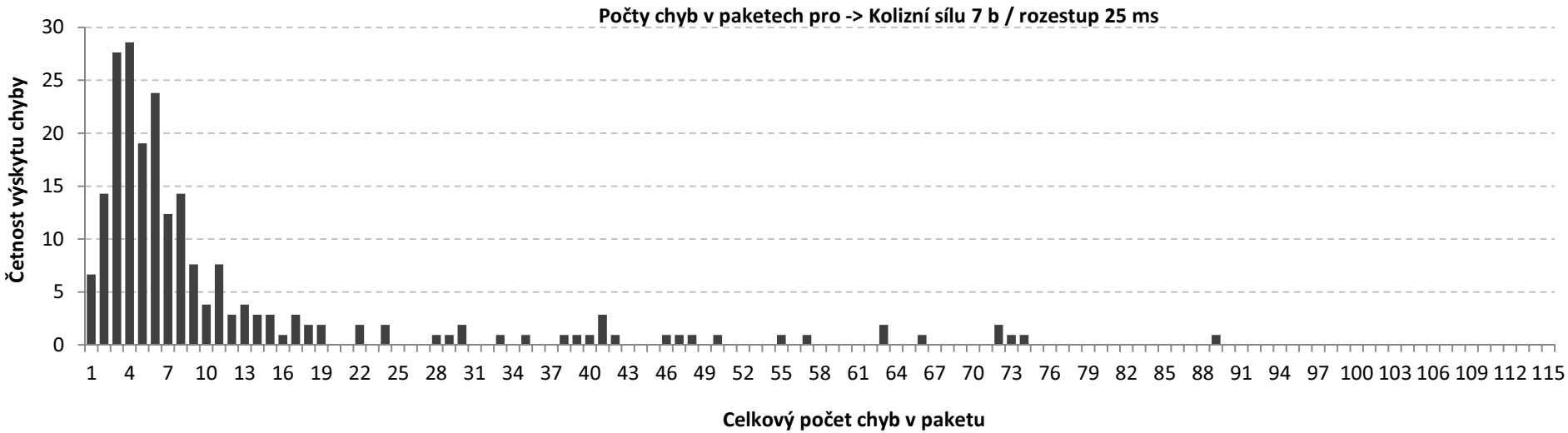

Distribuce chyb pro -> Kolizní sílu 7 b / rozestup 1 ms
Četnost výskytu chyby
0
5
10
15
20
25
30
35
1
16
31
46
61
76
91
106
121
136
151
166
181
196
211
226
241
256
271
286
301
316
331
346
361
376
391
406
421
436
451
466
481
496
511
526
541
556
571
586
601
616
631
646
661
676
691
706
721
736
751
Pozice bitu v paketu

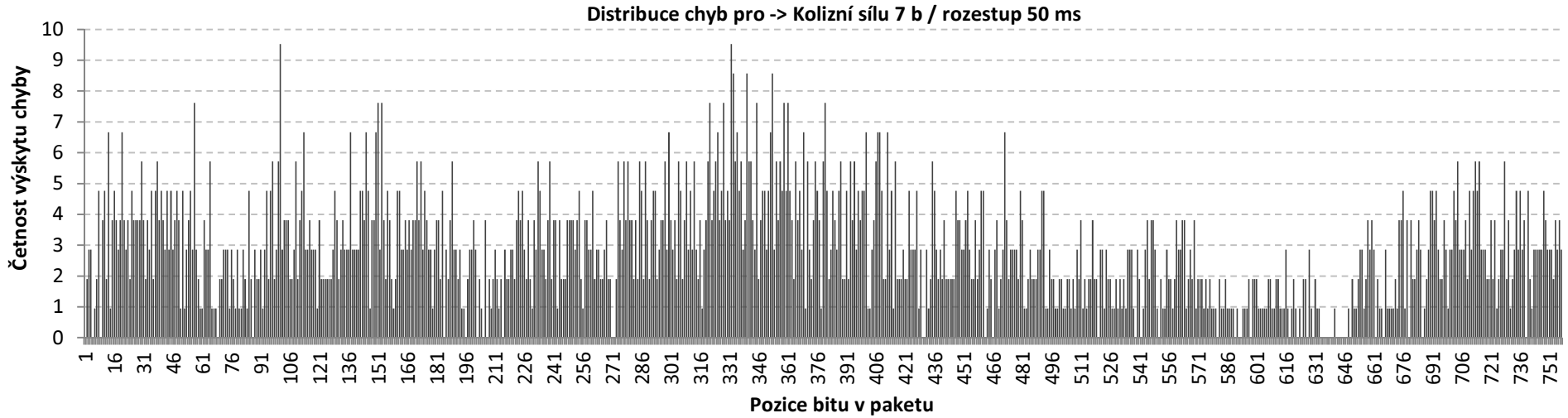

Počty chyb v paketech pro -> Kolizní sílu 7 b / rozestup 1 ms
Četnost výskytu chyby
0
5
10
15
20
25
30
35
40
45
1
4
7
10
13
16
19
22
25
28
31
34
37
40
43
46
49
52
55
58
61
64
67
70
73
76
79
82
85
88
91
94
97
100
103
106
109
112
115
Celkový počet chyb v paketu

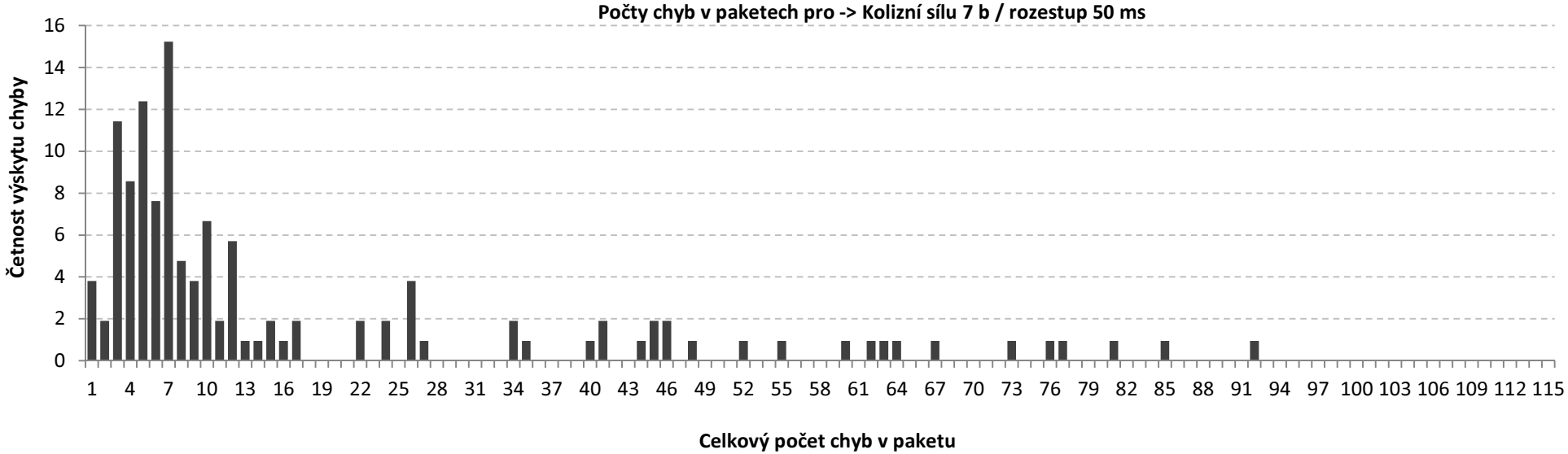

Distribuce chyb pro -> Kolizní sílu 7 b / rozestup 10 ms
Četnost výskytu chyby
0
5
10
15
20
25
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256 271 286 301 316 331 346 361 376 391 406 421 436 451 466 481 496 511 526 541 556 571 586 601 616 631 646 661 676 691 706 721 736 751
Pozice bitu v paketu

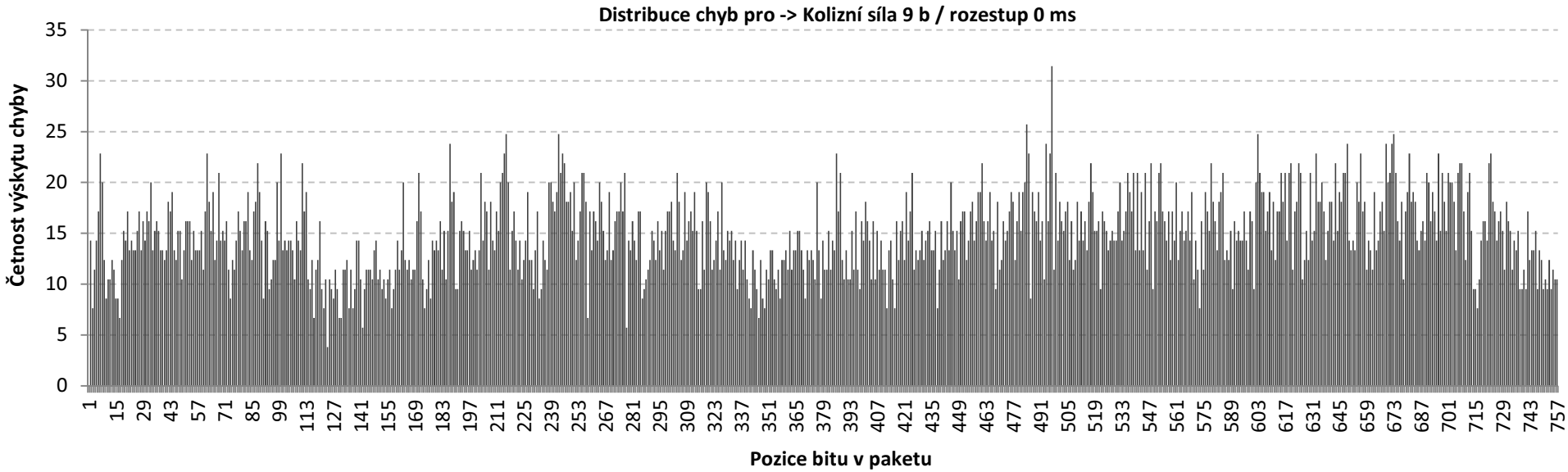

Počty chyb v paketech pro -> Kolizní sílu 7 b / rozestup 10 ms
Četnost výskytu chyby
0
5
10
15
20
25
30
35
40
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115
Celkový počet chyb v paketu

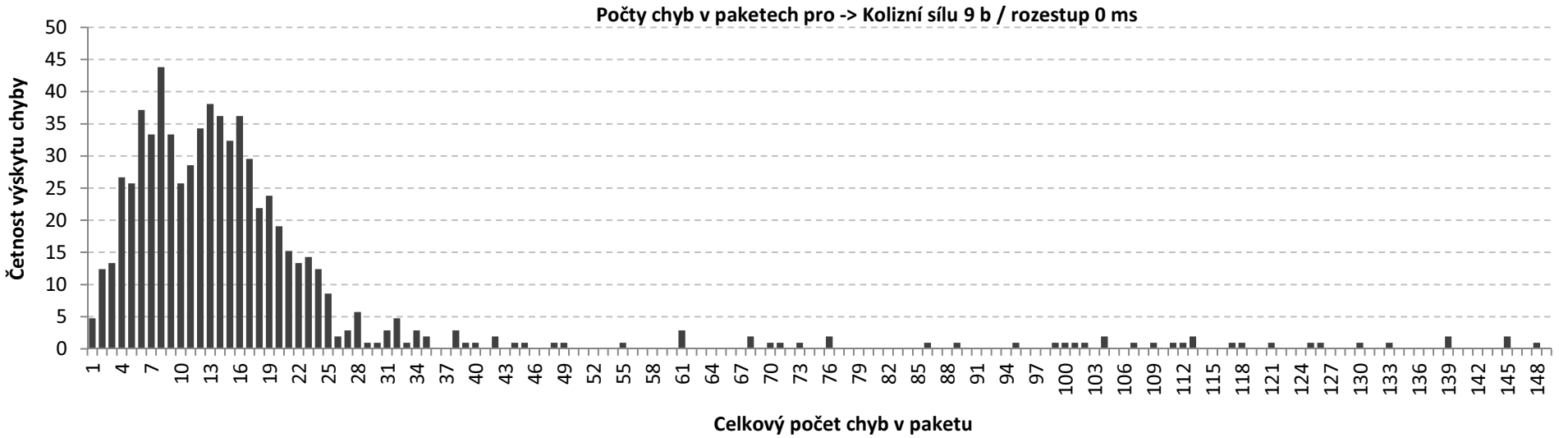

Distribuce chyb pro -> Kolizní sílu 7 b / rozestup 25 ms
Četnost výskytu chyby
0 1 2 3 4 5 6 7 8 9 10
1 16 31 46 61 76 91 106 121 136 151 166 181 196 211 226 241 256 271 286 301 316 331 346 361 376 391 406 421 436 451 466 481 496 511 526 541 556 571 586 601 616 631 646 661 676 691 706 721 736 751
Pozice bitu v paketu

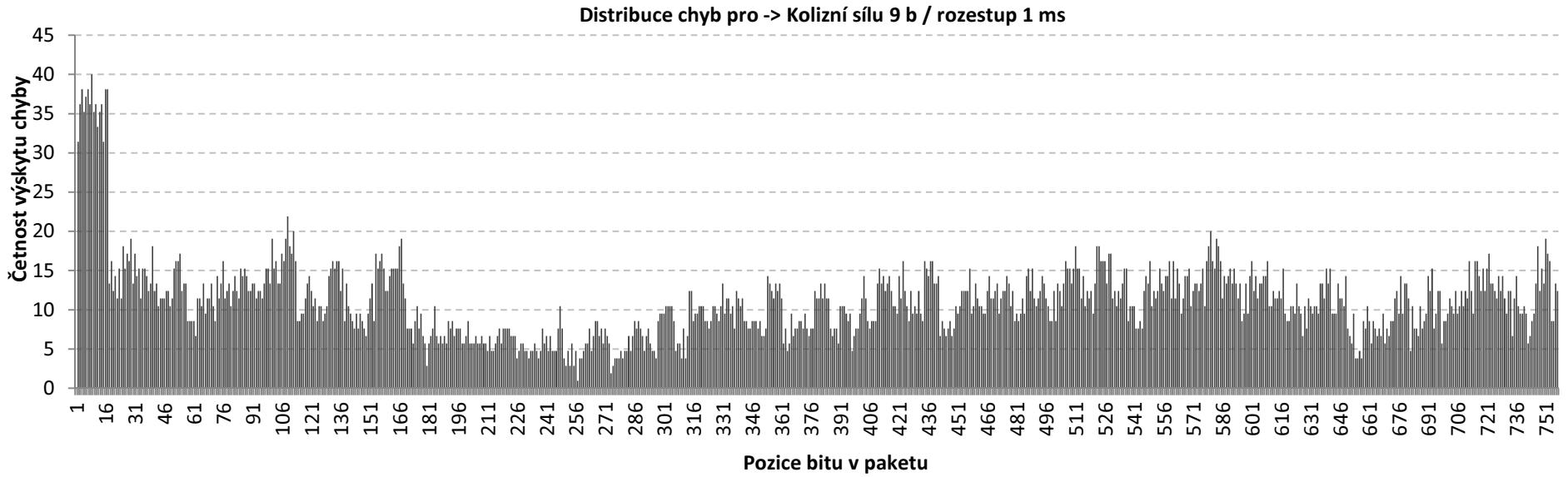

Počty chyb v paketech pro -> Kolizní sílu 7 b / rozestup 25 ms
Četnost výskytu chyby
0 5 10 15 20 25 30
1 4 7 10 13 16 19 22 25 28 31 34 37 40 43 46 49 52 55 58 61 64 67 70 73 76 79 82 85 88 91 94 97 100 103 106 109 112 115
Celkový počet chyb v paketu

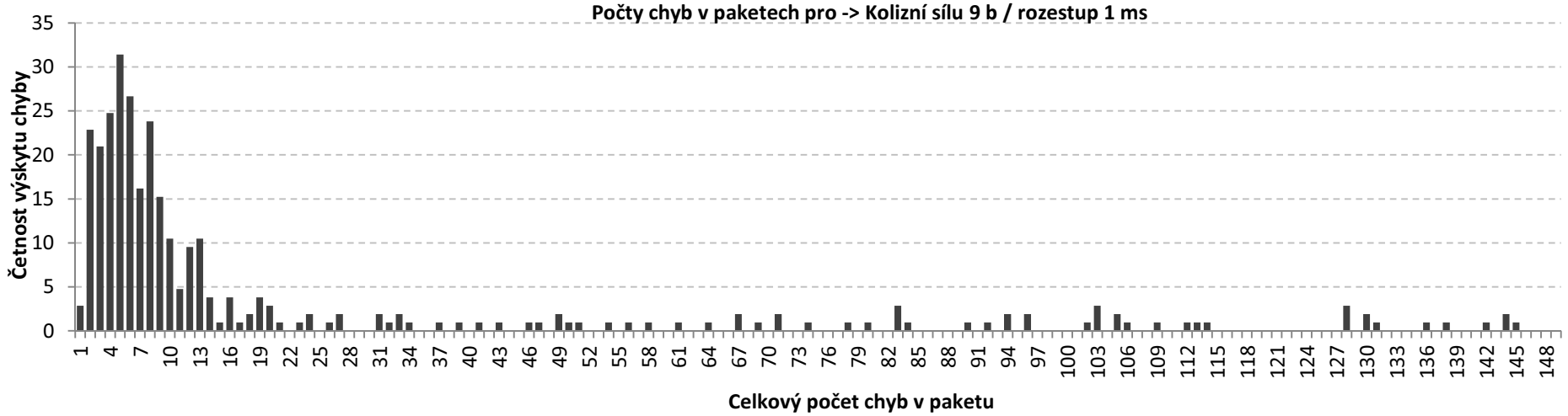

Distribuce chyb pro -> Kolizní sílu 7 b / rozestup 50 ms
Četnost výskytu chyby
Pozice bitu v paketu

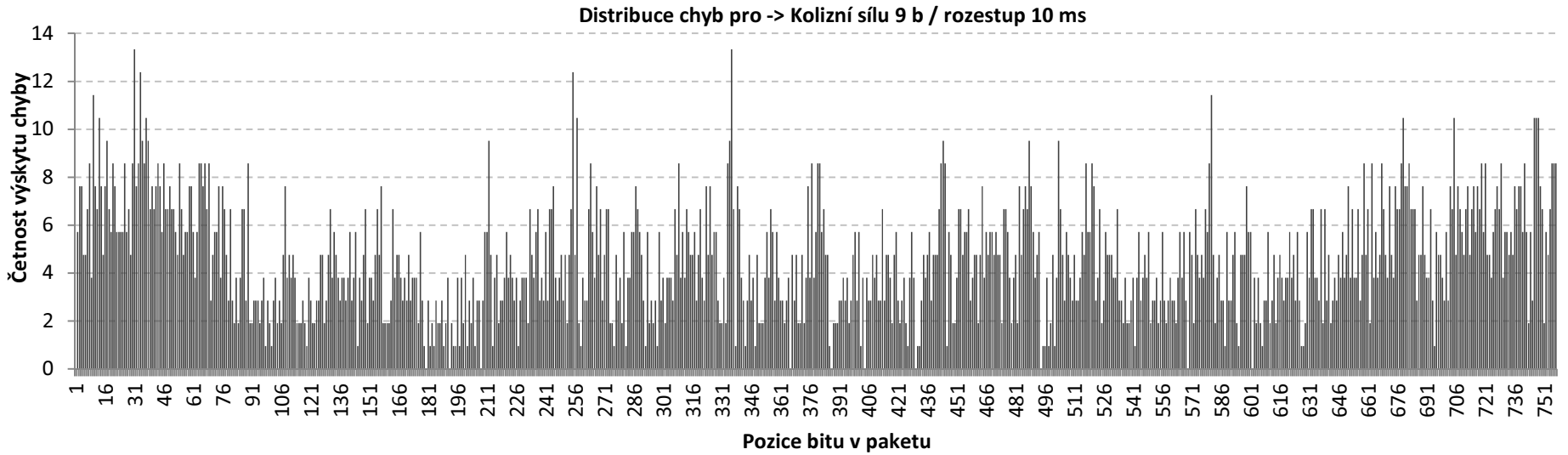

Počty chyb v paketech pro -> Kolizní sílu 7 b / rozestup 50 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

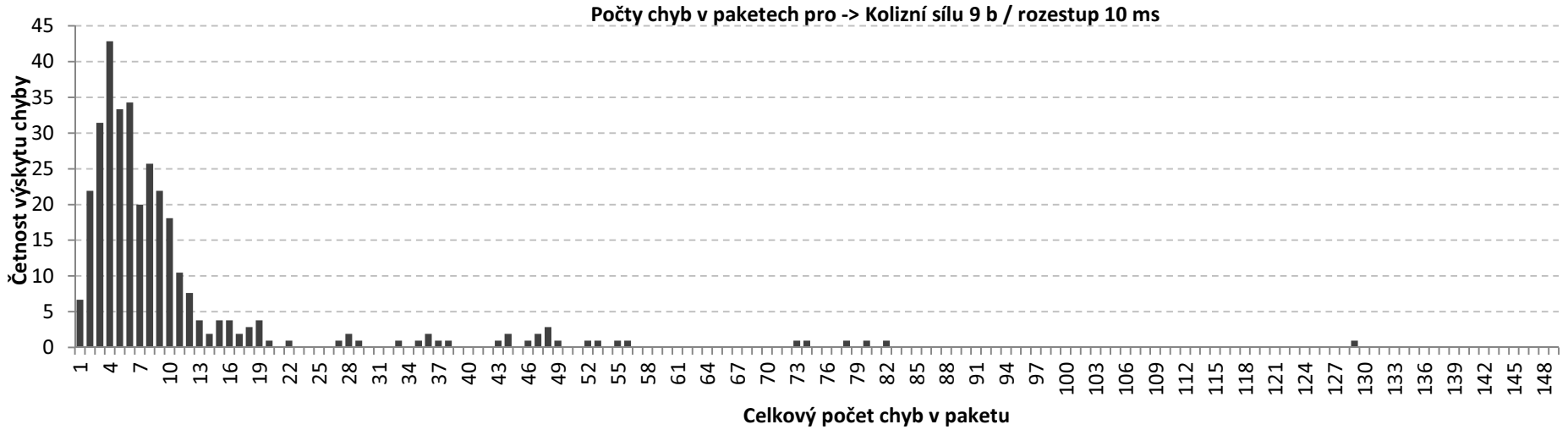

Distribuce chyb pro -> Kolizní síla 9 b / rozestup 0 ms
Četnost výskytu chyby
Pozice bitu v paketu

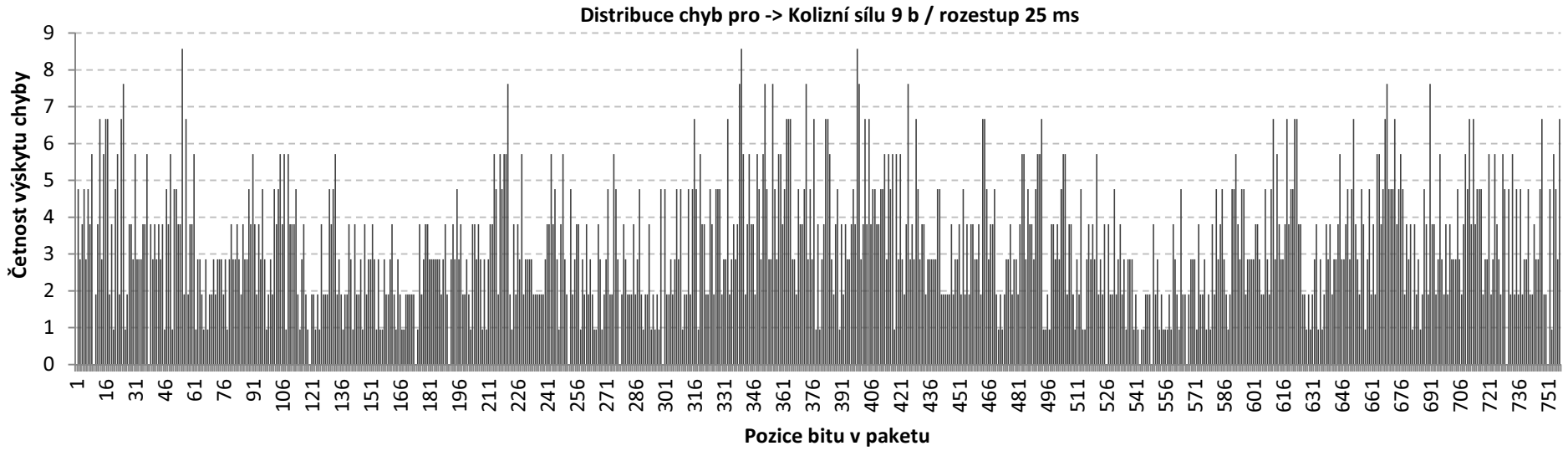

Počty chyb v paketech pro -> Kolizní sílu 9 b / rozestup 0 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

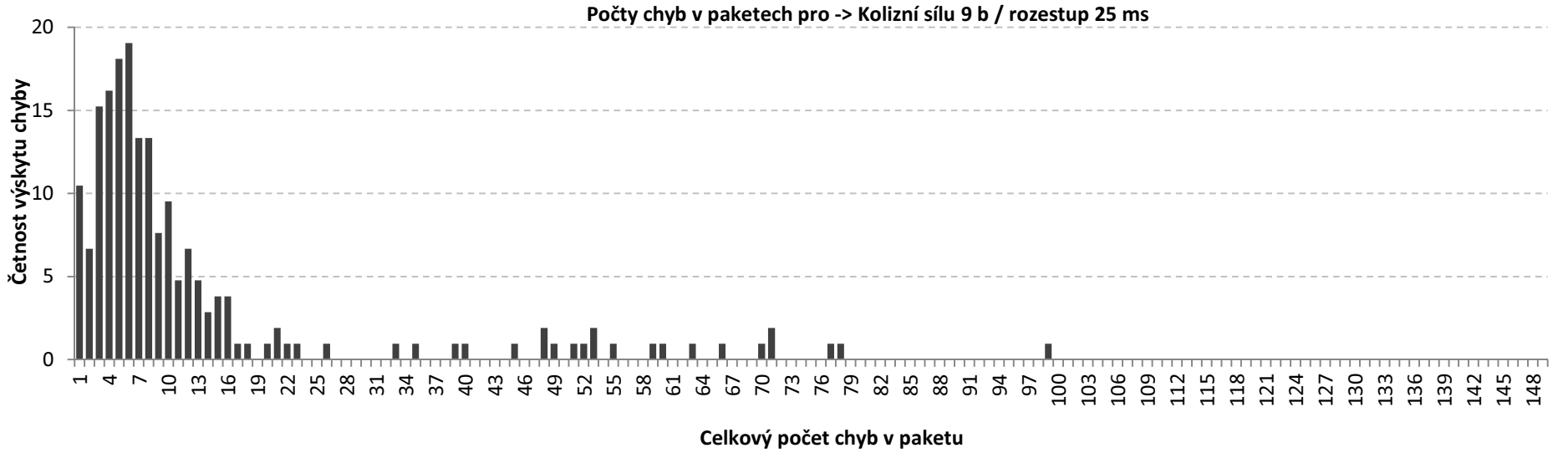

Distribuce chyb pro -> Kolizní sílu 9 b / rozestup 1 ms
Četnost výskytu chyby
Pozice bitu v paketu

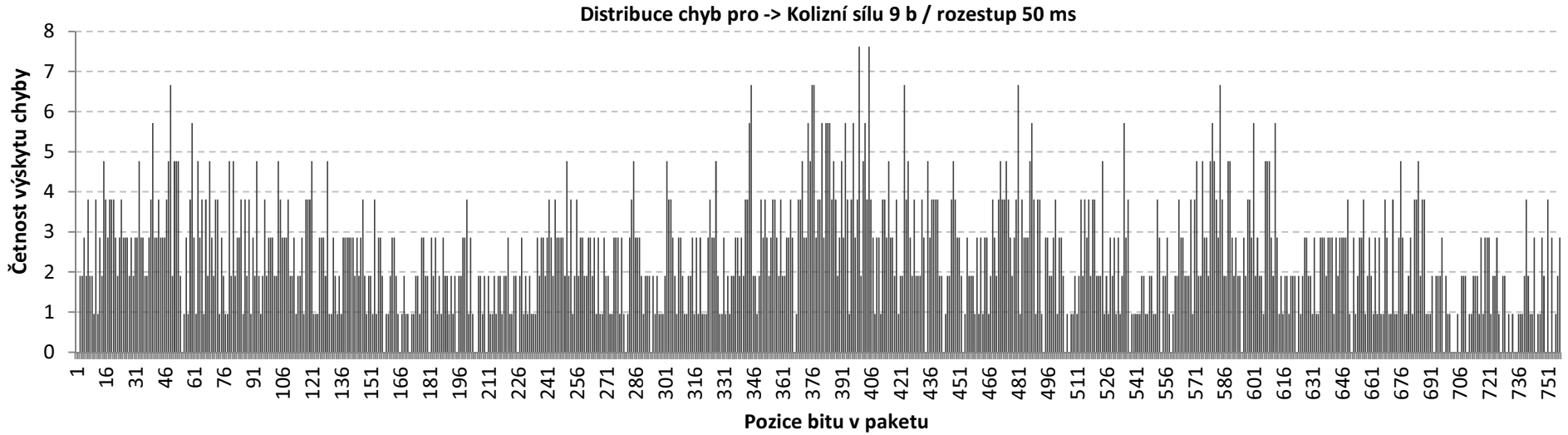

Počty chyb v paketech pro -> Kolizní sílu 9 b / rozestup 1 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

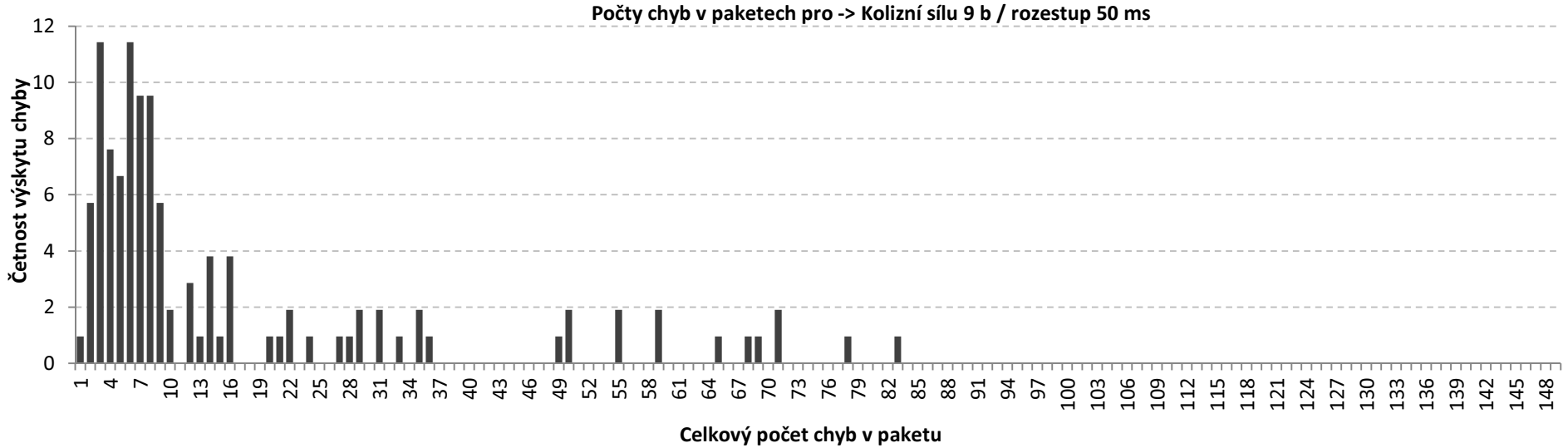

Distribuce chyb pro -> Kolizní sílu 9 b / rozestup 10 ms
Četnost výskytu chyby
Pozice bitu v paketu

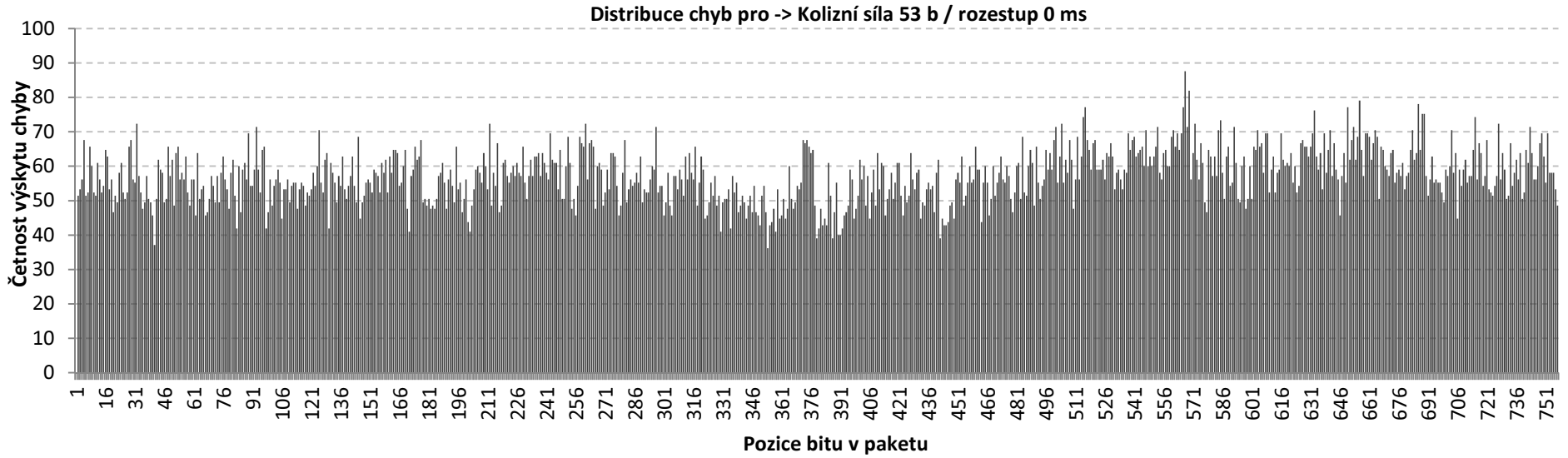

Počty chyb v paketech pro -> Kolizní sílu 9 b / rozestup 10 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

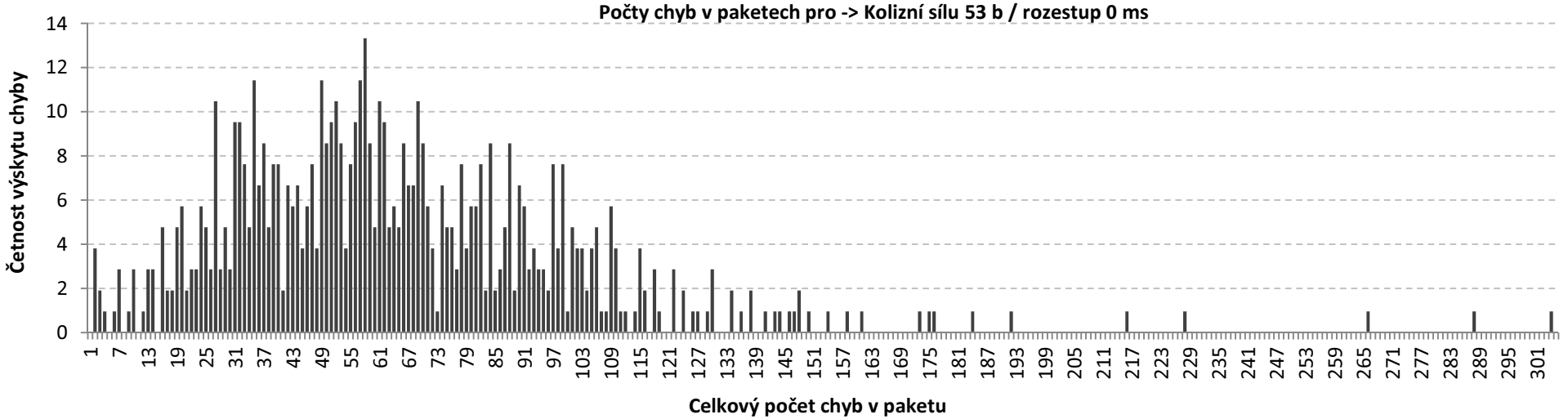

Distribuce chyb pro -> Kolizní sílu 9 b / rozestup 25 ms
Četnost výskytu chyby
Pozice bitu v paketu

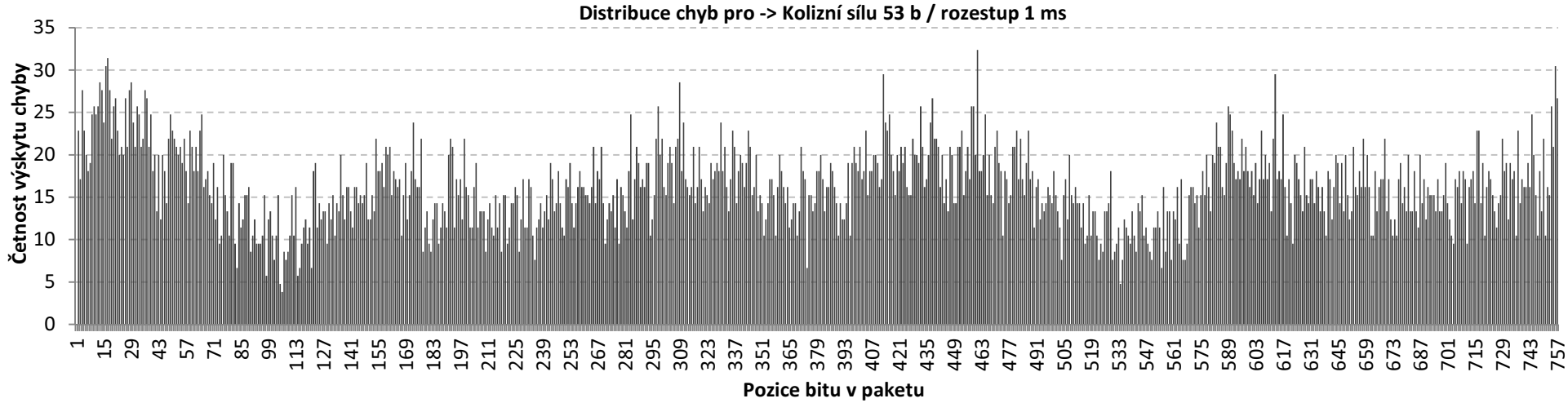

Počty chyb v paketech pro -> Kolizní sílu 9 b / rozestup 25 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

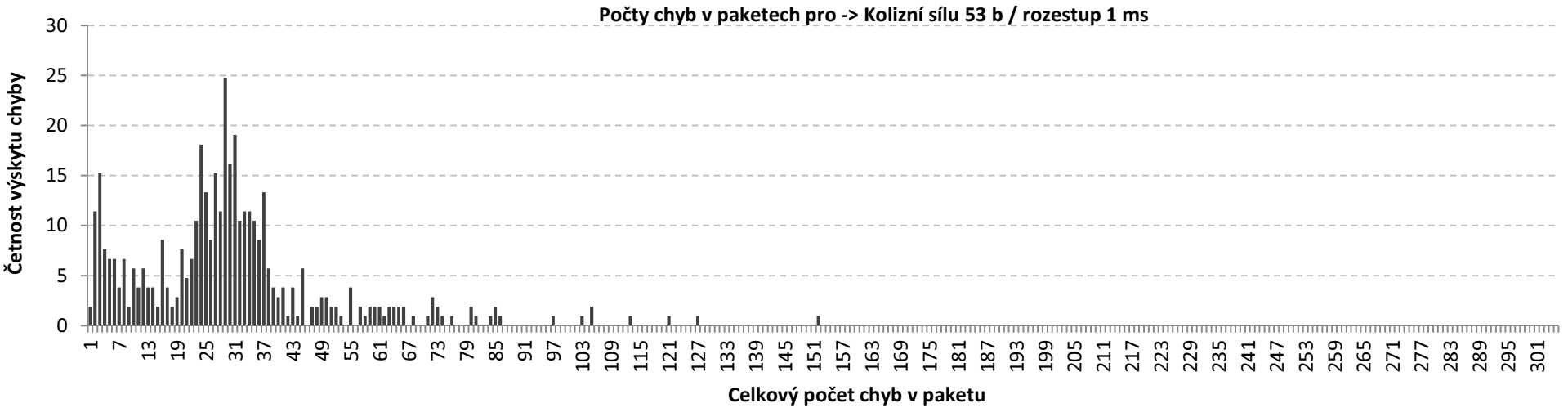

Distribuce chyb pro -> Kolizní sílu 9 b / rozestup 50 ms
Četnost výskytu chyby
Pozice bitu v paketu

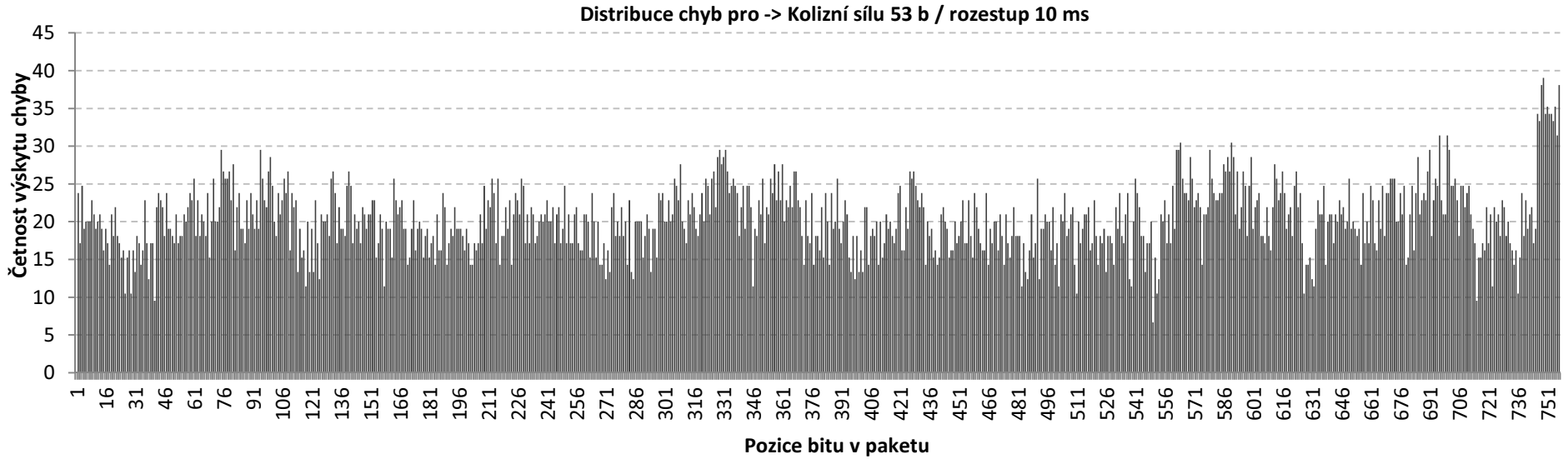

Počty chyb v paketech pro -> Kolizní sílu 9 b / rozestup 50 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

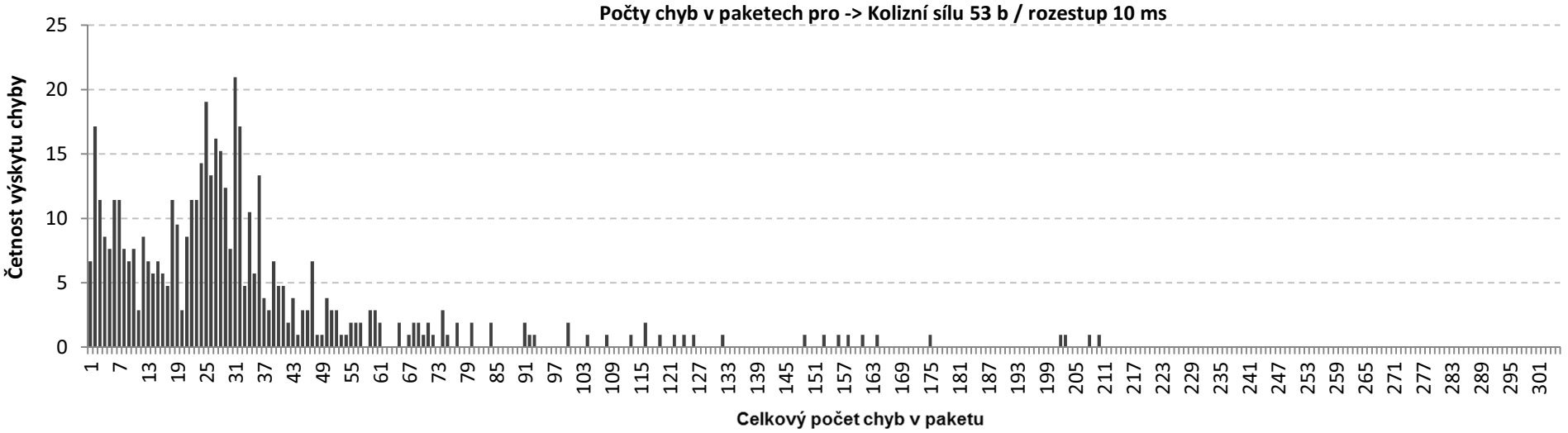

Distribuce chyb pro -> Kolizní síla 53 b / rozestup 0 ms
Četnost výskytu chyby
Pozice bitu v paketu

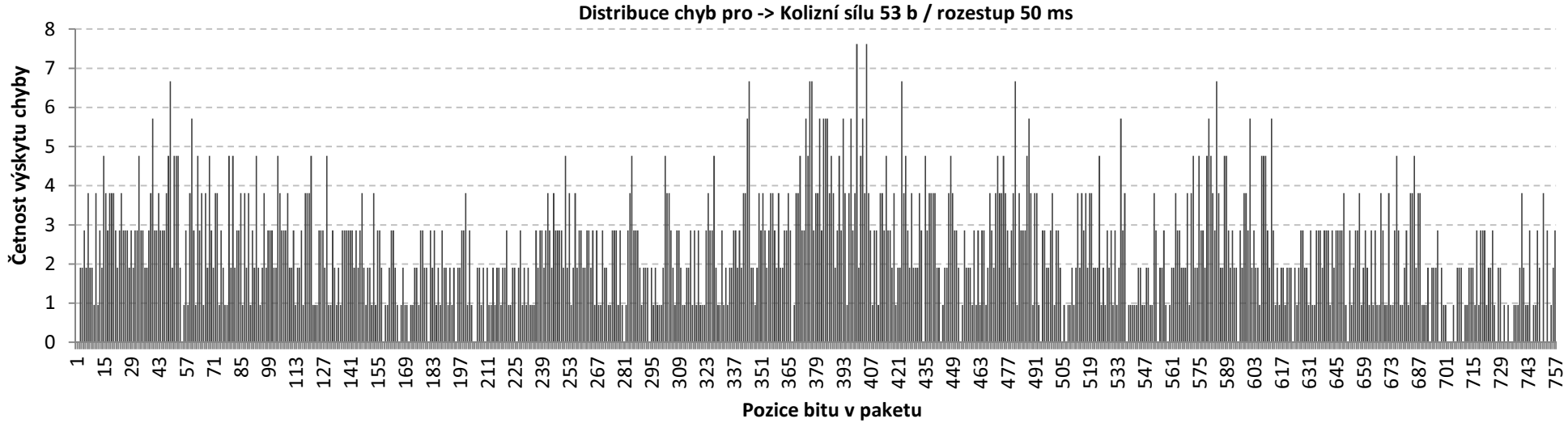

Počty chyb v paketech pro -> Kolizní sílu 53 b / rozestup 0 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

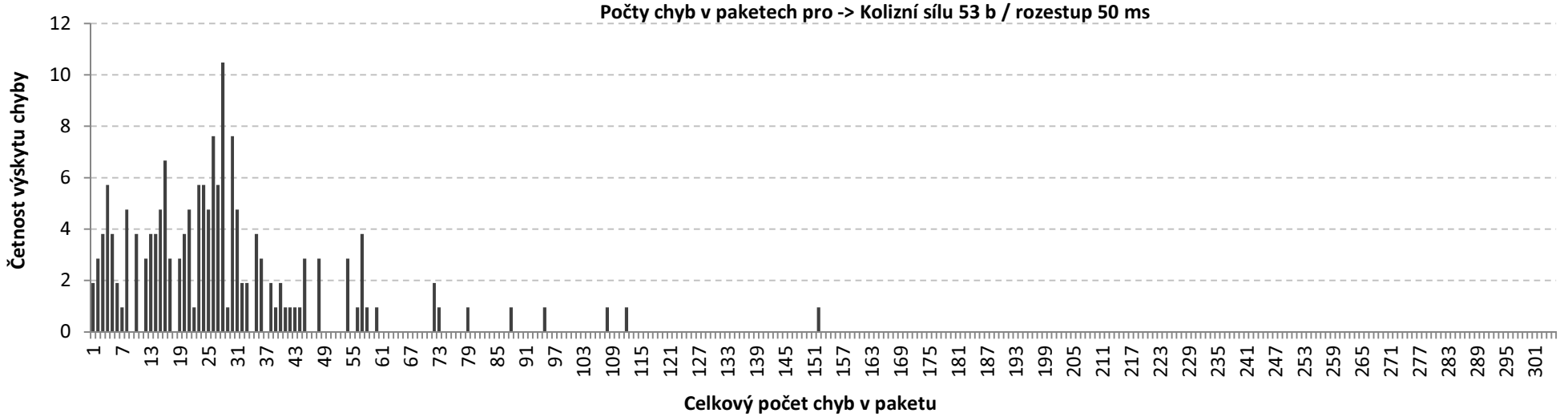

Distribuce chyb pro -> Kolizní sílu 53 b / rozestup 1 ms
Četnost výskytu chyby
Pozice bitu v paketu


Počty chyb v paketech pro -> Kolizní sílu 53 b / rozestup 1 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

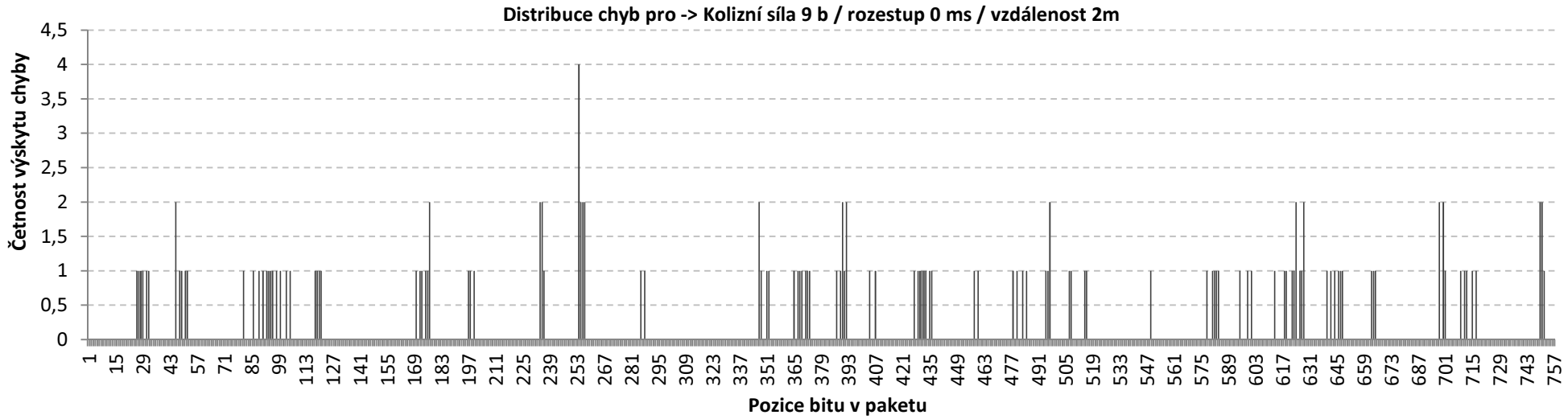

Distribuce chyb pro -> Kolizní sílu 53 b / rozestup 10 ms
Četnost výskytu chyby
Pozice bitu v paketu

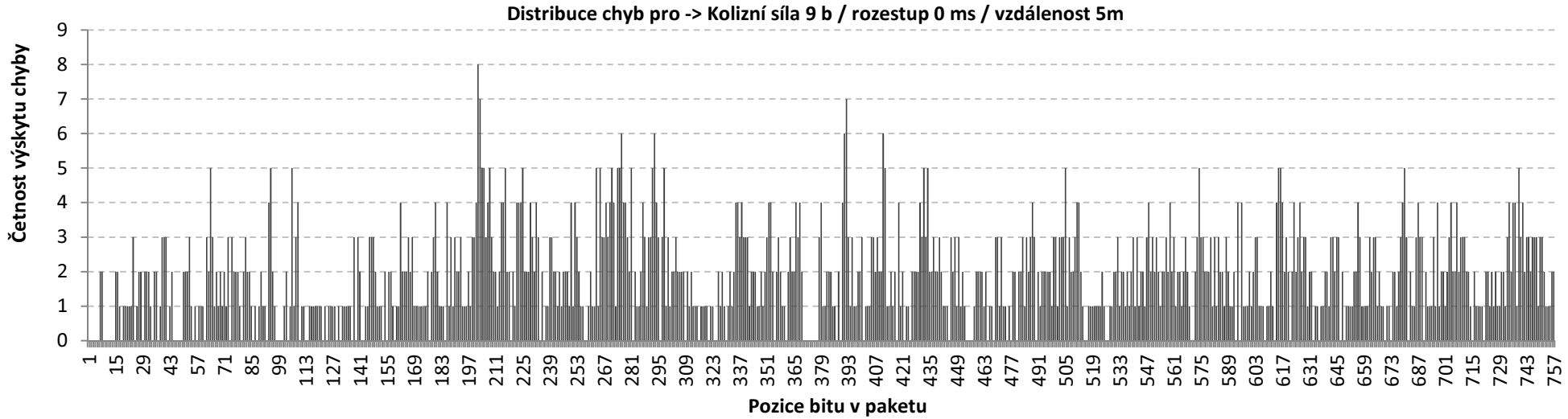

Počty chyb v paketech pro -> Kolizní sílu 53 b / rozestup 10 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

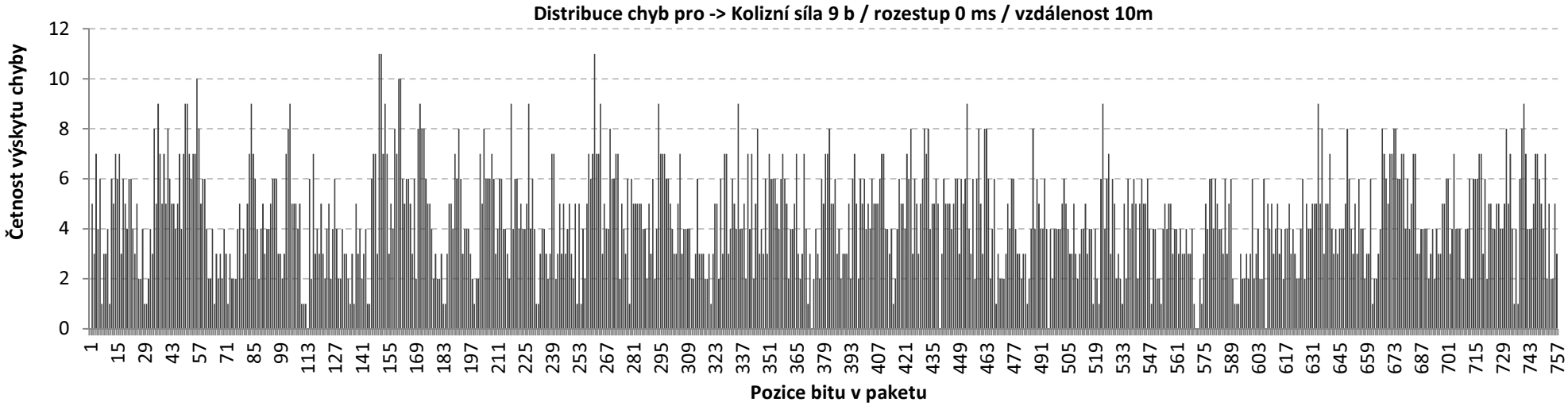

Distribuce chyb pro -> Kolizní sílu 53 b / rozestup 50 ms
Četnost výskytu chyby
Pozice bitu v paketu


Počty chyb v paketech pro -> Kolizní sílu 53 b / rozestup 50 ms
Četnost výskytu chyby
Celkový počet chyb v paketu

II. Příloha – Grafické výstupy k měření v bodě 5.3.3.2






Distribuce chyb pro -> Kolizní síla 9 b / rozestup 0 ms / vzdálenost 10m
Četnost výskytu chyby
0
2
4
6
8
10
12
1
15
29
43
57
71
85
99
113
127
141
155
169
183
197
211
225
239
253
267
281
295
309
323
337
351
365
379
393
407
421
435
449
463
477
491
505
519
533
547
561
575
589
603
617
631
645
659
673
687
701
715
729
743
757
Pozice bitu v paketu

III. Rutiny spojené s Hammingovým kódováním a Blokovým prokladáním

```
void decode_bin_buf (void){
uint8 i;
uint8 target_byte=0;

 for(i=0; i<95; i++){Xbuf_tmp[i]=0;}
  bit_cursor = 7;
  byte_cursor = 0;

  for (i=0; i < 54; i++){

    H_data[0]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[1]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[2]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[3]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[4]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[5]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[6]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();

      dehamming_bin_buf ();

    if(H_data[3]){Set_Bit_buf(target_byte, 0);}
    if(H_data[4]){Set_Bit_buf(target_byte, 1);}
    if(H_data[5]){Set_Bit_buf(target_byte, 2);}
    if(H_data[6]){Set_Bit_buf(target_byte, 3);}

    H_data[0]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[1]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[2]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[3]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[4]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[5]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();
    H_data[6]=Read_Bit_buf(byte_cursor, bit_cursor);
      Generate_sequential_position ();

      dehamming_bin_buf ();

    if(H_data[3]){Set_Bit_buf(target_byte, 4);}
    if(H_data[4]){Set_Bit_buf(target_byte, 5);}
    if(H_data[5]){Set_Bit_buf(target_byte, 6);}
    if(H_data[6]){Set_Bit_buf(target_byte, 7);}
    target_byte++;
  }
}
//*********************************************************************
void dehamming_bin_buf (void){
uint8 i, j, syndrom[3];

    for(i=0;i<3;i++){
      syndrom[i]=0;
        for(j=0;j<7;j++) {syndrom[i]+=(H_data[j]*hmatrix[i][j]);}
      syndrom[i]=syndrom[i]%2;
    }
```

```
   for(j=0;j<7;j++){
     if((syndrom[0]==hmatrix[0][j]) && (syndrom[1]==hmatrix[1][j]) && (syndrom[2]==hmatrix[2][j])){break;}
   }
   if(j != 7){H_data[j]=!H_data[j];
   }
}
//*********************************************************************
void code_bin_buf (void){
uint8 i;
uint8 bits[4];
 for(i=0; i<95; i++){Xbuf_tmp[i]=0;}
 bit_cursor = 7;
 byte_cursor = 0;

 for (i=0; i < 54; i++){
     bits[0] = Xbuf[i] & 0x01  ?  1 : 0;
     bits[1] = Xbuf[i] & 0x02  ?  1 : 0;
     bits[2] = Xbuf[i] & 0x04  ?  1 : 0;
     bits[3] = Xbuf[i] & 0x08  ?  1 : 0;
     hamming_bin_buf(bits[0],bits[1],bits[2],bits[3]);

     bits[0] = Xbuf[i] & 0x10  ?  1 : 0;
     bits[1] = Xbuf[i] & 0x20  ?  1 : 0;
     bits[2] = Xbuf[i] & 0x40  ?  1 : 0;
     bits[3] = Xbuf[i] & 0x80  ?  1 : 0;
     hamming_bin_buf(bits[0],bits[1],bits[2],bits[3]);
 }
}
//*********************************************************************
void hamming_bin_buf (uint8 bit1, uint8 bit2, uint8 bit3, uint8 bit4){
uint8 i, j, coded[7], data[4];

data[0]=bit1; data[1]=bit2; data[2]=bit3; data[3]=bit4;

   for(i=0;i<7;i++){
     coded[i]=0;
     for(j=0;j<4;j++){coded[i]+= (data[j] * gmatrix[j][i]);}
     if(1==coded[i]%2){Set_Bit_buf(byte_cursor, bit_cursor);}
     Generate_sequential_position ();
   }
}
//*********************************************************************
void interleave_bin_buf (void){
uint8 i;
bit_cursor = 0;
byte_cursor = 0;
  for (i=0; i < 94; i++){

    Write_interleaved_bit(Xbuf[i] & 0x01  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x02  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x04  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x08  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x10  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x20  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x40  ?  1 : 0);
    Write_interleaved_bit(Xbuf[i] & 0x80  ?  1 : 0);
  }
    Write_interleaved_bit(Xbuf[94] & 0x10  ?  1 : 0);
    Write_interleaved_bit(Xbuf[94] & 0x20  ?  1 : 0);
    Write_interleaved_bit(Xbuf[94] & 0x40  ?  1 : 0);
    Write_interleaved_bit(Xbuf[94] & 0x80  ?  1 : 0);
}
//*********************************************************************

void Write_interleaved_bit (uint8 value){
 uint8 tmp=0;

 byte_cursor = bit_cursor >> 3;    //deleno 8
 tmp = bit_cursor %8;              //zbytek po deleni 8

 if(value==1){Set_Bit_buf(byte_cursor, tmp);}

 if(bit_cursor==755){bit_cursor = 0; byte_cursor = 0; line_cnt = 1;}
```

```
  bit_cursor = bit_cursor + intlvr_size;
  if(bit_cursor>755){bit_cursor=line_cnt++;}

}
//*********************************************************************
uint8 Generate_interleaved_position (){
uint8 tmp=0;
  byte_cursor = bit_cursor >> 3;    //deleno 8
  tmp = bit_cursor %8;              //zbytek po deleni 8
  if(bit_cursor==755){bit_cursor = 0; byte_cursor = 0; line_cnt = 1;}
  bit_cursor = bit_cursor + intlvr_size;
  if(bit_cursor>755){bit_cursor=line_cnt++;}

  return(tmp);
}
//*********************************************************************
void Generate_sequential_position (){
    bit_cursor--;

    if(bit_cursor > 100){
      bit_cursor = 7;
      byte_cursor ++;
    }}
//*********************************************************************
void Set_Bit_buf(uint8 byte, uint8 bit){
  switch(bit){
  case 0:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x01;
    break;
  case 1:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x02;
    break;
  case 2:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x04;
    break;
  case 3:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x08;
    break;
  case 4:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x10;
    break;
  case 5:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x20;
    break;
  case 6:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x40;
    break;
  case 7:
    Xbuf_tmp[byte] = Xbuf_tmp[byte] | 0x80;
    break;
  }
}
//*********************************************************************
uint8 Read_Bit_buf(uint8 byte, uint8 bit){
uint8 tmp;
  switch(bit){
  case 0:
    tmp = Xbuf[byte] & 0x01  ?  1 : 0;
    break;
  case 1:
    tmp = Xbuf[byte] & 0x02  ?  1 : 0;
    break;
  case 2:
    tmp = Xbuf[byte] & 0x04  ?  1 : 0;
    break;
  case 3:
    tmp = Xbuf[byte] & 0x08  ?  1 : 0;
    break;
  case 4:
    tmp = Xbuf[byte] & 0x10  ?  1 : 0;
    break;
  case 5:
    tmp = Xbuf[byte] & 0x20  ?  1 : 0;
    break;
```

```
  case 6:
    tmp = Xbuf[byte] & 0x40  ?  1 : 0;
    break;
  case 7:
    tmp = Xbuf[byte] & 0x80  ?  1 : 0;
    break;
  }
  return(tmp);}
//*********************************************************************
void deinterleave_bin_buf (void){
uint8 i;
bit_cursor = 0;
byte_cursor = 0;
line_cnt = 1;
   for (i=0; i < 94; i++){

     Xbuf_tmp[i] = 0;
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x01;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x02;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x04;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x08;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x10;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x20;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x40;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[i] = Xbuf_tmp[i] | 0x80;}
   }

    Xbuf_tmp[94] = 0;//nejprve nulovani cilove pozice
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[94] = Xbuf_tmp[94] | 0x10;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[94] = Xbuf_tmp[94] | 0x20;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[94] = Xbuf_tmp[94] | 0x40;}
    if(Read_Bit_buf(byte_cursor, Generate_interleaved_position ())) {Xbuf_tmp[94] = Xbuf_tmp[94] | 0x80;}
}
//*********************************************************************
```

IV. Zdrojový kód programu pro analýzu dat ze ZigBee transceiverů.

```
Dim active, BINpacket, ones, zeros, zero_seq, 0, 00, 000, 0000, 00000, 000000, 0000000
Dim SbitL1, SbitL2, SbitL3, SbitL4, SbitL5, SbitL6, SbitL7
Dim sync, sync_last
Dim filename1, filename2, bin_array(), bin_input(), ErrStat(756, 1)

Private Sub Command1_Click()

MSComm1.RThreshold = 1
MSComm1.InputLen = 1
MSComm1.Settings = "38400,N,8,1"
MSComm1.DTREnable = False
MSComm1.Handshaking = comRTSXOnXOff
MSComm1.CommPort = Combo1.ListIndex
MSComm1.PortOpen = True


For i = 0 To 1000
filename1 = "C:\RAWbinary"
filename1 = filename1 & i
filename1 = filename1 & ".txt"

  If Dir(filename1) = "" Then
  Open filename1 For Output As #1
  Exit For
  End If
Next i

End Sub

Private Sub Command2_Click()
Dim errs
errs = 0
CLR_mem
CommonDialog1.Filter = "Apps (*.txt)|*.txt|All files (*.*)|*.*"
CommonDialog1.DefaultExt = "txt"
CommonDialog1.DialogTitle = "Select File"
CommonDialog1.ShowOpen

Open CommonDialog1.FileName For Input As #3
Do While Not EOF(3)
Rem----------------------------------------
  Line Input #3, packt
  packet = packt
  If Not packet = "" Then
  Label6.Caption = Label6.Caption + 1
  For i = 1 To Len(packet) - 4
  DoEvents
  Sbit = Mid(packet, i, 1)
    If Sbit = 0 Then
      errs = errs + 1
      zeros = zeros + 1
      ErrStat(i, 1) = ErrStat(i, 1) + 1

    ElseIf Sbit = 1 Then
      ones = ones + 1
    End If

Rem-----------------------------------------------
    If Sbit = 0 Then
    zero_seq = zero_seq + 1
    End If

    If Sbit = 1 And zero_seq > 0 Then
    If zero_seq = 1 Then 0 = 0 + 1
    If zero_seq = 2 Then 00 = 00 + 1
    If zero_seq = 3 Then 000 = 000 + 1
    If zero_seq = 4 Then 0000 = 0000 + 1
    If zero_seq = 5 Then 00000 = 00000 + 1
    If zero_seq = 6 Then 000000 = 000000 + 1
    If zero_seq = 7 Then 0000000 = 0000000 + 1
    zero_seq = 0
```

```
    End If


  Next i

  ErrStat(errs, 0) = ErrStat(errs, 0) + 1
  errs = 0
  End If
Rem-----------------------------------------
Loop
Close #3



Label2.Caption = "1: " & ones & "x"
Label3.Caption = "0: " & zeros & "x"
Label5.Caption = "Total: " & (zeros + ones) / 8 & " Bytes"
Label4.Caption = "Bit error rate:  " & Round(zeros / (zeros + ones) * 100, 4) & " %"
Label7.Caption = "1x0: " & O
Label8.Caption = "2x0: " & OO
Label9.Caption = "3x0: " & OOO
Label10.Caption = "4x0: " & OOOO
Label11.Caption = "5x0: " & OOOOO
Label12.Caption = "6x0: " & OOOOOO
Label13.Caption = "7x0: " & OOOOOOO
Exit Sub

err:
msg = MsgBox("Chyba! Prázdný soubor!", vbOKOnly, "Packet Stream Analysis")
End Sub



Private Sub Command3_Click()
If MSComm1.PortOpen = True Then
MSComm1.PortOpen = False
Close #1
Close #2
End If
   BINpacket = ""
   active = 0
   Label1.Caption = "Logged segments: " & active
End Sub

Private Sub Command4_Click()

Dim err, errIN, srs, d7_108(1), d14_54(1), d21_36(1), d28_27(1), d36_21(1), d54_14(1), d108_7(1), avgErr_IN(1), MaxErr
err = 0
errIN = 0
avgErr_IN(1) = 0: avgErr_IN(0) = 0
CLR_mem

Label6.Caption = 0
Rem On Error GoTo Err
CommonDialog1.Filter = "Apps (*.txt)|*.txt|All files (*.*)|*.*"
CommonDialog1.DefaultExt = "txt"
CommonDialog1.DialogTitle = "Select File"
CommonDialog1.ShowOpen

Label40.Caption = "File: " & Mid(CommonDialog1.FileName, Len(CommonDialog1.FileName) - 14, 11)

Open CommonDialog1.FileName For Input As #2
Do While Not EOF(2)
Rem-----------------------------------------
   Line Input #2, packt

   packet = packt
   If Not packet = "" Then
   Label14.Caption = Label14.Caption + 1
   For i = 1 To Len(packet) - 4
   DoEvents
      Sbit = Mid(packet, i, 1)
      If Sbit = 0 Then errIN = errIN + 1
      ReDim Preserve bin_input(1 To i)
      bin_input(i) = Sbit
   Next i
    If errIN > 0 Then
```

```
    avgErr_IN(0) = avgErr_IN(0) + errIN: avgErr_IN(1) = avgErr_IN(1) + 1: Label17.Caption = Round(avgErr_IN(0) / avgErr_IN(1), 1)
      If MaxErr < errIN Then MaxErr = errIN
    Else
    Label39.Caption = Label39.Caption + 1
    End If
    Label42.Caption = MaxErr
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(7, 108)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label19.Caption = Label19.Caption + 1: d7_108(0) = d7_108(0) + err: d7_108(1) = d7_108(1) + 1: Label20.Caption =
Round(d7_108(0) / d7_108(1), 1)
    If errIN > 0 And err = 0 Then Label16.Caption = Label16.Caption + 1

    err = 0
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(14, 54)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label21.Caption = Label21.Caption + 1: d14_54(0) = d14_54(0) + err: d14_54(1) = d14_54(1) + 1: Label24.Caption =
Round(d14_54(0) / d14_54(1), 1)
    If errIN > 0 And err = 0 Then Label22.Caption = Label22.Caption + 1
    err = 0
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(21, 36)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label23.Caption = Label23.Caption + 1: d21_36(0) = d21_36(0) + err: d21_36(1) = d21_36(1) + 1: Label26.Caption =
Round(d21_36(0) / d21_36(1), 1)
    If errIN > 0 And err = 0 Then Label25.Caption = Label25.Caption + 1
    err = 0
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(28, 27)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label27.Caption = Label27.Caption + 1: d28_27(0) = d28_27(0) + err: d28_27(1) = d28_27(1) + 1: Label29.Caption =
Round(d28_27(0) / d28_27(1), 1)
    If errIN > 0 And err = 0 Then Label28.Caption = Label28.Caption + 1
    err = 0
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(36, 21)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label30.Caption = Label30.Caption + 1: d36_21(0) = d36_21(0) + err: d36_21(1) = d36_21(1) + 1: Label32.Caption =
Round(d36_21(0) / d36_21(1), 1)
    If errIN > 0 And err = 0 Then Label31.Caption = Label31.Caption + 1
    err = 0
'-----------------------------------------------------------------------------------------------------------------------------------------------------------
    cpy_array
    Call deInterleave_bin_array(54, 14)
    DoEvents
    deCode
       For i = 1 To UBound(bin_array)
          If bin_array(i) = 0 Then err = err + 1
       Next i
    If err > 0 Then Label33.Caption = Label33.Caption + 1: d54_14(0) = d54_14(0) + err: d54_14(1) = d54_14(1) + 1: Label35.Caption =
Round(d54_14(0) / d54_14(1), 1)
```

```
   If errIN > 0 And err = 0 Then Label34.Caption = Label34.Caption + 1
   err = 0
'-------------------------------------------------------------------------------------------------------------------------------------------------------------
   cpy_array
   Call deInterleave_bin_array(108, 7)
   DoEvents
   deCode
      For i = 1 To UBound(bin_array)
         If bin_array(i) = 0 Then err = err + 1
      Next i
   If err > 0 Then Label36.Caption = Label36.Caption + 1: d108_7(0) = d108_7(0) + err: d108_7(1) = d108_7(1) + 1: Label38.Caption =
Round(d108_7(0) / d108_7(1), 1)
   If errIN > 0 And err = 0 Then Label37.Caption = Label37.Caption + 1
   err = 0
'-------------------------------------------------------------------------------------------------------------------------------------------------------------

   errIN = 0
   End If
Rem-----------------------------------------
Loop
Close #2

savedata
End Sub
Private Sub savedata()


Open "c:/out.csv" For Output As #7
Print #7, Label14.Caption & ";" & Label39.Caption & ";" & Label17.Caption & ";" & Label19.Caption & ";" & Label16.Caption & ";" &
Label21.Caption & ";" & Label22.Caption & ";" & Label23.Caption & ";" & Label25.Caption & ";" & Label27.Caption & ";" &
Label28.Caption & ";" & Label30.Caption & ";" & Label31.Caption & ";" & Label33.Caption & ";" & Label34.Caption & ";" &
Label36.Caption & ";" & Label37.Caption
Close #7
End Sub

Private Sub cpy_array()
ReDim bin_array(1 To 756)

For b = 1 To UBound(bin_input)
bin_array(b) = bin_input(b)
Next b
End Sub


Private Sub showk()
filename1 = "C:\save.txt"
Open filename1 For Output As #5

For x = 1 To UBound(bin_array)
Write #5, bin_array(x)
Next x

Close #5
End Sub
Private Sub deInterleave_bin_array(W As Integer, H As Integer)

Dim tmp_array(108, 108) As String, cursor As Integer
Rem ReDim tmp_array(H, W)
If Not UBound(bin_array) = 756 Then MsgBox ("BinBuf ma blbou velikost - " & UBound(bin_array))

cursor = 0
For j = 1 To W
   For k = 1 To H
   cursor = cursor + 1
   tmp_array(k, j) = bin_array(cursor)
   Next k
Next j

cursor = 0
For k = 1 To H
   For j = 1 To W
   cursor = cursor + 1
   bin_array(cursor) = tmp_array(k, j)
   Next j
Next k

End Sub
```

```
Public Sub deCode()
   For i = 0 To 107
      Call deHamming(bin_array(i * 7 + 1), bin_array(i * 7 + 2), bin_array(i * 7 + 3), bin_array(i * 7 + 4), bin_array(i * 7 + 5), bin_array(i
* 7 + 6), bin_array(i * 7 + 7), i + 1)
   Next i
   ReDim Preserve bin_array(1 To 432)
End Sub


Public Sub deHamming(x1, x2, x3, x4, x5, x6, x7, srs)
Dim s1, s2, s3
Dim syndrome

p1 = x4 Xor x5 Xor x6 Xor x7
p2 = x2 Xor x3 Xor x6 Xor x7
p3 = x1 Xor x3 Xor x5 Xor x7


syndrome = p1 * 4 + p2 * 2 + p3 * 1


   Select Case syndrome
   Case 0: GoTo ok
   Case 1: If x1 = "0" Then x1 = "1" Else x1 = "0"
   Case 2: If x2 = "0" Then x2 = "1" Else x2 = "0"
   Case 3: If x3 = "0" Then x3 = "1" Else x3 = "0"
   Case 4: If x4 = "0" Then x4 = "1" Else x4 = "0"
   Case 5: If x5 = "0" Then x5 = "1" Else x5 = "0"
   Case 6: If x6 = "0" Then x6 = "1" Else x6 = "0"
   Case 7: If x7 = "0" Then x7 = "1" Else x7 = "0"
   End Select

ok:

bin_array(srs * 4 - 3) = x3
bin_array(srs * 4 - 2) = x5
bin_array(srs * 4 - 1) = x6
bin_array(srs * 4) = x7

End Sub


Private Sub CLR_mem()
Label6.Caption = 0
Label14.Caption = 0
zero_seq = 0
active = 0
zeros = 0
ones = 0
OOOOOOO = 0
OOOOOO = 0
OOOOO = 0
OOOO = 0
OOO = 0
OO = 0
O = 0
MaxErr = 0
BINpacket = ""
Label2.Caption = "1: "
Label3.Caption = "0: "
Label5.Caption = "Total: "
Label4.Caption = "Bit error rate:  "
Label7.Caption = "1x0: "
Label8.Caption = "2x0: "
Label9.Caption = "3x0: "
Label10.Caption = "4x0: "
Label11.Caption = "5x0: "
Label12.Caption = "6x0: "
Label13.Caption = "7x0: "

Label17.Caption = "0"
Label16.Caption = "0"
Label19.Caption = "0"
Label20.Caption = "0"
```

```
Label21.Caption = "0"
Label22.Caption = "0"
Label24.Caption = "0"

Label23.Caption = "0"
Label25.Caption = "0"
Label26.Caption = "0"

Label27.Caption = "0"
Label28.Caption = "0"
Label29.Caption = "0"

Label30.Caption = "0"
Label31.Caption = "0"
Label32.Caption = "0"

Label33.Caption = "0"
Label34.Caption = "0"
Label35.Caption = "0"

Label36.Caption = "0"
Label37.Caption = "0"
Label38.Caption = "0"

Label39.Caption = "0"
Label42.Caption = "0"

For i = 0 To 756
ErrStat(i, 1) = 0
ErrStat(i, 0) = 0
Next i

End Sub

Private Sub Command5_Click()
CommonDialog1.FileName = "out"
CommonDialog1.Filter = "Apps (*.csv)|*.csv|All files (*.*)|*.*"
CommonDialog1.DefaultExt = "csv"
CommonDialog1.DialogTitle = "Select File"
CommonDialog1.ShowOpen

Open CommonDialog1.FileName For Output As #8

For i = 0 To 756
Print #8, ErrStat(i, 0) & ";" & ErrStat(i, 1)
Next i

Close #8

End Sub



Private Sub Form_Load()
CLR_mem
For i = 0 To 4
Combo1.AddItem "Com" & i
Combo1.ListIndex = Combo1.NewIndex
Next i
End Sub



Private Sub MSComm1_OnCommSSSS()
If MSComm1.CommEvent = comEvReceive Then

sdata = Asc(MSComm1.Input)

sync = sdata
If sync = 0 And sync_last = 0 Then
    If Len(BINpacket) > 700 Then
    Print #1, BINpacket
    Print #1, vbNewLine
    ElseIf Len(BINpacket) < 700 Then
    Print #2, BINpacket
    Print #2, vbNewLine
    End If
```

```
   BINpacket = ""
   active = active + 1
   Label1.Caption = "Logged segments: " & active
End If
sync_last = sdata

If sdata = 0 Then Exit Sub


BINpacket = BINpacket & Dec2Bin(sdata)

End If
End Sub


Private Sub MSComm1_OnComm()
If MSComm1.CommEvent = comEvReceive Then

sdata = Asc(MSComm1.Input)
BINpacket = BINpacket & Dec2Bin(sdata)

      If Len(BINpacket) = 756 Then
      Print #1, BINpacket
      Print #1, vbNewLine
      BINpacket = ""
      End If

   active = active + 1
   Label1.Caption = "Logged segments: " & active

End If
End Sub


Function Dec2Bin(ByVal n As Long) As String
Do Until n = 0
   If (n Mod 2) Then Dec2Bin = "1" & Dec2Bin Else Dec2Bin = "0" & Dec2Bin
   n = n \ 2
Loop

Do Until (Len(Dec2Bin) = 8)
Dec2Bin = "0" & Dec2Bin
Loop

End Function
```

V. Výsledky analýzy dat při volbě rozměru prokládací matice

	Úspěšnost prokladačů při koexistenci s kolizní sílou ping paketů 7 b					
	Rozestup	*0 ms*	1 ms	10 ms	25 ms	50 ms
	Ø chybovost	21,2	17,7	14,2	12,4	11,7
Prokladač	108/7	80%	78%	87%	89%	86%
	54/14	78%	70%	83%	85%	84%
	36/21	76%	71%	83%	84%	81%
	28/27	75%	69%	81%	83%	81%
	21/36	55%	52%	61%	63%	66%
	14/54	53%	50%	61%	65%	60%
	7/108	36%	28%	38%	43%	41%

	Úspěšnost prokladačů při koexistenci s kolizní sílou ping paketů 9 b					
	Rozestup	*0 ms*	1 ms	10 ms	25 ms	50 ms
	Ø chybovost	18,3	27,3	10,6	13,3	16,4
Prokladač	108/7	70%	77%	90%	85%	76%
	54/14	68%	68%	75%	82%	71%
	36/21	72%	72%	73%	83%	71%
	28/27	72%	70%	72%	83%	72%
	21/36	45%	49%	57%	60%	55%
	14/54	41%	44%	55%	57%	60%
	7/108	20%	27%	35%	37%	41%

	Úspěšnost prokladačů při koexistenci s kolizní sílou ping paketů 53 b				
	Rozestup	*0 ms*	1 ms	10 ms	50 ms
	Ø chybovost	67,7	30	32,3	27,8
Prokladač	108/7	14%	78%	74%	74%
	54/14	6%	35%	43%	40%
	36/21	3%	25%	27%	25%
	28/27	3%	17%	20%	21%
	21/36	2%	16%	20%	19%
	14/54	1%	14%	16%	15%
	7/108	1%	11%	12%	11%

	Úspěšnost prokladaču při rušení z komutátoru točivého stroje					
	Ø chybovost	2,5	5,7	17,1	24,6	31
Prokladač	108/7	98%	84%	22%	16%	7%
	54/14	100%	92%	39%	30%	11%
	36/21	100%	96%	74%	41%	20%
	28/27	100%	98%	73%	40%	17%
	21/36	100%	98%	68%	33%	20%
	14/54	100%	92%	49%	28%	12%
	7/108	99%	91%	32%	8%	3%

VI. Software programu pro Simulaci ZigBee protokolu

```
Private Sub Command2_Click()
Command2.Enabled = False
Dim pkt_cursor, hexpkt

''DSSS pokud je zaskrtnuto
  If Check1.Value = 1 Then
    import_pkt_MATLAB
    deSpread_pkt
    clr_file
    Export_pkt
  End If
import_pkt
hexpkt = Pkt_BinToHex
Label7.Caption = 0: Label8.Caption = 0: Label9.Caption = 0: Label10.Caption = 0

pkt_cursor = 1
For i = 1 To (Len(hexpkt) / 2 / pkt_length)
err = Check_pkt(Mid(hexpkt, pkt_cursor, pkt_length * 2))
If err > 0 Then Label8.Caption = Label8.Caption + 1
If err = 10 Or err = 11 Or err = 110 Or err = 111 Then Label10.Caption = Label10.Caption + 1
If err = 100 Or err = 101 Or err = 110 Or err = 111 Then Label9.Caption = Label9.Caption + 1
pkt_cursor = pkt_cursor + pkt_length * 2
Label7.Caption = Label7.Caption + 1
DoEvents
Next i

Command2.Enabled = True
End Sub

Private Sub Command8_Click()
Dim F As String, result As String
Dim fls As Integer
Command8.Enabled = False
fls = 0

Clear_results

F = Dir$(App.Path & "/in_*.*")

Do While LenB(F) > 0
Scan_file = App.Path & "/" & F
Call Command2_Click

result = Mid(F, 4, 9) / 1000000 & ";" & Label7.Caption & ";" & Label8.Caption & ";" & Label9.Caption & ";" & Label10.Caption &
";" & (Label8.Caption / Label7.Caption)
Export_results (result)

F = Dir$

fls = fls + 1
Command8.Caption = "Scanned " & fls & " fls"
Loop

Command8.Enabled = True
End Sub

Private Sub Form_Load()
CRC_init
pkt_length = Payload_len + 32
Text1.Text = "500"
Check1.Value = 1
Check2.Value = 1
Check3.Value = 1
Option1.Value = True
Scan_file = App.Path & "/in.txt"

If Not Command = "" Then Timer1.Enabled = True
End Sub

'preambule 32 nul + SOF delimiter 8bit (hE5) + delka paketu 8bit
```

```
'prokladac je 28*27 a generuje 756bit

Private Sub Command5_Click()
import_pkt
Spread_pkt
clr_file
Export_pkt
End Sub

Private Sub Command6_Click()
import_pkt
deSpread_pkt
clr_file
Export_pkt
End Sub

Private Sub Command7_Click()
Command7.Enabled = False
Dim pkt_cursor, hexpkt

''DSSS pokud je zaskrtnuto
  If Check1.Value = 1 Then
    import_pkt
    deSpread_pkt
    clr_file
    Export_pkt
  End If

import_pkt
hexpkt = Pkt_BinToHex

Label7.Caption = 0: Label8.Caption = 0: Label9.Caption = 0: Label10.Caption = 0
pkt_cursor = 1
For i = 1 To (Len(hexpkt) / 2 / pkt_length)
err = Check_pkt(Mid(hexpkt, pkt_cursor, pkt_length * 2))
If err > 0 Then Label8.Caption = Label8.Caption + 1
If err = 10 Or err = 11 Or err = 110 Or err = 111 Then Label10.Caption = Label10.Caption + 1
If err = 100 Or err = 101 Or err = 110 Or err = 111 Then Label9.Caption = Label9.Caption + 1
pkt_cursor = pkt_cursor + pkt_length * 2
Label7.Caption = Label7.Caption + 1
DoEvents
Next i

Command7.Enabled = True
End Sub

Private Sub Command3_Click()
Command3.Enabled = False
clr_file
For i = 1 To Text1.Text
If Check2.Value = 0 Then StringToBinary_pkt (Gen_Standard_Pkt) Else StringToBinary_pkt (Gen_Coded_Pkt)

Export_pkt
Next i

''DSSS pokud je zaskrtnuto
  If Check1.Value = 1 Then
    import_pkt
    Spread_pkt
    clr_file
    Export_pkt
  End If
Command3.Enabled = True

End Sub

Private Sub Interleave_bin_array(W As Integer, H As Integer)

Dim tmp_array(108, 108) As String, cursor As Integer
Rem ReDim tmp_array(H, W)
If Not UBound(bin_array) = 756 Then MsgBox ("BinBuf ma blbou velikost - " & UBound(bin_array))

cursor = 0
For j = 1 To H
```

```
   For k = 1 To W
   cursor = cursor + 1
   tmp_array(k, j) = bin_array(cursor)
   Next k
Next j

cursor = 0
For k = 1 To W
   For j = 1 To H
   cursor = cursor + 1
   bin_array(cursor) = tmp_array(k, j)
   Next j
Next k

End Sub

Private Sub deInterleave_bin_array(W As Integer, H As Integer)
Dim tmp_array(108, 108) As String, cursor As Integer
Rem ReDim tmp_array(H, W)
If Not UBound(bin_array) = 756 Then MsgBox ("BinBuf ma blbou velikost - " & UBound(bin_array))

cursor = 0
For j = 1 To W
   For k = 1 To H
   cursor = cursor + 1
   tmp_array(k, j) = bin_array(cursor)
   Next k
Next j

cursor = 0
For k = 1 To H
   For j = 1 To W
   cursor = cursor + 1
   bin_array(cursor) = tmp_array(k, j)
   Next j
Next k

End Sub
''////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''DSSS rutiny

Private Sub deSpread_pkt()
Dim DSSS_word As Double
Dim Original_word As String
Dim diff As Integer, err As Integer
Dim DSSS_tmp() As String

DSSS_word = 0
Original_word = 0

For i = 1 To UBound(bin_array) / 32
   diff = 32
   err = 0
   For j = 1 To 32
      err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0000, j, 1))
   Next j
   If err < diff Then DSSS_orig = "0000": diff = err

   err = 0
   For j = 1 To 32
      err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1000, j, 1))
   Next j
   If err < diff Then DSSS_orig = "1000": diff = err

   err = 0
   For j = 1 To 32
      err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0100, j, 1))
   Next j
   If err < diff Then DSSS_orig = "0100": diff = err

   err = 0
   For j = 1 To 32
      err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1100, j, 1))
   Next j
```

```
If err < diff Then DSSS_orig = "1100": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0010, j, 1))
Next j
If err < diff Then DSSS_orig = "0010": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1010, j, 1))
Next j
If err < diff Then DSSS_orig = "1010": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0110, j, 1))
Next j
If err < diff Then DSSS_orig = "0110": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1110, j, 1))
Next j
If err < diff Then DSSS_orig = "1110": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0001, j, 1))
Next j
If err < diff Then DSSS_orig = "0001": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1001, j, 1))
Next j
If err < diff Then DSSS_orig = "1001": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0101, j, 1))
Next j
If err < diff Then DSSS_orig = "0101": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1101, j, 1))
Next j
If err < diff Then DSSS_orig = "1101": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0011, j, 1))
Next j
If err < diff Then DSSS_orig = "0011": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1011, j, 1))
Next j
If err < diff Then DSSS_orig = "1011": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_0111, j, 1))
Next j
If err < diff Then DSSS_orig = "0111": diff = err

err = 0
For j = 1 To 32
   err = err + (bin_array(j + DSSS_word) Xor Mid(DSSS_1111, j, 1))
Next j
If err < diff Then DSSS_orig = "1111": diff = err
```

```
   ReDim Preserve DSSS_tmp(1 To Original_word + 4)
   For j = 1 To 4
   DSSS_tmp(j + Original_word) = Mid(DSSS_orig, j, 1)
   Next j
   Original_word = Original_word + 4
   DSSS_word = DSSS_word + 32
Next i

ReDim Preserve bin_array(1 To UBound(DSSS_tmp))
For i = 1 To UBound(DSSS_tmp)
  bin_array(i) = DSSS_tmp(i)
Next i

End Sub

Private Sub Spread_pkt()
Dim ctverice, bits, dsss_seq

ctverice = 1: dsss_seq = 1
For i = 1 To UBound(bin_array) / 4
   bits = bin_array(ctverice) & bin_array(ctverice + 1) & bin_array(ctverice + 2) & bin_array(ctverice + 3)
   ctverice = ctverice + 4

   Select Case bits
   Case "0000": bits = DSSS_0000
   Case "1000": bits = DSSS_1000
   Case "0100": bits = DSSS_0100
   Case "1100": bits = DSSS_1100
   Case "0010": bits = DSSS_0010
   Case "1010": bits = DSSS_1010
   Case "0110": bits = DSSS_0110
   Case "1110": bits = DSSS_1110
   Case "0001": bits = DSSS_0001
   Case "1001": bits = DSSS_1001
   Case "0101": bits = DSSS_0101
   Case "1101": bits = DSSS_1101
   Case "0011": bits = DSSS_0011
   Case "1011": bits = DSSS_1011
   Case "0111": bits = DSSS_0111
   Case "1111": bits = DSSS_1111
   End Select

   For j = 1 To 32
      ReDim Preserve DSSS_tmp(1 To dsss_seq)
      DSSS_tmp(dsss_seq) = Mid(bits, j, 1)
      dsss_seq = dsss_seq + 1
   Next j
Next i

For i = 1 To UBound(DSSS_tmp)
   ReDim Preserve bin_array(1 To i)
   bin_array(i) = DSSS_tmp(i)
Next i
End Sub
''////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''File handling

Private Sub clr_file()
Open App.Path & "\out.txt" For Output As #1
Close #1
End Sub


Private Sub import_pkt_MATLAB()
Dim cnt, index, line

   Open Scan_file For Input As #1
        Line Input #1, line
        DoEvents
   Close #1

   index = 1
   For i = 1 To Len(line)
```

```
   DoEvents
   tmp = Mid(line, i, 1)
      If tmp = "0" Or tmp = "1" Then
               ReDim Preserve bin_array(1 To index)
               bin_array(index) = tmp
               index = index + 1
      End If
   Next i

End Sub

Private Sub import_pkt()
Dim cnt, line

   Open App.Path & "\out.txt" For Input As #1
   Do While Not EOF(1)
   cnt = cnt + 1
   DoEvents
   ReDim Preserve bin_array(1 To cnt)
   Line Input #1, bin_array(cnt)
   Loop
   Close #1

End Sub

Private Sub Export_pkt()

Open App.Path & "\out.txt" For Append As #1

For i = 1 To (UBound(bin_array))
   Print #1, bin_array(i)
Next i

Close #1
End Sub

Private Sub Clear_results()

Open App.Path & "\scan_results.csv" For Output As #1
Close #1
End Sub

Private Sub Export_results(result As String)

Open App.Path & "\scan_results_zb.csv" For Append As #1
   Print #1, result
Close #1
End Sub

''////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''paket generator
''generuje paket (4B-preamble, 1B-delimiter, 1B-delka, 95B-payload, 2B-CRC)
''+ je potreba generovat 8B na MAC Header, 8B NWK Header, 8B APS Header navic
''celkove je Kodovany paket s 95B payloadem dlouhy 123B

Private Function Gen_Standard_Pkt() As String
Dim pkt As String

pkt = Rnd_payload(Payload_len)      'Payload_len definovano v modulu
pkt = pkt & CRC_CCITT(pkt)
pkt = "AAAAAAAAAAAAAAAA" & pkt  'MAC Header
pkt = "AAAAAAAAAAAAAAAA" & pkt  'NWK Header
pkt = "AAAAAAAAAAAAAAAA" & pkt  'APS Header
pkt = "0000000000000000E566" & pkt      'PHY + SYNC Header
Gen_Standard_Pkt = pkt
End Function

Private Function Gen_Coded_Pkt() As String
Dim pkt As String

pkt = Rnd_payload(Coded_Payload_len)      'Payload_len definovano v modulu
pkt = pkt & CRC_CCITT(pkt)

StringToBinary_pkt (pkt)
```

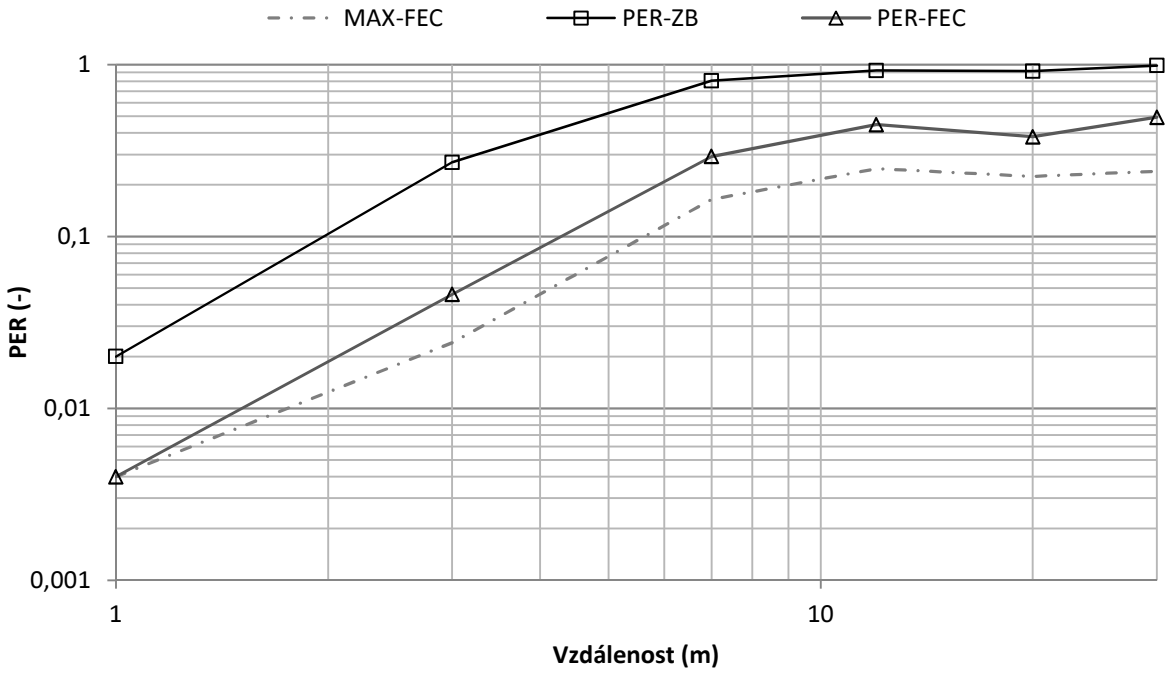
```
If Check2.Value = 1 Then Hamming_bin_array

If Check3.Value = 1 Then
      If Option2.Value = True Then Call Interleave_bin_array(36, 21)
      If Option1.Value = True Then Call Interleave_bin_array(108, 7)
   End If
''pridat dummy 4 bity
ReDim Preserve bin_array(1 To 760)
For i = 1 To 4: bin_array(756 + i) = 0: Next i

pkt = (Pkt_BinToHex)
'pkt len - 190 (380bit - 95B)

pkt = "AAAAAAAAAAAAAAAA" & pkt  'MAC Header
pkt = "AAAAAAAAAAAAAAAA" & pkt  'NWK Header
pkt = "AAAAAAAAAAAAAAAA" & pkt  'APS Header
pkt = "00000000E566" & pkt      'PHY + SYNC Header
Gen_Coded_Pkt = pkt
End Function

Private Function Rnd_payload(length) As String
Dim randnum As String

For i = 1 To length
DoEvents
Randomize
randnum = Hex(CByte(Rnd * &HFF))
If Len(randnum) < 2 Then randnum = "0" & randnum    'bylo li generovane cislo jednomistne, je treba doplnit nulu
Rnd_payload = Rnd_payload & randnum
Next i

End Function


Private Function Check_pkt(pkt As String) As Integer
Dim payload
Check_pkt = 0
payload = Len(pkt)

If Not Mid(pkt, 1, 12) = "00000000E566" Then Check_pkt = 100

''v pripade kodovanych paketu netreba kontrolovat chyby v hlavickach vyssich vrstev
If Check2.Value = 0 And Check3.Value = 0 Then
If Not Mid(pkt, 13, 48) = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" Then Check_pkt = Check_pkt +
10
End If

If Check2.Value = 1 Or Check3.Value = 1 Then
' volba Hammingova dekodovani
StringToBinary_pkt (Mid(pkt, 61, payload - 60))

ReDim Preserve bin_array(1 To 756)                    ''korekce 4 dummy bitu
   If Check3.Value = 1 Then
      If Option2.Value = True Then Call deInterleave_bin_array(36, 21)
      If Option1.Value = True Then Call deInterleave_bin_array(108, 7)
   End If
   If Check2.Value = 1 Then deHamming_bin_array
   pkt = Pkt_BinToHex

If Not Mid(pkt, 105, 108) = CRC_CCITT(Mid(pkt, 1, 104)) Then Check_pkt = Check_pkt + 1
Else

If Not Mid(pkt, payload - 3, 4) = CRC_CCITT(Mid(pkt, 61, payload - 64)) Then Check_pkt = Check_pkt + 1
End If
End Function

''////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''HAMMING
Private Sub Hamming_bin_array()

Dim cnt As Integer
Dim tmp_array() As String
cnt = 3
```

```
For i = 1 To UBound(bin_array)
DoEvents
   If cnt = 0 Then
   Call Hamming(bin_array(i - 3), bin_array(i - 2), bin_array(i - 1), bin_array(i))
   ReDim Preserve tmp_array(1 To i / 4 * 7)
   tmp_array((i / 4 * 7) - 6) = h1
   tmp_array((i / 4 * 7) - 5) = h2
   tmp_array((i / 4 * 7) - 4) = h3
   tmp_array((i / 4 * 7) - 3) = h4
   tmp_array((i / 4 * 7) - 2) = h5
   tmp_array((i / 4 * 7) - 1) = h6
   tmp_array((i / 4 * 7)) = h7
   cnt = 4
   End If
cnt = cnt - 1
Next i

For i = 1 To UBound(tmp_array)
ReDim Preserve bin_array(1 To i)
bin_array(i) = tmp_array(i)
Next i

End Sub

Private Sub deHamming_bin_array()
Dim cnt As Integer
Dim tmp_array() As String
cnt = 6

'ReDim Preserve bin_array(1 To 756)

For i = 1 To UBound(bin_array)
DoEvents
   If cnt = 0 Then
   Call deHamming(bin_array(i - 6), bin_array(i - 5), bin_array(i - 4), bin_array(i - 3), bin_array(i - 2), bin_array(i - 1), bin_array(i))
   ReDim Preserve tmp_array(1 To i / 7 * 4)
   tmp_array((i / 7 * 4) - 3) = h1
   tmp_array((i / 7 * 4) - 2) = h2
   tmp_array((i / 7 * 4) - 1) = h3
   tmp_array((i / 7 * 4)) = h4
   cnt = 7
   End If
cnt = cnt - 1
Next i

For i = 1 To UBound(tmp_array)
DoEvents
ReDim Preserve bin_array(1 To i)
bin_array(i) = tmp_array(i)
Next i
End Sub

Public Sub Hamming(x1, x2, x3, x4)
Dim p1, p2, p3

p1 = x1 Xor x2 Xor x4
p2 = x1 Xor x3 Xor x4
p3 = x2 Xor x3 Xor x4

h1 = p1
h2 = p2
h3 = x1
h4 = p3
h5 = x2
h6 = x3
h7 = x4

End Sub

Public Sub deHamming(x1, x2, x3, x4, x5, x6, x7)
Dim s1, s2, s3
Dim syndrome

p1 = x4 Xor x5 Xor x6 Xor x7
```

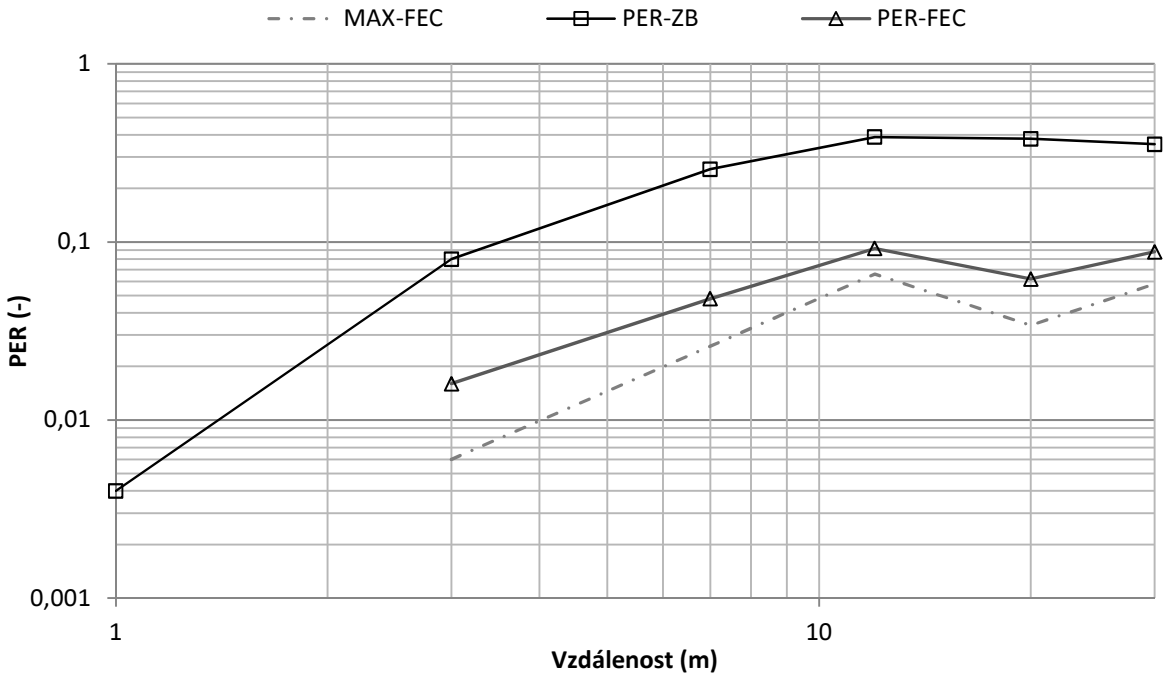
```
p2 = x2 Xor x3 Xor x6 Xor x7
p3 = x1 Xor x3 Xor x5 Xor x7

syndrome = p1 * 4 + p2 * 2 + p3 * 1

   Select Case syndrome
   Case 0: GoTo ok
   Case 1: If x1 = "0" Then x1 = "1" Else x1 = "0"
   Case 2: If x2 = "0" Then x2 = "1" Else x2 = "0"
   Case 3: If x3 = "0" Then x3 = "1" Else x3 = "0"
   Case 4: If x4 = "0" Then x4 = "1" Else x4 = "0"
   Case 5: If x5 = "0" Then x5 = "1" Else x5 = "0"
   Case 6: If x6 = "0" Then x6 = "1" Else x6 = "0"
   Case 7: If x7 = "0" Then x7 = "1" Else x7 = "0"
   End Select

ok:
h1 = x3
h2 = x5
h3 = x6
h4 = x7

End Sub

''////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''     CRC CCITT 16
''takhle funguje - data hexa po dvojicich ve stringu
''Text1.Text = CRC_CCITT("05ff")

Private Sub CRC_init()
Dim X As Long
Dim TempStr As String

Open App.Path & "\CRCTAB.txt" For Input As #1
   For X = 0 To 255
      Input #1, TempStr
      CRCTab(X) = CLng(TempStr)
   Next
Close #1
End Sub

Private Function CRC_CCITT(data As String) As String
Dim DataByte() As Byte
Dim DataSize As Long
Dim X
Dim index
Dim TempLng
Dim CRC

DataSize = (Len(data) / 2) - 1
ReDim DataByte(DataSize)
index = 1
For X = 0 To DataSize
   DataByte(X) = "&h" & Mid(data, index, 2)
   index = index + 2
Next

CRC = CLng("&hFFFF")

For X = 0 To DataSize
   TempLng = ((CRC \ 256) Xor DataByte(X))
   CRC = ((CRC * 256) And 65535) Xor CRCTab(TempLng)
Next
CRC_CCITT = Right("0000" & Hex(CRC), 4)

End Function

''////////////////////////////////////////////////////////////////////////////////////////////////////////////////
''podpurne

Function Num2Bin(ByVal n As Long) As String
Do Until n = 0
   If (n Mod 2) Then Num2Bin = "1" & Num2Bin Else Num2Bin = "0" & Num2Bin
   n = n \ 2
```

```
Loop
Do Until (Len(Num2Bin) = 8)
Num2Bin = "0" & Num2Bin
Loop
End Function


Private Function Pkt_BinToHex() As String
Dim biit, Byyte

Bit = 0: Byyte = 1
ReDim Preserve num_array(1 To Byyte): num_array(Byyte) = 0
For i = 1 To UBound(bin_array)
DoEvents
  num_array(Byyte) = num_array(Byyte) + bin_array(i) * 2 ^ (7 - Bit)
  Bit = Bit + 1
  If Bit = 8 Then Bit = 0: Byyte = Byyte + 1: ReDim Preserve num_array(1 To Byyte): num_array(Byyte) = 0
Next i
ReDim Preserve num_array(1 To UBound(num_array) - 1)
If Not Bit = 0 Then msg = MsgBox("nevyslo to na celej bajt", vbCritical, "Zase pruser...")

Pkt_BinToHex = ""
For i = 1 To UBound(num_array)
DoEvents
Byyte = Hex(num_array(i))
If Len(Byyte) < 2 Then Byyte = "0" & Byyte
Pkt_BinToHex = Pkt_BinToHex & Byyte
Next i

End Function

Private Sub StringToBinary_pkt(pkt As String)
Dim Byyte, index, bits


index = 1: Byyte = 1
For i = 1 To (Len(pkt) / 2)

  bits = Num2Bin("&h" & Mid(pkt, Byyte, 2))
  Byyte = Byyte + 2

    For j = 1 To 8
    DoEvents
      ReDim Preserve bin_array(1 To index)
      bin_array(index) = Mid(bits, j, 1)
      index = index + 1
    Next j
Next i

End Sub

Private Sub Check3_Click()
Option2.Value = True
End Sub

Private Sub Timer1_Timer()
Timer1.Enabled = False
Scan_file = App.Path & "/" & Command
Command2_Click
SNR = Mid(Command, 4, Len(Command) - 7)
SNR = Replace$(SNR, ".", ",")
On Error Resume Next
result = SNR & ";" & Label7.Caption & ";" & Label8.Caption & ";" & Label9.Caption & ";" & Label10.Caption & ";" &
(Label8.Caption / Label7.Caption)
Export_results (result)
End
End Sub
```

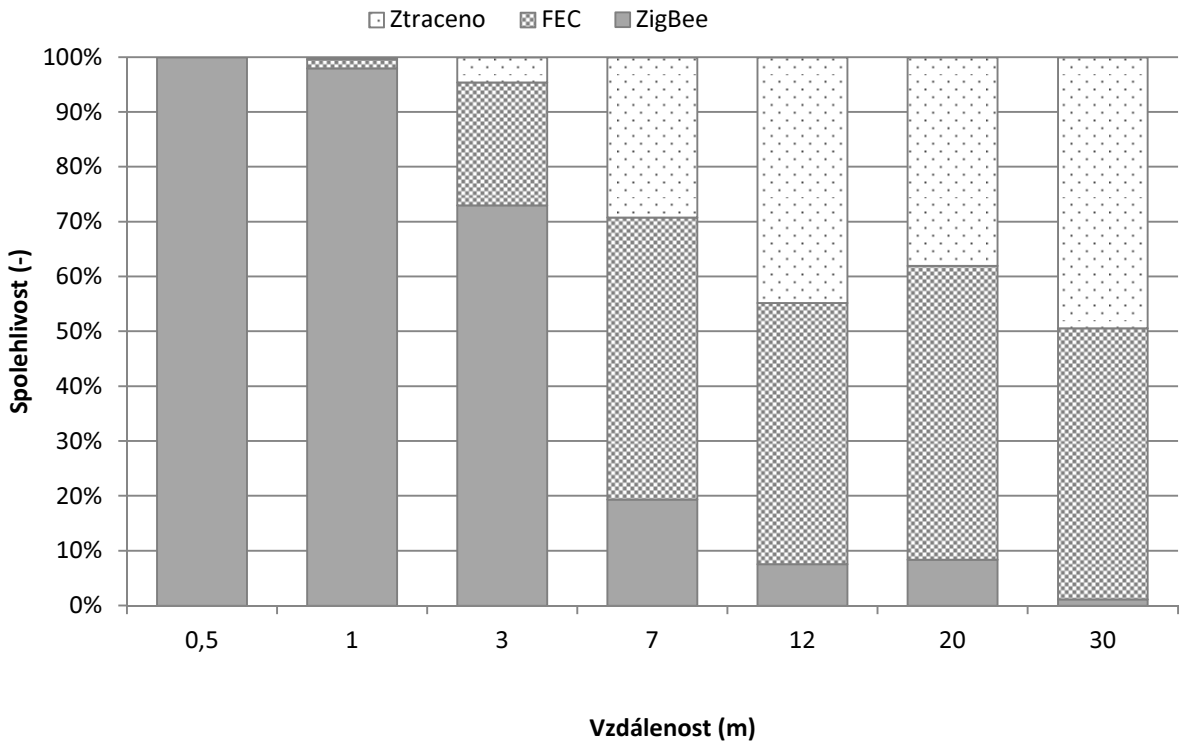
VII. Výsledky měření při koexistenci s WiFi 802.11g



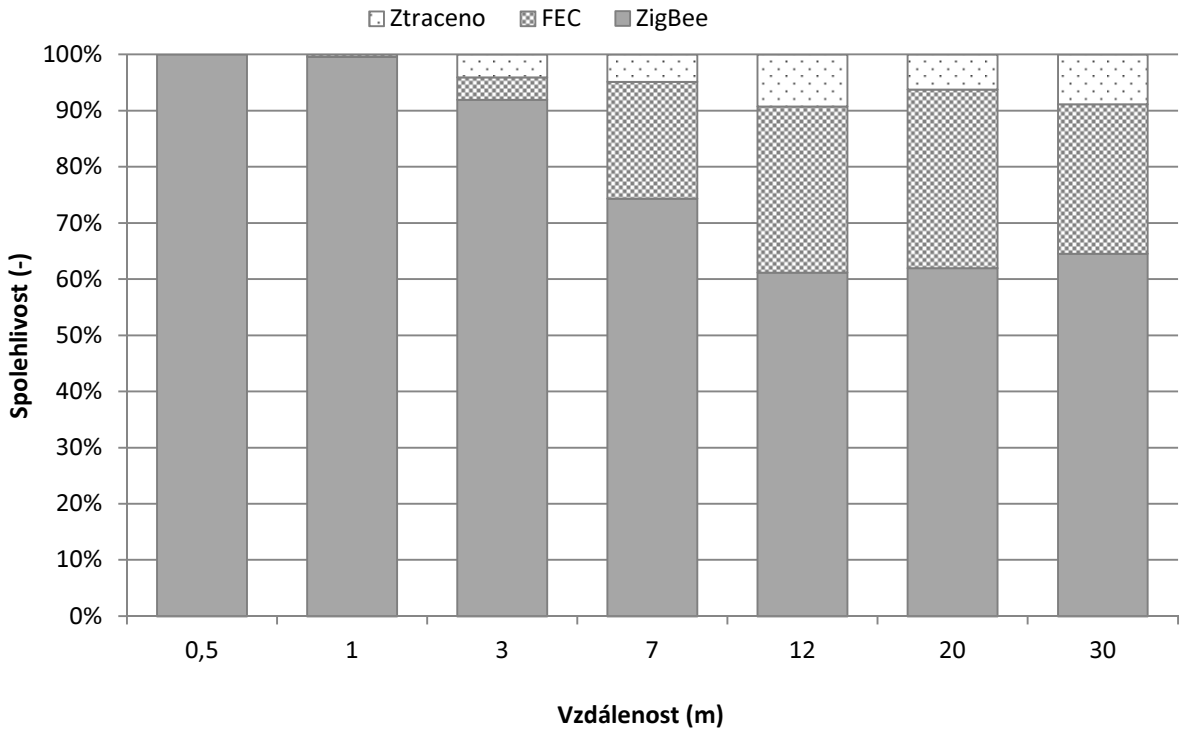
Obr. 1. Graf učinnosti FEC metody při koexistenci s payload 10 B a 0 ms



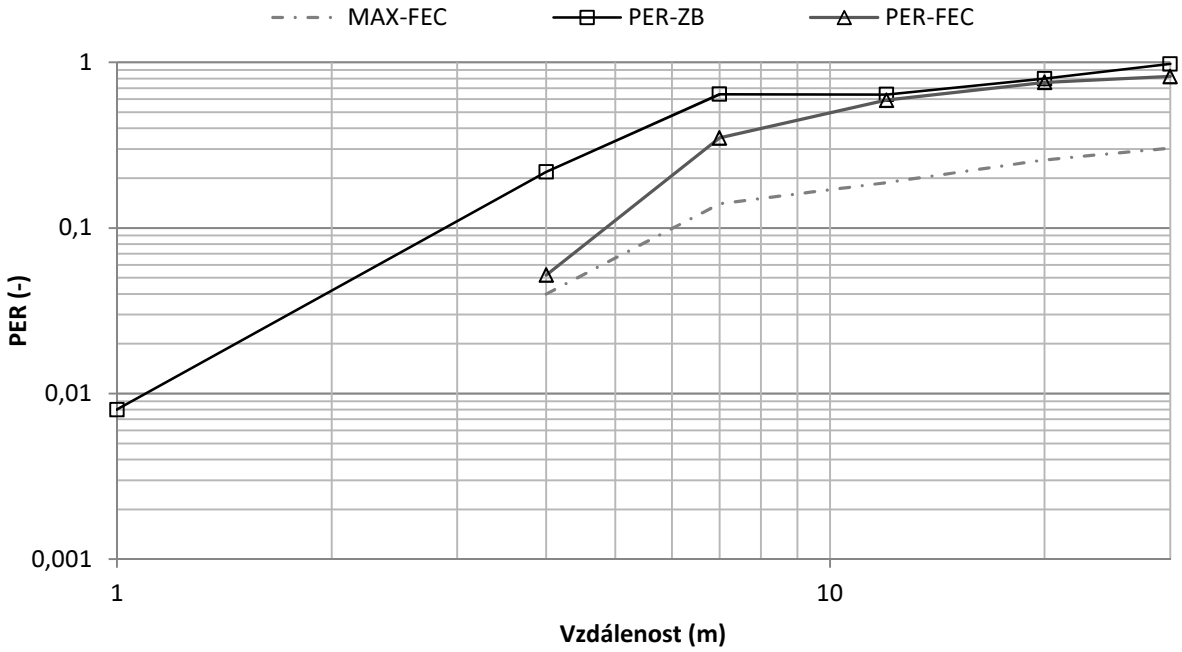
Obr. 2. Graf učinnosti FEC metody při koexistenci s payload 10 B a 1 ms



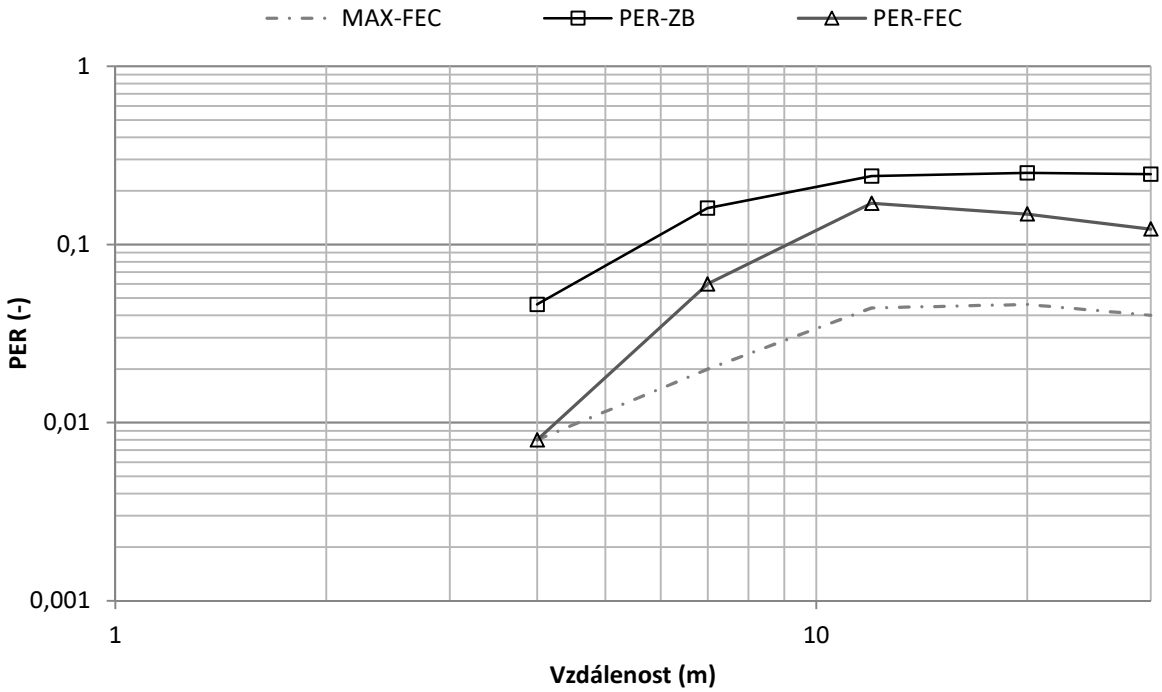
Obr. 3. Spolehlivost při koexistenci s payload 10 B a 0 ms



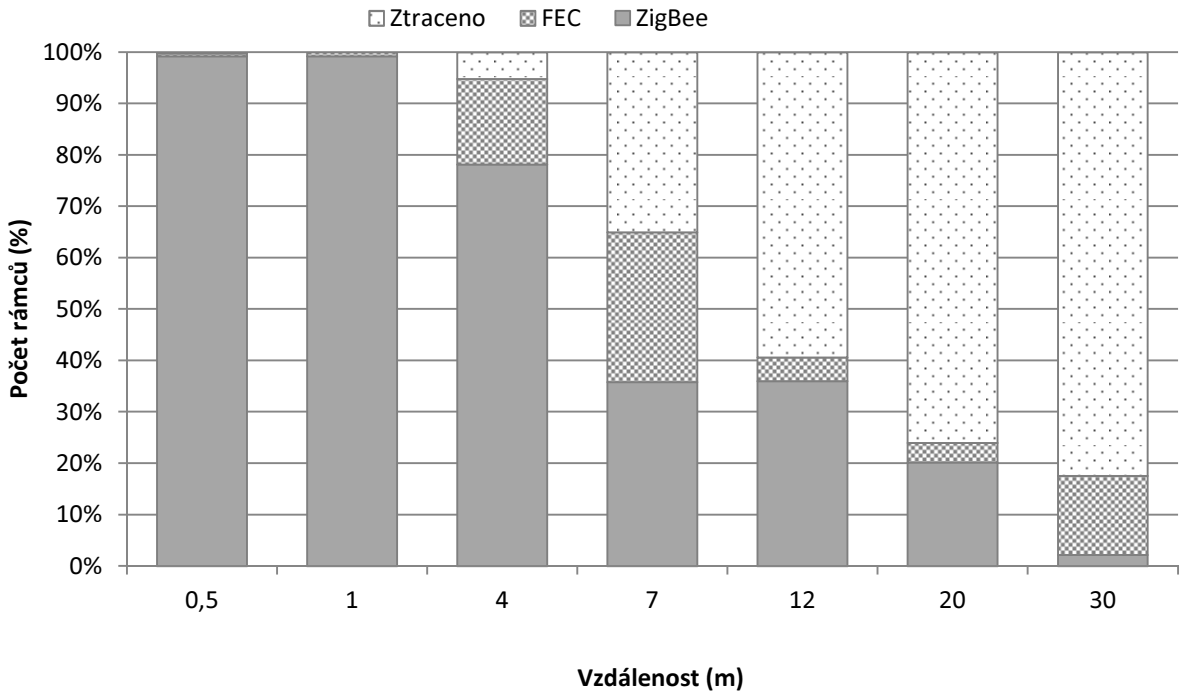
Obr. 4. Spolehlivost při koexistenci s payload 10 B a 0 ms



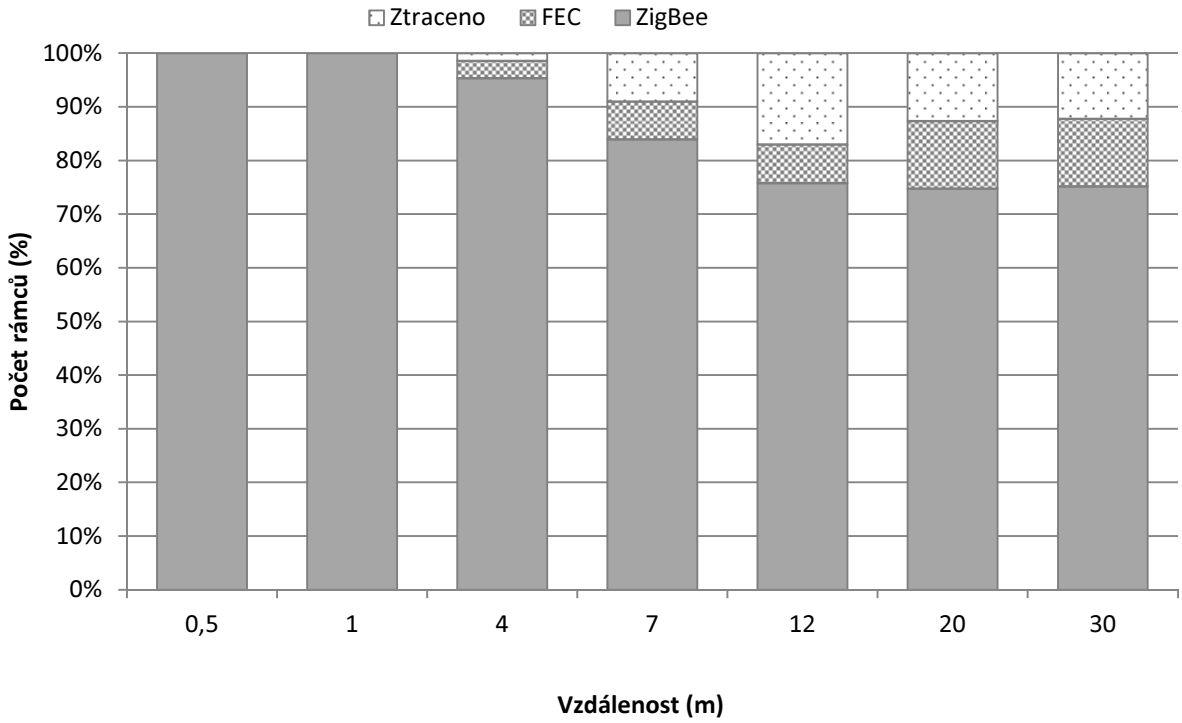
Obr. 5. Graf PER pro koexistenci s „500 B a 0 ms“



Obr. 6. Graf PER pro koexistenci s „500 B a 1 ms“



Obr. 7. Sloupcové vyjádření výsledků pro koexistenci s „500 B a 0 ms“



Obr. 8. Sloupcové vyjádření výsledků pro koexistenci s „500 B a 1 ms“