VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Vytvoření programového nástroje pro převod a zpracování naměřených dat z analyzátoru pro další využití.

Creating software tool for transfer and processing of mesurement data from analyzer for next usage

2010 Martin Prokeš

Prohlášení

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě dne 30.4. 2011 **……..……..**

Martin Prokeš

Poděkování

Tímto bych chtěl poděkovat vedoucímu své bakalářské práce, Ing. Přemyslu Merovi, PhD, za pomoc, poskytnuté materiály a veškerý čas, který mi věnoval.

Abstrakt

Tato práce je zaměřena na návrh a tvorbu aplikace sloužící ke zpracování a transformaci datových vstupů do podoby vhodné pro další zpracování. Vstupním souborem jsou data v textové podobě a výstupem pak strukturovaná data v tabulkách. Uživatel má možnost, pomocí nastavení filtrů a omezení, ukládat výstup dle různých parametrů . Vstupními daty jsou jednotlivé pakety zachycené při Bluetooth komunikaci, proto je v teoretické části této práce popsána problematika Bluetooth komunikace. Aplikace je vytvořena v objektově-orientovaném jazyce JAVA.

Klíčová slova

Bluetooth, protokol, paket, analýza, BTAnalyzer, java

Abstract

This thesis is focused on creating software tool for transfering and processing data input for next usage. Input text-oriented file is transformed into structured table. User may change output data form by changing filters and restrictions parameters. Input data consists of single packets of Bluetooth communication. So there is described Bluetooth technology in theoretical part of this thesis. Application is written in object-oriented programming language JAVA.

Keywords

Bluetooth, protocol, packet, analyze, BTAnalyzer, java

Seznam použitých symbolů a zkratek

ACL	-Asynchronous Connectionless Link – Asynchronní nespojově-orientované spojení
BT	-Bluetooth
CRC	-Cyclic Redundancy Check – Cyklický redundantní součet
DQPSK	-Diferencial Quadrature Phase Shift Keying – Diferenční kvadraturní fázové kódování
EDR	-Enchanced Data Rate – Vylepšený přenos dat
FEC	-Forward Error Correction – Dopředná korekce chyb
FHSS	-Frequency Hopping Spread Spectrum – Rozprostřené spektrum pomocí frekvenčních přeskoků
GFSK	-Gaussian Frequency Shift Keying – Gaussovské frekvenční klíčování
HEC	-Header Error Correction – Kontorla hlavičky proti chybám
HW	-Hardware
IEEE	-The Institute of Electrical and Electronics Engineers
ISM	-Industrial, Scientific and Medical
L2CAP	-Logical Link Control and Adaptation Protocol
LMP	-Link Management Protocol
OBEX	-Object exchange protocol
PSK	-Phase Shift Keying
SCO	-Synchronous Connection Oriented link – Synchrnonní spojově-orientované spojení
SW	-Software
WPAN	-Wireless Personal Area Network – Bezdrátová osobní síť

Obsah

1 Úvod

V bakalářské práci se zabývám tvorbou aplikace sloužící ke zpracování dat získaných pomocí HW&SW zařízení, které slouží pro zachytávání komunikace pomocí protokolu Bluetooth mezi zařízeními.

Studenti využívají v průběhu výuky nástroje sloužící k zachytávání paketů pro účely jejich dalšího zpracování a následné analýzy. I během krátké bezdrátové komunikace dojde k přenesení obrovského množství dat, která však jsou z větší části pro potřeby dalšího zpracování nadbytečná, je proto nutno tyto data následně ručně filtrovat a zdlouhavě získávat potřebné údaje. Cílem mé práce bylo navrhnout a realizovat nástroj, jenž by dle zvolených kritérií toto zpracování automatizoval.

V této práci se nejdříve seznámíme s popisem bezdrátové technologie Bluetooth. Jelikož je tato technologie v dnešní době velice rozšířená a využívaná, tak si uvedeme pouze stručný souhrn informací o její historii, vývoji a možnostech využití.

Další kapitola bude obsahovat popis jednotlivých typů paketů z datového výstupu Bluetooth protokolového analyzátoru, jelikož se jedná o stěžejní znalosti pro návrh výsledné aplikace a její správnou funkčnost.

V hlavní části této práce se seznámíte s požadavky na výslednou aplikaci, postupem vypracování a rozborem algoritmů využitých ve vlastní aplikaci. Jedna z částí se věnuje ukázkám zpracování dat v různých podobách.

Závěrečná část popisuje výsledky dosažené v této prácí a snaží se nastínit další možnosti využití aplikace popisované v této práci.

2 Technologie Bluetooth

2.1 Popis

Bluetooth je bezdrátová technologie určena ke komunikaci dvou či více zařízení na krátké vzdálenosti. V dnešní době umožňuje efektivně nahradit množství různých propojovacích kabelů pomocí jediného rádiového spojení. Technologie Bluetooth pracuje stejně jako jiná velmi populární technologie Wi-Fi v bezlicenčním pásmu ISM. Bluetooth byl navržen tak, aby jej bylo možno využívat i na zařízeních s omezeným zdrojem energie (mobilní telefony, bezdrátové klávesnice/myši, sluchátka apod.) [6],[7]

2.2 Historie

První úvahy nad vznikem nové bezdrátové technologie byly vedeny v roce 1994 v jedné z divizí firmy Ericsson. Mělo se jednat o technologii, která by vedla k nahrazení kabelového propojení mezi dvěma mobilními telefony, případně mezi mobilním telefonem a jiným zařízením. Práci na specifikaci se ujala skupina BSIG, která byla založena v roce 1998 firmami IBM, Toshiba, Intel, Ericsson a Nokia. V dnešní době má tato skupina více než 10 000 členů. První specifikace byla touto vývojovou skupinou uveřejněna v roce 1999 ve verzi 1.0a. [6],[7]

2.3 Vývoj technologií

Verze 1.0a se ovšem potýkala s velkým množstvím chyb, například problémy s kompatibilitou či implementací pikosítí. Proto až základě specifikace Bluetooth 1.1 uveřejněné v roce 2001 byl přijat organizací IEEE dne 14. Ledna 2002 standart pod označením IEEE 802.15.1, který byl závazný pro výrobce mobilních zařízení z důvodu vzájemné kompatibility. Skupina standartů 802.15 popisuje sítě WPAN, což jsou bezdrátové sítě pokrývající malé (osobní) oblasti s nízkým vysílacím výkonem a nízkými pořizovacími náklady, kromě Bluetooth zde patří i ZigBee (802.15.4) nebo UWB (802.15.3a). Posledním standartem pod hlavičkou IEEE byla upravená verze specifikace Bluetooth 1.2 (IEEE 802.15.1a). Po tomto vydání bylo skupinou IEEE jednoznačně odhlasováno, že se nebude pokračovat ve spolupráci se skupinou BSIG, což fakticky znamenalo, že následující verze Bluetooth již nebudou standardizovány pod hlavičkou IEEE. [6],[7]

Verze protokolu Bluetooth, ze které jsem při přípravě podkladů k výsledné aplikaci vycházel byla 2.0 EDR, v současnosti se jedná o nejrozšířenější verzi Bluetooth protokolu a je využívaná ve většině zařízeních podporujících Bluetooth komunikaci.

Dalším významným vývojovým krokem skupiny BSIG je technologie Bluetooth ve verzi 3.0, která mnohonásobně zvyšuje přenosovou rychlost až na teoretických 24 Mbit/s, ovšem s výrazným rozdílem a to, že pro přenos dat jsou použitý technologie standartu 802.11 – WiFi.

V současné sobě nejnovější specifikací je verze 4.0 z července roku 2010. V této verzi se počítá k návratu k původnímu poslání této technologie, čímž je velmi nízká spotřeba a cena. Předpokládá se výdrž až jeden rok při využití knoflíkové baterie.

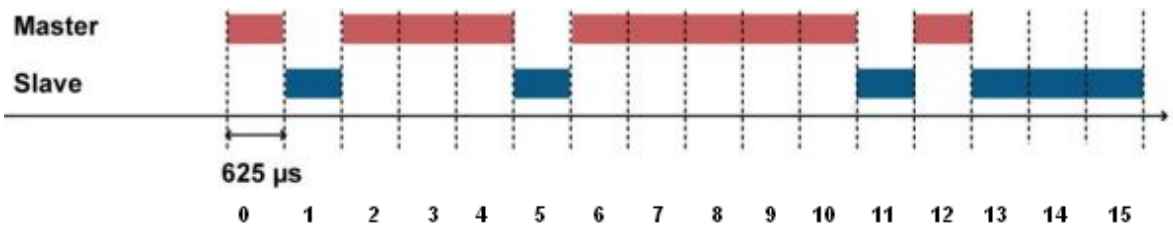
2.4 Protokol

V době vzniku této práce je nejrozšířenější a nejvyužívanější Bluetooth ve specifikaci Bluetooth v2.1 + EDR. Proto bude popis protokolu Bluetooth v této práci vycházet právě z této specifikace. Specifikace rozděluje protokol Bluetooth do jednotlivých vrstev odpovídajících vrstvám referenčního modelu OSI.

2.4.1 Fyzická vrstva

Fyzická vrstva OSI modelu je u Bluetooth rozdělena na 2 části – RF(rádiové rozhraní) a baseband vrstvu, tyto vrstvy slouží k vlastnímu přenosu informací mezi zařízeními a jejich zabezpečení proti rušení. Jak již zbylo zmíněno výše technologie Bluetooth pracuje v bezlicenčním pásmu ISM, pro které je vymezena frekvence 2,4 – 2,5 GHz, samotné Bluetooth pak využívá frekvence 2400 až 2483,5 MHz, které je rozděleno na 79 jednotlivých kanálů o šířce 1 MHz v rozmezí 2402 až 2480 MHz, které jsou ohraničeny ochranným pásmem. Bluetooth využívá metody Frequency Hopping Spread Spectrum – FHSS k potlačení možných interferencí v pásmu ISM, které je v současnosti velice hojně využívané s čímž souvisí jeho velká zarušenost. FHSS je založena na změně nosné frekvence dle pseudonáhodné sekvence, která musí být před začátkem vysílání dohodnuta mezi vysílačem a přijímačem. Výpočet této sekvence je určen MAC adresou a stavem řídících hodin master zařízení, tímto je zajištěna odlišná sekvence pro různé pikosítě a jejich možnost vzájemné koexistence. U této metody dochází k 1600 přeskokům za vteřinu. Ke změně frekvence dochází po odeslání každého paketu. Od verze 1.2 je využíváno Adaptivního FHSS, které umožňuje master zařízení vyřadit z posloupnosti frekvencí ty, které jsou nejvíce zarušeny, což vedlo k vyššímu potlačení interferencí a tím k nárůstu výkonu. [4]

Pro podporu duplexního provozu je využita metoda Time Division Duplex (TDD), při níž smí master zahájit komunikaci pouze na začátku sudého time slotu a zařízení typu slave pouze na začátku lichého time slotu, viz obrázek 2.1 [1]



Obázek 2.1 - Timesloty

Ve verzi je 2.0 je využita modulační metoda GFSK, jedná se o modulaci s klíčováním pomocí frekvenčního zdvihu s využitím Gaussova filtru, který slouží k zúžení výsledné šířky pásma. Pomocí této modulace lze dosáhnout rychlosti 1Mbps. Pro zvýšení navýšení přenosové rychlosti se dále používá modulace PSK ve dvou variantách a to π-4/DQPSK pro rychlosti do 2Mbps nebo

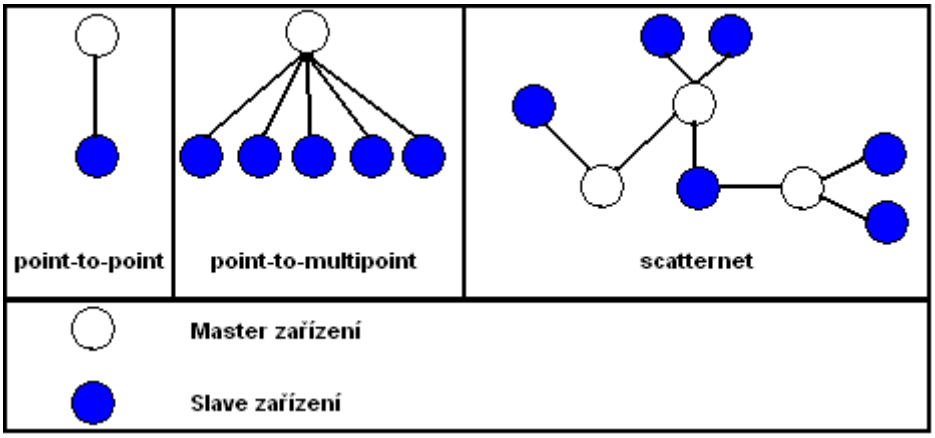
8DPSK pro rychlosti až 3Mbps. Při využití těchto modulací je pak se pak technologie označuje Bluetooth v2.0+EDR .

2.4.2 Linková vrstva

I tato vrstva je rozdělena v případě Bluetooth protokolu do dvou podvrstev. Jedná se o vrstvu LMP. která je zodpovědná za správu vzájemného propojení Bluetooth zařízení a vrstvu L2CAP, která slouží ke komunikaci mezi vyššími vrstvami a vrstvou Baseband.

Nejdříve si uvedeme informace o vrstvě LMP, která má na starost udržování spojení. Jednotlivá Bluetooth zařízení se mohou navzájem propojit, tím dojde ke vzniku piconetu (česky pikosít). Vztah zařízení v takové síti je postaven na principu master – slave, kde master řídí spojení (tok dat, posloupnost přeskoků). Podle vzájemného vztahu mezi zařízeními rozlišujeme následující 3 druhy pikosítí (Obr. 2.2):

- Point – to – point - síť kde jsou pouze dvě zařízení jedno typu master druhé je typu slave
- Point – to – multipoint – síť, ve které se nachází jedno zařízení typu master a až 7 zařízení typu slave, která jsou řízena master zařízením
- Scatternet – síť propojující navzájem několik pikosítí, mohou mít společné slave zařízení, nebo master zařízení jedné sítě může být v jiné síti slave zařízením



Obrázek 2.2 - pikosítě

Druhou částí je výše zmiňovaná podvrstva L2CAP, která sestavuje virtuální přenosové kanály mezi koncovými zařízeními, jenž poté slouží k samotnému simultánnímu přenosu informací (např. přenos dat nebo přenos hlasu). Dále připravuje data pro Baseband vrstvu a monitoruje spojení pro spojově a nespojově orientované služby. Vrstva L2CAP však sama přistupuje pouze k ACL (nespojově orientovaným asynchronním) přenosům vrstvy Baseband. ACL spojení jsou využívána k přenosu dat, která nejsou náchylná na zpoždění, ale hlavní důraz je kladen na integritu dat. Kvůli tomu je zajištěno opětovné vysílání paketů, které dorazily poškozené nebo nedorazily vůbec. Jednotlivé typy paketů budou detailně popsány v kapitole 3. [4]

Druhou možností přenosu je spojení SCO (synchronní spojově orientované), tento typ spojení je primárně určen pro hlasové přenosy, proto z důvodu snížení latence spojení není tento přenos

poskytován vrstvou L2CAP ,ale samostatně přistupuje na Baseband vrstvu. Další omezení latence je dosaženo neopakováním přenosu při poškození dat. K omezení množství chyb se používá metoda FEC (dopředná chybová korekce), která však díky vkládání redundantních dat vede ke snížení výsledné přenosové rychlosti. [1]

2.4.3 Vyšší vrstvy

Posledním protokolem patřícím do tzv. Bluetooth core [5], což je soubor povinně implementovaných protokolů v každém BT zařízení je protokol SDP, tento protokol slouží k rozpoznání jednotlivých zařízení a zjištění informací o službách, které BT zařízení poskytuje. Toto je veliké plus pro výrobce BT zařízení jelikož není potřeba implementovat kompletní sadu BT protokolů, ale jen určité moduly pro ně využitelné, což opět samozřejmě šetří náklady.

Jelikož původním posláním BT technologie bylo nahrazení klasický kabelů, tak došlo k implementaci protokolu RFCOMM, který umožňuje emulaci sériového portu RS-232, bez velkých úprav je proto možno využívat jiné přejaté protokoly jako je TCP/IP , OBEX nebo AT commands.

2.5 Bluetooth profily

Profily slouží k určení technických možností jednotlivých zařízení a k určení jejich vzájemné slučitelnosti. Profily určují využití jednotlivých protokolů pro potřeby komunikace. Toto umožňuje implementaci jen těch protokolů, které jsou pro dané zařízení potřebná – např. BT myš nepotřebuje implementovat protokoly pro přenos hlasu jako například BT sluchátka. [8]

Základním profilem je GAP (Generic Access Profile), definuje 3 základní úkoly, které jsou platné i pro všechny ostatní profily. 1) Zavedení definic, doporučení a základních požadavků pro přístupové procedury. 2)Popisuje chování zařízení ve standby a připojovacím módu, aby bylo možno navázat spojení. 3) Kódování schémat, jmen procedur a parametrů.[2]

Dalším důležitým profilem je SDAP (Service Discovery Aplication Profile), který slouží k lokalizaci služeb v dosahu BT zařízení a jejich nabídnutí uživateli.

Velký soubor profilů vychází ze společného profilu SPP (Serial Port Profile) – profil popisuje možnost propojení zařízení pomocí sériového portu. Profily spadající do této oblasti jsou např. HFP (Hands-free profile), HP (Headset profile) nebo GOEP (Generic Object Exchange Profile), který opět obsahuje skupinu profilů sloužících k přenosu objemných dat. [2]

2.6 Sestavení spojení

Pod pojmem sestavení spojení je obsažen celý proces od vyhledání okolních zařízení v dosahu, jejich synchronizaci a následný vznik pikonetu. Proces vyhledávání zařízení se nazývá inquiry. Zařízení ve stavu inquiry naslouchá na jednotlivých frekvencích, zda jiné zařízení nevysílá svůj inquir požadavek. Je-li inquiry požadavek zachycen, pak zařízení odešle zpět požadavek page, ve kterém odesílá hodnotu svých vnitřních hodin a svou adresu zařízení. Každé standardní Bluetooth zařízení může pracovat v režimu MASTER i v režimu SLAVE, proto je nutno nejprve určit v jakém vztahu zařízení budou. Zařízením MASTER se stane to zařízení, které požadavek

iquiry vyslalo jako první a SLAVE pak zařízení, které na tento požadavek odpovědělo. Poté dojde k vytvoření FHSS sekvence pro následnou komunikaci. [1]

Jelikož je udržování spojení energeticky dosti náročné může proto po navázání spojení dojít k přechodu SLAVE do jednoho ze tří nízko-energetických módů – jedná se o sniff, hold a park.

Sniff mód – pokud se SLAVE dostane do sniff módu, dojde ke snížení četnosti naslouchání přijímačem. [1]

Hold mód – Pokud není potřeba přenášet žádná data může MASTER zařízení nastavit SLAVE mód hold. Kdy neprobíhá žádný přenos a SLAVE zařízení je řízeno svými vlastními vnitřními hodinami. Při tomto stavu dojde k faktickému vystoupení z pikonetu až do doby expirace hold módu kdy SLAVE zařízení musí opět provést standardní komunikační a udržovací procesy.[1]

Park mód – Ke komunikaci mezi MASTER a SLAVE zařízením dochází pouze v periodických prodloužených intervalech kdy dojde k desynchronizaci vnitřních hodin podle MASTER zařízení.[1]

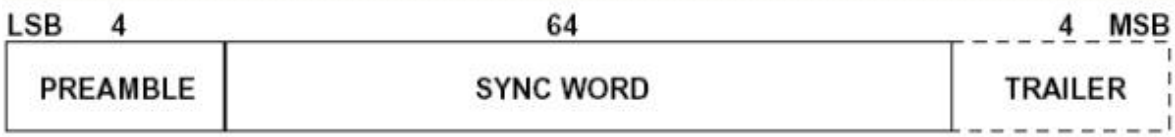
3 Pakety Bluetooth

3.1 Paket

Data mezi zařízeními jsou posílaná ve formě paketů, za jejichž sestavení a předání rádiové vrstvě je zodpovědná Baseband vrstva. Jednotlivé bity paketu jsou na rádiové rozhranní odesílány v pořadí od nejméně významného bitu (LSB) po nejvýznamnější (MSB). Základní struktura paketu je stejná pro všechny aplikace a je tvořena třemi částmi – přístupovým kódem, záhlavím a užitečnou informací (viz obrázek 3.1).

Obrázek 3.1 - Paket

Přístupový neboli access code je jedinou povinnou části paketu a slouží k resynchronizaci vnitřních hodin SLAVE zařízení, synchronizaci bitů a bitových slov a k identifikaci v rámci piconetu. Přístupový kód opět můžeme rozdělit na 3 části – Preambule, synchronizační slovo a Trailer (Obrázek 3.2). Acces code může mít délku 68 nebo 72 bitů. Délku 68 bitů má v případě, že Access code není následován hlavičkou ani daty. Tento paket pak slouží pouze k synchronizaci a hlavička neobsahuje část Trailer. [1]



Obrázek 3.2 - Access Code

Preambule může nabývat pouze hodnot 1010 nebo 0101 v závislosti na LSB synchronizačního slova. Základem pro výpočet synchronizačního slova je adresa zařízení tzv. BD_ADDR o délce 48 bitů. Trailer pak slouží k určení kompenzace napětí.

Header slouží pro řízení provozu na úrovni Link Manageru, slouží k rozlišení zařízení v síti, určuje typ paketu a obsahuje příznaky k řízení přenosu (Obrázek 3.3).

Obrázek 3.3 - Header

AM_ADDR slouží k rozlišení cílového zařízení. Pokud zařízení typu SLAVE obdrží od MASTER a paket s neodpovídající AM_ADDR může ukončit příjem ihned po načtení hlavičky a vyčkat na další příslušný time slot.[1]

TYPE umožňuje definovat až 16 různých typů paketu. Podle typu lze rozpoznat počet time slotů obsazených jedním paketem což opět umožňuje zařízením, kterým není paket určen, zastavit příjem na delší dobu, která je závislá od počtu využitých time slotů.[1]

FLOW určuje možnost přijímače přijímat další pakety. Například pokud přijímač nestíhá zpracovávat příchozí informace a jeho buffer je zaplněn, dojde k nastavení FLOW = 0 u potvrzujícího paketu, což znamená, že zařízení přijímá pouze určité typy paketů (ID, POLL, NULL). (viz kapitola 3.2) [1]

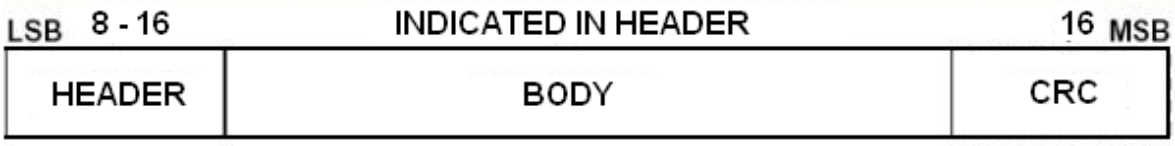
ARQN slouží k oznámení informace zdroji, že paket byl přijat správně (ARQN=1) nebo se vyskytla chyba (ARQN=0).

SEQN zabraňuje duplikaci paketů, střídavě musí nabývat hodnot 0/1.

HEC slouží ke kontrole integrity hlavičky.

Ačkoliv je celková délka hlavičky pouze 18 bitů (10 informačních + 8 HEC), pro zajištění správného přenosu je však použito trojnásobně opakující kódování a z tohoto důvodu je ve výsledku přeneseno 54 bitů.

Poslední a největší částí paketu je PAYLOAD – vlastní nesená data, délka nesené informace je v závislosti na typu paketu různá a může nabývat hodnot 0-2745 bitů. PAYLOAD můžeme rozdělit opět na 3 části a to Header, Body a CRC code.[1]



Obrázek 3.4 - Payload

Struktura PAYLOAD je různá a závisí na typu paketu, zda li se jedná o paket SCO, ACL nebo FHS. Délka vlastního těla paketu je definovaná v hlavičce PAYLOADu, zvětšení délky je umožněno využitím více slotových paketů. Další informace k problematice rozdělení paketů získáte v následující kapitole. CRC součet pak slouží jako jednoduchý nástroj pro kontrolu přenesených dat.[1]

3.2 Typy paketů

Jednotlivé typy paketů můžeme standardně rozdělit do tří kategorií. Pakety, které slouží pro komunikaci na ACL spojení, pakety pro SCO spojení a na pakety společné pro oba typy spojení.

3.2.1 Společné pakety

- ID

Paket o délce 68 bitů, využíván pro vyhledání zařízení (Inquiry), navázání spojení (Paging) nebo potvrzení spojení (Response). Nese informace o přístupovém kódu zařízení (DAC) nebo o přístupovém kódu pro vyhledání (IAC). Jelikož paket neobsahuje hlavičku ani data je možno jej odeslat až dvakrát během jednoho časového slotu, což urychluje vyhledávání zařízení v okolí.[1][4]

- NULL

Paket má délku 126 bitů a je složen pouze z přístupového kódu a hlavičky. Slouží k přenosu informací o spojení, k potvrzení správného příjmu nebo k informaci o stavu přijímacího bufferu (FLOW). [1][4]

- POLL

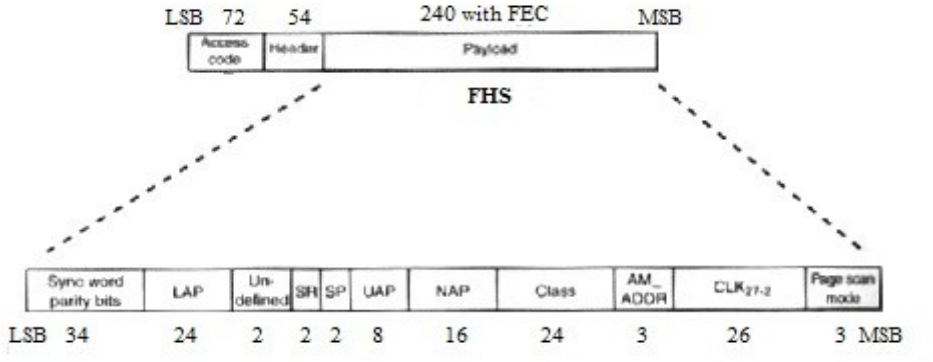
Strukturou a délkou je shodný s paketem NULL. Na rozdíl od NULL paketu však musí být přijímací stranou potvrzen. [1][4]

- FHS

Jedná se o synchronizační paket předávající adresu zařízení (BD_ADDR) a vnitřní hodiny zdrojového zařízení za účelem synchronizace sekvence frekvenčních přeskoků. Délka celého paketu je 366 bitů z čehož je 144 bitů vlastních dat doplněných o 16 bitů kontrolního součtu FCS dohromady zabezpečených pomocí 2/3 FEC ochrany (dohromady 240 bitů). Zbylých 126 bitů je tvořeno hlavičkou a přístupovým kódem. Jeden FHS paket je přenesen během jednoho timeslotu. [1][4]

- DM1

Slouží pro podporu řídících zpráv, může přenášet i normální uživatelská data. Na SCO spojích může přerušit informační tok pro zaslání řídících informací. Velikost paketu je 366 bitů a je přenesen během jednoho timeslotu. [1][4]



Obrázek 3.5 - Struktura FHS paketu

3.2.2 SCO pakety

- HV1,2,3

Jedná se o pakety sloužící k přenosu hlasu. Všechny typy mají stejnou délku nesené informace (PAYLOAD), ale liší se podle použité FEC ochrany. HV1 - 10 bytů informace při použití 1/3 FEC ochrany, to odpovídá 1,25 ms řeči při kvalitě 64kbps. HV2 – 20 bytů informace při použití 2/3 FEC ochrany – 2,5ms řeči při 64 kbps. HV3 – přenáší 30 bytů informace nechráněné pomocí FEC, odpovídá 3,75 ms řeči. Celková délka paketu včetně hlavičky a přístupového kódu je 366 bitů a paket je přenesen během jednoho timeslotu. [1][4]

- DV

Data – voice paket umožňuje kombinovaný přenos užitečných dat i hlasu. Pro hlasovou část je zde vyčleněno 80 bitů bez ochrany FEC. Hlasová část se chová jako klasický SCO paket – nikdy není znova přenášen. Pro data je pak vyčleněno 150 bitů, které obsahují 2/3 FEC ochranu.Datová část je po přijetí kontrolována a při chybě znovu přenášena. Délka paketu je maximálně 366 bitu a paket je přenesen během jednoho timeslotu. [1][4]

3.2.3 ACL pakety

- DM 1,3,5

Data-medium rate paket je určen pouze k přenosu dat. DM 1 paket může nést až 18 bytů užitečných dat. Data obsahují 16 bitový kontrolní součet CRC a jsou zabezpečeny 2/3 FEC kódováním. DM3 paket má stejnou strukturu, pro přenos paketu je však využito tří následujících timeslotů, čímž dojde k navýšení délky nesené informace až na 123 bytů. DM5 pak využívá 5 timeslotů a může nést informaci o délce až 226 bytů dat. [1][4]

- DH 1,3,5

Data-high rate paket vychází z paketu DM, data však nejsou kódována pomocí FEC, což vede k dalšímu nárustu množství přenášených dat. DH1 až 28 bytů v jednom timeslotu, DH3 až 185 bytů ve třech timeslotech a DH5 až 341 bytů v pěti timeslotech. [1][4]

- AUX1

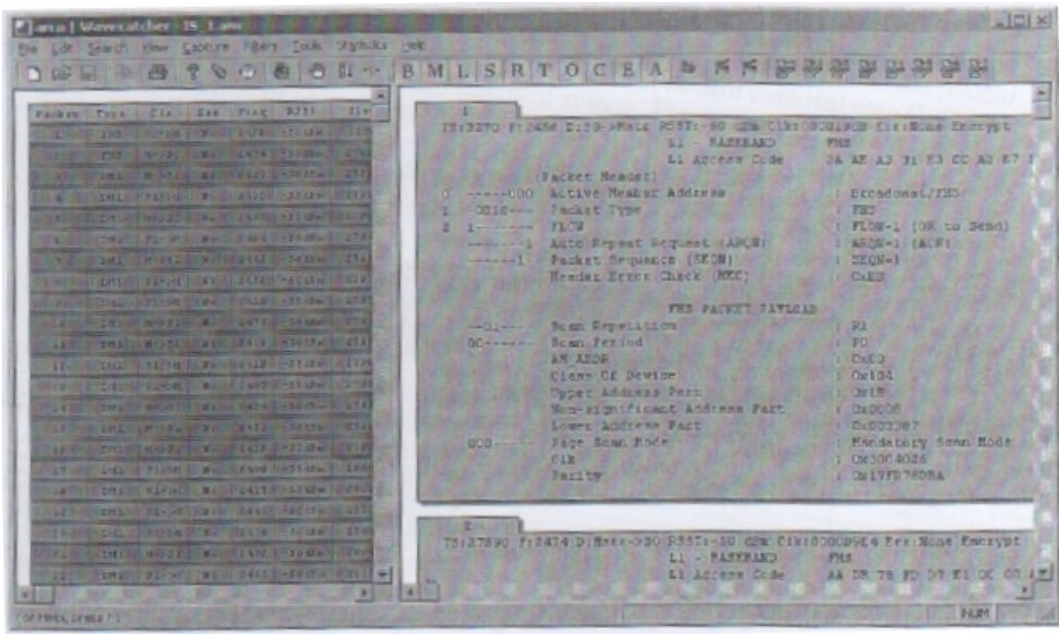
Struktura stejná jako u paketu DH1 avšak neobsahuje kontrolní součet CRC čímž dojde k nárůstu nesené informace na 30 bytů. Je přenášen během jednoho timeslotu. [1][4]

4 Tvorba aplikace

V této kapitole se seznámíte s celkovým průběhem vývoje výsledné aplikace. Celá kapitola bude rozdělena do několika částí, v nichž se budeme zvlášť věnovat analýze požadavků dle zadání této závěrečné práce a návrhem jejich zpracování, v další části se seznámíme se strukturou vstupních dat, jejichž zpracování je cílem této práce. Projdeme si uživatelské rozhraní a dále bude rozebrána funkčnost jednotlivých ovládacích prvků GUI.

4.1 Analýza požadavků a vstupu

Tato aplikace je určena pro zpracování výstupních dat pořízených pomocí Bluetooth protokolového analyzátoru arca|Wavecatcher, který slouží k zachytávání komunikace mezi zařízeními využívajícími technologii Bluetooth. Jedná se o hardwarové zařízení pro vlastní detekci přenosu a aplikaci sloužící ke zpracování dat zachycených analyzátorem. Při využití proprietárního software arca|Wavecatcher je možná jednoduchá aplikace různých filtrů sloužících k rychlému procházení a třídění získaných paketů dle požadavků uživatele (Např. podle typu paketu nebo podle účastníků komunikace).[3] Viz obrázek 4.1 zobrazující zachycené pakety v přehledné tabulce.

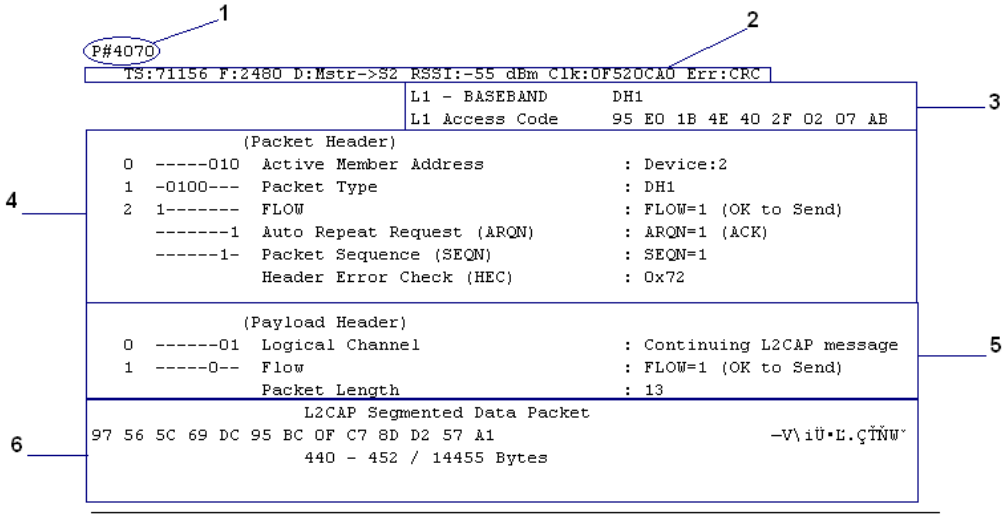


Obrázek 4.1 - Arca|Wavecatcher

Výsledná aplikace by měla umožňovat podobné možnosti práce se zaměřením především na filtrování paketů podle jejich typu a směru přenosu. Další zpracovávanou součástí by mělo být automatizované získávání informací o přenosu. Tímto se rozumí především doba přenosu, počet jednotlivých přenesených paketů nebo průměrná úroveň. Mezi požadavky v zadání je i možnost grafického výstupu. Mělo by se jednat například o grafické zaznamenání průběhu frekvence v závislosti na čase. Nebo zachycení průběhu změny přijímací úrovně signálu v čase.

Smyslem aplikace je umožnit automatizovanou práci s naměřenými daty i mimo prostředí arca|Wavecatcher.

Pro další práci se získanými daty se využívá funkce aplikace arca|Wavecatcher Export to Decode [3], která provede uložení získaných dat do textové podoby. Struktura tohoto souboru je zobrazená na obr.4.2.



Obrázek 4.2 - Struktura textového souboru

Každý paket je zaznamenán stejně, jako je zobrazeno na obrázku 4.2. Jednotlivé pakety jsou v souboru odděleny horizontální čárou.

1 – Číslo paketu

2 – Informace získané z Baseband vrstvy protokolu (Čas, Frekvence, Směr, Úroveň RSSI, Hodnota CLK zařízení, informace o výskytu chyby

3 – Access code a typ paketu

4 – Obsah záhlaví paketu

5 – Obsah záhlaví užitečné informace

6 – vlastní přenášená data

Tato struktura se může lišit podle typu paketu a nastavení aplikace Arca|Wavecatcher. V prvním případě může dojít k absenci částí 4,5 a 6 v případě paketu typu ID nebo části 5,6 pokud se jedná o paket typu POLL. Ve druhém případě je formát ovlivněn možností vypnout dekódování informací Baseband vrstvy v aplikaci Arca|Wavecatcher. A paket obsahuje pouze záhlaví – část 1 a 2 a zbytek paketu je nedekódován a zobrazen v hexadecimálním tvaru. Obě výše popsané možnosti jsou zobrazeny na obrázku 4.3.

Pro návrh výsledné aplikace jsou podstatné pouze části 1, 2 a 3 jelikož obsahují všechny potřebné informace pro další zpracování. Dalším zpracováním je myšlena filtrace jednotlivých paketů dle volby uživatele (podle směru nebo typu paketu) a zapsání dat do výstupního souboru ve formě tabulky. Dále aplikace statistických funkcí např. doba přenosu, počet přenesených paketů, četnost výskytu paketů apod.

```
P#2062
   TS:59612 F:2461 D:Mstr->S2 RSSI:-50 dBm Clk:0F51B270 Err:CRC
                              L1 - BASEBAND      ID
                              L1 Access Code     95 E0 1B 4E 40 2F 02 07 AB
```

```
P#2073
   TS:59678 F:2410 D:Mstr->S2 RSSI:--- Clk:0F51B2F4 Err:None
                              L1 - BASEBAND      POLL
                              L1 Access Code     95 E0 1B 4E 40 2F 02 07 AB
            (Packet Header)
   0  -----010  Active Member Address             : Device:2
   1  -0001---  Packet Type                       : POLL
   2  1-------  FLOW                             : FLOW=1 (OK to Send)
      -------1  Auto Repeat Request (ARQN)        : ARQN=1 (ACK)
      ------0-  Packet Sequence (SEQN)            : SEQN=0
                Header Error Check (HEC)         : 0x8F
```

```
P#2
   TS:000:00:00.001786 F:2402 D:Mstr->S0 RSSI:--- Clk:09DE65AD Err:None Encrypt
   Raw Hex Packet
   95 3B 0E D0 F7 10 00 00 AB 00                          •;.Đ÷...«.
```

Obrázek 4.3 – Různé typy paketů (ID, POLL a vypnuté dekódování)

Výsledná zpracovaná data budou dále sloužit pro další analýzu průběhu komunikace a ověření teoretických znalostí získaných studiem protokolu Bluetooth.

Pro další využití nejsou podstatná data v částech 4 až 6 (dle obrázku 9) a proto mohou být ze zpracování vypuštěna. Vypuštěná data tvoří dle odhadu 60 až 80 % z celkového množství dat vložených na vstupu. Toto velice snižuje celkovou jak paměťovou tak výpočetní náročnost.

4.2 Struktura a prostředí aplikace

Pro tvorbu aplikace byl zvolen programovací jazyk JAVA. Hlavním důvodem vedoucím k této volbě byla multiplatformita výsledné aplikace. Při tvorbě aplikace byl kladen důraz především na rychlost zpracování dat, v další řadě na vytvoření přehledného uživatelského rozhraní pro pohodlnou práci.

Pro tvorbu aplikace byla zvolena architektura Model-View-Controller (MVC), jedná se o architekturu, která rozděluje aplikaci na datový model, uživatelské rozhraní a řídící logiku. Jedná

se o tři nezávislé komponenty, modifikace jedné z komponent má minimální vliv na komponentu jinou.

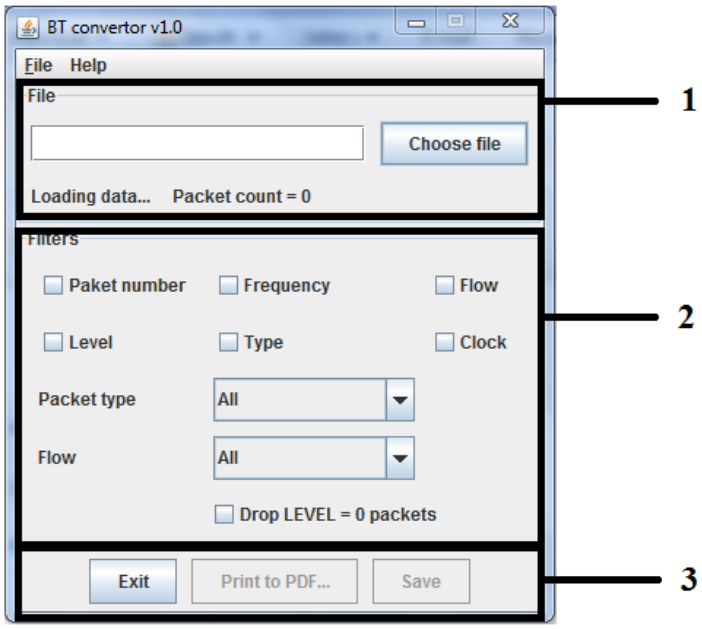
- **Model** - doménově specifická reprezentace informací, s nimiž aplikace pracuje. V tomto případě se bude jednat především o třídu zpracování datového vstupu a načtení jednotlivých paketů.

-View – převádí data reprezentována modelem do podoby vhodné k interaktivní prezentaci uživateli. Ve výsledné aplikaci se bude jednat o uživatelské rozhraní sloužící k ovládání aplikace.

-Controller - reaguje na události (typicky pocházející od uživatele) a zajišťuje změny v modelu nebo v pohledu.

Pro uživatelské rozhraní byla zvolena struktura jednoho formuláře. Hlavním důvodem byla jednoduchost a intuitivní ovládání, což by při použití několika formulářů pro různá nastavení bylo narušeno. Formulář můžeme rozdělit do 3 logicky oddělených částí (viz. Obrázek 4.4).

1) Volba souboru a textová zpráva o průběhu zpracování
2) Nastavení filtrů
3) Uložení výstupu



Obrázek 4.4 – Náhled formuláře

4.3 Spuštění aplikace

Aplikace se skládá z balíku BTanalyzer.jar a externích knihoven uložených v adresáři lib. Z důvodu pohodlnější distribuce je aplikace včetně knihoven přenášena v ZIP archívu.

1) Nejprve extrahujte archív do zvolené složky

2) V příkazovém řádku zadejte příkaz v následujícím formátu

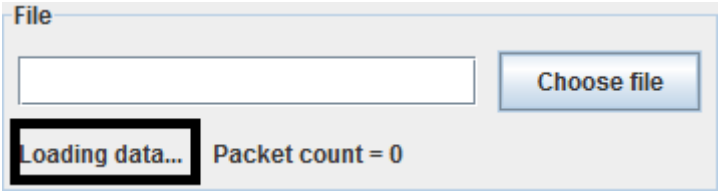
java –jar X:\cesta_kadresari\BTanalyzer.jar

Pro správnou funkčnost aplikace je nutno mít na PC nainstalován minimálně balík Java Runtime Enviroment ve verzi JRE6u24. Dostupné na http://java.com/en/download/index.jsp.

4.4 Ovládání

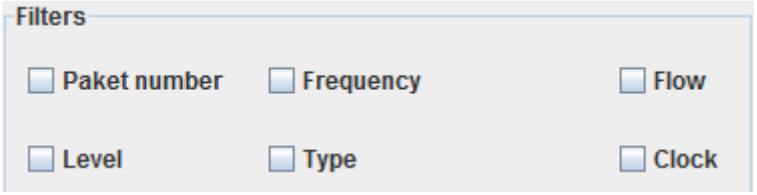
1) Otevření souboru – Stisknutím tlačítka (Open…) otevřete dialogové okno pro volbu souboru se vstupními daty. Podporovány jsou soubory v textovém formátu s koncovkou *.txt, musí se jednat o výstupní soubory aplikace Arca|Wavecatcher při použití funkce Export To Decode.

2) Zpracování souboru – Po volbě vstupního soboru dojde ke zpracování dat, o jehož průběhu jste informování textovou zprávou (viz Obr. 4.5) Po úspěšném načtení souboru se nastaví počítadlo paketů na příslušnou hodnotu.



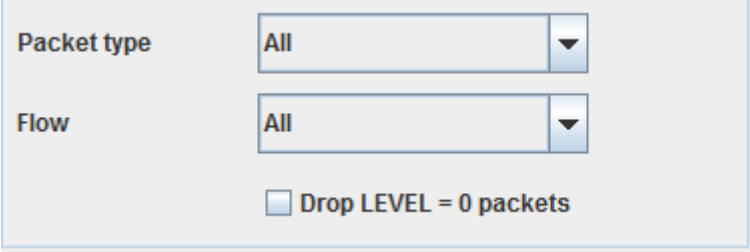
Obrázek 4.5 – Textová zpráva

3) Atributy – Jednotlivá zaškrtávací políčka označují příslušné atributy paketů. Označením jednotlivých políček ovlivníte strukturu výstupních dat. Pokud není některá z voleb aktivní (nelze ji zaškrtnout), pak nelze tento atribut ze zdrojového souboru získat (Viz kap. 4.1)



Obrázek 4.6 - Atributy

4) Filtry – Aplikace analýzou vstupních dat získá množinu všech možných hodnot vyskytujících se u atributů Type a Flow. Uživatel může z nabídky zvolit příslušný typ nebo směr paketu. Pouze pakety se zvolenými hodnotami budou zahrnuty v konečném výstup. Pomocí zaškrtávacího políčka „Drop Level = 0 Packet“ dojde k vypuštění paketu, u kterých nebyla během přenosu zjištěna úroveň signálu.



Obrázek 4.7 - Filtry

5) Uložení výstupu – Po stisknutí tlačítka (Save…) dojde k otevření dialogového okna sloužícího k volbě adresáře a zadání názvu výstupního soboru. O průběhu ukládání jste opět informování textovou zprávou v horní části aplikace. Druhou možností je uložení informací o zvolených paketech ve formátu PDF.

4.5 Externí knihovny

Jelikož je aplikace zaměřena pouze na zpracování dat bylo využito externích knihoven pro zápis výstupu do souboru typu XSL a PDF. V této části se s těmito knihovnami blíže seznámíme.

4.5.1 Zápis do PDF

Jedná se o knihovnu umožňující tvorbu a manipulaci s dokumenty PDF. V základní verzi se jedná o knihovnu psanou v jazyce JAVA, která však z důvodu velké popularity je pravidelně převáděna (portována) i do jazyka C#, kde se vyskytuje pod názvem iTextSharp. Knihovna umožňuje tvorbu dokumentů podle přednastavených šablon nebo automatický převod z XML souborů. Pro aplikaci postačilo využití základní funkce zápisu do dokumentu po jednotlivých odstavcích. V aplikaci byla použita verze iText 2.1.5. Knihovna je distribuována pod licencí GNU AGPLv3 umožňující její využití za podmínky veřejného šíření zdrojového kódu aplikace. Příklad syntaxe :

```
Document document = new Document();
PdfWriter.getInstance(document, new FileOutputStream("HelloWorld.pdf"));
document.open();
document.add(new Paragraph("Hello World"));
document.close();
```

Mezi další možné využití knihovny iText patří online tvorba PDF interaktivních formulářů v reálném čase. Možnost spojování několika PDF dokumentů atd. Více informací lze získat na webu www.itextpdf.com

4.5.2 Zápis do XLS

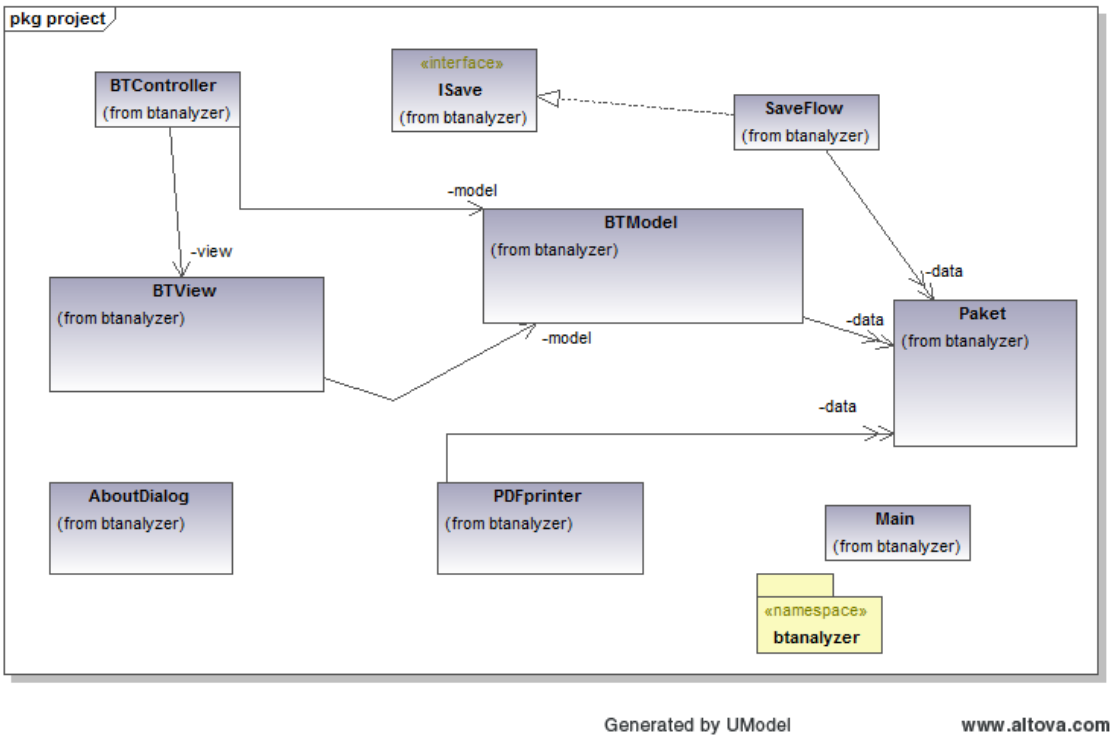
Java Excel API je open-source knihovnou sloužící ke čtení, zápisu a úpravě dokumentů tabulkových kalkulátorů ve formátu XLS. Mimo klasického zápisu na disk umožňuje také odeslání výstupu pomocí HTTP, zápis do databáze nebo využití jiného socketu. V aplikaci je využita knihovna jxl 2.6.10. Knihovna je distribuována pod licencí GNU LGPL. Příklad syntaxe:

```
import jxl.*;
import jxl.write.*;

WritableWorkbook workbook = Workbook.createWorkbook(new File("output.xls"));
WritableSheet sheet = workbook.createSheet("First Sheet", 0);
Label label = new Label(0, 2, "A label record");
sheet.addCell(label);
workbook.write();
workbook.close();
```

4.6 Rozbor

Následující obrázek zachycuje pomocí UML modelování vztahy jednotlivých tříd v balíku BTAnalyzer. Základní kostru tvoří skupina tříd MVC modelu a to BTController, BTView a BTModel. Z obrázku je dále patrné, že třída BTModel pro své zpracování dat využívá dalších tříd Paket, SaveFlow a PDFprinter.



Obrázek 4.8 – UML diagram

V následujících podkapitolách budou rozebrány požité třídy a popsány některé zajímavé části zdrojového kódu.

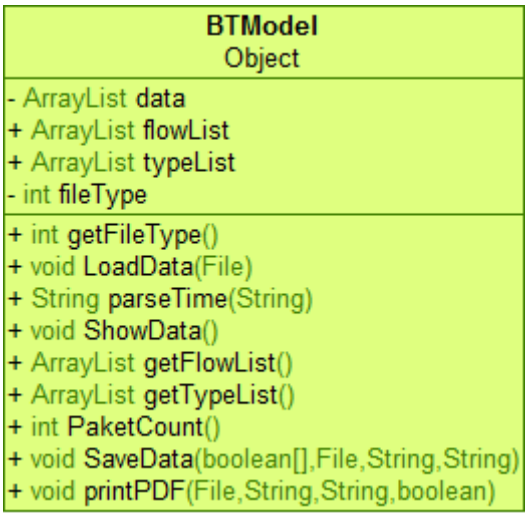
4.6.1 Třída BTModel

Třída BTModel obstarává načtení vstupního souboru, jeho zpracování a následný zápis dat do souboru ve formátu XSL nebo PDF.

```
public class BTModel {

    private ArrayList<Paket> data = new ArrayList<Paket>();
```

```
    ArrayList<String> flowList = new ArrayList<String>();
    ArrayList<String> typeList = new ArrayList<String>();
    private int fileType;
…
```



Obrázek 4.9 - BTModel

V kolekci *data* jsou uloženy jednotlivé pakety. Kolekce *flowList* a *typeList* obsahují seznamy typů a směrů paketů vyskytujících se ve zpracovávaném souboru. Metoda *LoadData(File name)* prochází jednotlivé řádky vstupního souboru zadaného v parametru a získává z něj jednotlivé pakety, jenž poté ukládá do kolekce *data* jako instanci třídy *Paket.* Jednotlivé atributy paketu získá rozdělením řetězce metodou *str.split(" ").*

```
public void LoadData(File name)
{
…
while ((str = br.readLine()) != null) {
            if (str.startsWith("P#")) {
                Paket pak;
                pNumber = str.substring(2, str.length());
                str = br.readLine();
                splitter = str.split(" ");
                int len = splitter[3].length();
                String tmp =splitter[3].substring(3,len);
                pTime =parseTime(tmp);
                pFlow = splitter[5].substring(2);
                if(!flowList.contains(pFlow))
                {
                    flowList.add(pFlow);
                }
                pFreq = splitter[4].substring(2);
                pLevel = splitter[6].substring(5);
                str = br.readLine();
                str = str.trim().substring(13).trim();

                pType = str;
                if(pType.length() <=1){
                    pType = "N/A";
                }
```

```
                if(!typeList.contains(pType))
              {
                    typeList.add(pType);
                }
                pak = new Paket(pNumber, pFreq, pFlow, pLevel,pTime, pType);
                data.add(pak);
            }
        }
…
}
```

Metoda *SaveData(boolean[] arg, File fName, String pFlowFilter,String pTypeFilter)* vytváří instance třídy *SaveFlow* a předává jí parametry potřebné pro zpracování výstupu ve formátu XSL. Nakonec volá metodu *save.saveFile()* jenž provede vlastní zápis do souboru.

```
public void SaveData(boolean[] arg, File fName, String pFlowFilter,String
pTypeFilter) throws IOException, WriteException
 {
    ArrayList<Paket> tmpData=(ArrayList<Paket>) data.clone();
    Collections.copy(tmpData, data);
    SaveFlow save = new SaveFlow(arg, fName, pTypeFilter, pFlowFilter, tmpData);
    save.saveFile();
}
```

Metoda printPDF(File file,String pFlowFilter,String pTypeFilter,boolean errorCorection) vytváří instance třídy PDFprinter a předává jí parametry potřebné pro tisk výstupního souboru ve formátu PDF

```
public void printPDF(File file,String pFlowFilter,String pTypeFilter,boolean
errorCorection) throws FileNotFoundException, DocumentException
{
    String [] typeArray = typeList.toArray(new String [typeList.size()]);
    String [] flowArray = flowList.toArray(new String [typeList.size()]);
    PDFprinter printer = new PDFprinter(data, pFlowFilter, pTypeFilter,
errorCorection, typeArray,flowArray,fileType);
    printer.saveToPdf(file);
}
```

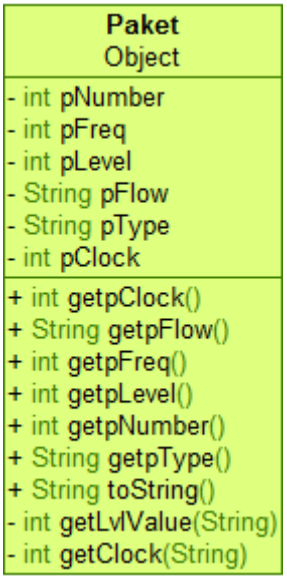
4.6.2 Třída Paket

Tato třída reprezentuje jednotlivé pakety. U každého paketu jsou uloženy jeho atributy - číslo paketu (*pNumber*), vysílací frekvence (*pFreq*), vysílací úroveň (*pLevel*), směr paketu (*pFlow*), typ paketu (*pType*) a časová stopa (*pClock*). Kromě konstruktoru obsahuje tato třída metody přistupující k jednotlivým atributům. Metoda getLvlValue() řeší případné nečíselné hodnoty v atributu úrovně signálu, které se mohou vyskytnout v případě chyby při přenosu paketu nebo špatném dekódování hlavičky paketu. Níže je ukázka třídy a konstruktoru:

```
public class Paket{
   private int pNumber;
   private int pFreq;
   private int pLevel;
   private String pFlow;
   private String pType;
   private int pClock;
```

```
   Paket(String   pNumber,String   pFreq,String   pFlow,   String   pLevel,String
pClock,String pType)

{
   this.pNumber = Integer.parseInt(pNumber);
   this.pLevel = getLvlValue(pLevel);
   this.pFreq = Integer.parseInt(pFreq);
   this.pFlow = pFlow;
   this.pType = pType;
   this.pClock=getClock(pClock);
 }
...
}
```



Obrázek 4.10 - Paket

4.6.3 Třída PDFprinter

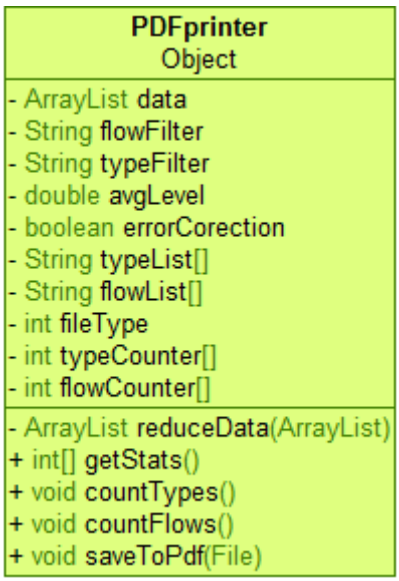
Hlavním úkolem této třídy je zápis statistických informací o získaných datech do výstupního souboru formátu PDF. Třída pracuje s celou kolekcí *data,* proto je nutné postarat se o vymazání nepotřebných paketů dle nastavení filtru podle zadání uživatele. Toto má za úkol metoda *ArrayList<Paket> reduceData(ArrayList<Paket> data)* jejímž vstupním parametrem je zdrojová kolekce paketů a výstupem nová kolekce redukovaná o nepotřebné pakety. Metoda prochází jednotlivé záznamy kolekce data a nejprve je testuje zde je atribut *pFlow* roven hodnotě nastaveného filtru, pokud tato podmínka vyhovuje je paket přidán do nové kolekce v opačném případě je paket zahozen. Poté následuje stejný proces pro kontrolu atributu *pType*. Kontrola typu paketu:

```
if(!(typeFilter.equals("All")))
       {
       reducedData = new ArrayList<Paket>();
       for(Paket p: flowReducedData)
            {
            if(p.getpType().equals(typeFilter)) reducedData.add(p);
            }
       }
```

Metoda *getStats()* provádí vlastní statistické operace nad daty. Jedná se o výpočet průměrné hodnoty úrovně signálu, zjištění počátečního a koncového času přenosu a celkovou dobu trvání měřeného spojení.

Dalšími statistickými metodami jsou tyto dvě – *countTypes()* a *countFlows()* – metody procházejí kolekci paketů a počítají četnost výskytu pro každý z vyskytujících se typů paketu, respektive směru paketu v případě *countFlows()*.

```
public void countFlows()
{
 flowCounter = new int [flowList.length];
 Iterator it = data.iterator();
 while(it.hasNext())
       {
       Paket pak = (Paket)it.next();
       for(int i=0;i<flowList.length;i++)
             {
              if(pak.getpFlow().equals(flowList[i]))
                    {
                    flowCounter[i]++;
                    }
             }
       }
}
```



Obrázek 4.11 - PDFprinter

Metodou obstarávající samotný zápis do souboru PDF je metoda *saveToPdf(File file),* jako parametr je jí předáván soubor do kterého bude proveden zápis. Tato metoda využívá externí knihovny iText (viz Kapitola 4.5.1). Do výstupu postupně zapisuje datum pořízení výstupu získaného metodou *Date(),* celkový počet paketů, informace o aktivních filtrech, počáteční a koncový čas přenosu, dobu trvání přenosu a průměrnou úroveň při přenosu. A nakonec podle typu analyzovaného souboru (viz kapitola 4.1), seznam jednotlivých typů paketů a směrů přenosu s počtem jejich výskytů.

4.6.4 Třída SaveFlow

Třída SaveFlow slouží k vytvoření objektu primárně zapisujícího zpracovaná data ve formě tabulky do souborů formátu XSL. Tento formát je primárně určen pro práci s daty v tabulkových procesorech např. Microsoft Excel nebo OpenOffice.org Calc. Pro zápis je využito externí knihovny jxl (4.5.2). Tato třída opět pracuje s celou kolekcí paketů, proto je nutno nejdříve aplikovat filtrování dle volby uživatele. O toto se starají následující tři metod: *aplyTypeFilter()*, *aplyFlowFilter()* a *aplyErrorFilter()*. První dvě metody zpracovávají data podobně jako metoda *reduceData()* zmíněná v předchozí kapitole, proto zde nebudou podrobně rozebírány. Metoda *aplyErrorFilter()* prochází redukovanou kolekci paketů a vypouští další pakety, u kterých se během zpracování nepodařilo zjistit úroveň signálu (pFlow = 0). Metoda *saveFile()*, obsluhuje vlastní zápis dat do souboru na disku. Nejdříve vytvoří instanci nového pracovního dokumentu:

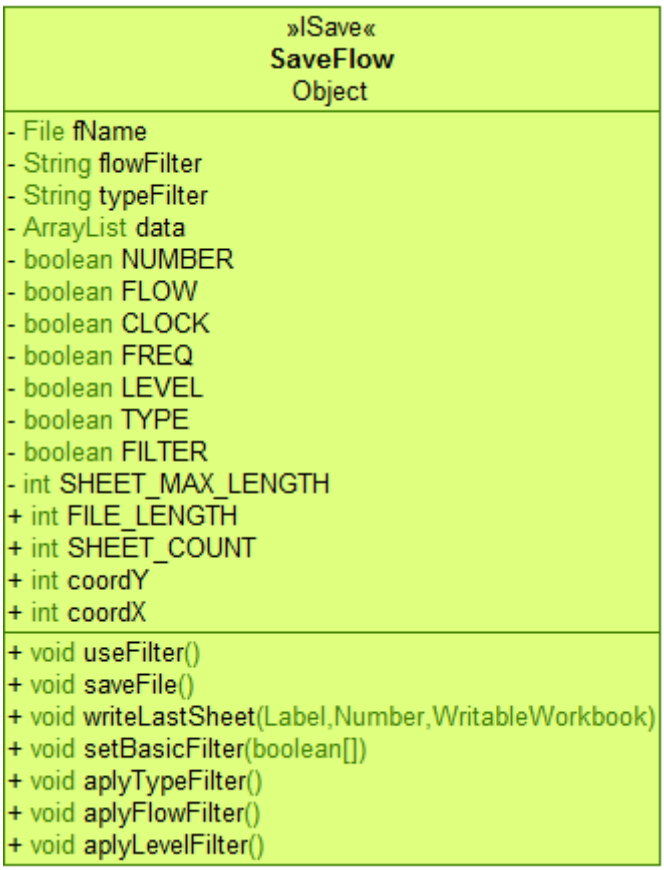
```
WritableWorkbook workbook = Workbook.createWorkbook(fName);
WritableSheet sheet = workbook.createSheet("Sheet 1", 0);
```

Dále dle zvolených filtrů vytvoří záhlaví pro jednotlivé sloupce :

```
if (NUMBER) {
   lbl1 =new Label(coordX++,coordY,"Number");
   sheet.addCell(lbl1);
   }
if (FREQ) {
   lbl1 =new Label(coordX++,coordY,"Freq");
   sheet.addCell(lbl1);
   }
if (FLOW) {
   lbl1 =new Label(coordX++,coordY,"Flow");
   sheet.addCell(lbl1);
   }
if (LEVEL) {
   lbl1 =new Label(coordX++,coordY,"Level");
   sheet.addCell(lbl1);
   }
if (TYPE) {
   lbl1 =new Label(coordX++,coordY,"Type");
   sheet.addCell(lbl1);
   }
if (CLOCK) {
   lbl1 =new Label(coordX++,coordY,"Clock");
   sheet.addCell(lbl1);
   }
```

A následně prochází filtrovanou kolekcí a zapisuje do výstupu zvolené hodnoty pro jednotlivé pakety. Každý řádek v tabulce odpovídá jednomu paketu. Následně je celý výstup zapsán a dokument je uzavřen.

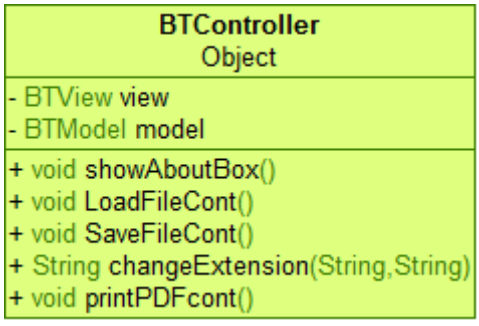
```
workbook.write();
workbook.close();
```



Obrázek 4.12 - SaveFlow

4.6.5 Třída BTController

Třída BTController je zodpovědná za zachytávání událostí, které provádí uživatel v rámci uživatelského rozhraní a ošetřuje vznik možných chyb při manipulaci s daty. Nejčastěji se může jednat o neoprávněný zápis do souboru (např. pokud je soubor právě využíván) nebo dojde-li k chybě při zpracovávání vstupních dat.



Obrázek 4.13 - BTController

Tato třída v MVC modelu složí jako jediné spojení mezi modelem a uživatelským rozhraním, což je patrné z konstruktoru této třídy:

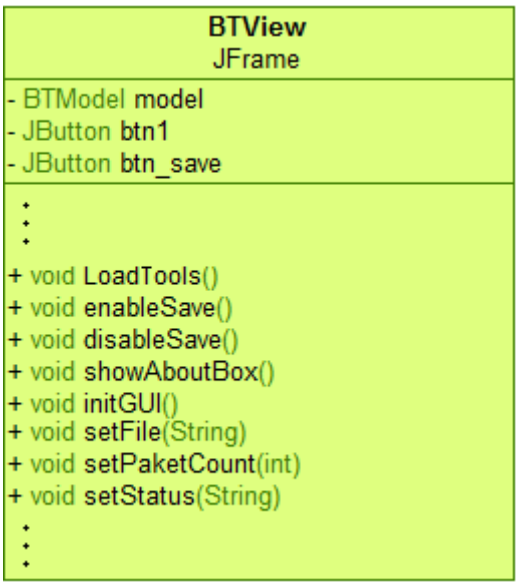
```
BTController(BTView view, BTModel model){
```

```
    this.view = view;
    this.model = model;
    view.addBtn1Listener(new BtnLoadListener());
    view.addBtn_saveListener(new Btn_saveListener());
    view.addBtn_printListener(new Btn_printListener());
    view.addmiLoadListener(new miLoadListener());
    view.addmiSaveListener(new miSaveListener());
    view.addmiPrintListener(new miPrintListener());
    view.addmiEndListener(new miEndListener());
    view.addmiAboutListener(new miAboutListener());
}
```

4.6.6 Třída BTView

Třída BTView slouží v MVC modelu k vytvoření uživatelského rozhraní umožňující ovládat a řídit zpracování dat. Důležitou metodou této třídy je metoda *initGUI()*. Jenž vytváří instance všech prvků potřebných pro ovládání a nastavení aplikace a řídí jejich rozmístění pomocí layoutu. Mezi ovládacími a zobrazovacími prvky jsou objekty tříd Panel, Button, Label, ComboBox, CheckBox, Menu a MenuBar. Metoda *showAboutBox()* je volána při stisknutí položky About v menu Help a vede k zobrazení dialogu informujícího o vlastní aplikaci.

```
public void showAboutBox()
{
      if(aboutBox == null)
          {
           aboutBox = new AboutDialog(this,true);
           aboutBox.show();
           }else
          {
           aboutBox.show();
           }
 }
```



Obrázek 4.14 - BTView

4.7 Ukázka výstupu

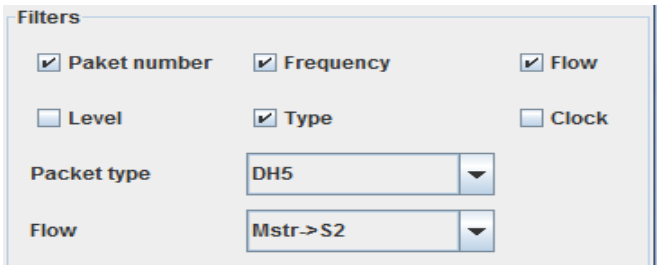
V této podkapitole budou prezentovány ukázky výstupních souborů zpracovaných aplikací BTanalyzer.

Tabulka je základním výstupním formátem aplikace. Tato forma byla zvolena z toho důvodu, aby umožnila další pohodlné zpracování dat. Pomocí funkcí tabulkových procesorů lze jednoduše získat další statistická data – četnost výskytu jednotlivých frekvencí, poměr výskytů jednotlivých typů paketů apod. Data zpracovaná do tabulek jsou také vhodná pro tvorbu grafů. Jedinou nevýhodou tohoto formátu je maximální povolený rozsah XLS dokumentu jenž je omezen na 65536 řádků. Proto pokud je množství výstupních paketů větší pak je do výstupu zapsáno pouze prvních 65536 paketů. Na obrázku níže vidíte část zpracovaného souboru, u kterého bylo nastavení filtrů následující (Obr. 4.10)

4.7.1 Tabulka XLS

	A	B	C	D	E
1	Number	Freq	Flow	Type	
2	2060	2421	Mstr->S2	DH5	
3	2061	2441	Mstr->S2	DH5	
4	2063	2425	Mstr->S2	DH5	
5	2064	2439	Mstr->S2	DH5	
6	2065	2406	Mstr->S2	DH5	
7	2066	2479	Mstr->S2	DH5	
8	2067	2445	Mstr->S2	DH5	
9	2068	2463	Mstr->S2	DH5	
10	2070	2404	Mstr->S2	DH5	
11	2072	2443	Mstr->S2	DH5	
12	2078	2479	Mstr->S2	DH5	
13	2081	2412	Mstr->S2	DH5	
14	2083	2426	Mstr->S2	DH5	
15	2123	2439	Mstr->S2	DH5	
16	2124	2457	Mstr->S2	DH5	
17	2126	2429	Mstr->S2	DH5	
18	2127	2405	Mstr->S2	DH5	
19	2132	2449	Mstr->S2	DH5	
20	2137	2455	Mstr->S2	DH5	
21	2138	2412	Mstr->S2	DH5	
22	2141	2408	Mstr->S2	DH5	

Obrázek 4.15 – XLS náhled



Obrázek 4.16 – Nastavení filtrů

4.7.2 PDF výstup

Tento výstup slouží pouze jako zdroj informací o analyzovaném souboru. Obsahuje informace o datu zpracování, počtu paketů, době průběhu přenosu a nastavení voleb pro filtrování výstupu. Ukázka výstupního PDF dokumentu je na obrázku (Obr.4.11). Struktura dokumentu se může lišit z důvodu různého typu vstupních dat. Pokud se nepodařilo získat informace o typu paketů pak výstupní PDF dokument neobsahuje část „Paket type stats“. Pokud se pouze u některých paketů nepodařilo získat informaci o typu paketu pak je tento paket označen N/A (not available).

Bluetooth analyzer
Printed : Tue Apr 19 01:57:08 CEST 2011
Packet count :7797
Selected flow : All
Selected type : All
Begin : 2 ms
End : 96566 ms
Time :96564 ms
Avarage level :NaN dBm
Packet type stats :
ID : 2887
FHS : 1
N/A : 1686
POLL : 3080
DH5 : 125
DH3 : 2
DH1 : 14
DM5 : 1
DM1 : 1
Packet flow stats :
Mstr->S0 : 1975
S0->Mstr : 1
S7->Mstr : 1
S3->Mstr : 6
Mstr->S4 : 1
S6->Mstr : 2
Mstr->S2 : 5808
Mstr->S5 : 1
Mstr->S3 : 2

Obrázek 4.17 – Ukázka PDF dokumentu

4.8 Porovnání vstup/výstup

Jelikož je i během krátkého Bluetooth spojení přeneseno obrovské množství paketů, vede toto k nárůstu objemu výstupních dat aplikace Arca|Wavecatcher. Toto klade vysoké výpočetní nároky na zpracování dat. Velikost testovacích souborů přesahovala 50 MB při délce přenosu 99 vteřin. V následující tabulce je zobrazen poměr objemu vstupních a výstupních dat a doba zpracování souboru.

Vstupní soubor			Výstupní soubor	
Velikost(MB)	Načtení (ms)	Počet paketů	Velikost(MB)	Zápis (ms)
11	860	7797	0,9	1450
60,5	6060	63144	7,4	5770
89	7830	76410	9,8	7130

Tabulka 4.1 – Zpracování dat

5 Závěr

Výsledná aplikace umožňuje načíst vstupní data, která musí být připravena v přesně daném formátu, jenž poskytuje jako svůj výstup aplikace arca|Wavecatcher. Aplikace je schopna rozpoznat jednotlivé pakety v souboru a získat o nich všechny dostupné informace potřebné pro další zpracování. Aplikace umí řešit výskyty různých chyb, buďto obsažených ve vstupních souborech nebo vzniklých při zpracování. Struktura modelu MVC umožňuje nezávislou změnu vnitřní logiky zpracování dat bez nutnosti zasahovat do vzhledu uživatelského rozhraní aplikace. Aplikace umožňuje uložit dva typy výstupních dat. Tabulková data, která jsou vhodná pro další využití uživatelem. Druhým výstupem je PDF dokument, jenž lze považovat za meta-data k tabulkám obsahující informace o jejich obsahu. Aplikace byla testována na široké množině vstupních dat, při kterých došlo vždy k bezproblémovému načtení vstupu. V průběhu testování se vyskytl problém s omezením počtu zapisovaných řádku ve výstupním tabulkovém souboru, kde je maximální počet řádku jednoho sešitu 65536. Tento problém se podařilo vyřešit tvorbou výstupních tabulek zapsaných do více stran souboru XLS. Jelikož se jedná o nástroj, jenž zpracovává velmi specifická data, je jeho jiné využití takřka vyloučeno. Mezi možnosti dalšího rozšíření by mohl patřit návrh a zpracování dodatečných nástrojů pro tvorbu grafů nebo hlubších analýz ze zpracovaných dat.

6 Literatura

[1] MORROW, Robert. *Bluetooth Operation and Use*. New York(USA) : McGraw-Hill, c2002. xviii, 567 s. ISBN 0-07-138779-X.

[2] GRATTON, Dean A. *Bluetooth profiles : the definitive guide*. Upper Saddle River(USA) : Prentice Hall, c2003. xx, 569 s. ISBN 0-13-009221-5.

[3] Arca Technologies. *Bluetooth Protocol Analyzer*. California (USA) : Arca Technologies, 2002. 62 s.

[4] *Palowireless : Wireless Resource Center* [online]. c2011 [cit. 2011-04-26]. Dostupné z WWW: <www.palowireless.com>.

[5] *Specification of the Bluetooth System : Core v1.2* [online]. USA : Bluetooth SIG, 05 November 2003 [cit. 2011-04-26]. Dostupné z WWW: <www.bluetooth.com>.

[6] Bluetooth SIG. *Bluetooth* [online]. c2011 [cit. 2011-04-26]. Dostupné z WWW: <www.bluetooth.com>.

[7] Bluetooth SIG. *Bluetooth SIG* [online]. c2011 [cit. 2011-04-26]. Dostupné z WWW: <www.bluetooth.org>.

[8] TICHA, Ondřej. *Bluetooth* [online]. [s.l.], 2004. 6 s. Semestrální práce. Západočeská univerzita v Plzni, Katedra informatiky a výpočetní techniky. Dostupné z WWW: <http://www.kiv.zcu.cz/~simekm/vyuka/pd/zapocty-2004/bluetooth-ticha/>.

7 Seznam obrázků a tabulek

Obrázky:

Tabulky

8 Přílohy

Seznam příloh na přiloženém CD:

- Zdrojový kód aplikace ve složce /src
- Spustitelná aplikace ve složce /dist včetně externích knihoven /lib
- Dokumentace JAVADOC ve složce /javadoc
- Elektronická podoba závěrečné práce