VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SNIFFER PRO KOMUNIKACI PO ENERGETICKÉM VEDENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

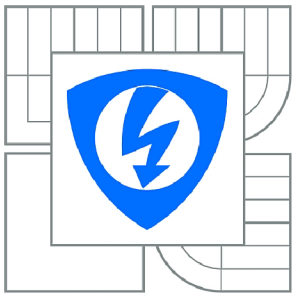
AUTOR PRÁCE
AUTHOR

RADEK ZAPLETAL

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SNIFFER PRO KOMUNIKACI PO ENERGETICKÉM VEDENÍ

SNIFFER FOR POWER LINES COMMUNICATION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

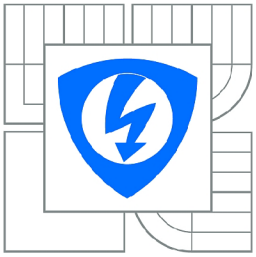
AUTOR PRÁCE — RADEK ZAPLETAL

AUTHOR

VEDOUCÍ PRÁCE — Ing. LEŠEK FRANEK

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Radek Zapletal	***ID:***	147005
Ročník:	3	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Sniffer pro komunikaci po energetickém vedení

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit sniffer pro protokol MT39 z oblasti inteligentních sítí. Zadání lze shrnout do následujících bodů:

1. Literární rešerše problematiky snifferů
2. Literární rešerše problematiky Smart Grid a Smart Meteringu
3. Návrh univerzálního snifferu pro inteligentní sítě
4. Realizace snifferu
5. Ověření snifferu
6. Popis jak rozšířit sniffer o další protokoly

DOPORUČENÁ LITERATURA:

OREBAUGH, A., G. RAMIREZ a J. BEALE. Wireshark & Ethereal Network Protocol Analyzer Toolkit. Elsevier Science, 2006. ISBN 9780080506012. Dostupné z: http://books.google.cz/books?id=-AdTE9S3kigC

Termín zadání:	10.2.2014	***Termín odevzdání:***	26.5.2014

Vedoucí práce: Ing. Lešek Franek
Konzultanti bakalářské práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá návrhem a realizací univerzální aplikace pro zachytávání, analýzu a filtrování komunikace na energetické síti (PLC). Data poskytuje hardwarový sniffer na sériové lince RS-232. Aplikace bude využívána v oblasti inteligentních sítí a Smart Meteringu. Součástí práce je popis těchto oblastí včetně problematiky snifferů i návrh několika variant řešení s výběrem optimálního a popisem realizace.

KLÍČOVÁ SLOVA

Sniffer, Inteligentní síť, Smart Metering, Přenos dat po elektrické síti, Wireshark, Knihovna Qt, C++, RS-232, Lua

ABSTRACT

This work deals with the design and development of universal software packet analyzer for capturing, analysis and filtering Power-line communication (PLC). The input data are provided by hardware packet analyzer over RS-232 interface. The application will find use in the area of Smart Grids and Smart Metering. This work describes both of these areas including sniffers and suggests several possible solutions with a selection of the best one and description of the development process.

KEYWORDS

Packet analyzer, Smart Grid, Smart Metering, Power-line communication, Wireshark, Qt framework, C++, RS-232, Lua

ZAPLETAL, Radek *Sniffer pro komunikaci po energetickém vedení*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2014. 49 s. Vedoucí práce byl Ing. Lešek Franek

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Sniffer pro komunikaci po energetickém vedení“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce panu Ing. Lešku Franekovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Cílem této bakalářské práce je navrhnout, realizovat a prakticky ověřit softwarovou aplikaci - Univerzální sniffer pro energetické sítě na Windows platformě. Ten má být schopen zachytávat pakety proudící po energetických sítích (tzv. PLC komunikaci) a následně umožnit jejich celkovou analýzu zahrnující kompletní dekódování použitého komunikačního protokolu, přehledné zobrazení zachycených paketů, vyhledávání a filtrování zachycené komunikace a případné uložení dat pro pozdější použití. Výsledný program by měl být univerzální, tedy nebýt navržen pouze pro jeden daný komunikační protokol, ale umožňovat co nejsnazší doplnění podpory případných dalších PLC protokolů.

Práce je realizována ve spolupráci s firmou ModemTec, s. r. o. a v rámci jejího řešení má být současně implementován i firmou vyvinutý PLC protokol MT39. Hardwarová část snifferu je již k dispozici v podobě zařízení, poskytujícího zachycená surová data na sériové lince.

K úspěšnému vyřešení zadaných cílů bude třeba seznámit se s principy PLC komunikace a inteligentních sítí, dále nastudovat problematiku snifferů a jejich používání a poté zvolit vhodný postup a technologie k samotné realizaci aplikace.

Aplikace bude následně moci být v praxi používána při dalším vývoji a zlepšování komunikačních protokolů pro energetické sítě či pro diagnostické účely již existujících sítí, kde značně urychlí analýzu zachycených dat, která by jinak bylo nutné ručně dle jednotlivých bajtů dohledávat v dokumentaci k protokolům. Hlavní uplatnění nalezne v rámci v současné době se rozvíjejících technologií Smart Grids a Smart Meteringu.

1 SNIFFERY

Pojmem sniffer se označuje počítačový program (tedy softwarový sniffer) či hardwarová součást, která slouží k zachytávání, možnému ukládání a případné další analýze provozu na různých typech digitálních sítí či komunikačních tras. Někdy bývá též označován pojmem síťový či paketový analyzátor. Hardwarový i softwarový sniffer bývají většinou dvě vzájemně se doplňující entity, kde hardwarová část provádí samotný sběr dat ze sítě a softwarová umožňuje jejich analýzu.

Cílem procesu síťové analýzy je dekódovat data předávaná po síti v daných protokolech a umožnit uživateli jejich zobrazení v čitelné podobě pro pohodlnou orientaci v datovém provozu. Jelikož sniffery zachycují veškerá data proudící danou sítí, může jejich použití případnými útočníky znamenat i velké bezpečnostní riziko, jelikož jejich přítomnost je velmi obtížné detekovat a mohou být umístěny prakticky kdekoliv [1].

Sniffery bývají využívány k různorodým činnostem, mezi které patří: [2]

- Analýza problémů na sítích
- Detekce narušení sítě útočníkem
- Detekce zneužívání sítě k jiným než povoleným užitím jejími uživateli
- Monitorování využívané šířky pásma sítě
- Vytváření statistik o provozu sítě
- Ladění během vývoje síťových protokolů
- Zjišťování hesel či jiných citlivých údajů přenášených po síti
- Filtrace nestandardního obsahu od běžné síťové komunikace
- Zdroj dat pro běžný dohled nad sítí
- Reverzní inženýrství proprietálních protokolů

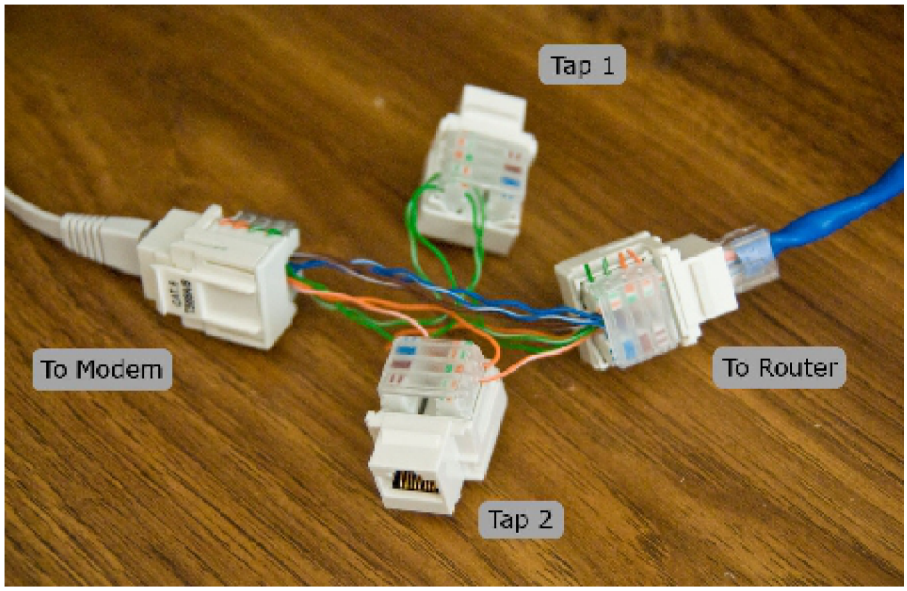
1.1 Hardwarové sniffery

Hardwarovým snifferem je většinou pasivní zařízení, které je připojeno k odbočení ze sítě. Odbočení může v praxi vypadat i jen jako paralelní napojení na komunikační vodiče. Nejedná se o nijak konkrétně specifikovaný výrobek a jeho různá provedení se od sebe mohou i velmi lišit na základě fyzického média pro které je určen. Takový sniffer však pouze přijímá data na daném médiu a sám o sobě do komunikace na síti nevstupuje. Zachycená data může následně ukládat do vnitřní paměti či je přímo poskytovat přes své komunikační rozhraní k dalšímu zpracování softwarovému snifferu.

V následující části budou pro názornost krátce představeny některé existující používané hardwarové sniffery pro různá komunikační média.

1.1.1 HW sniffer pro síť Ethernet

V případě nejrozšířenější sítě Ethernet jako HW sniffer postačí i zcela běžná síťová karta osobního počítače. Zde se sice jedná o aktivní zařízení, nicméně fyzicky mívá připojeny pouze vodiče pro příchozí data. V případě jednoduché síťové odbočky znázorněné na obrázku 1.1 by se tato síťová karta pouze propojila do konektoru Tap 1 nebo Tap 2 v závislosti na požadovaném směru zachycovaných dat. Takto připojená síťová karta musí navíc podporovat přepnutí do tzv. promiskuitního režimu, ve kterém karta akceptuje veškerý provoz na síti - tedy včetně paketů určených jiným adresátům.



Obr. 1.1: Jednoduché odbočení ke snifferu pro síť Ethernet [3]

V případě, kdy existuje požadavek na zachytávání pouze vlastní komunikace daného počítače, není nutné ani vytvářet síťovou odbočku, ta je nahrazena možností ovladačem síťové karty softwarově kopírovat veškerou komunikaci i ke snifferu spuštěnému na tomto počítači. V případě sběrnicového zapojení sítě bylo takto možné zachytávat komunikaci na celé síti, v současnosti, kdy je však většina sítí již řešena topologií typu hvězda s aktivními směrovacími prvky však často síťová karta jiné než pro ni určené pakety neobdrží.

1.1.2 HW sniffer pro počítačovou klávesnici PS/2

Tento typ hw snifferu je typickou ukázkou použití snifferů útočníkem pro získání citlivých údajů. Sniffer (ukázaný na obrázku 1.2) má podobu běžného PS/2 konektoru, o který se pouze prodlouží konektor připojené PC klávesnice a běžný uživatel

jen těžko takto připojené zařízení odhalí.

Uvedený sniffer si poté ukládá veškeré úhozy připojené klávesnice do své vnitřní paměti a po odstranění snifferu si může útočník tato data jednoduše přečíst a dostává tak do rukou kompletní napsaný text, tedy i uživatelská jména, hesla a jiné údaje. Vzhledem ke snadné dostupnosti a ceně takového zařízení pohybující se okolo 600 Kč [4] se jedná o velmi reálnou hrozbu pro citlivé osobní i firemní údaje.

Obr. 1.2: HW sniffer pro počítačovou klávesnici PS/2 [4]

1.1.3 HW sniffer pro energetickou síť

Tento typ snifferu slouží k zachytávání komunikace na energetické síti přenášené technologií Power Line Communication. Jeho připojení se provádí jednoduchým připojením do rozvodů 230 Voltů. Na obrázku 1.3 je zobrazen tento typ snifferu vyvinutý firmou ModemTec, který zachycená data nijak neukládá ani neanalyzuje, pouze je okamžitě předává na sériové rozhraní k dalšímu zpracování. Právě tento sniffer bude využíván jako zdroj dat pro výslednou aplikaci vyvinutou v rámci této práce.

Obr. 1.3: HW sniffer firmy ModemTec pro energetické sítě [5]

1.2 Softwarové sniffery

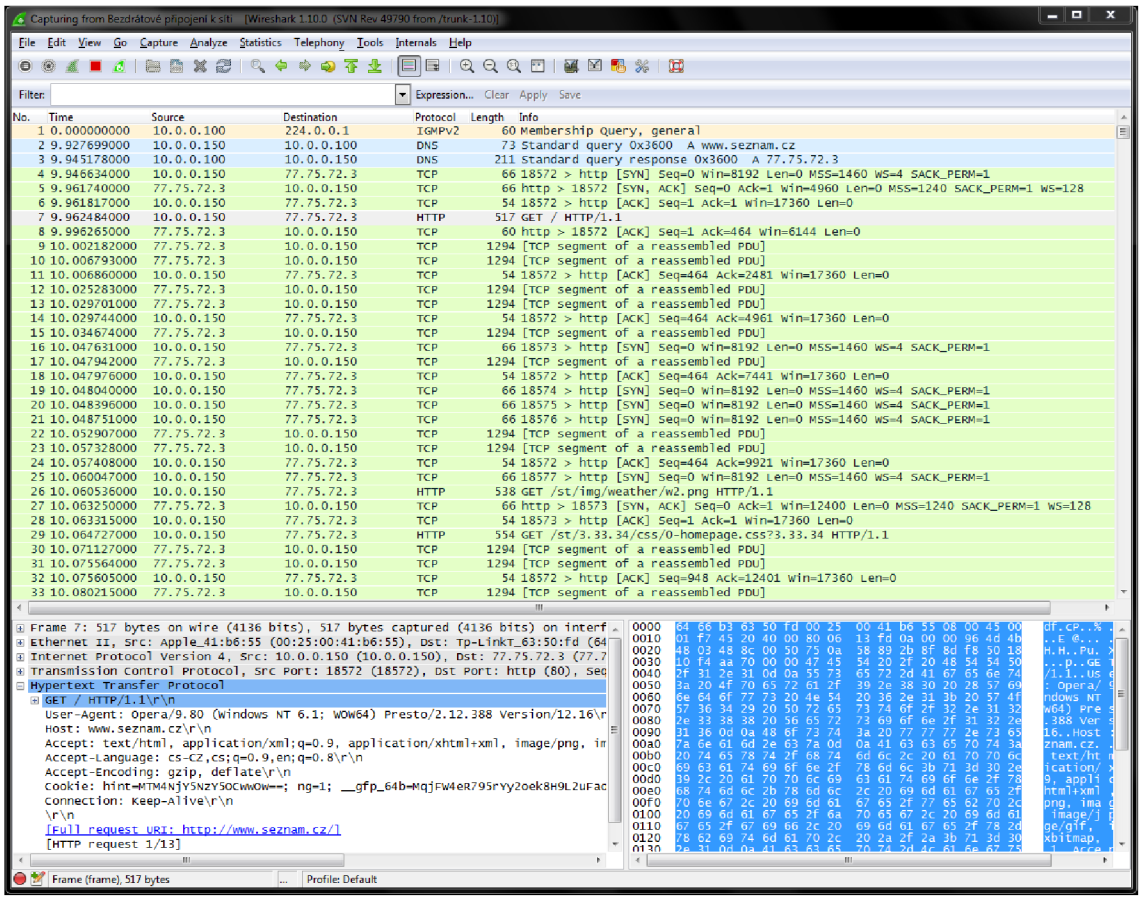
Softwarový sniffer má nejčastěji podobu počítačového programu instalovaného na příslušném počítači. Nejdůležitějším parametrem softwarových snifferů je seznam protokolů, který je daný sniffer schopen dekódovat. Další důležitou vlastností je rozsah funkcí, která nám daný program poskytuje. Díky rozličným požadavkům na používání snifferu bývá jejich celková funkčnost poměrně rozsáhlá a zahrnující především následující:

- Podporovaná zařízení, ze kterých je daný sniffer schopný přijímat data
- Podpora filtrování zachycených dat i s rozlišením vlastností jednotlivých protokolů
- Možnost ukládat a opětovně načítat zachycenou komunikaci
- Identifikace chyb v přenášených datech (dle specifikace protokolů)
- Statistiky přenesených dat (dle typů paketů, odesílatele, příjemce)
- Generování grafů ze statistik
- Podpora tisku
- Rekonstrukce jednotlivých spojení mezi vybranými uzly
- Kvalita dekódování protokolů – míra detailnosti parsování dat
- Přehledné a přizpůsobitelné uživatelské rozhraní, barevné rozlišování paketů
- Možnost doplňovat dekodéry pro zatím neobsažené protokoly

Existující SW sniffery mají dnes již ve většině případů velmi podobné časem prověřené uživatelské rozhraní, skládající se ze seznamu po sobě jdoucích zachycených paketů (online aktualizovaného) se základními informacemi, detailního zobrazení vybraného paketu v hexadecimální podobě a dekódovaných informací o datech v daném paketu, viz obrázek 1.4.

1.2.1 Univerzálnost snifferů

Na rozdíl od hardwarových snifferů, kde to díky odlišnostem komunikačních médií možné není, je většina dnes existujících sw snifferů koncipována jako univerzální, což vyjadřuje schopnost snifferu analyzovat různé protokoly a případně i možnost sniffer pohodlnou cestou rozšířit o podporu dalších protokolů. Tato koncepce je zcela pochopitelná, jelikož i velmi odlišné komunikační protokoly mají v přenášených datech vždy tyto společné vlastnosti – čas příchodu daného paketu, délka (počet bajtů) dat obsažených v paketu, samotná data paketu. Samotná komunikace tedy musí být pouze paketově orientovaná. I toto kritérium se dá nicméně v nouzi obejít považováním každého bajtu za jednotlivý paket. Rozhraní snifferů jsou tedy navržená



Obr. 1.4: Ukázka uživatelského rozhraní softwarového snifferu – Wireshark

pouze pro tyto základní parametry komunikace a veškerou další logiku obstarávají dekodéry jednotlivých protokolů.

1.2.2 Princip dekódování komunikace ze surových dat

V rámci dat, která jsou do softwarového snifferu doručena, většinou není předem známo v jakém protokolu jsou data jednotlivými pakety přenášena. Dekodéry jednotlivých protokolů tedy obsahují i schopnost detekovat, zda se jedná o právě tento konkrétní protokol a to z informací poskytnutou nadřazenou vrstvou dat či z charakteristických znaků protokolu. Díky faktu, že síťová komunikace je nejčastěji prováděna pomocí dat zabalených do jednotlivých vrstev vybraných z modelu ISO/OSI, probíhá dekódování formou detekce protokolu nejsvrchnější vrstvy, její zpracování a předání vrstvou přenášených dat k detekci a zpracování další vrstvy obdobným způsobem.

Dekodéry jednotlivých protokolů fungují na principu parsování vstupního bufferu jednotlivých bajtů dat a jejich transformace do čitelné struktury odpovídající

specifikaci daného protokolu.

1.2.3 Přehled snifferů

Jednotlivé sniffery je možné rozdělit do dvou skupin podle provedení uživatelského rozhraní pouze pro příkazovou řádku nebo s plnohodnotným grafickým uživatelským rozhraním (GUI). Programy pro příkazový řádek jsou vhodné spíše pro automatizované zpracování zachycených paketů, zatímco zástupci z druhé skupiny jsou pro ruční analýzu síťového provozu značně přehlednější a pohodlnější. I tyto GUI programy však bývají pouhou nástavbou nad zachytávacími programy pro příkazovou řádku.

V současné době již existuje celá řada různě kvalitních softwarových snifferů. Náplní zbytku této kapitoly bude přehled a srovnání funkcí několika vybraných programů – Wireshark, Microsoft Network Monitor a WinDump.

Wireshark

Program Wireshark je bezplatný softwarový síťový analyzátor s otevřeným zdrojovým kódem. Jedná se o dnes vůbec nejpoužívanější softwarový sniffer [6]. Program je vyvíjen již od roku 1998, kdy byl znám pod pojmenováním Ethereal. Díky patentovým sporům o toto jméno však roku 2006 obdržel nynější název. Wireshark je vyvíjen jako multiplatformní aplikace a funguje pod operačními systémy Windows, Linux, OS X, BSD a Solaris. Jedná se o zástupce snifferů s grafickým uživatelským rozhraním, k dispozici je nicméně i verze pro příkazovou řádku zvaná T-Shark [7].

V současné době podporuje dekódování více než 141 000 informací ze zhruba 1000 protokolů [8]. Mezi podporované patří např. ATM, ARP, BitTorrent, CAN, DHCP, DNS, Ethernet, FTP, GIT, GPRS, GSM, HTTP, I2C, ICQ, IGMP, IMAP, IP, Jabber, NetBIOS, POP, SMTP, SSL, TCP, Telnet či UDP.

Wireshark pro svoji zachytávací část vyžaduje knihovnu pcap (pro Windows implementována jako WinPcap, pro Linux libpcap) a je tedy schopen přijímat data z rozhraní, která jsou v této knihovně podporována.

Ukázka uživatelského rozhraní se zachycenou komunikací HTTP požadavku na úvodní stránku serveru seznam.cz je zobrazena na obrázku 1.4. Je zde vidět seznam zachycených paketů s barevným rozlišením dle identifikovaného protokolu, krátké shrnutí významu každého paketu ve sloupci Info a informační pole dekódovaná z jednotlivých vrstev paketu.

Jelikož množství zachycené komunikace je často velmi rozsáhlé, má Wireshark zabudovanou podporu pro dva typy filtrování. Prvním je tzv. Capture Filter, který

filtruje pakety hned v okamžiku jejich příchodu a všechny, které nevyhovují zadaným pravidlům, jsou zcela zahozeny. Druhým typem filtru je Display filter, který je aplikován až následně a filtruje pakety, které budou ve výsledku zobrazeny v hlavním seznamu paketů. Podobnou koncepci filtrů využívají i další softwarové analyzátory. Display filter se ve Wiresharku zadává formou textového řetězce obsahující definici podmínek filtrace, tedy např. pro zobrazení výhradně komunikace, kde je cílová adresa v IP protokolu 1.2.3.4, jej stačí zadat jako: „ip.dst==1.2.3.4“. Dostupná informační pole pro filtry jsou interaktivně napovídána, takže jejich zadávání je velmi pohodlné.

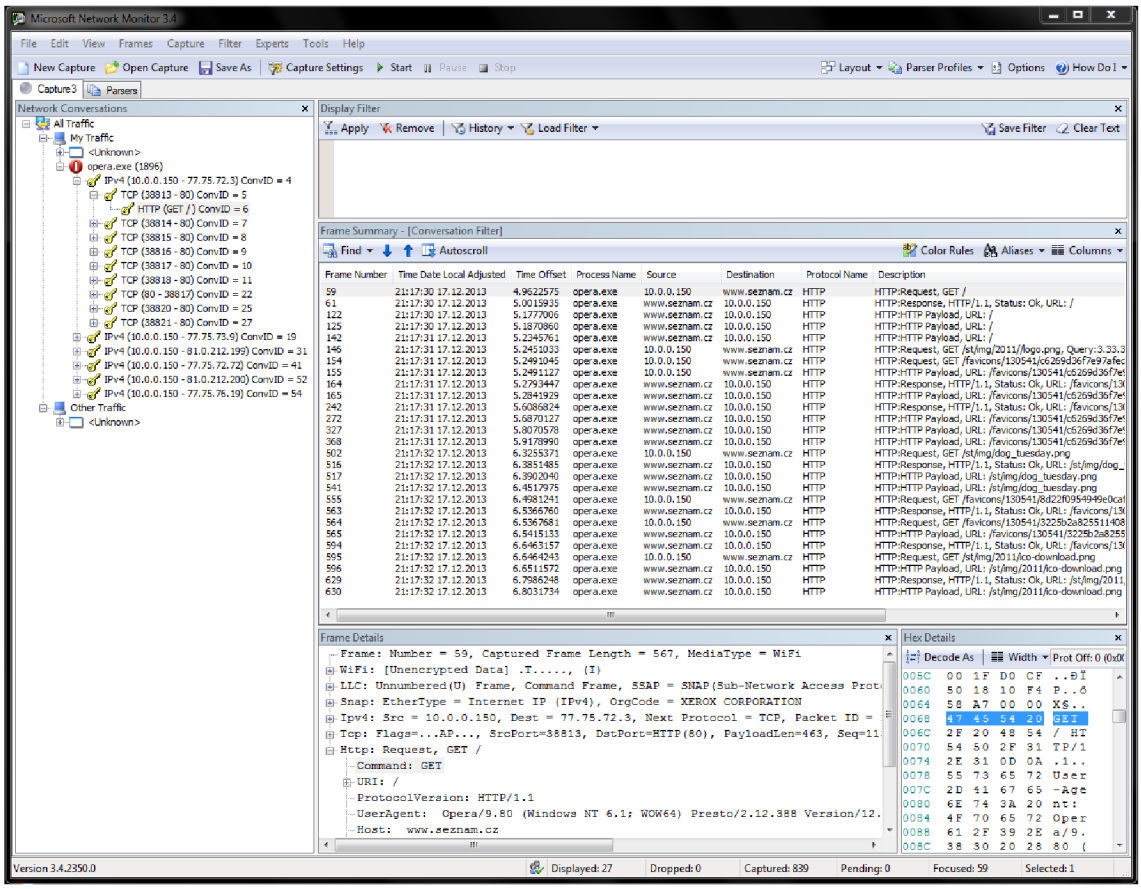
Wireshark disponuje i množstvím pokročilých funkcí, mezi které patří např.:

- Spojování jednotlivých paketů do celkové komunikace proběhlé mezi dvěma koncovými body
- Detekce VoIP telefonátů včetně integrované funkce pro přehrání obsahu
- Zachytávání komunikace na USB sběrnici
- Zobrazení seznamu koncových bodů na síti
- Na Linuxu schopnost přepnutí bezdrátových rozhraní do monitorovacího režimu a následně zachycení veškeré probíhající bezdrátové komunikace
- Automatická dekomprese dat zabalených gzip komprimací
- Dešifrování některých protokolů např IPsec, SSL/TLS, WEP či WPA/WPA2
- Statistiky přenosu dat dle jednotlivých protokolů i s grafickým výstupem
- Možnost vytváření pluginů pro dekódování dosud nepodporovaných protokolů

Microsoft Network Monitor

Program Microsoft Network Monitor je softwarový sniffer vyvinutý firmou Microsoft pro operační systém Windows a je k dispozici zdarma ke stažení na internetových stránkách firmy. Aplikace již od roku 2010 není dále vyvíjena a na podzim 2013 byl uvolněn její plánovaný nástupce Microsoft Message Analyzer, který však má podstatně vyšší systémové nároky. I starší Network Monitor je však co do rozsahu funkčnosti velmi propracovaný sniffer nabízející i některé v jiných programech nepodporované funkce. [9] [10]

Mezi základní funkce patří samozřejmě zachytávání paketů z dostupných síťových rozhraní, aplikace Capture a Display filtrů a podpora více než 300 zpracovatelných protokolů včetně některých proprietálních firmy Microsoft. K dispozici je i možnost doprogramovat si vlastní dekodéry protokolů a API pro přístup k celému zachytávacímu a parsovacímu rozhraní. Program je schopný otevírat i ukládat souborový formát .cap definovaný knihovnou pcap, takže zachycená data je možné následně



Obr. 1.5: Microsoft Network Monitor s filtrováním dle procesu a konverzace

zpracovat i jiným síťovým analyzátorem, který jej rovněž podporuje (např Wiresharkem).

Aplikace má např. v porovnání s Wiresharkem mimo jiné následující užitečné funkce navíc:

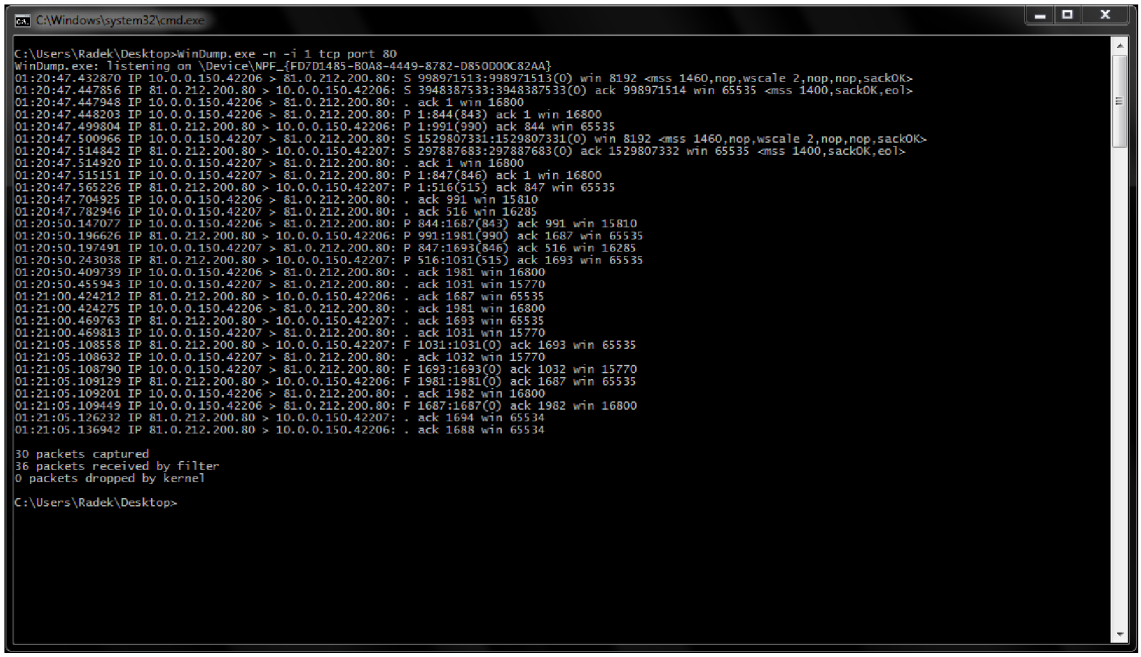
- Možnost simultánně zachycovat a zpracovávat komunikaci v oddělených záložkách, kde je pro každou možné vybrat příslušná vstupní síťová rozhraní
- Automatické seskupování paketů do jednotlivých konverzací koncových bodů a protokolů a jejich přehledné živé zobrazení v levé stromové struktuře
- V případě lokální komunikace zobrazení i názvu procesu, ke kterému daná komunikace náleží, včetně příslušného filtrování
- Schopnost přepnutí Wifi síťové karty ve Windows do monitorovacího režimu a zachytávání originální vrchní 802.11 vrstvy paketů – ve snifferech používajících knihovnu WinPCap je tento druh komunikace zachytáván jako Ethernetová komunikace po drátech. [11]

WinDump

Aplikace WinDump vznikla portováním softwarového snifferu tcpdump běžně dostupného v operačních systémech Linux pod systém Windows. Jedná se o síťový analyzátor určený pouze pro příkazovou řádku funkčně závislý na knihovně pcap. Program má otevřený zdrojový kód s aktivní vývojářskou komunitou a je šířený pod bezplatnou licencí BSD [12].

Program má sice oproti svým grafickým protějškům funkce omezenější, nicméně pro řadu základních analýz síťového provozu plně postačuje a s výhodou jej můžeme použít i vzdáleně např. v rámci ssh sezení. K podporovaným funkcím patří Capture filtry, automatický DNS překlad IP adres, volitelné zobrazení celého obsahu paketů či ukládání zachycené komunikace do .cap souborového formátu. Protokoly jsou podporovány pouze základní, z nejpoužívanějších se jedná o: Ethernet, FDDI, IP, IPv6, ARP, TCP a UDP. [13]

Ukázka výstupu programu je na obrázku 1.6. Zde použitý příklad zobrazuje spuštění s aktivním Capture filtrem, který přijímá pouze pakety TCP protokolu s číslem portu 80 (část tcp port 80), se zákazem překladu IP adres (-n) a jako vstupní síťové zařízení je vybráno to s pořadovým číslem 1 (část -i 1). Výstup je ve formě lineárního seznamu zachycených paketů.



Obr. 1.6: Výstup zachycené komunikace v programu WinDump

2 SMART GRIDS A SMART METERING

2.1 Smart Grids

Smart Grids (česky inteligentní sítě) jsou moderní elektrické sítě, které dokáží sledovat výrobu i odběr jednotlivých energií (elektrické energie, plynu, vody, tepla) a na jejich základě výrobu i spotřebu regulovat v reálném čase, v místním i globálním měřítku a zajistit efektivní přenos k odběratelům. Jako přenosová média se využívají silové elektrické či komunikační sítě. [14]

Sítě pracují na principu oboustranné komunikace mezi výrobními zdroji a jednotlivými spotřebiteli či spotřebiči. Na základě informací o jejich chování je možné automatizovaně zajistit vyšší efektivitu, spolehlivost, ekonomičnost a trvalou udržitelnost výroby a distribuce energií.

V první fázi je nutné mít k dispozici data o aktuálním stavu sítě, která jsou přenášena do řídících center výrobců a distributorů a následně předávána inteligentními sítěmi zpět k samotným spotřebitelům, kteří tak mají možnost aktivně sledovat svoji aktuální spotřebu, znát aktuální náklady a moci díky tomu optimalizovat svoji spotřebu. Díky historickým datům mohou také analyzovat spotřebu z dlouhého období a výběrem spotřebičů s nižší spotřebou energie podniknout kroky ke snížení celkových nákladů. Je také možné vyhnout se tradičnímu modelu ročního fakturačního období s pravidelnými zálohami a platit tak vždy pouze skutečně spotřebovanou energii či zvolit si tarif co nejvíce efektivní vzhledem k chování spotřebitele.

Možnost toku dat opačným směrem, tedy od distributorů k zákazníkům, poté navíc umožňuje na dálku a automatizovaně měnit parametry zákaznické přípojky jako jsou používaný tarif, hodnota jističe či úplně odpojit zákazníka v případě neplacení za spotřebované energie nebo při vzniklé krizové situaci v síti pro zabránění jejímu kolapsu. [15]

Hlavní přednosti inteligentních sítí pro dané cílové skupiny je možné shrnout následujícím způsobem:

Zákazníci

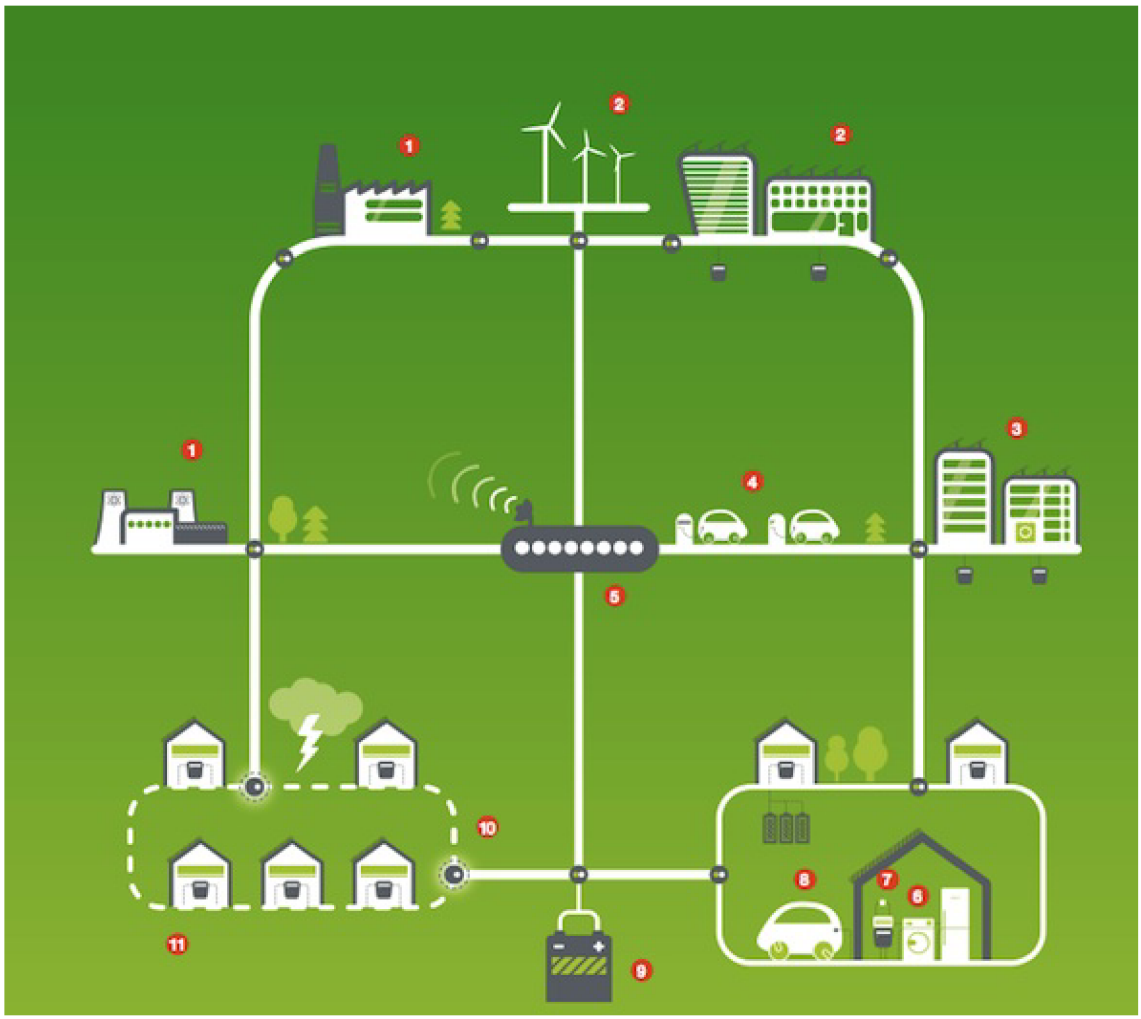
- okamžitá znalost aktuální spotřeby energie a placené ceny
- historické záznamy o spotřebě pro hledání optimalizací
- hrazení pouze skutečné spotřeby bez měsíčních záloh

Dodavatelé

- aktuální detailní stav sítě umožňující efektivní regulaci výroby a spotřeby
- automatický odečet spotřeby zákazníků
- možnost snadného odpojení neplatičů či během krize

2.1.1 Komponenty Smart Grids

Základní princip inteligentních sítí znázorňující požadovaný cílový stav je zobrazen na obrázku 2.1.



Obr. 2.1: Cílový stav Smart Grids [16]

Jednotlivě očíslované součásti v cílovém stavu mají následující význam: [16]

1. Tradiční velkokapacitní elektrárny
2. Alternativní zdroje energie – větrné farmy, solární panely
3. Kogenerační jednotka v místě spotřeby energie
4. Elektromobil včetně infrastruktury dobíjecích míst
5. Automatizované kontrolní centrum – slouží k ovládání sítě za pomocí v reálném čase získávaných informací ze sítě

6. Smart Home (chytrý domov) – schopnost automatického zapojování a odpojování spotřebičů k distribuční soustavě dle aktuálního stavu sítě, možnost odložení spotřeby vybraných spotřebičů na dobu mimo špičku odběru elektrické energie
7. Smart Meter – nainstalovaná inteligentní měřidla podporující oboustranný přenos informací po inteligentní síti a začlenění jednotlivých zákaznických spotřebičů rovněž do této sítě
8. Elektromobil – zde v podobě, která využívá jeho vnitřní akumulátor jako zásobník energie pro vyrovnávání sítě – v případě přebytků energie se dobíjí, při nedostatku je část jeho energie odčerpávána zpět do sítě.
9. Skladování energie – elektřina vyrobená během nižší spotřeby se uloží do baterií a spotřebovává později ve špičce
10. Dálkové ovladače a senzory – mají za úkol detekovat kolísání energie a případné poruchy v energetické síti, přičemž mohou postižené části automaticky izolovat od zbytku sítě a tím zabránit kolapsu soustavy
11. Izolovaná část distribuční sítě – díky možnosti flexibilního přesměrování toku elektřiny a izolování problémových míst se minimalizují dopady vzniklých poruch a výpadků pouze na dané postižené místo

Z uvedených informací je patrné, že celá koncepce je důležitou součástí pro možnost postupného zvyšování podílu obnovitelných zdrojů (solární panely, větrné farmy, kogenerační jednotky) na výrobě elektrické energie, jelikož často obtížná možnost předvídání množství jimi vyrobené energie a velké kolísání výkonů přímo vyžadují mít možnost okamžitě reagovat na tyto rychlé změny ovládáním připojených zátěží. S tím souvisí i celková decentralizace výroby elektrické energie z několika velkých elektráren do mnoha malých zdrojů v řádech desítek kW až MW. [16]

2.1.2 Smart Grids v ČR

V České republice začala s rozvojem inteligentních sítí společnost ČEZ roku 2010 v rámci pilotního projektu FutureMotion. Projekt je plánován až do roku 2015 a má za cíl vyzkoušet na vhodné lokalitě celý koncept inteligentních sítí a posoudit jeho přínos pro případné celostátní nasazení. Za testovací lokalitu tzv. „Smart Region“ bylo vybráno východočeské Vrchlabí, jelikož se jedná o menší ucelenou lokalitu, kde již existují vhodně připojitelné obnovitelné zdroje energie, prostor pro kogenerační jednotky i možnost vyzkoušet koncept E-Mobility (elektromobilita). Dále budou částečně pokryty i Pardubice, Jeřmanice a Hradec Králové. V současné době již bylo nainstalováno okolo 35 500 inteligentních elektroměrů. [17]

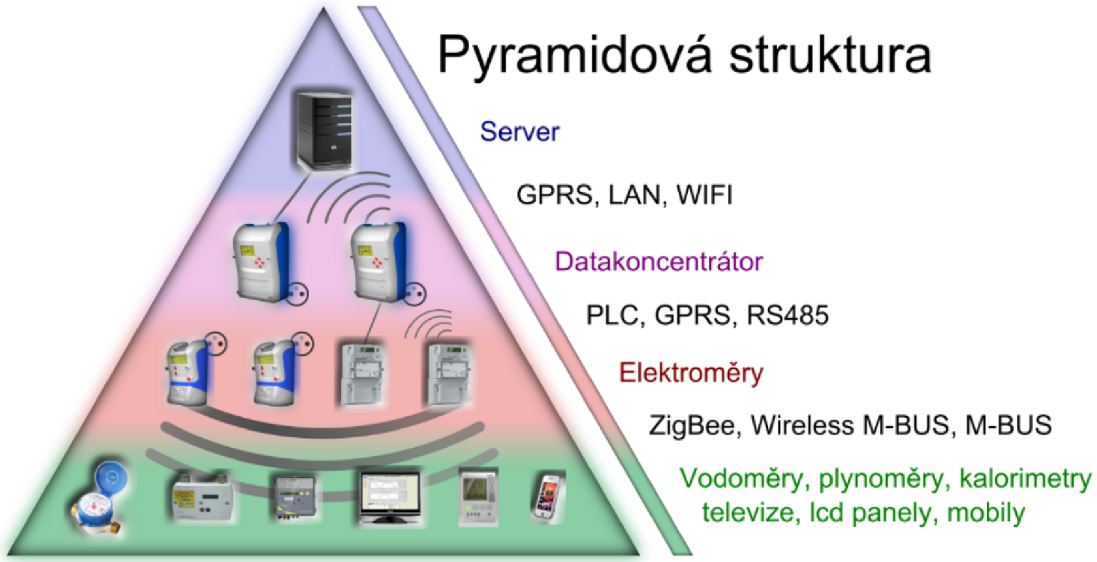
2.2 Smart Metering

Smart Meteringem označujeme koncept inteligentního měření, kdy je měřidlo určité veličiny vybaveno komunikačním rozhraním umožňujícím obousměrnou komunikaci mezi měřidlem a centrálním bodem. Tento koncept umožňuje automatizovaný sběr naměřených dat, jejich vyhodnocení a případné další řízení na základě těchto dat. Jedná se o jednu ze základních součástí inteligentních sítí. [17]

Smart Metering v inteligentních sítích vyjadřuje převážně měření a práci se spotřebou elektrické energie. Důvody pro rozvoj této oblasti jsou převážně zvýšení stability sítě a zabránění hromadným výpadkům. Přechod domácnosti na Smart Metering tedy nutně začíná výměnou stávajícího elektroměru za moderní inteligentní model tzv. Smart Meter.

2.2.1 Struktura Smart Meteringu

Struktura inteligentního měřícího systému má pyramidový charakter (zobrazena na obrázku 2.2) a skládá se ze čtyř úrovní, které mezi sebou vzájemně spolupracují.



Obr. 2.2: Pyramidová struktura technologie Smart Metering [15]

Vodoměry, plynoměry, kalorimetry atd.

Uvedená zařízení slouží k měření různorodých fyzikálních veličin s výjimkou elektrické energie. Dále se tedy může jednat i o teploměry, hladinoměry a další. Na-

měřená data předávají pomocí komunikačního rozhraní dále do elektroměru. Pro tuto komunikaci jsou využívány technologie ZigBee, M-BUS či Wireless M-BUS. Pokročilejší varianty měřidel mohou podporovat i povelování či vzdálenou konfiguraci směrem z elektroměru. [15]

Televize, LCD panely, mobilní telefony

Tato zařízení mají v inteligentních sítích za úkol informovat spotřebitele o aktuální spotřebě energie, využívaném tarifu, ceně za spotřebovávanou energii či libovolné další dostupné informaci. Zákazníkovi je tak umožněno přizpůsobit se aktuální situaci a dosáhnout maximální efektivity a úspory nákladů. [15]

Elektroměry

Elektroměry slouží primárně k měření spotřeby elektrické energie, monitorování napětí v instalačním místě a zaznamenávání dalších různých typů událostí týkajících se dané přípojky. Elektroměry také přijímají data z přiřazených měřidel a spolu s vlastními naměřenými daty je mohou předávat dále do data koncentrátoru v rámci PLC komunikace, kdy příslušný modem může být již rovnou součástí elektroměru či modulárně doplnitelný. V místech, kde PLC komunikace není použitelná se nahrazuje GPRS komunikací, rádiovým přenosem či v průmyslu metalickým vedením M-BUS, Ethernet, RS-485 apod. Elektroměr současně i přijímá data od data koncentrátoru a může tak provádět funkce jako spínání vnitřních relé, úplné odpojení zákazníka, změna tarifu a další. [15]

Tyto moderní elektroměry poskytují odběratelům výrazně lepší přehled o spotřebované elektřině a to přímo v reálném čase. Vylepšují tak stávající dvoustavový koncept hromadného dálkového ovládání (HDO) o možnost tvorby libovolného množství na míru šitých tarifů pro maximální optimalizaci výroby a spotřeby elektrické energie. Hlavní rozdíly oproti stávajícím elektroměrům jsou následující:

- **Běžný elektroměr** - tento typ elektroměru využívá dosud většina domácností v ČR. Data s naměřenou spotřebou elektřiny jsou průběžně ukládána do tzv. registru. Tento registr musí být každý rok ručně odečten pracovníkem distribuční společnosti a teprve na jeho základě je zpětně vyhotoveno vyúčtování. Nepodporuje žádný sběr statistických informací o spotřebě.
- **Inteligentní elektroměr** - měří spotřebu taktéž průběžně a zaznamenává ji do své paměti. Navíc však může monitorovat i kvalitu napětí v síti (přepětí, podpětí, odchylky frekvence), detekovat pokusy o manipulaci s elektroměrem

či mnohé jiné funkce. Naměřená data se zcela automaticky skrze inteligentní sítě přenášejí k distributorovi energie, kde jsou využita k řízení sítě. Ukázka inteligentního elektroměru je na obrázku 2.3. [17]

Obr. 2.3: Inteligentní elektroměr Meterus [18]

Data koncentrátory

Data koncentrátory fungují jako brány mezi komunikací po elektrické či rádiové síti a jinými typy komunikace, nejčastěji TCP/IP. Jelikož PLC komunikace není schopná projít trafostanicí, umisťují se data koncentrátory s výhodou právě sem do stávajících instalací. Jeden data koncentrátor přitom typicky obsluhuje kolem 100 elektroměrů, v hustých zástavbách je však možné se dostat i k číslu 1000. Data koncentrátor již přímo komunikuje se servery pomocí LAN, WiFi či GPRS spojení. [15]

Obr. 2.4: Data koncentrátor firmy ModemTec [5]

Server

Na vrcholu pyramidy jsou umístěny servery, které přijímají data z data koncentrátorů, vhodným způsobem je zpracují a odesílají výsledné řídící příkazy opět data koncentrátorům. Servery mohou obsluhovat pracovníci dohledového centra a předcházet tak případným problémům se sítěmi. [15]

3 NÁVRH UNIVERZÁLNÍHO SNIFFERU PRO INTELIGENTNÍ SÍTĚ

3.1 Definice požadavků na sniffer

Požadavky na výsledný softwarový sniffer vycházejí čistě z procesů, při kterých bude tento sniffer používán, tak aby jeho využití mělo pro uživatele maximální přínos. Mezi zásadní požadavky byly tedy definovány:

- Příjem dat pomocí sériové linky RS-232 s možností její konfigurace
- Funkce pro rozdělení přijatých dat na jednotlivé pakety
- Přehledné zobrazení přijatých paketů v reálném čase
- Rekonstrukce informací z paketů dle jejich typu a přehledné zobrazení všech těchto dat
- Možnost filtrování zobrazené komunikace dle extrahovaných dat
- Možnost přijatá data pouze ukládat do souboru pro pozdější zpracování
- Zobrazení statistik přijatých dat
- Možnost barevného odlišení zobrazených paketů dle typu
- Univerzálnost z hlediska snadné implementace dalších protokolů
- Použití na platformě Microsoft Windows
- Případný vývoj pomocí jazyka C++ a multiplatformních Qt knihoven

3.2 Možné varianty řešení

V rámci analýzy řešení bylo zjištěno, že možných variant návrhu a realizace tohoto snifferu existuje více a to i zcela odlišných. Tato kapitola se tedy zabývá výčtem i detailnějším popisem navržených řešení s uvedením výhod a nevýhod každé z nich. Jako hlavní kritérium bylo zvoleno splnění všech výše definovaných požadavků a následný poměr časová náročnost vývoje / výsledný rozsah funkčnosti. Navržená řešení jsou následující:

3.2.1 Použití již existujícího snifferu s napsáním pluginu pro parsing paketů

Nabízí se využít již existující softwarový sniffer, např. některý z představených výše v kapitole o snifferech. Tato varianta má velkou výhodu, že tyto softwarové sniffery jsou již od počátku psány jako univerzální a většinou již splňují velké množství z požadovaných funkcí. Časová náročnost vývoje by tedy závisela jen na tvorbě

příslušných pluginů pro parsování vlastních protokolů. V úvahu by připadaly jak nejpoužívanější Wireshark, tak i Microsoft Network Monitor. Díky velkému rozšíření těchto snifferů mezi uživateli by navíc mohli být bez jakéhokoliv zaučení okamžitě používány.

Zásadním problémem tohoto řešení se však stává požadavek na příjem dat z RS-232 linky. Sériové rozhraní je totiž na rozdíl od většiny síťových protokolů bajtově orientované a existující paketově orientované softwarové sniffery s ním tedy pochopitelně neumějí pracovat. Existují určité programy specializující se na zachytávání dat výhradně ze sériového portu, zde však zase chybí funkčnost požadovaná specifikací výše a žádný vhodný kandidát tak zde nalezen nebyl.

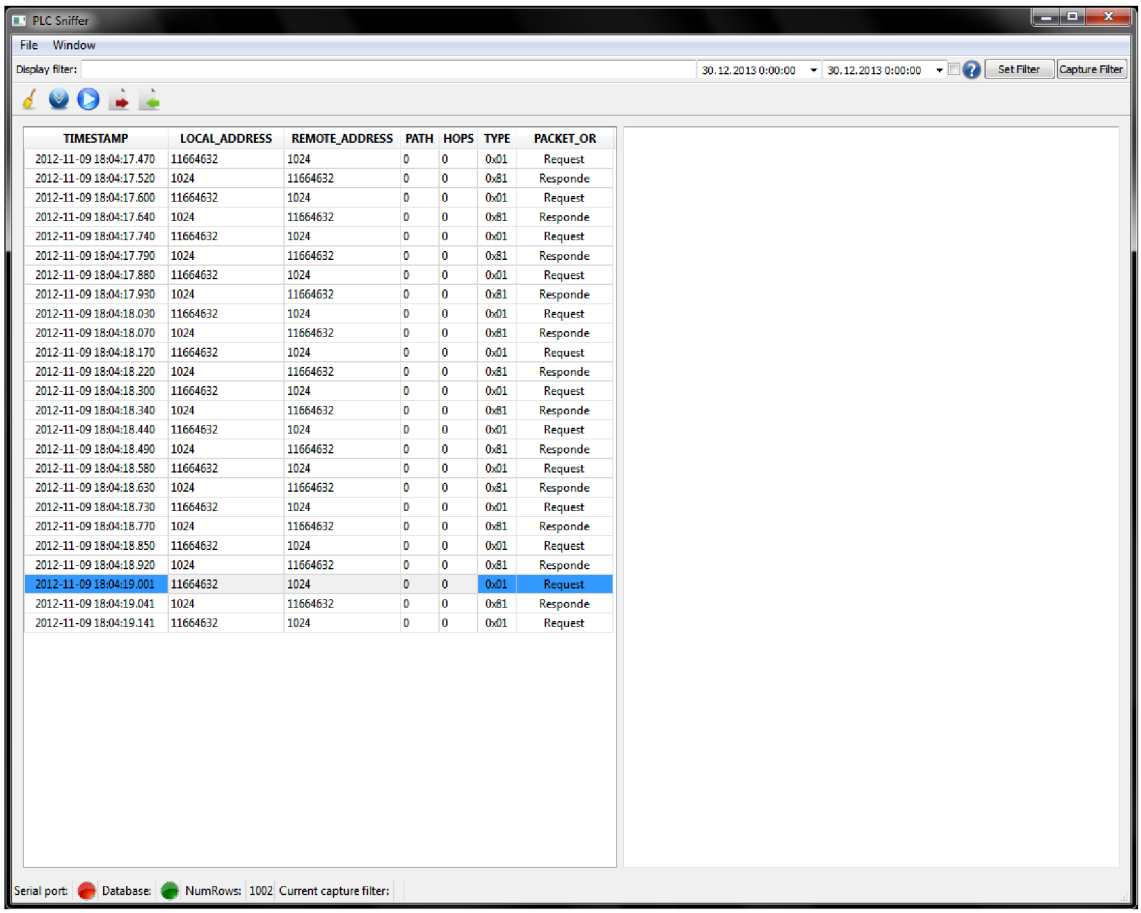
3.2.2 Pokračování ve vývoji snifferu firmy ModemTec

Firma ModemTec, v jejíž spolupráci je tato práce realizována, má již k dispozici základ vlastního softwarového snifferu vyvíjeného zcela od počátku pro použití k zachytávání PLC komunikace ze sériové linky. Projekt je vytvářen v Qt prostředí. Tato varianta řešení tedy spočívá v podstatném rozšíření funkčnosti stávajícího programu o všechny specifikované funkce. Výhoda tohoto řešení spočívá v možnosti maximálního možného přizpůsobení výsledné aplikace vlastním požadavkům, nevýhodou je naopak největší požadavek na čas strávený vývojem. Je třeba zdůraznit, že pro dosažení funkčnosti obdobné např. programu Wireshark by bylo třeba skutečně velké množství vývojářské kapacity.

Stávající stav projektu je zobrazen na obrázku 3.1. Analýzou zdrojových kódů bylo zjištěno, že implementovány jsou funkce pro jednoduché filtrování paketů, základní konfigurace a GUI, práce se soubory a XML definice typů paketů. Zatím ale není spolehlivě dořešena např. synchronizace RS-232 komunikace či vůbec podpora pro různé protokoly. Za nepříliš šťastné se dá považovat současné řešení zobrazeného seznamu paketů, kdy zřejmě kvůli paměťové náročnosti je zobrazeno vždy maximálně 25 paketů a je možné přejít pouze na další / předchozí stránku. Chybí zde vizuální informace o poloze v seznamu zachycených paketů a možnost rychle a intuitivně přecházet na požadovanou polohu v případě velkého množství dat. Zdrojové kódy jsou prakticky nekomentované, takže navázat na existující kód by zabralo o něco více času.

3.2.3 Vytvoření datové brány pro Wireshark a napsání parsovacích pluginů

Studiem dokumentace programu Wireshark bylo zjištěno, že tento sniffer jako jediný podporuje příjem dat pomocí tzv. Named Pipe - „pojmenované roury“. Jedná



Obr. 3.1: Rozpracovaný PLC sniffer firmy ModemTec

se o pojmenované místo (buffer) v paměti fungující na principu FIFO fronty, které je možné sdílet mezi procesy. V případě vytvoření této pojmenované roury vlastní aplikací a předáním jejího jména Wiresharku pro otevření je poté možné do Wiresharku v reálném čase dodávat libovolná data.

Principem tohoto řešení by tedy bylo vytvoření vlastní aplikace, která by zajišťovala příjem bajtů ze sériové linky, následně by tuto posloupnost bajtů byla schopná dle nastaveného protokolu rozdělit na jednotlivé pakety a ty poté pomocí pojmenované roury online předávat Wiresharku, který by sloužil jako zobrazovací jednotka. Do Wiresharku by se také musely doplnit pluginy pro parsování dat z jednotlivých paketů. Výhodou tohoto řešení je nižší časová náročnost a možnost využít všechny existující funkce Wiresharku. Jako nevýhodu zde lze spatřit nutnost používat dvě aplikace současně a z toho plynoucí nutnost přepínání během spouštění / pozastavení / ukončení příjmu dat.

3.3 Výběr výsledného řešení

Je zřejmé, že varianta číslo 1 není realizovatelná. Na výběr tedy zbývají pouze další varianty. Při uvážení kritéria poměru časové náročnosti vývoje / výsledného rozsahu funkčnosti lze očekávat, že 3. varianta řešení bude nejvýhodnější. Její náročnost vývoje je nižší, jelikož je třeba implementovat pouze RS-232 bránu a rozsah funkčnosti odpovídá minimálně všem funkcím obsaženým ve Wiresharku, což je pro praktické použití zcela dostačující. Kritéria uvedená na začátku této kapitoly jsou jinak splněna.

K nevýhodě třetí varianty - přepínání mezi aplikacemi – bylo navíc experimentálně zjištěno, že je pomocí detekce chyb při zápisu do pojmenované roury možné ve vytvořené aplikaci detekovat použití tlačítka Wiresharku pro ukončení zachytávání dat. Tímto částečně odpadá i problém nutného přepínání aplikací.

K další fázi návrhu byla tedy vybrána varianta č. 3, která bude dále analyzována.

3.4 Návrh GUI aplikace

Grafické uživatelské rozhraní (GUI) aplikace bylo navrženo v souladu s požadavky na funkčnost aplikace a je prezentováno na obrázku 3.2. Z důvodu zkušeností autora práce s prostředím Microsoft Visual Studio byl návrh GUI proveden právě v tomto prostředí jako jednoduchá .NET aplikace, i když výsledný sniffer bude později realizován v Qt prostředí.

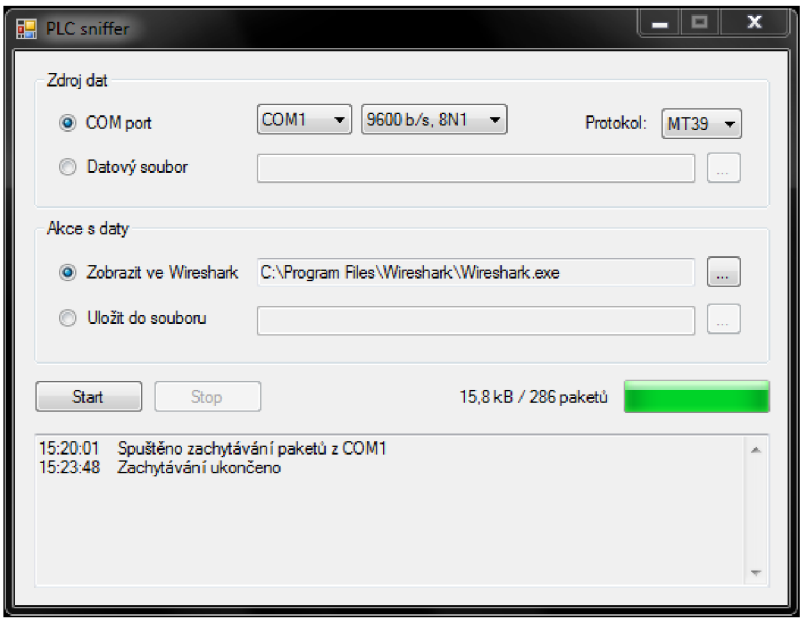
V horní části okna je k dispozici přepínací pole pro výběr a konfiguraci zdroje dat, se kterými bude program pracovat. Tím může být samotná RS-232 linka, tedy COM port, či uložený datový soubor pro zpracování dříve zachycených dat.

Níže je poté volena akce s předanými daty. Ty bude možné přeposílat pomocí pojmenované roury do Wiresharku, jehož exe soubor musí být tedy aplikaci známý, či v druhém případě bude možné ukládat data ze sériového portu přímo do souboru a až později analyzovat.

Následují tlačítka Start, Stop pro ovládání programu, poté jednoduchá statistika přijatých bajtů a nalezených paketů. Progress bar na pravé straně slouží pro rychlou vizuální identifikaci stavu aplikace. Nedílnou součástí je i log důležitých nastálých událostí ve spodní části.

3.5 Tvorba pluginu pro Wireshark

Při implementaci podpory vlastních protokolů do Wiresharku (tzv. dissectorů) jsou k dispozici dvě možnosti. První je napsání těchto pluginů v jazyce C++, jejich kompilace do DLL knihovny a následné překopírování do složky s pluginy Wiresharku.



Obr. 3.2: GUI navržené aplikace

Toto řešení není ideální, jelikož vyžaduje nastavení celého pracovního prostředí pro překlad Wiresharku, což je komplikované a časově náročné. [19] Navíc při každé menší změně protokolu by bylo nutné vše znovu rekompilovat.

Lepší řešení nabízí vestavěná podpora skriptovacího jazyka Lua přímo ve Wiresharku. Lua je poměrně jednoduchý skriptovací jazyk, který ale pro realizaci dissectorů zcela postačuje a navíc nám dává možnost vyhnout se jakékoliv kompilaci. Pro registraci pluginu současně se spuštěním Wiresharku, stačí následující příkaz:

```
wireshark.exe -X lua_script:"plugin.lua" -k -i "\\.\pipe\roura"
```

Druhá část příkazu ukazuje i možnost, pomocí které je Wireshark rovnou při spuštění napojen na zadanou pojmenovanou rouru a aktivováno zachytávání.

3.6 RS-232 knihovna

Jelikož bude aplikace realizována v Qt prostředí, bylo by vhodné zvolit i multiplatformní Qt knihovnu pro přístup k sériovému portu. Oficiální Qt komunitou vyvinutý je modul pod názvem QtSerialPort. Tato knihovna byla testována autorem práce, avšak chování funkcí pro synchronní čtení ze sériového portu se vůbec neshodovalo s očekávaným ani dokumentovaným chováním. Po nahlédnutí do zdrojových kódu pro Windows platformu bylo zjištěno, že modul využívá poměrně nestandardní metodu pro práci se sériovým portem, což by mohlo vysvětlovat uvedené chování a nebude tedy použit.

Další variantou je volně dostupná knihovna qextserialport, jelikož však pro Qt 5 není k dispozici jiná než beta-verze, nebude rovněž použita.

Z těchto důvodů bylo rozhodnuto zvolit pro přístup k sériovému portu samotné Win32 API funkce – CreateFile, ReadFile atd.

Jelikož hranice paketů přicházejících na sériový port mohou být často rozlišeny pouze z časových mezer mezi jednotlivými bajty, lze pro toto oddělení využít tzv. timeouty pro čtení mezi jednotlivými příchozími znaky. V případě, že je daný timeout překročen, byla právě zaznamenána hranice daného paketu. V souvislosti s tímto postupem a provedenými testy je nutné zmínit, že na Windows platformě se nepodařilo dosáhnout oddělení paketů v případě rozestupů kratších než 10 ms, což naznačují i některé neoficiální veřejné zdroje. [20] V případě kratších intervalů by tedy bylo nutné určovat hranice paketů jiným způsobem – např. pomocí kontroly často obsaženého kontrolního součtu, či se spolehnout na fakt, že již první přijatý bajt je prvním bajtem paketu (uvedený postup je použit ve výše popsaném základu ModemTec snifferu).

3.7 Formát dat pro Wireshark

Do pojmenované roury s připojeným Wiresharkem je nutno dodávat data v jasně specifikovaném formátu. Tento formát je definován knihovnou libpcap a skládá se z globální hlavičky na úplném začátku komunikace a dále z dat jednotlivých paketů, každý uvozen vlastní hlavičkou. [21]

Globální hlavička má tuto podobu:

```
typedef struct pcap_hdr_s {
    guint32 magic_number;  /* povinně 0xa1b2c3d4 */
    guint16 version_major; /* verze formátu (nyní 2) */
    guint16 version_minor; /* verze formátu (nyní 4) */
    gint32  thiszone;      /* časový posun v sekundách vůči UTC */
    guint32 sigfigs;       /* přesnost časových známek */
    guint32 snaplen;       /* maximální délka jednoho paketu */
    guint32 network;       /* identifikátor typu sítě */
} pcap_hdr_t;
```

Definice hlavičky jednotlivých paketů:

```
typedef struct pcaprec_hdr_s {
    guint32 ts_sec;    /* časová známka v sekundách */
    guint32 ts_usec;   /* časová známka v mikrosekundách */
```

```
    guint32 incl_len;  /* počet bajtů dat paketu za hlavičkou */
    guint32 orig_len;  /* původní délka dat paketu */
} pcaprec_hdr_t;
```

3.8 Celkový návrh fungování snifferu

Sniffer bude vytvořen v Qt prostředí s třídou hlavního okna postavené na QMainWindow. Program při startu automaticky načte a nastaví ovládací prvky do podoby v jaké byly před posledním řádným ukončením a to pomocí jejich uložení do konfiguračního souboru. Dojde k naplnění seznamu dostupných COM portů a inicializaci registrovaných protokolů, které budou všechny poděděny z bázové třídy Protocol, která zajistí potřebnou abstrakci pro nezávislost na jednom konkrétním protokolu.

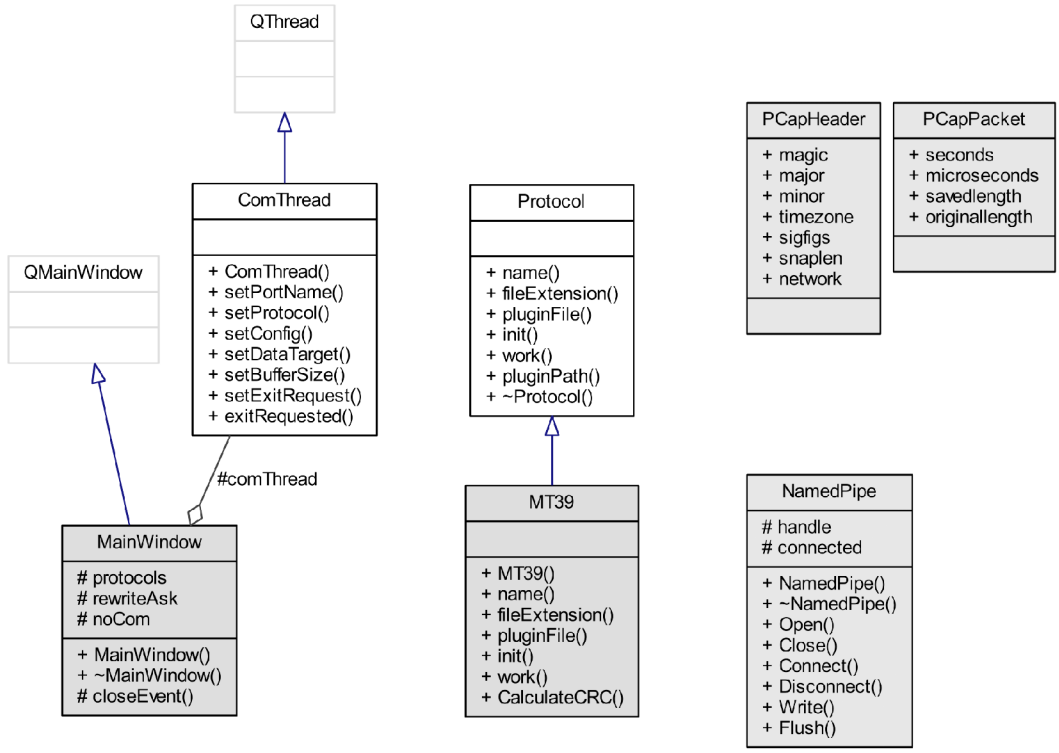
Po stisknutí tlačítka start a vybraném zdroji dat RS-232 bude spuštěno samostatné vlákno, které se bude starat o přijímání bajtů z portu a jejich transformaci na pakety. V případě současného zvolení cíle Wireshark bude vytvořena zadaná pojmenovaná roura a spuštěn program Wireshark s příkazovou řádkou, která zajistí jeho připojení na rouru. Zápis dat do roury bude probíhat opět v dalším samostatném vlákně, aby nedošlo k zablokování GUI aplikace. Zajištěna bude potřebná synchronizace vláken a aktualizace stavů v GUI.

4 REALIZACE SNIFFERU

4.1 Postup vývoje

V první fázi bylo nutné provést stažení a instalaci balíku mutliplatformních Qt knihoven, se kterými bylo současně nainstalováno i vývojové prostředí QtCreator použité pro celý následný vývoj, který probíhal v operačním systému Windows 7, 64 bitová edice. Pro správu verzí zdrojových kódů aplikace byl použit systém GIT. Následně byl v QtCreatoru založen nový projekt typu „Program s uživatelským rozhraním Qt“.

Dalším krokem bylo namodelování grafického uživatelského rozhraní do hlavního okna aplikace v souladu s dříve provedeným návrhem. Následovalo propojení GUI s obslužnými funkcemi pomocí definice slotů pro jednotlivé signály a postupné naprogramování všech funkcí aplikace, jejíž nejdůležitější principy budou dále detailněji rozebrány. Výsledná aplikace funguje za spolupráce pěti vytvořených tříd a dvou struktur, jejichž podoba je zachycena na obrázku 4.1.



Obr. 4.1: Diagram tříd vytvořeného PLC snifferu

4.2 Popis hlavních tříd snifferu

4.2.1 MainWindow

Třída MainWindow reprezentuje hlavní okno aplikace a jejím základním úkolem je příslušně reagovat na signály vyslané od GUI. Zajišťuje tak následující činnosti:

- Správa seznamu podporovaných protokolů, který se vytváří při startu aplikace
- Správa přednastavených konfigurací COM portu
- Načtení seznamu dostupných COM portů při startu aplikace
- Povolování / zakazování jednotlivých prvků GUI v závislosti na vybraném režimu
- Kontrola všech zadaných údajů před spuštěním vybrané činnosti
- Načtení a ukládání aktuálního nastavení aplikace do .ini souboru pro zachování při dalším spuštění
- Vytvoření pracovního vlákna pro zachytávání a ukládání komunikace a předání nastavených parametrů
- Blokování ukončení aplikace za běhu vlákna s předáním požadavku o ukončení
- Logování přijatých zpráv do připraveného textového pole

4.2.2 ComThread

Ve třídě ComThread je naprogramována hlavní činnost aplikace, tedy datová brána mezi COM portem a vybraným cílem. Vlákno je spuštěno třídou MainWindow s předáním všech parametrů nastavených přes GUI. V závislosti na nich provede třída inicilizaci cílového souboru nebo komunikační roury se zapsáním PCapHeader hlavičky a otevření vybraného COM portu. V případě roury je současně spuštěn Wireshark, který se na ni automaticky připojí.

Následně je v nekonečné smyčce volána pracovní funkce work() vybraného protokolu. Ta v případě nalezení paketu zpětně volá předanou funkci pro zápis paketu do vybraného cíle a to ve formátu PCapPacket. Současně ve smyčce probíhá každých 100 ms aktualizace zobrazovaných statistik a kontrola ukončovacího požadavku. Synchronizace s GUI vláknem je provedena pomocí signálů umisťovaných do jeho fronty zpráv, v opačném směru poté pomocí mutexu, jelikož pracovní vlákno nemá frontu zpráv. Podstatně rychlejší kritické sekce (Win32 API funkce InitializeCriticalSection, EnterCriticalSection a další) zde nebylo možné využít, jelikož nejsou v Qt knihovně multiplatformě implementovány. Mutexem je ochráněn přístup k vlákny sdílené proměnné vyjadřující požadavek GUI na ukončení pracovního vlákna.

Pouze v případě, kdy byl přes GUI vybrán režim otevření již v souboru zachycených dat ve Wiresharku, není toto vlákno aktivováno, jelikož postačí zavolat spuštění Wiresharku s příslušnými parametry v příkazové řádce.

4.2.3 Protocol

Třída Protocol je z většiny pouze abstraktní definicí požadavků, které musí splňovat odvozené třídy jednotlivých podporovaných protokolů. Detailněji je tato třída popsána v kapitole dokumentující postup přidání nového protokolu.

4.2.4 MT39

Třída obsahuje veškerou funkčnost nutnou pro zpracování příchozích dat z COM portů do formátu jednotlivých paketů protokolu MT39. Za zmínění stojí princip synchronizace na první bajt paketu. Jelikož protokol MT39 používá konstantní délku všech paketů a součástí dat je i kontrolní CRC součet, byla zde implementována synchronizační metoda, která přepočítává kontrolní součty na všech možných počátečních pozicích a až při souhlasném výpočtu CRC provede synchronizaci začátku dat paketu na tuto pozici.

4.2.5 NamedPipe

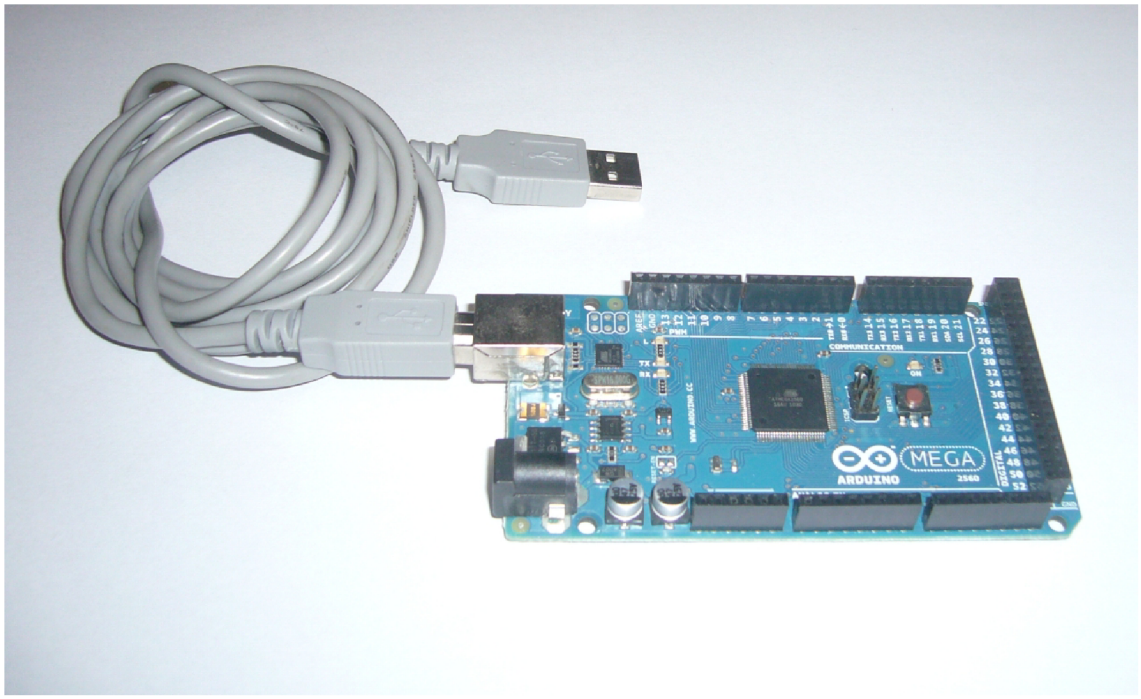
Tato třída reprezentuje objektové zapouzdření Win32 API funkcí pro práci s pojmenovanými rourami - CreateNamedPipe, ConnectNamedPipe, WriteFile a dalších. S její instancí pracuje ComThread v případě, že byl jako datový cíl zvolen online přenos do Wiresharku.

4.3 Simulátor PLC modemu

Jelikož vytvářená aplikace PLC snifferu má jako jednu z hlavních činností čtení dat z COM portu a jejich následné zpracování, bylo nutné vymyslet, jak příchozí komunikaci na portu během vývoje simulovat, aby program nebyl vytvářen bez možnosti okamžitého testování. Využít reálný hardwarový PLC sniffer by totiž znamenalo nutnost sestavení kompletní demonstrační PLC sítě, bylo tedy vhodnější nalézt jednodušší alternativu.

K vyřešení tohoto problému byla vybrána platforma Arduino, kterou měl autor práce k dispozici z dřívějších experimentů a to v provedení Mega2560. Jedná se o prototypovací platformu, založenou na procesoru Atmel ATmega2560, vyobrazena je na obrázku 4.2.

Zařízení se k PC připojuje rozhraním USB, přes které je virtuálně simulován přenos dat rozhraním RS232. Programování desky se provádí pomocí software Arduino Software IDE v jazyce C/C++, přičemž k dispozici je softwarová knihovna Wiring, pomocí které je možné velmi efektivně vytvářet programy využívající dostupné funkce procesoru.



Obr. 4.2: Platforma Arduino Mega2560 použitá k simulaci komunikace

Program nahraný do Arduina simulující paketovou PLC komunikaci na síti může mít poté např. následující podobu:

```
byte first[8]  = {0xb1,0x92,0xd3,0x40,0xd1,0x61,0x07,0x08};
byte second[8] = {0x10,0x15,0x27,0xa0,0xff,0x12,0xb1,0xc7};

void setup() {
      Serial.begin(115200);
}

void loop() {
      for(int x=0;x<8;x++) Serial.write(first[x]);
      delay(100);
```

```
        for(int x=0;x<8;x++) Serial.write(second[x]);
        delay(80);
}
```

V uvedené ukázce je provedena inicializace sériového rozhraní a následně jsou v nekonečné smyčce cyklicky odesílány dva předdefinované pakety proložené různými časovými prodlevami. V případě potřeby lze takto do simulované komunikace obdobně vkládat i záměrně chybné bajty pro ověření správné funkce kontroly CRC, synchronizace paketů a filtrování poškozených dat, což bylo v rámci vývoje rovněž realizováno.

4.4 Odlišnosti od návrhu

Během vývoje snifferu byly zjištěny některé specifické vlastnosti, kvůli kterým bylo nutné provedený návrh aplikace mírně upravit. Jedná se o následující změny:

4.4.1 RS-232 knihovna

V první fázi byla tato část snifferu vytvořena v souladu s návrhem, tedy obsluhou komunikace přímo pomocí Win32 API funkcí. Z důvodu preference firmy ModemTec, aby byla tato část programu multiplatformní a byla možná případná snadná portace zachytávacího systému na jiné platformy, bylo však tedy nakonec hledáno náhradní řešení.

Jelikož dříve navrhovaná knihovna qextserialport postoupila během vývoje snifferu do Release Candidate verze, bylo rozhodnuto o jejím využití a aplikace podle toho příslušně upravena.

Knihovna nabízí dvě metody práce s portem:

- QextSerialPort::EventDriven - třída při příchodu dat generuje signál
- QextSerialPort::Polling - na dostupnost dat je nutné se dotazovat

Je důležité poznamenat, že metoda Pollingu se při testech ukázala jako zcela nefunkční pro realizaci čtení malých dávek dat z portu (řádově desítky bytů). Například při scénáři, kdy je v příchozím vyrovnávacím bufferu portu uloženo 5 bytů, tato hodnota zjištěna a zavolána funkce pro načtení uvedených 5 bytů, vlákno aplikace se na desítky vteřin zablokuje ve čtecí funkci. Třída QextSerialPort je poděděna ze třídy QIODevice a nahlednutím do zdrojových kódů bylo zjištěno, že v Polling režimu se tato třída nehledě na žádané množství dat k přečtení pokouší vždy načíst celý svůj vnitřní buffer, který má řádově větší velikost a protože se čeká až tato

data na port dorazí, funkce se zablokuje. Z uvedeného důvodu bylo nakonec možné použít výhradně režim EventDriven a pravděpodobně se jedná i o příčinu zjištěných problémů během dříve zmíněných testů knihovny QtSerialPort.

4.4.2 Zapisovací vlákno

Původně navržený princip práce s daty obsahoval jedno vlákno pro čtení z portu a druhé vlákno pro zápis do pojmenované roury. V průběhu vývoje se však potvrdilo, že zápisem do připojené roury nemůže dojít k zablokování vlákna a bylo tedy zvoleno řešení obsahující pouze jedno vlákno, které při načtení paketu jej samo zapíše do roury či vybraného souboru. Zápis do souboru je na dnešních PC rovněž o několik řádů rychlejší než maximální rychlost příchozích dat na COM portu. Pro kontrolu byla navíc přidána funkce zjištění zaplnění čtecího bufferu COM portu a případné varovné hlášení. Tento stav je však spíše teoretický a v praktických testech nebyl nikdy zaznamenán.

4.4.3 GUI progress bar

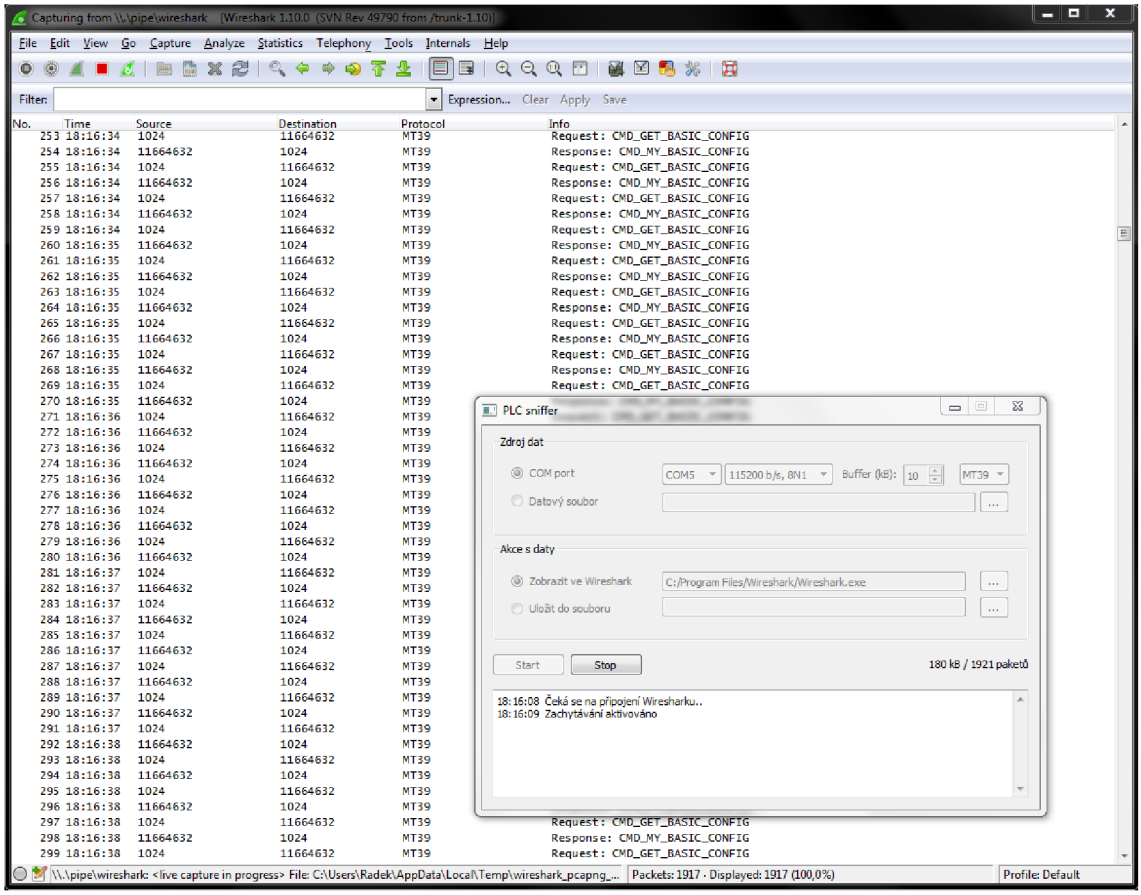
V návrhu GUI byl zakreslen ukazatel průběhu, který měl vizuálně zobrazovat aktuální stav aplikace. Cílem bylo během zachytávání využít tzv. marquee režim, kdy není nastavena žádná specifická hodnota a ukazatel zobrazuje opakující se animaci. Během testů tohoto režimu se však ukázalo, že při jeho vykreslení Qt knihovnou vznikají nepříjemné grafické artefakty a byl proto z návrhu odstraněn.

4.5 Ověření snifferu

Cílem vývoje snifferu byla v reálném provozu fungující aplikace, zachytávací část snifferu tedy musela být testována i ve skutečných podmínkách s připojeným hardwarovým PLC snifferem (modemem). V první fázi po dokončení této zachytávací části s podporou ukládání do souboru byl tedy rozpracovaný sniffer předán k testu firmě ModemTec, kde byla provedena zkouška fungování. Sniffer se podařilo úspěšně zprovoznit a provést ukázkové zachytávání dat na síti v laboratoři s protokolem MT39. Tato data byla následně využita během dalšího vývoje při tvorbě Lua pluginu pro Wireshark. Část snifferu pracující s COM portem byla tedy tímto úspěšně ověřena.

Po dokončení celé aplikace včetně MT39 pluginu prováděl ověření fungování všech zbylých funkcí autor práce s výsledkem, že vyvinutá aplikace úspěšně komunikuje s

programem Wireshark, kde také korektně pracuje vytvořený dissector pro protokol MT39. Tyto funkce již není třeba ověřovat v laboratoři a je tedy možné prohlásit, že ověření všech funkcí bylo úspěšné. Ukázka celkového fungování je přiložena na obrázku 4.3.



Obr. 4.3: Výsledný PLC sniffer během aktivovaného zachytávání dat

4.6 Rozšiřování snifferu o další protokoly

Jelikož navržený softwarový sniffer měl být univerzální ve vztahu k podporovaným protokolům, je nutné zmínit i postup, jakým tyto nové protokoly do snifferu implementovat. S touto myšlenkou byl celý sniffer vyvíjen, uvedený proces byl tedy cíleně navržen pro maximální rychlost a jednoduchost. Přidání nového protokolu se tedy skládá z následujících kroků:

Vytvoření třídy dědící z třídy Protocol s implementací těchto metod:

- QString name() - metoda má vracet zobrazovaný název protokolu
- QString fileExtension() - metoda vracející příponu asociovanou s daným protokolem. Jelikož se podle přípony rozlišují jednotlivé protokoly, musí být v rámci ostatních protokolů unikátní
- QString pluginFile() - metoda vracející název souboru s Lua pluginem pro tento protokol. Pluginy musí být umístěny ve složce dissectors u hlavního .exe souboru aplikace.
- bool init(QextSerialPort *port) - volitelně překrytelná metoda volaná během startu zachytávání komunikace, ve které má protokol možnost nastavit si v parametru předaný sériový port dle svých dalších potřeb.
- bool work(QextSerialPort *port, function packetFunc, function logFunc) - hlavní cyklicky volaná metoda, ve které má protokol možnost provádět čtení z portu, analýzu načtených bajtů a rozdělení na pakety. Pro každý identifikovaný paket má metoda zavolat v parametru předanou funkci packetFunc, ve které si sniffer již sám zpracuje předaný paket do formátu PCapPacket a následně jej odešle do vybraného cíle. Do parametrů funkce stačí tedy předat ukazatel na data paketu a délku (počet bytů) těchto dat. V metodě work se dále nesmějí volat dlouhodobě blokující operace, jinak by nebylo možné vlákno korektně ukončovat a sniffer by se stal neresponzivním. Návratová hodnota metody má být standardně true, v případě chyby lze však předat false a pracovní vlákno bude okamžitě ukončeno. Voláním logFunc("zpráva") lze navíc předávat zprávy k zobrazení v GUI.

Vytvoření Lua pluginu pro Wireshark:

- Soubor s pluginem musí odpovídat výše nastavenému pluginFile()
- Doporučený minimální obsah pluginu:

```
custom_proto = Proto("MT39","PLC protocol MT39")

function custom_proto.dissector(buffer,pinfo,tree)
  local subtree = tree:add(custom_proto,buffer(),"MT39 packet")
  pinfo.cols.protocol = "MT39"
end

dlt_table = DissectorTable.get("wtap_encap")
dlt_table:add(wtap["USER3"],custom_proto)
```

- Na prvním řádku ukázky je vytvořena instance nového protokolu, parametry jsou název a jeho krátký popis. Dále je definována funkce dissector, kterou Wireshark volá pro každý zpracovávaný paket. Parametr buffer zapouzdřuje samotná data paketu, pomocí pinfo.cols je možné nastavovat hodnoty sloupců v zobrazeném lineárním seznamu paketů a tree slouží k vytvoření stromové struktury s detailními naparsovanými informacemi o jednotlivém paketu.
- Zásadní částí pluginu jsou uvedené dva příkazy na konci. Díky konstantě USER3, na jejíž hodnotu sniffer nastavuje i pole network ve struktuře PCapHeader, dojde ke správnému spárování dissectoru s daty předanými Wiresharku.
- Samotné parsování informací z paketu se zapisuje dále uvnitř funkce dissector. Níže je uvedena ukázka zpracování protokolu, kde má paket v prvních dvou bajtech uloženou adresu cíle. Prvním parametrem funkce add je označení polohy dat (počáteční pozice a délka), druhým samotný zobrazený text v detailu paketu. Volání le_uint() převede data na zadané pozici v bufferu na číslo s uvažováním kódování little endian.

```
subtree:add(buffer(0,2),"Target: ".. buffer(0,2):le_uint())
```

- V pluginu je také možné definovat pole, podle kterých bude možné display filtrem ve Wiresharku filtrovat zobrazené pakety. Níže uvedená ukázka vytvoří pole s názvem mt39.target, pod kterým bude dostupné ve filtru. Pole bude mít popis "Target", datový typ 16 bitového čísla bez znaménka (uint16) v little endian (_le), hodnota bude extrahována z prvních dvou bajtů bufferu (0,2) a zobrazována bude v desítkové soustavě (base.DEC).

```
-- před funkcí dissector
custom_proto.fields.target = ProtoField.uint16("mt39.target",
                                "Target", base.DEC)

-- uvnitř funkce dissector
subtree:add_le(custom_proto.fields.target, buffer(0, 2))
```

- Jelikož množství podporovaných datových typů a funkcí je v Lua pluginech velmi široké, je vhodné vycházet při tvorbě z dostupné dokumentace k Wiresharku [22], kde jsou všechny přípustné funkce přehledně popsány. Rovněž lze doporučit zhlédnutí hotového komentovaného pluginu pro protokol MT39, na kterém je demonstrována většina nejčastěji využívaných operací.

Registrace nového protokolu v PLC snifferu :

- Do souboru mainwindow.cpp vložit na začátek: #include "NovyProtokol.h"
- V konstruktoru MainWindow() zavolat: AddProtocol(new NovyProtokol());

Po provedení uvedených kroků stačí již pouze aplikaci překompilovat a nový protokol bude od této chvíle plně podporován.

4.7 Přehled funkcí v programu Wireshark

Wireshark podporuje s načtenými daty velké množství funkcí, k nejužitečnějším a často používaným je zde krátce popsáno jejich použití. Všechny lze samozřejmě aplikovat na přijaté PLC pakety.

- Filtrování - do textového pole Filter stačí zadat příslušný omezující logický výraz např.: mt39.target == 18.
- Barevné rozlišení paketů - podrobné nastavení lze vyvolat z hlavního menu: View > Coloring Rules.
- Konfigurace zobrazení - lze zcela volně konfigurovat zobrazené panely a sloupce v seznamu paketů: Edit > Preferences > User Interface > Columns, Layout.
- Vyhledávání v datech - možné podle textového řetězce, display filtru či zadaných hexadecimálních dat. Dostupné z Edit > Find Packet.
- Synchronizace polohy v textovém detailu paketu s hexadecimálním výpisem - po výběru pole myší v jednom z panelů se v druhém automaticky vybere související údaj.
- Současné zobrazení detailů více paketů - po dvojkliku na paket v seznamu se otevře nové okno s jeho detaily. Těchto oken je možné otevřít libovolný počet.
- Statistiky zachycených dat - v textové formě, případně časový graf. Lze vyvolat z hlavního menu: Statistics > Summary, IO Graph
- Tisk, Exporty do jiných formátů (XML, C pole, CSV) - z hlavního menu: File > Print, Export.

4.8 Možnosti dalšího vývoje

Jelikož je vytvořený sniffer adaptérovou aplikací, která propojuje dva původně nekompatibilní datové toky a pro samotnou analýzu zachycených dat je použit velmi dobře funkčně vybavený Wireshark, není zde příliš velký prostor pro další vylepšování aplikace.

Jednou z možných cest dalšího vývoje by se nicméně mohlo stát zrušení nutnosti kompilovat kvůli novým protokolům celý program a umožnit díky oddělení do samostatně načítaných dynamických knihoven jejich přidání / odebrání pouhým zkopírováním do složky programu.

Jinou variantou by bylo vložit do snifferu podporu runtime modulu pro jazyk Lua a i samotnou funkčnost pro vytváření paketů z přijatých bytů programovat rovněž v tomto skriptovacím jazyce. Tím by kompilace taktéž zcela odpadla a navíc by bylo možné efektivně použité algoritmy upravovat.

Rovněž je vhodné zmínit, že ačkoliv byla aplikace vytvořena k užití v souvislosti s PLC komunikací, není rozhodně na tento typ komunikace omezena a je možné ji psaním vlastních pluginů využít pro zcela libovolnou komunikaci zachytitelnou na sériovém portu.

ZÁVĚR

Cílem této bakalářské práce bylo seznámit se s principy snifferů, inteligentních sítí, Smart Meteringu a následně navrhnout, realizovat a prakticky ověřit univerzální softwarový sniffer pro inteligentní sítě.

V první kapitole této práce byla podrobně popsána problematika snifferů jak hardwarových, tak i softwarových včetně ukázek konkrétních produktů a použití.

Ve druhé kapitole byly popsány inteligentní sítě, jejich prvky a výhody pro uživatele, struktura celé sítě a problematika Smart Meteringu.

Dále byly navrženy celkem tři varianty řešení, z nichž pouze dvě byly nakonec realizovatelné a z nich vybrána vhodná varianta řešení, tedy vytvoření datové brány pro Wireshark a doprogramování parsovacích pluginů. Rovněž byly specifikovány a otestovány technologie pro samotnou realizaci a navrženo GUI a chování výsledné aplikace.

Ve čtvrté kapitole byla podrobně rozebrána provedená realizace snifferu v souladu s navrhovaným postupem a celkové objektové řešení výsledné aplikace. Popsány byly rovněž důvody, proč se v některých oblastech musel návrh poupravit, metoda pro testování funkcí aplikace i bez dostupného hardwarového PLC snifferu a postup, kterým lze do snifferu přidávat podporu nových protokolů.

Sniffer byl rovněž úspěšně ověřen v laboratořích firmy ModemTec s reálnými hardwarovými sniffery a i druhá část komunikace s programem Wireshark na protokolu MT39 byla úspěšně otestována. Vytvořený produkt je tedy v souladu s definovanými požadavky a může být zaveden do praktického používání.

LITERATURA

[1] OREBAUGH, A., G. RAMIREZ a J. BEALE. *Wireshark & Ethereal Network Protocol Analyzer Toolkit.* Elsevier Science, 2006. ISBN 9780080506012. Dostupné z URL: <http://books.google.cz/books?id=-AdTE9S3kigC>.

[2] Packet analyzer. *Wikipedia* [online]. 2013 [cit. 2013-11-05]. Dostupné z URL: <http://en.wikipedia.org/wiki/Packet_analyzer>.

[3] Passive network tap. *Hack a Day* [online]. 2008 [cit. 2013-11-05]. Dostupné z URL: <http://hackaday.com/2008/09/14/passive-networking-tap/>.

[4] KeyKatcher 64K PS/2 Hardware Keylogger. *Amazon.com* [online]. 2013 [cit. 2013-11-05]. Dostupné z URL: <http://www.amazon.com/KeyKatcher-64K-PS-Hardware-Keylogger/dp/B004ZLV1UI>.

[5] ModemTec. *ModemTec.cz* [online]. 2013 [cit. 2013-12-21]. Dostupné z URL: <http://www.modemtec.cz>.

[6] Protocol Analyzers/Sniffers. *Softpedia* [online]. 2013 [cit. 2013-11-06]. Dostupné z URL: <http://www.softpedia.com/catList/193,0,1,0,1.html>.

[7] Wireshark. *Wikipedia* [online]. 2013 [cit. 2013-11-05]. Dostupné z URL: <http://en.wikipedia.org/wiki/Wireshark>.

[8] Wireshark. *Wireshark.org* [online]. 2013 [cit. 2013-11-06]. Dostupné z URL: <http://www.wireshark.org>.

[9] Microsoft Network Monitor. *Wikipedia* [online]. 2013 [cit. 2013-11-12]. Dostupné z URL: <http://en.wikipedia.org/wiki/Microsoft_Network_Monitor>.

[10] Microsoft Message Analyzer. *Microsoft.com* [online]. 2013 [cit. 2013-11-12]. Dostupné z URL: <http://www.microsoft.com/en-us/download/details.aspx?id=40308>.

[11] The Wireshark Wiki. *Wireshark Wiki* [online]. 2013 [cit. 2013-11-07]. Dostupné z URL: <http://wiki.wireshark.org>.

[12] tcpdump. *Wikipedia* [online]. 2013 [cit. 2013-11-05]. Dostupné z URL: <http://en.wikipedia.org/wiki/Tcpdump>.

[13] WinDump – Manual. *WinPcap* [online]. 2013 [cit. 2013-11-20]. Dostupné z URL: <http://www.winpcap.org/windump/docs/manual.htm>.

[14] Inteligentní sítě. *Wikipedie* [online]. 2013 [cit. 2013-12-05]. Dostupné z URL: <http://cs.wikipedia.org/wiki/Inteligentní_sítě>.

[15] FRANEK, Lešek. *Data koncentrátor pro chytré sítě: diplomová práce.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2012. 114 s. Vedoucí práce Ing. Pavel Kučera, Ph.D.

[16] Inteligentní sítě – Česká republika nezůstává pozadu. *Ekologické bydlení* [online]. 2010 [cit. 2013-12-05]. Dostupné z URL: <http://www.ekobydleni.eu/energie/inteligentni-site-ceska-republika-nezustava-pozadu>.

[17] Pilotní projekty ČEZ Smart Grids. *FUTUR/E/MOTION* [online]. 2013 [cit. 2013-12-07]. Dostupné z URL: <http://www.futuremotion.cz/smartgrids/cs/index.html>.

[18] Smart meter. *Wikipedia* [online]. 2013 [cit. 2013-12-12]. Dostupné z URL: <http://en.wikipedia.org/wiki/Smart_meter>.

[19] Creating Your Own Custom Wireshark Dissector. *CodeProject* [online]. 2007 [cit. 2013-12-20]. Dostupné z URL: <http://www.codeproject.com/Articles/19426/Creating-Your-Own-Custom-Wireshark-Dissector>.

[20] Comm timeout resolution. *Skupiny Google* [online]. 2003 [cit. 2013-12-21]. Dostupné z URL: <https://groups.google.com/forum/#!topic/comp.os.ms-windows.programmer.win32/UDqAi9O8vMA>.

[21] Development LibpcapFileFormat. *The Wireshark Wiki* [online]. 2013 [cit. 2013-12-21]. Dostupné z URL: <http://wiki.wireshark.org/Development/LibpcapFileFormat#Global_Header>.

[22] Lua. *The Wireshark Wiki* [online]. 2014 [cit. 2014-05-18]. Dostupné z URL: <http://wiki.wireshark.org/Lua>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

GUI grafické uživatelské rozhraní

PLC Power-line Communication

FIFO First In, First Out

CRC Cyclic redundancy check

API Application Programming Interface

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

- Zdrojové kódy vytvořené aplikace
- Elektronická verze práce