TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Nástroje pro testování protokolu IEC-104 pro automatizaci distribučních sítí

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 – Informační technologie
Autor práce: **Michael Sucharda**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Tools for Testing of IEC-104 Protocol for Distribution Grid Automation

Bachelor thesis

Study programme: B2646 – Information Technology
Study branch: 1802R007 – Information Technology
Author: **Michael Sucharda**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: 2015/2016

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Michael Sucharda**
Osobní číslo:	**M12000183**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Nástroje pro testování protokolu IEC-104 pro automatizaci distribučních sítí**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se podrobně s funkcemi protokolu IEC 60870-5-104 a s vybranými implementacemi v konkrétních terminálech, routerech ELVAC RTU a COM.TOM, případně jiných vhodných.
2. Na existujících aplikacích otestujte a zanalyzujte základní funkce a vlastnosti tohoto protokolu.
3. Vyberte vhodné softwarové řešení pro implementaci vlastního klienta, který prostřednictvím tohoto protokolu získá, archivuje a jednoduše vizualizuje měřená data.
4. Diskutujte výhody a nevýhody navrženého řešení, možnosti dalšího rozvoje a využití navržené aplikace.

Rozsah grafických prací: dle potřeby dokumentace

Rozsah pracovní zprávy: cca 30–40 stran

Forma zpracování bakalářské práce: tištěná/elektronická

Seznam odborné literatury:

[1] ELVAC A.S. Příručka uživatele RTU7.4, RTU7K a RTU7KL. Rev 6; 5/14. Ostrava, 2014.

[2] OLSEN, Ricardo. QTester104: IEC 60870-5-104 protocol tester. Sourceforge.net [online]. [cit. 2015-10-09]. Dostupné z: http://sourceforge.net/projects/qtester104/

[3] BECK IPC. IEC 60870-5-104 Controlled Station: from com.tom WEB-PLC 8.0.2. 2015. Dostupné také z: http://www.com-tom.de/download/other/Functionality_IEC60870-5-104_com.tom_WEB-PLC_V04.pdf

[4] SCHWARZ, Karlheinz. Comparison of IEC 60870-5-101/-103/-104, DNP3, and IEC 60870-6-TASE.2 with IEC 61850. 2002. Dostupné také z: www.nettedautomation.com

[5] SANCHEZ, G., I. GOMEZ, J. LUQUE, J. BENJUMEA a O. RIVERA. IEEE. Using Internet Protocols to Implement IEC 60870-5 Telecontrol Functions. 2009. Dostupné také z: http://ieeexplore.ieee.org/

Vedoucí bakalářské práce: Ing. Jan Kraus, Ph.D.
Ústav mechatroniky a technické informatiky

Datum zadání bakalářské práce: 10. října 2015

Termín odevzdání bakalářské práce: 16. května 2016

prof. Ing. Václav Kopecký, CSc.
děkan

Technická univerzita v Liberci L.S. - 14 -

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 16. 5. 2016

Podpis:

Anotace

Tato bakalářská práce se zabývá protokolem IEC 60870 používaným ve SCADA systémech. Zejména se zaměřuje na části 5-104 a 5-101 protokolu IEC 60870. V teoretické části se zabývá seznámením s protokolem IEC 60870 a jeho srovnáním s některými jinými protokoly používanými ve SCADA systémech. Popsány jsou některé existující implementace protokolu IEC 60870. Praktická část se zaměřuje na implementaci vlastní klientské a serverové aplikace protokolu IEC 60870-5-104. Následně pak na porovnání vlastních aplikaci s vybranými již existujícími aplikacemi.

Abstract

This bachelor thesis deals with IEC 60870 protocol used in SCADA systems. It focus especially on parts 5-104 and 5-101 of protocol IEC 60870. The theoretical part consists of learning about protocol IEC 60870 and it's comparison with some other protocols used in SCADA systems. There are also mentioned some already existing implementations of protocol IEC 60870. The practical part comprises of implementation of client and server applications of IEC 60870-5-104 protocol and their comparison with selected already existing applications.

Klíčová slova

IEC 60870-5-104, IEC 60870-5-101, SCADA, klientská aplikace, serverová aplikace

Key Words

IEC 60870-5-104, IEC 60870-5-101, SCADA, client application, server application

Obsah

Seznam obrázků

1 Úvod

Téma bakalářské práce, kterou jsem zvolil, se zabývá testování části 5-104 protokolu IEC 60870. Protokol IEC60870 se zabývá definicí norem pro dálkové řízení SCADA systému, a to zejména v elektrických sítích. SCADA systémy, z anglického Supervisory Control and Data Acquisition, neboli „dohled a sběr dat“, jsou systémy používané pro dálkové řízení územně rozsáhlých systémů, zahrnujících prakticky cokoliv od průmyslového závodu, až po národní rozvodní sítě. Běžnými součástmi těchto systémů jsou vzdáleně řiditelné jednotky (RTU), nebo programovatelné logické kontroléry (PLC). Většina kontrolních akcí je prováděna těmito zařízeními. Například PLC může řídit průtok v chlazení v části výrobního procesu. Ovšem SCADA systém umožňuje operátorovi nastavit hodnoty pro tento průtok a určit zobrazení nebo zaznamenání výstražných podmínek, jako je ztráta průtoku nebo vysoká teplota. Sběr dat pak začíná v RTU nebo PLC a zahrnuje přečtení těchto dat z měřidel a informací o stavu měřícího zařízení, které jsou zkombinovány tak aby operátor v řídící místnosti mohl rozhodnout o úpravě nebo přepsání normálních funkci RTU/PLC.

Pro účely komunikace přenosu dat a řízení v těchto systémech existují různé komunikační protokoly, které definují standardy pro podobu datových rámců, definují metody přenosu dat, komunikační sítě pro přenos těchto dat, metody řízení a přenosu řídících příkazu. Ve většině případu existují v těchto systémech dva typy stanic, stanice řídící a stanice řízené. Komunikace tak funguje buďto na principu master-slave nebo klient-server. Já se ve své práci zabývám protokolem definovaným v normě IEC 60780-5.

V teoretické části práce se zabývám popisem protokolu, zejména pak jeho pro moji práci nejdůležitějších částí 5-101 a 5-104. Popisuji zde zejména podobu přenášených dat definovanou v tomto protokolu. Seznámení se s tímto protokolem bylo jedním z cílů mé práce a odrazovým můstkem pro splnění dalších části. Také jsou zde uvedeny některé nejznámější podobné komunikační protokoly a uveden důvod proč ve své práci používám právě IEC 60870-5-104 Dále je zde také popis dalších použitých prostředků k seznámení s problematikou. Tedy nástroje umožňující analýzu a rozbor datových paketu přenášených sítí Ethernet a uspořádaných podle definic protokolu. Nebo o aplikace umožňující praktické navázání komunikace pomocí tohoto protokolu a vysílání řídích signálu.

V praktické části se pak dále seznamuji se strukturou dat přenášených protokolem IEC 60870 za pomoci již existujících řešení dané problematiky, jedná se zejména o volně dostupné aplikace řešící serverovou nebo klientskou část komunikace a také o jednotky RTU od firmy ELVAC a.s. Tato zařízení umožňují dálkový sběr dat a řízení pomocí několika protokolu, kdy jedním z nich je i IEC 60870-5-104.

Další část práce tvoří popis mého vlastního jednoduchého řešení klientské aplikace umožňující příjem a zobrazení přijatých dat. Moje klientská aplikace také umožňuje vyslání některých řídících příkazů definovaných v protokolem IEC 60870. Dále popisuji svou vlastní, velmi jednoduchou serverovou aplikaci, která mi následně umožnila provést zátěžový test mé klientské aplikace. Následuje tedy popis tohoto experimentu.

Nakonec stručně porovnávám svoje aplikace s již existujícími řešeními, které jsem v průběhu své práce využil. Toto srovnání provádím zejména z hlediska funkčnosti a ovladatelnosti mých aplikací. Na základě tohoto srovnání pak poukazuji na výhody a nevýhody mých řešení.

2 Teoretická část

V této části práce je obecně představen protokol IEC 60870, podrobněji rozebrány jeho části 60870-5-101 a 60870-5-104, které jsou zejména důležité pro tuto práci. Dále jsou představeny protokoly funkcí podobné IEC 60870 a je provedeno srovnání. Poté jsou obecně představeny další prostředky, použité dále v práci. Jedná se hlavně o již hotové aplikace implementující komunikaci prostřednictvím protokolu IEC 60870, které byli použity k nastudování a pochopení základních funkci protokolu.

2.1 Obecné představení normy IEC 60870

Norma IEC 60870 (u nás zavedená jako ČSN EN 60870) definuje systém pro dálkovou kontrolu, supervisory control and data acquisition, neboli SCADA systémy. Takové systémy jsou používány pro kontrolu elektrických přenosových sítí a jiných geograficky rozsáhlých systémů. Na specifikaci protokolů pracovala technická skupina TG 03 technického výboru TC 57 Mezinárodní elektrotechnické komise IEC, přičemž práce na ní začala v roce 1988. IEC 60870 má 6 části.

První dvě části IEC 60870-1 a IEC 60870-2 se zabývají obecnými ustanoveními (všeobecné zásady, výklad zvláštních výrazů) a provozními podmínkami (napájení a elektromagnetická kompatibilita, klimatické a další neelektrické vlivy).

Třetí část, IEC 60870-3, popisuje elektrické charakteristiky rozhraní, a to zejména podmínky pro rozhraní, které musí být splněny, aby mohly různé prvky spojené navzájem vytvořit funkční síť.

Čtvrtá část IEC 60870-4 se zabývá charakteristikami, které mají vliv na provoz systémů dálkového ovládání a souvisejí s vlastnostmi aplikace a funkcemi zpracování dat. Dále stanovuje soubor pravidel pro hodnocení a specifikaci požadavků na výkon.

Pro mojí práci nejdůležitější je pátá část IEC 60870-5 specifikující přenosové protokoly pro dálkové ovládání zařízení a soustav, které jsou založeny na sériovém přenosu binárně kódovaných dat a jsou určeny pro dohled a ovládání geograficky rozlehlých procesů.

Normy IEC 60870-5 vycházejí z modelu master-slave a specifikují funkce pro systémy dálkového ovládání. Dvě nejdůležitější z nich jsou, Report By Exception a mechanizmus přiřazovaní časových značek.

V komunikačních systémech s modelem master-slave je jedna řídící jednotka master, která odesílá požadavky všem svým podřízeným jednotkám, slaves. Každá podřízená jednotka reaguje na ty požadavky, které jsou jí určeny.

Pro jednotku master je často velmi důležité dozvědět se co nejrychleji o situaci, kdy na jednotce slave došlo k nějaké mimořádné události, tj. změnila se hodnota příslušné proměnné. K tomu je určena funkce Report By Exception (RBE). Ta umožňuje jednotce slave požádat o komunikaci s jednotkou master.

Norma IEC 60870-5 má několik částí, z nichž prvních pět je:

- IEC 60870-5-1 *Formáty přenosového rámce* – asynchronní přenos dat s linkovými protokoly *half-duplex* a *full-duplex,* standardy pro kódování, formátování a synchronizování datových rámců s hodnotami proměnných a metody zajištění integrity dat.
- IEC 60870-5-2 *Procedury spojového přenosu* – procedury pro sériový přenos kódovaných digitálních dat.
- IEC 60870-5-3 *Obecná struktura aplikačních dat* – pravidla pro strukturování jednotek aplikačních dat v přenosových rámcích.
- IEC 60870-5-4 *Definice a kódování aplikačních informačních prvků* – pravidla pro definování informačních prvků, zvláště digitálních a analogových procesních proměnných, často používaných v systémech dálkového řízení.
- IEC 60870-5-5 *Základní aplikační funkce* – standardy pro zajištění interoperability různých zařízení elektrizační soustavy.

Část IEC 60870-5-101 je Společná norma pro úkoly dálkového ovládání. Jejím cílem je umožnit funkční interoperabilitu mezi kompatibilními systémy. Publikována byla v roce 1995, a doplněna v roce 1998, kdy byl definován také přenos úplné časové značky.

Části IEC 60870-5-102 a IEC 60870-5-103 specifikují další společné standardy. První část specifikuje přenos integrovaných součtů hodnot v elektrizačních soustavách, druhá pak informační rozhraní ochran.

Část IEC 60870-5-104 s názvem Síťový přístup pro IEC 60870-5-101 používá normalizované transportní profily. Zjednodušeně řečeno, zatímco část 5-101 specifikuje mechanismy přenosu dat, část 5-104 stanovuje (nebo v některých případech jen doporučuje) jejich použití v běžných komunikačních sítích; z nich nejpopulárnější je Ethernet s TCP/IP spojením.

Část 6 skupiny standardů IEC 60780 se týká protokolů pro dálkové ovládání kompatibilních se standardy ISO a doporučeními ITU-T. Od části 5 se liší zejména tím, že vychází z modelu klient-server a je využívána pro větší systémy zpracování dat a počítačové sítě. Jde především o přenos informací mezi jednotlivými dispečinky elektrizační soustavy.

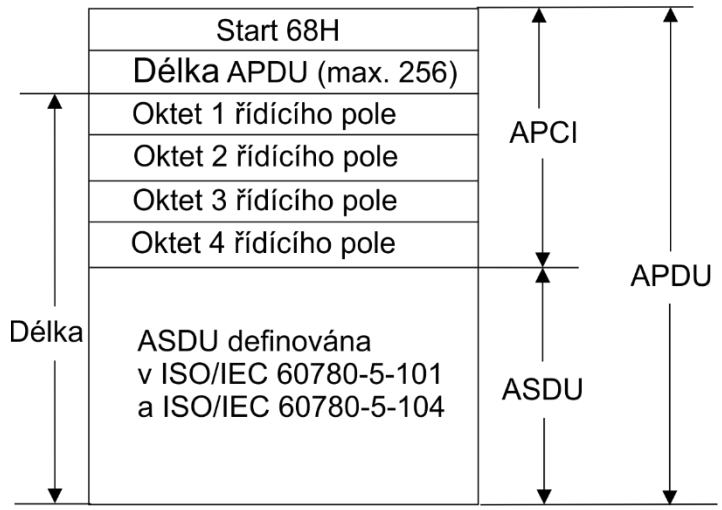
Zde bych rád upozornil na skutečnost, že řídící jednoty případně aplikace master, jsou při používání těchto norem označovány jako klienti. Pod pojmem server pak vystupují jednotky řízené, tedy ty které se starají o sběr a posílání dat.

2.2 Podrobnější představení částí 5-101 a 5-104 protokolu IEC 60870

Jak již bylo zmíněno výše část 5-101 definuje metody přenosu dat mezi jednotlivými zařízeními a část 5-104 určuje použití těchto metod v běžných komunikačních sítích. ASDU, jednotka dat aplikační služby (z anglického Application Service Data Unit), definovaná podle části 5-101 tak například neobsahuje žádný stop a start mechanizmus nebo mechanizmus pro ochranu před ztrátou nebo duplikací dat. K tomuto účelu definuje část 5-104 APCI, řídící informace aplikačního protokolu (z anglického Application Protol Control Information). ASDU a APCI pak dohromady tvoří celek označovaný APDU, to znamená jednotka dat aplikačního protokolu (z anglického Application Protocol Data Unit).

2.2.1 Část 5-104

Pro stanovení začátku ASDU obsahuje každá APCI následující vymezující prvky: start znak, stanovení délky ASDU a řídící pole (viz obrázek).



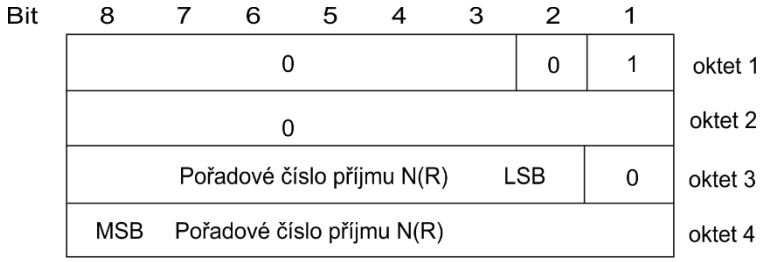
Obrázek 1: APDU definované podle normy

Start 68H definuje začátek toku dat. Délka APDU definuje délku hlavní části APDU, což zahrnuje čtyři oktety řídícího pole v APCI a celé ASDU. Prvním počítaným oktetem je první oktet řídícího pole a posledním počítaným oktetem je poslední oktet ASDU. Maximální délka ASDU je tak 249, neboť maximální hodnota délky pole APDU je 253 (255 – oktet start a oktet délky) a délka řídícího pole jsou 4 oktety. Řídící pole definuje informaci pro ochranu před ztrátou a duplikací zpráv, začátek a konec přenosu a kontrolu přenosového spojení. Používají se 3 typy formátu řídícího pole.[1]

Formát pro provádění číslovaných informačních přenosů (I formát). První bit oktetu 1 řídícího pole nastaven na 0, definuje I formát. APDU s I formátem vždy obsahuje ASDU[1]. Řídící informace o formátu je uvedena na obrázku.

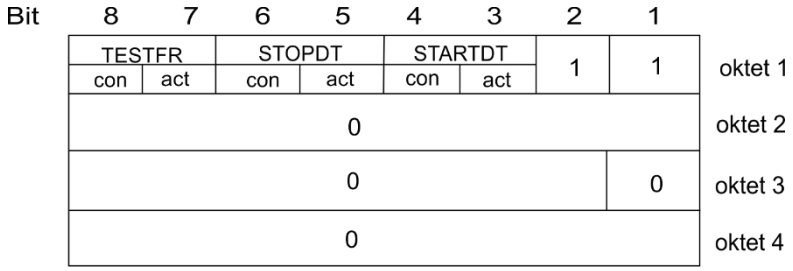
Obrázek 2: Řídící pole typu I formát

Formát pro provádění číslované kontrolní funkce (S formát). První bit oktetu 1 řídícího pole nastaven na 1 a druhý bit nastaven na 0 definuje S formát. APDU s S formátem obsahují pouze APCI[1].



Obrázek 3: Řídící pole typu S formát

Formát pro provádění nečíslované řídící funkce (U formát). První bit oktetu 1 řídícího pole nastaven na 1 a druhý bit nastaven na 1 definuje S formát. APDU s U formátem obsahuje pouze APCI. Řídící informace U formátu uvedena na obrázku. Kdykoliv může být aktivní pouze jedna funkce TESTFR (test rámce), STOPDT (ukončení přenosu dat) nebo STARTDT (zahájení přenosu dat).[1]



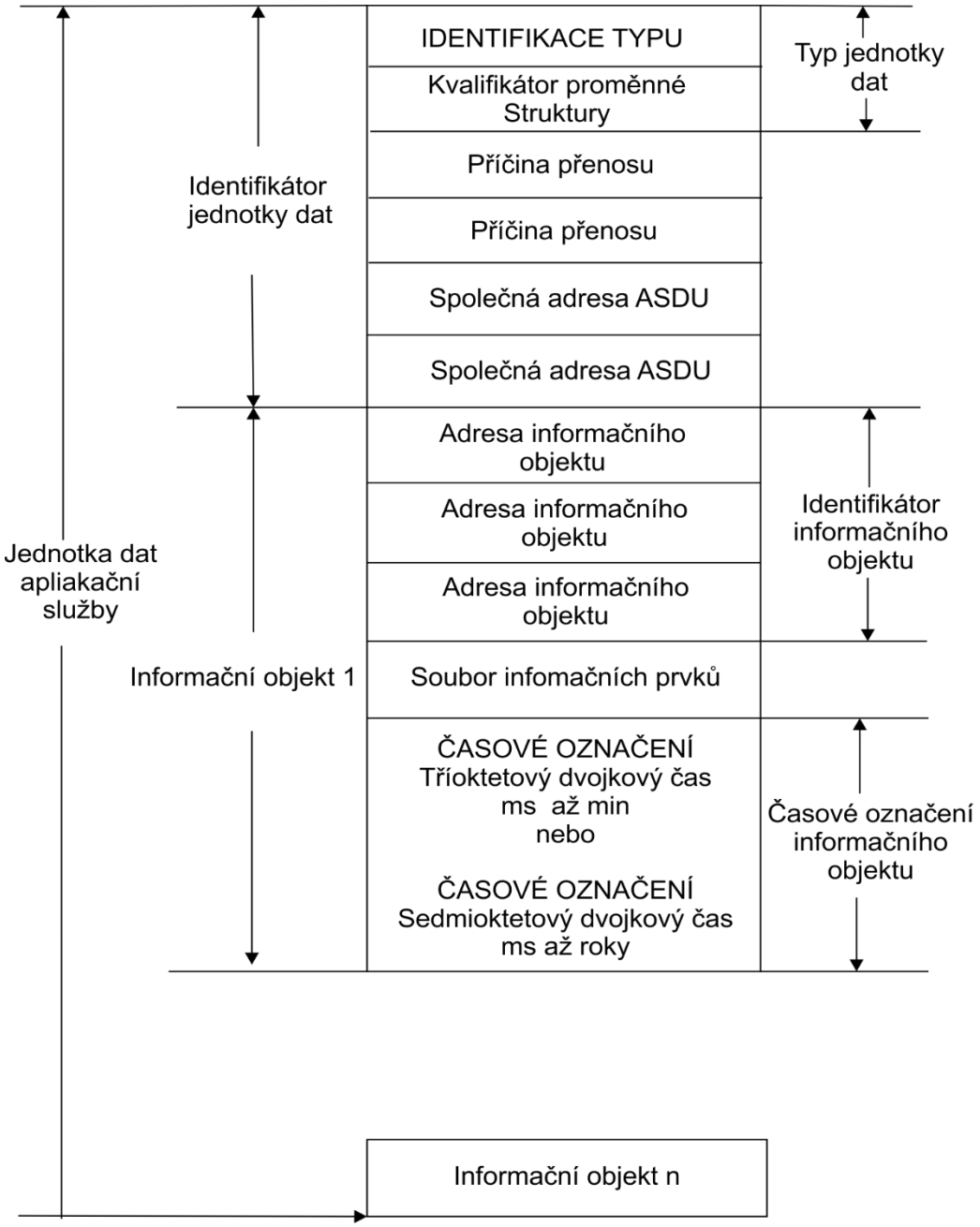
Obrázek 4: Řídící pole typu U formát

V případě, že spojení mezi řídící a řízenou stanicí je již zahájeno, avšak není používáno, je možné ho v obou směrech pravidelně testovat vysíláním zkušebních APDU, TESTFR = act, které přijímací stanice potvrzuje vysláním TESTFR = con. Obě stanice mohou zahájit zkušební proceduru po stanoveném časovém úseku, v němž nedošlo k přenosu dat.

Příkazy STARTDT, zahájení přenosu dat, a STOPDT, ukončení přenosu dat, používá řídící stanice k řízení přenosu dat z řízené stanice. Při vytvoření spojení mezi stanicemi není automaticky zahájen přenos dat. Výchozí nastavení je tedy STOPDT. V tomto stavu nevysílá řízená stanice žádná data s výjimkou nečíslovaných řídích funkcí a potvrzení takových funkcí. Řídící stanice musí aktivovat přenos uživatelských dat po daném spojení vysláním STARTDT act. Na tento příkaz řízená stanice odpoví STARTDT con. Pokud není STARTDT potvrzen, dojde k ukončení daného spojení mezi stanicemi.

Veškerá neodbavená uživatelská data z řízené stanice jsou vyslána až po STARTDT con. V případě přepnutí aktivního spojení na jiné spojení vyšle řídící stanice STOPDT act, řízená stanice zastaví přenos dat a vyšle STOPDT con. Po přijetí STOPDT con může řídící stanice dané spojení ukončit.

2.2.2 Část 5-101



Obrázek 5:Struktura ASDU: Počet oktetů příčiny přenosu je 1 nebo 2. Počet oktetů společné adresy ASDU je 1 nebo 2. Počet oktetů adresy informačního objektu je 1,2 nebo 3. Toto musí být v systému pevně nastaveno.

Identifikace typu definuje strukturu, typ a formát následujících *informačních objektů*. Je definován jako číslo od 1 do 255. *Informační objekty* jsou rozlišeny různými čísly *identifikace typu*. Hodnota 0 se nepoužívá a v normě je definován rozsah hodnot 1 až 127. Hodnoty 128 až 255 definovány nejsou. Hodnoty v rozsahu 136 až 255 pak mohou nezávisle definovat uživatelé normy[2].

Konkrétně pak:

- Rozsah 0 až 44 slouží pro provozní informace ve směru sledování.
- Rozsah 45 až 69 slouží pro provozní informace ve směru ovládání.
- Rozsah 70 až 99 slouží pro systémové informace ve směru sledování.
- Rozsah 100 až 109 složí pro systémové informace ve směru ovládání.
- Rozsah 110 až 119 souží pro parametr ve směru sledování.
- Rozsah 120 až 127 slouží pro přenos souborů.

Dále pak například:

- 9 označuje měřenou normalizovanou hodnotu
- 13 značí naměřenou hodnotu s pohyblivou řádovou čárkou
- 45 je jednoduchý povel
- 100 označuje dotazový povel
- 102 značí povel čtení
- 103 značí povel pro časovou synchronizaci
- 104 označuje zkušební povel

Kvalifikátor proměnné struktury určuje počet informačních objektů nebo prvků obsažených v ASDU na prvních 7 bitech (tedy 0 až 127). Osmý bit pak určuje adresování informačních objektů a prvků.[2]

Příčina přenosu identifikuje na prvních 6 bitech příčinu přenosu. 7. bit slouží pro rozlišení kladného a záporného potvrzení aktivace. Na 8. bitu je definován zkušební bit u těch ASDU, které byli definovány během zkoušky. Následující oktet je nepovinný a slouží k udání adresy původce ASDU, přičemž hodnota 0 je standardní a slouží k definování provozních informací, které musí byt přeneseny na všechny části decentralizovaného systému, hodnoty 1 až 255 lze použít pro adresování specifické části systému, na kterou je vrácena příslušná informace. Příkladem konkrétních hodnot může

být hodnota 1 značící periodické přenosy, hodnota 3 značící spontánní přenosy nebo hodnoty 6 a 7 značící aktivaci, respektive potvrzení aktivace.

Společná adresa ASDU určuje adresu konkrétní stanice. Jedná se o parametr, který je pro systém pevně nastaven. Hodnota 0 se nepoužívá. Maximální hodnota, tedy 255 nebo 65535 (v závislosti na použití jednoho nebo obou oktetů) je globální adresa sloužící pro všechny stanice.

Adresa informačního objektu je parametr, který je v systému pevně nastaven. Ve směru ovládání se tato hodnota používá pro místo určení, ve směru sledování pak pro určení zdroje. ASDU s nedefinovanými adresami informačního objektu řídící stanice nebere v potaz. Třetí oktet se používá pouze v případe strukturování adresy.[2]

Dále v ASDU následují informační prvky, které jsou strukturovány podle definic v IEC 60870-5-4. Sem spadá v zásadě ta nejpodstatnější část celého přenosu, tedy přenášená hodnota. Druhy přenášených informací definované podle normy jsou různé. Jedná se například o jednobitovou nebo dvoubitovou informaci, normalizovanou hodnotu, hodnotu s měřítkem, krátké číslo s pohyblivou řádovou čárkou. Dále jsem, pak spadají povely např. jednoduchý povel, dvojpovel a různá časová označení informačního objektu.

Dále je zařazen i kvalitativní deskriptor, což je informace skládající se z 5 bitů, které poskytují řídící stanici dodatečné informace o kvalitě informačního objektu. Tato informace se skládá z:

- Přeplněno/nepřeplněno – indikuje, zda hodnota informačního objektu nepřekročila stanovený rozsah hodnot.
- Blokováno/neblokováno – hodnota je pro přenos blokována.
- Zaměněno/nezaměněno – hodnota určena automaticky nebo vstupem obsluhy.
- Neaktuální/aktuální – hodnota je aktuální pokud její poslední aktualizace proběhla úspěšně.
- Neplatné/platné - pokud je po zaznamenání hodnoty zjištěn nenormální stav zdroje této informace, je označena jako neplatná, hodnota informačního objektu pro tento stav není definovaná.

2.3 Podobné komunikační protokoly a srovnání

2.3.1 DNP3

DNP (Distributed Network Protocol) byl původně vyvinut společností GE Harris Energy Control Systems Canada Ltd. (původně Westronic Inc.) v roce 1990. Na konci roku 1993 bylo vlastnictví protokolu a zodpovědnost za další definování specifikací DNP3 převedeno na DNP User Group a dokument sady specifikací DNP 3.0 Basic 4 byl uvolněn do veřejné domény. V roce 1995 byla vytvořena technická komise pro prozkoumání vylepšení protokolu a jejich navrhnutí ke schválení obecné uživatelské skupině. Jedním z nejdůležitějších úkolů této komise bylo vydat „DNP Subset Definition" dokument, který stanovil standardy pro implementace DNP3. DNP3 je založen na ranné práci IEC Technické komise 57 (TC-57), která vedla ke vzniku IEC 60870-5. DNP3 je součástí IEEE Standardu 1379, a byl vyvinut speciálně pro SCADA aplikace. Je navržen pro přenos relativně malých paketů spolehlivým způsobem.[4]

DNP3 je inteligentní otevřený protokol, který umožňuje:

- Posílání dotazu a odpovědí s více datovými typy v jedné zprávě.
- Rozdělení zprávy do více rámců k zajištění výborné detekce chyb a zotavení se z nich.
- Přidání pouze změněných dat do odpovědi.
- Přiřazení priority datům a dotazování dat periodicky podle jejich priority.
- Posílat zprávy bez předchozího dotazu.
- Synchronizaci času a standardní časové formáty.
- Více masterů a peer-to-peer operací.
- Uživatelům definovat vlastní objekty zahrnující přenosu souborů.

DNP3 je vrstvený protokol, který místo aby se držel 7. vrstvy OSI protokolu, přiklání se kvůli jednoduší implementaci k 3 vrstvému standardu navrženému IEC. Ovšem DP3 přidává čtvrtou pseudo-transportní vrstvu, která umožnuje rozdělení zpráv. Fyzická vrstva se zabývá mediem, přes které jsou data přenášena. V základu je DNP3 definován přes jednoduchou sériovou linku, jako je například RS-232. Datová vrstva se zabývá logickým spojením mezi odesílatelem a příjemcem a vylepšuje zvládání chyb. Toho je dosaženo uvedením každého datového rámce hlavičkou a vkládáním 16-bitové CRC, každých 16 bytů. Pseudo-transportní vrstva rozděluje zprávy aplikační vrstvy na

více datových rámců. Aplikační vrstva se stará o odpovědi na přijaté dotazy a vytváří zprávy na základě potřeby a dostupných dat. [5][6]

2.3.2 IEC 61850

IEC 61850 je standart pro komunikaci a modelování data při designu automatizovaných elektrických rozvodných sítí. Protokol IEC 61850 je součástí IEC TC-57. IEC 61850 popisuje způsob komunikace zařízení a související systémové požadavky. Pro popis dat a procesu, které mají být implementovány a kontrolovány, používá objektově orientovaný model. Použití objektově orientovaného přístupu poskytuje větší flexibilitu vývojářům a usnadňuje použití pro uživatele. Hlavním stavebním prvkem protokolu IEC 61850 je abstrakce definic datových prvků a služeb, to znamená vytváření dat/objektů a služeb nezávisle na jakémkoli jiném přenosovém protokolu. Tyto abstraktní definice poté umožňují mapování datových objektů a služeb, na kterýkoliv jiný protokol, který splňuje požadavky pro přenos dat a služeb. IEC 61850 pak definuje způsob mapování na Manufacturing Message Specification (MMS) a také mapování Sample Measured Values na Ethernetový datový rámec. Za nejdůležitější způsob přenosu dat definovaných v IEC 61850 je považován GOOSE (Generic Object Oriented Substation Event). Jedná se o rychlou nespojovou komunikační službu, která je využívána k přenosu časově kritických dat, kdy rychlost a bezpečnost přenosu jsou dosažený opakovaným odesláním zprávy.

Z hlediska celého systému je vyžadováno značné množství nastavení, aby všechny částí správně spolupracovaly. S cílem usnadnit tento proces a omezit množství lidských chyb definuje protokol IEC 61850 Substation Configuration Language (SLC). Jedná se jazyk pro konfiguraci zařízení používajících IEC 61850, založený na XML. Každé zařízení musí poskytovat SCL soubor popisující jeho konfiguraci. IEC 61850 poskytuje obsáhlý model shodný pro všechny výrobce a typy zařízení, jak mají zařízení organizovat data. To eliminuje velkou část nastavování jednotlivých zařízení, protože ta jsou často schopna se nastavit sama na základě nahraného SCL souboru.[7][8]

2.3.3 Modbus

Modbus byl publikovaný v roce 1979 společností Modicon (nyní Schneider Electric) pro použití s PLC. Vývoj a aktualizace jsou od roku 2004 řízeny Modbus Organization. Modbus je komunikační protokol nacházející se v 7. vrstvě modelu ISO/OSI. Modbus umožňuje komunikaci klient/server mezi zařízeními připojenými na

různých typech sítí a sběrnic. V současné době podporuje řadu komunikačních medii, např. sériové linky typu RS-232, RS-422 a RS-485, optické a radiové sítě nebo síť Ethernet s využitím protokolu TCP/IP (obvykle nazýváno Modbus/TCP). Komunikace probíhá metodou požadavek/odpověď a požadovaná funkce je specifikována pomocí kódu funkce, který je součástí požadavku.

Datový model Modbusu je založen na sadě tabulek, kdy definovány jsou čtyři základní tabulky:

- Diskrétní vstupy: 1-bit pouze pro čtení.
- Cívky: 1-bit pro čtení/zápis.
- Vstupní registry: 16-bitové slovo pouze čtení.
- Uchovávací registry: 16-bitové slovo pro čtení/zápis.

Každá z tabulek může mít až 65536 položek.[9]

Modbus se chová ke všem datum jako k přednastaveným hodnotám. Modbus čte hodnoty z rozsahu svých vstupů (nebo výstupů) provedením jednoho dotazu na přečtení každého rozsahu nebo typu. Standartní Modbus nemá koncept „eventů“ (přechodné indikace) nebo času. Data, která nejsou získána během čtení, jsou ztracena ve chvíli, kdy jsou přepsána novým polem dat. Protože Modbus podporuje zpětné čtení výstupů, hodně zařízení implementuje pouze výstupní typ dat, tedy všechny vstupy jsou čteny a adresovány tak, jako by se jednalo výstupy. Velká část výrobců Modbus zařízení přidává vlastní rozšíření funkčnosti nad rámec standartního Modbusu. Tato skutečnost a běžné využívání výstupů jako vstupů někdy značně stěžuje spolupráci i jednoduchých Modbus zařízení.[10]

2.3.4 Srovnání

V zásadě lze říči, že co se poskytování základních real-time informací ve SCADA systémech týče, poslouží každý z těchto protokolů stejně. Modbus/TCP, DNP3 a IEC 60870 by byly dokonce schopny poskytnout stejnou funkcionalitu i pro řízení celého SCADA systému, nejen pro získávání dat. Ovšem Modbus má jisté nedostatky, které způsobily, že byl pro daná použití nevhodný. Prvním problém byla neexistence autority, která by stanovala normu pro protokol Modbus, a tím pádem vznikla možnost, přidávat vlastní funkce a vylepšení k Modbusu ze strany výrobců různých zařízení. To ovšem vedlo ke špatné kompatibilitě zařízení od různých výrobců, nebo dokonce i různých sérii

zařízení od stejného výrobce. Dalšími značnými nedostatky Modbusu byla nemožnost přiřadit k odeslaným datum důvod přenosu, indikátor kvality dat a časovou značku.

Protokoly DNP3 a IEC 60870-5 mají v zásadě stejnou funkcionalitu. To je způsobeno tím, že oba vznikaly pro použití v sítích pro distribuci elektřiny a hlavně tím, že DNP3 je založen na brzkém návrhu protokolu IEC 60870-5. Nicméně i mezi těmito dvěma protokoly lze nalézt rozdíly. IEC 60870 používá adresu pro spojení a adresu pro data, oproti tomu DNP3 používá jen adresu spojení. To spolu s možností různé délky adres dává IEC 60870-5 větší pružnost v adresovacím sytému. Oba protokoly umožnují peer-peer komunikaci, ovšem u IEC 60870-5 je omezena pouze na point-to-point. DNP3 používá proměnou délku datových rámců, zatímco IEC 60870-5 používá fixní i proměnou délku. Pokud je u IEC 60870-5 použita fixní délka datového rámce jsou v porovnání s DNP3 vytvářeny jednoduché a krátké zprávy. DNP3 dovoluje kontrolu nad více body v jedné zprávě. Oproti tomu IEC 60870-5 pouze nad jedním. IEC 60870-5 kombinuje data a příkazy v kódování typu, DNP3 má rozdílné kódy pro data a funkce. DNP3 umožňuje poslat v jedné zprávě více datových typů, IEC 60870-5 pouze jeden typ dat ve zprávě.[11]

IEC 61850 je často označován jako vyspělejší komunikační protokol díky objektově orientovaným možnostem modelování dat. Starší protokoly, z důvodu limitací nastavených infrastrukturou, často řešily co největší minimalizaci objemu přenášených dat. IEC 61850 vytvářený v době, kdy již tyto limitace prakticky vymizely, je zaměřen spíše na to, jak data správně vytvářet a organizovat. Navíc se nejedná o přenosový protokol v pravém slova smyslu. Sám totiž nedefinuje, jak data přenášet po komunikačních sítích, ale definuje, jak data vytvořená pomocí něj přenést jinými protokoly (MMS, IEC 60870-5-104 atd.). IEC 61850 tak poskytuje některé značné výhody oproti starším protokolům. Místo adresování dat pomocí indexů používá hierarchii jmen. Každý informační prvek má své jméno, které ho zároveň popisuje. Je lépe rozšířitelný a poskytuje lepší možnosti uživatelského nastavení. V neposlední řadě také poskytuje úplný popis zařízení v XML formátu (ve vývoji pro DNP3).[12][13]

2.3.5 DLMS/COSEM

Z komunikačních protokolů také stojí za zmínku protokol DLMS/COSEM. DLMS (Device Language Message Specification) je sada standardů vyvinutá

a spravovaná DLMS User Association. COSEM (Companion Specification for Energy Metering) definuje transportní a aplikační vrstvu DLMS. Tento protokol definuje rozhraní pro přístup k datům na měřících zařízeních, a je zejména vytvořen pro měřící zařízení v energetickém průmyslu. Lze ho použít i pro jiná např. teplotní čidla. DLMS/COSEM využívá konceptů OSI modelu pro výměnu dat mezi měřidly a sběrnou stanicí. Základními charakteristikami tohoto protokolu jsou:

- K měřícím zařízením může být přistupováno z více stran (klientů a třetích stran).
- Zajištění efektivity: kompresí dat a možností výběru, ke kterým datům, se bude přistupovat.
- Pro více měřidel na stejném místě je možné zřídit jeden přístupový bod.
- Možnost použít různá komunikační media (lokální sítě, WAN sítě).
- Bezpečnost připojení je zajištěna použitím kryptografické ochrany pro xDLMS zprávy a COSEM data.

Komunikace mezi měřidlem a stanicí pro sběr dat probíhá na principu klient/server. Roli klienta zastává stanice pro sběr dat, roli serveru pak měřidlo. Klient posílá požadavky na data a server na něj odpovídá, případně je možné, aby server sám začal komunikaci za účelem informování klienta o nastalých událostech nebo zaslání dat za určitých, předem definovaných podmínek.[14]

Od protokolů zmíněných výše se tedy DLMS/COSEM liší především tím, že se jedná o protokol čistě pro sběr dat a neřeší ovládání vzdálených stanic. V praxi je pak používán spolu s protokolem IEC 60870-5-104 k přístupu k datům měřičů napětí.

2.4 Použité existující implementace IEC 60870

Pro lepší pochopení a vyzkoušení vlastností protokolu, následné testování a porovnání vlastních výsledků bylo využito několika existujících implementací protokolu.

Přístroje RTU od firmy ELVAC a. s. Jedná se o přístroje určené ke vzdálenému řízení, měření a sběr dat v energetických sítích. Tato zařízení poskytují širokou škálu komunikačních rozhraní, na kterých implementují různé komunikační protokoly včetně pro tuto práci důležitého IEC 60780-5-104.[15]

IEC Server je volně dostupná aplikace napsaná v jazyce Java, která simuluje serverovou stranu komunikace pomocí protokolu IEC 60780. Tato aplikace umožňuje

simulovat posílání různých typu hodnot. Také simuluje reakci na některé příkazy od řídící stanice definované podle protokolu.[16]

QTester104 je volně dostupná aplikace napsaná v jazyce C++ za pomoci frameworku QT, která umožňuje testování protokolu IEC 60780-5-104. Tato aplikace může být použita k navázání spojení se zařízením, příjmem dat z tohoto zařízení a posílání některých příkazu zařízení.[17]

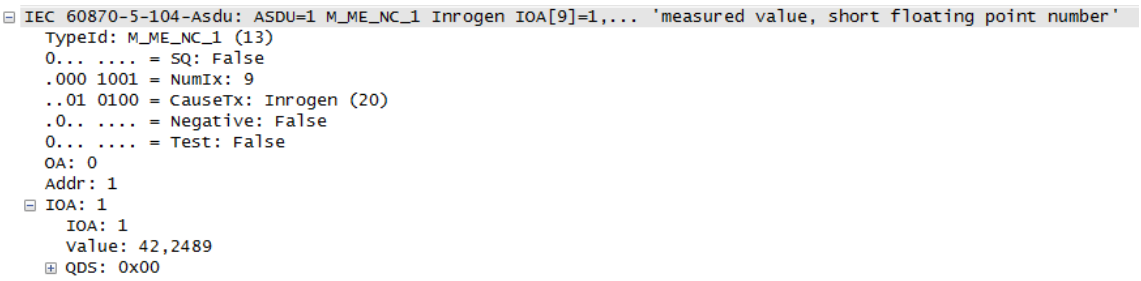
IEC Master je aplikace od společnosti ELVAC, jejíž demoverzi jsem měl k dispozici. Tato aplikace umožnuje navázání spojení se zařízením, příjem dat a řízení zařízení pomocí příkazů.

Wireshark je aplikace umožňující odchytávání a analýzu paketů na síťových spojeních. Používá se k monitorování a testování počítačových sítí a také k testování různých komunikačních protokolu. Tato aplikace umožňuje také filtrování komunikace pomocí protokolu IEC 60780-5-104. Dále je také schopna jednoduše zobrazit obsah a strukturu dat přenášených pomocí tohoto protokolu.[18]

Knihovna j60870 od organizace OpenMUC je knihovna pro programovací jazyk Java implementující funkce a metody potřebné pro vytváření aplikací, které komunikují protokolem IEC 60780-5-4. Tato knihovna byla použita k navržení a implementaci mé vlastní aplikace pro komunikaci protokolem IEC 60870-5-104. [19]

3 Vlastní řešení

Před realizací mé vlastní aplikace jsem se nejprve lépe seznámil s praktickou funkčností protokolu IEC 60870 a to zejména se strukturou přenášených dat. K tomu jsem využil jednotek RTU od firmy ELVAC, které jsem měl nastavené na cyklické zasílání měřených hodnot. Tyto jednotky byly také schopné reagovat na řídící příkazy pro přečtení dat. V případě, že jsem neměl k dispozici jednotkou RTU, využíval jsem aplikaci IEC Server. Jedná se o volně dostupnou aplikaci v jazyce Java, která simuluje některé základní funkce serveru IEC 60870-5-104. Jako klientské aplikace pro serverovou stranu poskytnutou RTU jednotkami jsem využil aplikace IEC Master a QTester104. IEC Master, k jehož trial verzi jsem měl přístup, je klientská aplikace protokolu IEC 60780 přímo od firmy ELVAC. QTester104 je pak volně dostupná implementace klientské aplikace v programovacím jazyce C/C++ s využitím knihovny Qt pro grafické rozhraní. Dále jsem při tomto testování použil program Wireshark, který umožňuje nejen sledovat provoz na síťové kartě, ale také filtrovat pouze komunikace pomocí protokolu IEC 60780. Ve Wiresharku jsem tedy mohl přímo vidět, jak přesně komunikace mezi klientem a serverem v protokolu IEC 60870 probíhá a také zde zobrazit strukturu posílaných APCI a ASDU po jednotlivých bitech, i s významem konkrétních bitů, jak je definováno v protokolu. Díky tomu jsem celou komunikaci a strukturu daných dat pochopil lépe, než pouze po seznámení se přímo s normou.



Obrázek 6: Ukázka zobrazení ASDU v programu wireshark.

3.1 Vlastní aplikace

K implementaci vlastních aplikací jsem použil knihovnu j60870 od organizace OpenMUC pro programovací jazyk Java. Jedná se o volně dostupnou knihovnu pod licencí GPL. Při své práci jsem využíval verzi 0.9 této knihovny, ovšem v průběhu práce

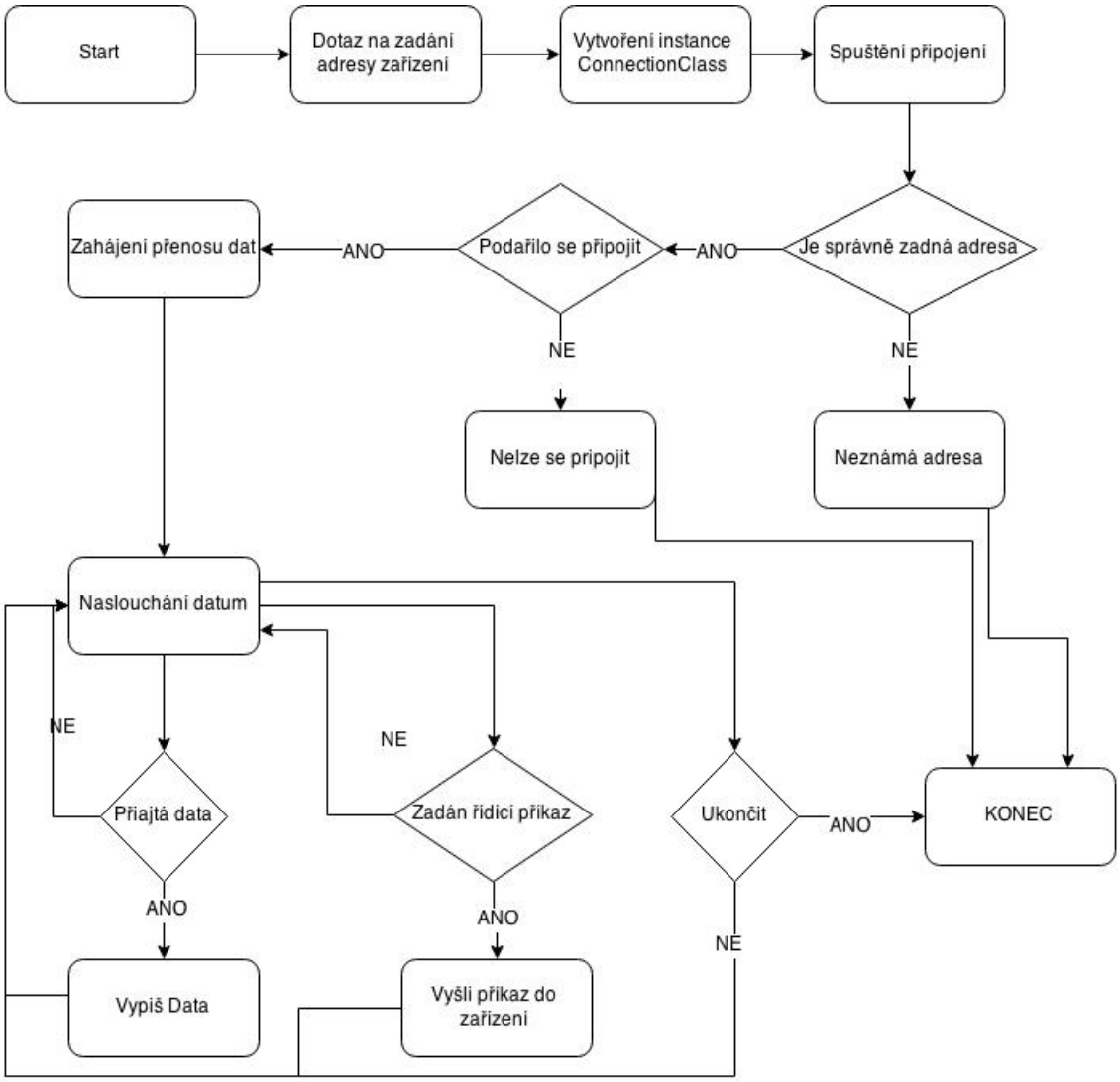
(24. února 2016) byla vydána již verze 1.0, která opravuje 3 chyby v knihovně, na které jsem při své prací nenarazil. Tato knihovna implementuje všechny funkce potřebné k navázání spojení mezi klientem a serverem. A také funkce pro posílání ASDU přenášející data a ASDU pro přenos řídících příkazů.

3.1.1 Klientská aplikace

Mojí klientskou aplikaci tvoří čtyři třídy: třída CleintMain, která řídí celou apliakci, třída ConnectionClass, která vytváří spojení se serverem a umožňuje kontrolu tohoto spojení, třída ConnectionListener, která se stará o naslouchání příchozím ASDU, a třída Graph, která umožňuje sledovat vybranou příchozí hodnotu v grafu.

Program po spuštění čeká na zadání příkazů. Prvním příkazem, který je potřeba zadat je příkaz pro připojení, který jako argumenty vyžaduje adresu serveru a port, na kterém server naslouchá (standardně se jedná o port 2404). Po zadání toho příkazu dojde k ověření správnosti adresy a případně překladu adresy za pomocí funkcí knihovny InetAddress. V případě, že je zadána neplatná adresa, dojde k vyvolání výjimky UnknownHostException. Pokud ověření adresy proběhlo v pořádku, dojde k vytvoření instance třídy ClientSap z knihovny j60870, která slouží k nastavení všech parametrů potřebných k připojení se k IEC 60870-5-104 serveru. Poté již dojde k pokusu o navázání spojení se serverem. K tomu je využívá funkce connnect třídy ClientSap, která v případě úspěchu vrací aktivní spojení. V případě neúspěšného pokusu o připojení je vyvolána výjimka informující uživatele o nastalé situaci. Po úspěšném připojení je po vytvořeném spojení vyslán příkaz pro začátek přenosu dat. Zároveň je přiřazena instance třídy ConnectionListener, která pak naslouchá příchozím ASDU na daném spojení. Po aktivním spojení je pak možné vysílat další řídící příkazy jako je například příkaz pro čtení určité hodnoty, nebo nastavení řídící proměnné. Ve své aplikaci jsem pro testování implementoval řídící příkazy:

- Dotazový povel, který z řízené stanice získá veškerá data.
- Příkaz čtení, který z řízené stanice získá jednu hodnotu, určenou adresou informačního objektu dané hodnoty v systému.
- Jednoduchý povel, který slouží ke změně stavu řízené jednotky.
- Povel pro časovou synchronizaci.
- Zkušební povel sloužící k otestování úplnosti smyčky mezi řídící a řízenou stanicí.
- Příkaz pro nastavení parametru a aktivaci parametru.



Obrázek 7: Diagram naznačují průběh mého vlastního programu.

Třída ConnectionListener implementuje rozhraní ConnectionEventListener, které je nutné implementovat tam, kde chceme naslouchat příchozím ASDU. Pro účely naslouchání je zde definovaná funkce newASDU, která nově příchozí ASDU vypíše na obrazovku a uloží přijatou hodnotu, adresu informačního objektu a čas, kdy k přijetí došlo do souboru. V případě, že chce uživatel sledovat jednu konkrétní hodnotu v grafu, spustí graf příkaz v hlavním programu a jako parametr tomuto příkazu zadá adresu informačního objektu dané hodnoty. Dojde k vytvoření instance třídy Graph, která se o vykreslení grafu stará a hodnoty jsou jí zasílány z funkce newASDU.

Protokol IEC 60870-5-104 definuje několik základních aplikačních funkcí pro sběr dat ze vzdálených stanic a jejich řízení. Jedná se tyto funkce:

- Inicializace stanice: slouží k uvedení stanice do správného provozního stavu před zahájením komunikace.
- Sběr dat na výzvu: slouží k získání dat ze vzdálené stanice vysláním příkazu pro čtení.
- Cyklický přenos dat: umožňuje řízené stanici cyklicky odesílat data, aniž by byla dotázáná řídící stanicí.
- Zachycení událostí: informuje řídící stanici o důležitých a neočekávaných událostech. Jedná se o funkci s vysokou prioritou.
- Celkový dotaz: jedná se o příkaz vyslaný řídící stanicí zejména po ztrátě spojení nebo inicializaci stanice.
- Časová synchronizace: příkaz vyslaný řídící stanicí k synchronizaci časových veličin. Tuto funkci lze použít pouze, pokud je maximální zpoždění v síti menší než požadovaná přesnost.
- Přenos povelů: s pomocí této funkce může řídící stanice ovládat řízenou stanici.
- Přenos celkových součtů: slouží k dálkovému odečtu veličin integrovaných podle určeného parametru, např: času.
- Zavádění parametrů: umožňuje řídící stanici měnit některé parametry řízené stanice, např.: mezní hodnoty měřených veličin.
- Zkušební procedura: slouží k otestování úplnosti smyčky z řídící do řízené stanice a zase zpět.
- Přenos souborů: slouží k přenosu informačních veličin, které svým rozsahem přesáhnou maximální možnou délku ASDU (249 bytů). Takové veličiny jsou pak přenášeny po segmentech.[3][20]

Z těchto základních aplikačních funkcí ve své klientské aplikaci neimplementuji pouze funkci pro přenos souborů. Jinak moje aplikace obsahuje vždy alespoň část metod pro každou z těchto funkcí např.: norma definuje 3 různé typy pro vyslání příkazu nastavení parametru. Já ve své aplikaci pro testování implementuji pouze jeden z těchto typů. Na obr. 9 lze vidět postup připojení, příjem dat a vyslání příkazů z mé aplikace.

```
"C:\Program Files (x86)\Java\jdk1.7.0_55\bin\java" ...
For list of available commands use "help"
Connect 127.0.0.1 2404

Connected!

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 1
Short float value: 233.1
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Dotaz
Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: PERIODIC, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 1
Short float value: 233.1
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false
IOA: 2
Short float value: 55.5
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Recieved ASDU:
Type ID: 100, C_IC_NA_1, Interrogation command
Cause of transmission: ACTIVATION_TERMINATION, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 0
Qualifier of interrogation: 0

Recieved ASDU:
Type ID: 13, M_ME_NC_1, Measured value, short floating point number
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 1
Short float value: 244.2
Quality, overflow: false, blocked: false, substituted: false, not topical: false, invalid: false

Recieved ASDU:
Type ID: 37, M_IT_TB_1, Integrated totals with time tag CP56Time2a
Cause of transmission: SPONTANEOUS, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 3
Binary counter reading: 15, seq num: 0, carry: false, counter adjusted: false, invalid: false
Time56: Sun May 10 11:37:00 CET 16
```

Obrázek 8: Ukázka běhu mého vlastního programu. Zobrazuje oznámení o připojení, přijetí spontánně odeslaných dat a reakci na příkaz „Celkový dotaz“, další přijetí spontánně odeslaných dat.

3.1.2 Serverová aplikace

Následně jsem pro testování přistoupil k velmi jednoduché aplikaci, která simuluje serverovou stranu komunikace protokolem IEC 60870-5-104. Moje serverová aplikace pro spuštění požaduje jeden parametr, a sice číslo portu, na kterém má naslouchat příchozím spojením. Pokud dojde k navázání spojení z klientské aplikace, čeká server na výzvu ke startu přenosu dat. Po obdržení této výzvy začne cyklicky přenášet jednu hodnotu a naslouchá příchozím ASDU z klientské aplikace. V případě, že server obdrží ASDU z klientské aplikace, vypíše ho na obrazovku. To mi posloužilo k ověření, zda řídící ASDU vyslaná pomocí funkcí knihovny j60870 odpovídají normě IEC 60870-5-101. Dále pokud server příjme příkaz pro celkový dotaz, odpoví na něj vysláním ASDU obsahujícím náhodně definované hodnoty.

```
A client has connected using TCP/IP. Will listen for a StartDT request.
Got interrogation command:

Recived asdu:
Type ID: 100, C_IC_NA_1, Interrogation command
Cause of transmission: ACTIVATION, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 0
Qualifier of interrogation: 0



Recived asdu:
Type ID: 103, C_CS_NA_1, Clock synchronization command
Cause of transmission: ACTIVATION, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 0
Time56: 15-05-16 21:26:44:148



Recived asdu:
Type ID: 102, C_RD_NA_1, Read command
Cause of transmission: REQUEST, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 5



Recived asdu:
Type ID: 45, C_SC_NA_1, Single command
Cause of transmission: ACTIVATION, test: false, negative con: false
Originator address: 0, Common address: 1
IOA: 4
Single Command state on: true, selected: false, qualifier: 0
```

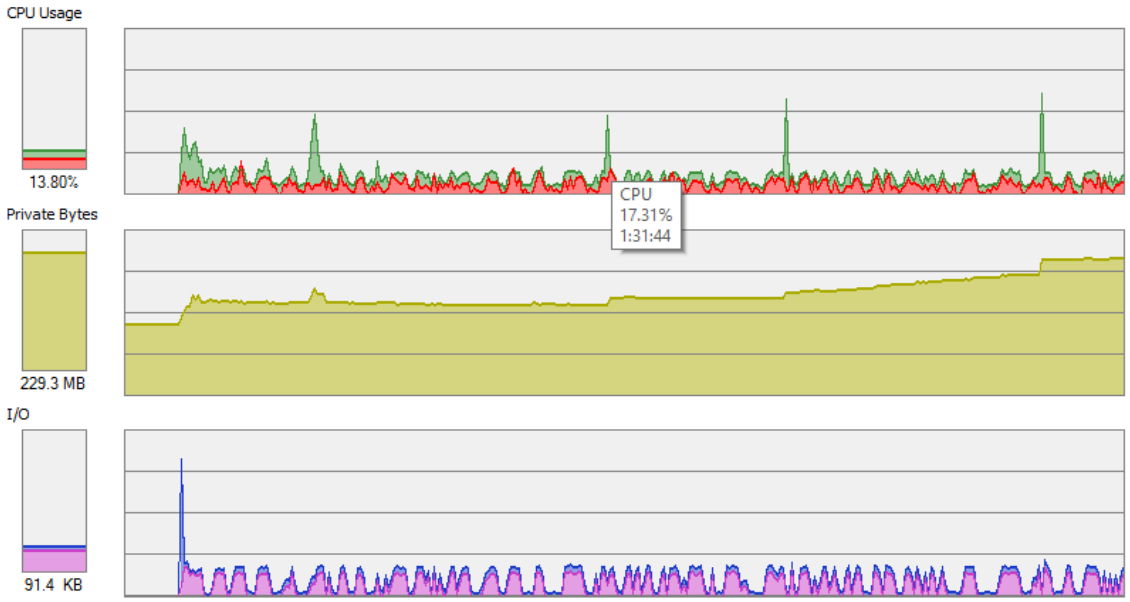
Obrázek 9: Ukázka příjmu řídích příkazů v serverové aplikaci. Lze vidět přijetí Celkového dotazu, příkazu pro synchronizaci času, příkazu pro čtení hodnoty a jednoduchého povelu.

3.2 Experiment

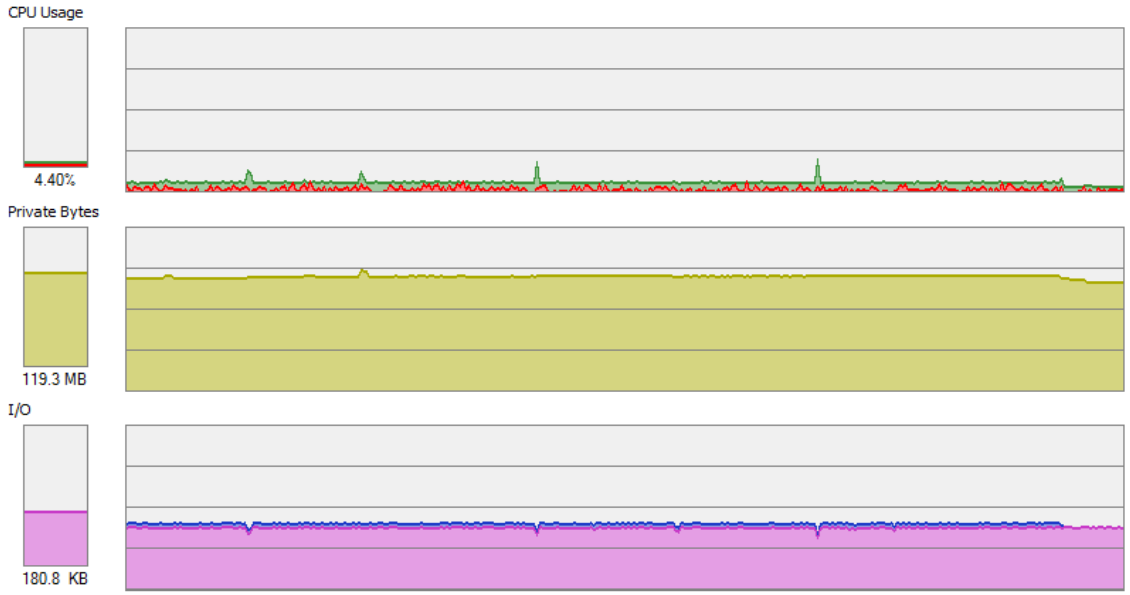
Po vytvoření vlastní klientské a serverové aplikace a otestování funkcí knihovny j60870 jsem se na návrh vedoucího práce rozhodl otestovat schopnosti mé klientské aplikace v podmínkách, které by více odpovídaly reálnému nasazení. Do této doby jsem ze své klientské aplikace navazoval vždy pouze jedno spojení s jedním serverem. V reálné situaci jsou ovšem z jednoho řídícího střediska sbírána data i z deseti tisíců stanic. Cílem tohoto pokusu tedy bylo zjistit, jaký objem příchozích ASDU bude moje aplikace ještě schopna obsloužit a kdy již přestane zvládat. Nejprve jsem tedy upravil serverovou aplikaci tak, aby jako svůj argument vyžadovala číslo portu, na který se má připojit. Standardně je pro tyto účely používán port 2404. Vzhledem k tomu, že jsem při svém testu potřeboval spustit více serverových aplikací na jednom počítači, bylo nutné, aby aplikace využívala i jiné porty. Jako druhý parametr aplikace používala číslo (reprezentují čas milisekundách), které určovalo, jak často bude server cyklicky odesílat data. Poté jsem ze serverové aplikace vytvořil spustitelný soubor JAR. Dále jsem vytvořil dávkový soubor, který mi daný JAR soubor umožnil spustit vícekrát jedním kliknutím na mnou předem určených portech. Následně jsem upravil i klientskou aplikaci tak, aby pro argumenty příkazu navázání spojení kromě adresy a portu vyžadovala také počet spojení, která se mají vytvořit. Vzhledem k tomu, že na počítači kde jsem spouštěl servery, byl dostatek nevyužitých portů jdoucích po sobě, mohl jsem servery vytvářet cyklicky na portech jsoucích po sobě. Stejně tak v cyklu vytvářet spojení na po sobě jdoucí porty. Pro testování jsem požil dvě verze své klientské aplikace, lišící se v nakládání s příchozími ASDU. První byla bez úprav, čili příchozí ASDU vypsala na standartní výstup a přijatou hodnotu ještě uložila do souboru. Druhá pak pouze vypisovala přijatá ASDU na standartní výstup.

Nejprve jsem potřeboval zjistit, jak vůbec poznat, že moje klientská aplikace již nezvládá zpracovávat příchozí ASDU. Jednou z funkcí knihovny je i zobrazení počtu nepotvrzených ASDU pro daná přepojení, ovšem vzhledem k tomu, že příjem ASDU je zde navržen tak, aby nedocházelo ke ztrátě dat, byla všechna ASDU potvrzována i v případě, že byl příjem přetížen. Ovšem samo ukládání nezpracovaných ASDU napomohlo vyřešení toho problému, protože v případě kdy aplikace nestíhala, rostly postupně s časem běhu její nároky na paměť, což jsem mohl snadno sledovat například pomocí programu Process Explorer [21]. Zároveň toto logovaní způsobilo, že pokud moje klientská aplikace nezvládala zpracovávat příchozí ASDU, pokračovala v tom i po

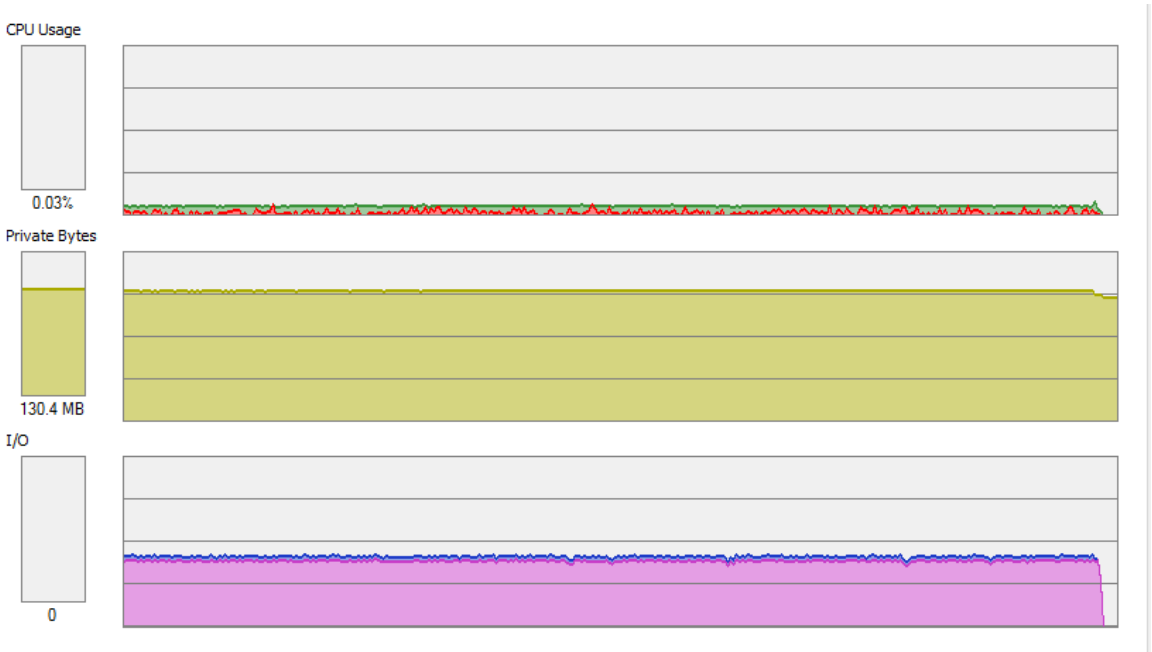
skončení přenosu dat ze serveru. Tedy dále vypisovala data, i když ze serverů už žádná nepřicházela. V případě, že zpracování zvládala, ukončilo se vypisování dat zároveň s ukončením serverů, což bylo možné sledovat v grafu I/O v programu Process Explorer.



Obrázek 10: Ukázka nárůstu nároku na paměť v případě, že klient nezvládá zpracovat příchozí ASDU.



Obrázek 11: Ukázka po ukončení vysílání dat ze serverů v případě, že klient nezvládal zpracovávat ASDU. Je zde možné vidět, že vstup již skončil, ale výstup dále pokračuje.



Obrázek 12: Ukázka po ukončení vysílání dat ze serverů v případě, že klient zvládat zpracovat ASDU. Je zde vidět, že výstup končí zároveň se vstupem.

Následným testováním jsem zjistil, že varianta s logováním hodnot do souboru je podle očekávání značně neefektivní a nezvládá zpracovat ani 200 příchozích ASDU za sekundu. Rozhodl jsem se tedy limit této varianty blíže nehledat. Pro variantu s pouhým vypisováním dat, jsem zjistil, že 620 ASDU za vteřinu ještě bezpečně zvládá. V případě 630 příchozích ASDU za sekundu již některá ASDU ukládala pro pozdější zpracování. Pro toto přesnější určování se ukázal lepší graf I/O programu Process Explorer, neboť nároky na paměť rostly jen velmi pomalu, ale na grafu I/O byl konec výstupu zároveň se vstupem jasně viditelný.

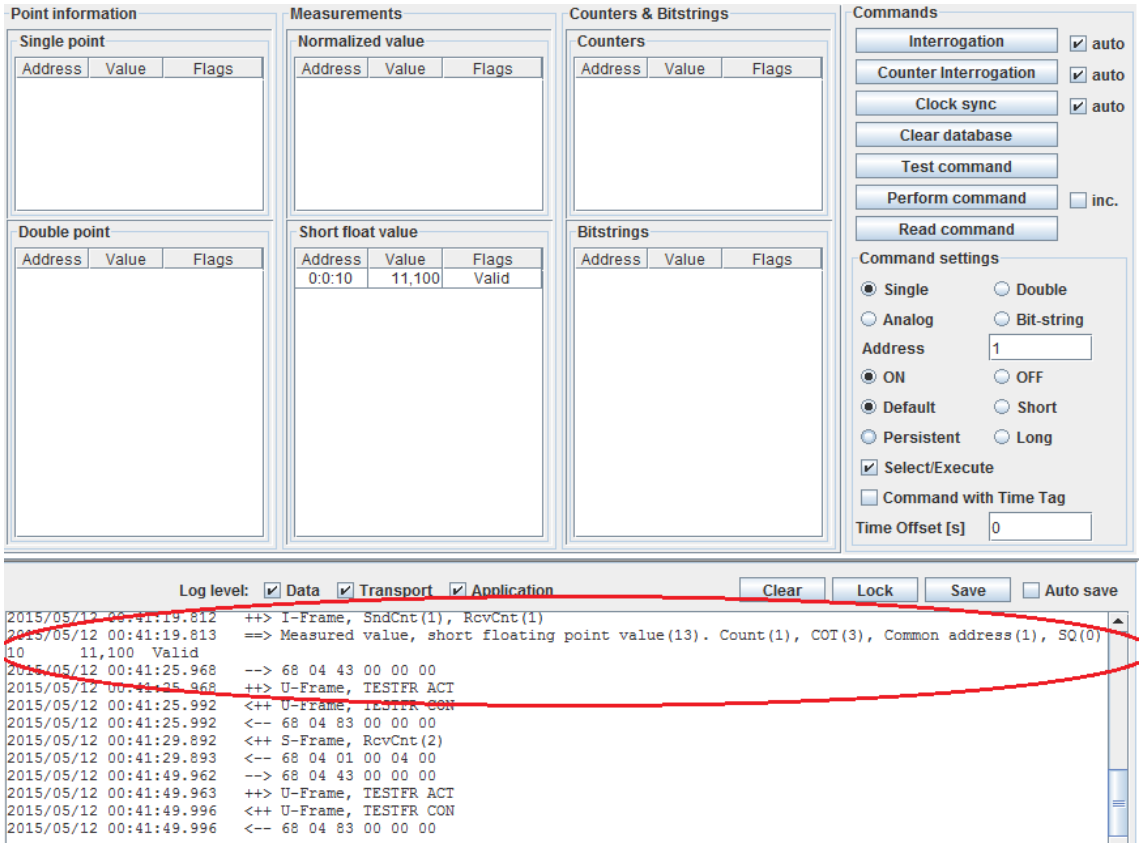
4 Zhodnocení výsledků

Podařilo se mi tedy vytvořit vlastní klientskou a jednoduchou serverovou aplikaci. Klientská aplikace je schopna navázat spojení se serverovou aplikací, zobrazit přijatá ASDU, uložit přijaté hodnoty do souboru a umožňuje sledování jedné vybrané hodnoty v grafu. Serverová aplikace je značně jednoduchá. Umožňuje navázat spojení s klientem, cyklicky odesílá jednu předdefinovanou hodnotu a reaguje na dotazový povel.

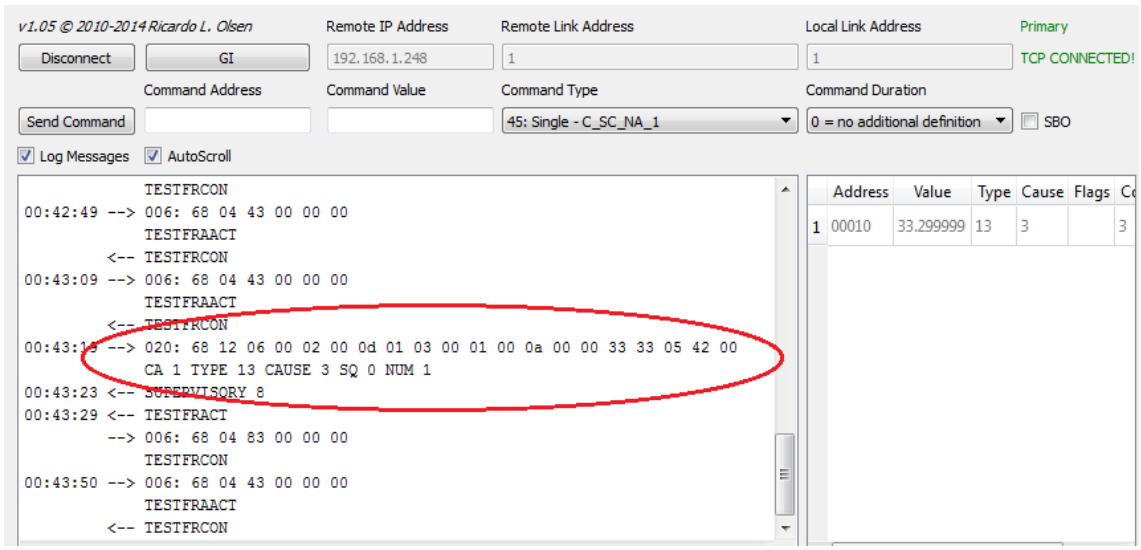
Co se porovnání mé klientské aplikace s aplikacemi QTester104 a IEC Master týče, je má aplikace značně jednoduší, protože funguje pouze jako „command line" aplikace, zatímco IEC Master a QTester104 mají grafické rozhraní. Grafické rozhraní je vhodnější zejména pro ovládání celé aplikace, kdy lze vyslat příkaz stiskem tlačítka a není potřeba psát příkaz. Zobrazení a sledování jednotlivých ASDU a hodnot v nich obsažených je v aplikacích QTester104 a IEC Master poměrně jasné a přehledné, zejména díky grafickému rozhraní. Ovšem moje aplikace v tomto ohledu, dle mého názoru, nijak výrazně nezaostává. Sice vypisuje jednotlivá ASDU pod sebe, ale vzhledem k tomu, že jsou všechny informace obsažené v ASDU jasně vypsány na několika řádcích, není jejich sledování obtížné. Obě zmíněné aplikace také hodnoty obsažené v ASDU aktualizují v zobrazení grafického rozhraní. V mojí aplikaci lze tedy lépe sledovat postupný vývoj přijatých hodnot. Některé z parametrů obsažených v ASDU dokonce nejsou v aplikacích QTester104 a IEC Master zobrazeny vůbec, a to zejména v případě aplikace QTester104. Jedná se například o parametry indikující zkoušku, kladné a záporné potvrzení, případně kvalitativní deskriptor. Celé ASDU pak tyto aplikace zobrazí pouze jako číselnou reprezentaci přijatých bytům. Oproti tomu moje aplikace zobrazí i význam přijatých dat. Také umožňuje navázat spojení s více servery najednou na rozdíl od aplikací QTester104 a IEC Master, a také ukládání přijatých dat do souboru a zobrazení vybrané hodnoty v grafu.

Moje serverová aplikace pak ve srovnání s aplikaci IECServer značně zaostává. Možnosti nastavení a vysílání různých dat jsou v aplikaci IECServer daleko lepší. Moje aplikace má naprosto minimální možnosti nastavení, jen port a rychlost odesílání testovací hodnoty. Nicméně pro základní otestování navázání spojení a příjmu cyklicky přenášených dat v klientovi poslouží. Také mi umožnila lépe vidět strukturu ASDU ve směru řízení.

V zátěžovém testu pak verze mé aplikace bez logování do souboru nedopadla úplně špatně, protože zvládla zpracovat relativně značné množství příchozích ASDU, konkrétně 620 za sekundu.



Obrázek 13: ukázka příjmu ASDU v programu IEC Master.



Obrázek 14: Ukázka příjmu ASDU v programu QTester104.

5 Závěr

V práci jsem se seznámil se základní podobou datových rámců vysílaných pomocí protokolu IEC 60780-5-104 a blíže pak definovaných v IEC 60870-5-101. Dále jsem se seznámil s tím co jednotlivé části těchto dat znamenají a k čemu daná informace slouží. Hlouběji jsem pak strukturu komunikace prozkoumal v praxi, pro tento účel se jako výborný pomocník ukázal program Wireshark. Také jsem se seznámil s některými jinými protokoly pro vzdálené řízení a sběr dat a porovnal je s IEC 60870-5-104.

K účelu získání dat správně naformátovaných podle protokolu IEC 60870-5-104 mi posloužily přístroje RTU od firmy ELVAC a. s. Tyto přístroje jsou po správném nastavení schopny poskytovat naměřená data v potřebném formátu.

Následně jsem se po seznámení s existujícími implementacemi protokolu rozhodl pro vlastní řešení zvolit knihovnu pro jazyk Java j60870 od organizace OpenMUC. Ta mi umožnila realizovat vlastní jednoduché řešení klientské aplikace. Následně jsem vytvořil vlastní jednoduchou serverovou aplikaci, která mi pomohla blíže se seznámit se strukturou příkazových ASDU. Poté jsem provedl experiment s cílem určit jaký objem příchozích ASDU je moje klientská aplikace schopna zvládnout a vyhodnotil výsledky tohoto experimentu.

Poté jsem řešení, která jsem vytvořil, stručně porovnal s některými již existujícími aplikacemi. I když je moje klientská i serverová aplikace značně jednoduší než řešení, se kterými jsem srovnával, pro základní testování příjmu dat a odeslání příkazů poslouží. V případě mé serverové aplikace jsem oproti aplikaci IECServer získal možnost podívat se na strukturu ASDU ve směru řízení v čitelnější podobě než je pouhý řetězec čísel.

Možnosti budoucího pokračování práce jsou poměrně rozsáhle. V první řadě by se jistě jednalo o vylepšení mé klientské aplikace, aby byla lépe ovladatelná, umožňovala lepší a přehlednější zobrazení přijatých dat a lepší možnosti odesílání příkazů. V neposlední řadě by jistě bylo dobré pokusit se zefektivnit její možnosti, co se objemu přijatých ASDU týče. Dalším krokem by mohlo být vytvoření dokonalejší aplikace pro serverovou stranu komunikace pomocí protokolu IEC 60870-5-104.

Použitá literatura

[1] *ČSN EN 6087-5-104. Systémy a zařízení pro dálkové ovládání - ČÁST 5-104: Přenosové protokoly - Síťový přístup pro IEC 60870-5-101 používající normalizované transportní profily*. 2007. Praha: Český normalizační institut.

[2] *ČSN EN 60870-5-101. Systémy a zařízení pro dálkové ovládání - Část 5-101: Přenosové protokoly - Společná norma pro základní úkoly dálkového řízení*. 2005. Praha: Český normalizační institut.

[3] *ČSN EN 60870-5-5. Systémy a zařízení pro dálkové ovládání - Část 5: Přenosové protokoly - Oddíl 5: Základní aplikační funkce*. 1998. Praha: Český normalizační institut.

[4] DNP3 Overview [online]. 2002 [cit. 2016-05-15]. Dostupné z: http://www.chipkin.com/articles/wp-content/uploads/2013/06/DNP3_Overview.pdf

[5] *A DNP3 Protocol Primer* [online]. 2005 [cit. 2016-05-15]. Dostupné z: http://www.dnp.org/AboutUs/DNP3%20Primer%20Rev%20A.pdf

[6] *DNP3: User and Reference Manual* [online]. 2007 [cit. 2016-05-15]. Dostupné z: https://scadahacker.com/library/Documents/ICS_Protocols/Control%20Microsystems%20-%20DNP3%20User%20and%20Reference%20Manual.pdf

[7] OZANSOY, Cagil R., Aladin ZAYEGH a Akthar KALAM. *The Application-View Model of the International Standard IEC 61850* [online]. 2009 [cit. 2016-05-15]. Dostupné z: http://ieeexplore.ieee.org/

[8] MACKIEWICZ, R. E. *Overview of IEC 61850 and Benefits* [online]. 2006 [cit. 2016-05-15]. Dostupné z: http://ieeexplore.ieee.org/

[9] *Modbus Application Protocol Specification* [online]. 2006 [cit. 2016-05-15]. Dostupné z: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf

[10] *Using DNP3 & IEC 60870-5 Communication Protocols In the Oil & Gas Industry* [online]. 2001 [cit. 2016-05-15]. Dostupné z: https://scadahacker.com

[11] Diferences between DNP3 and IEC 60870. CLARKE, Gordon R., Deon REYNDERS a Edwin WRIGHT. *Practical modern SCADA protocols: DNP3, 60870.5 and related systems* [online]. Boston: Newnes, 2004, s. 307-310 [cit. 2016-05-15]. ISBN 0750657995. Dostupné z: https://books.google.cz/books?id=ENqyW8fExswC&dq=DNP3&lr=&hl=cs&source=gbs_navlinks_s

[12] SCHWARZ, Karlheinz. *Comparison of IEC 60870-5-1001/-103/-104, DNP3 and IEC 60870-6-TASE.2 with IEC 61850* [online]. 2002 [cit. 2016-05-15]. Dostupné z: www.nettedautomation.com

[13] MOHAGHEGHI, S., J. STOUPIS a Z. WANG. *Communication Protocols and Networks for Power Systems-Current Status and Future Trends* [online]. 2009 [cit. 2016-05-15]. Dostupné z: http://ieeexplore.ieee.org/

[14] *DLMS/COSEM Architecture and Prorocols* [online]. 2014 [cit. 2016-05-15]. Dostupné z: http://dlms.com/documents/Excerpt_GB8.pdf

[15] ELVAC A.S. Příručka uživatele RTU7.4, RTU7K a RTU7KL. Rev 6; 5/14. Ostrava, 2014.

[16] *IEC Server* [online]. 2013. [cit. 2016-05-11]. Dostupné z: https://sourceforge.net/projects/iecserver/

[17] OLSEN, Ricardo. *Qtester104: IEC 60870-5-104 protocol tester. Sourceforge.net* [online]. [cit. 2016-05-11]. Dostupné z: http://sourceforge.net/projects/qtester104/

[18] *Wireshark* [online]. [cit. 2016-05-11]. Dostupné z: https://www.wireshark.org/

[19] *J60870 API* [online]. 2014. [cit. 2016-05-11]. Dostupné z: http://www.openmuc.org/projects/j60870/javadoc/

[20] SANCHEZ, G., I. GOMEZ, J. JUQUE, J. BENJUMEA a O. RIVERA. IEEE. Using Internet Protocols to Implement IEC 60870-5 Telecontrol Functions. [online]. 2009. Dostupné z: http://ieeexplore.ieee.org/

[21] RUSSINOVICH, Mark. Procces Explorer v16.12 [online]. [cit. 2016-5-11] Dostupné z: https://technet.microsoft.com/en-us/sysinternals/processexplorer

Příloha A – Ukázka průběhu komunikace klient server v programu Wireshark

Source	Destination	Protocol	Length	Info
192.168.1.235	192.168.1.248	104apci	60	<- U (STARTDT act)
192.168.1.248	192.168.1.235	104apci	60	-> U (STARTDT con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (0,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104asdu	70	<- I (0,1) ASDU=1 C_IC_NA_1 Act IOA=0
192.168.1.248	192.168.1.235	104asdu	70	-> I (1,1) ASDU=1 C_IC_NA_1 Req_NEGA IOA=0
192.168.1.235	192.168.1.248	104apci	60	<- S (2)
192.168.1.248	192.168.1.235	104asdu	74	-> I (2,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (3)
192.168.1.248	192.168.1.235	104apci	60	-> S (1)
192.168.1.248	192.168.1.235	104asdu	74	-> I (3,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (4)
192.168.1.248	192.168.1.235	104asdu	74	-> I (4,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (5)
192.168.1.248	192.168.1.235	104asdu	74	-> I (5,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (6)
192.168.1.248	192.168.1.235	104asdu	74	-> I (6,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (7)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (7,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (8)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (8,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (9)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (9,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (10)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (10,1) ASDU=1 M_ME_NC_1 Spont IOA=1
192.168.1.235	192.168.1.248	104apci	60	<- S (11)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR act)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104apci	60	-> U (TESTFR act)
192.168.1.235	192.168.1.248	104apci	60	<- U (TESTFR con)
192.168.1.248	192.168.1.235	104asdu	74	-> I (11,1) ASDU=1 M_ME_NC_1 Spont IOA=1

Příloha B – Ukázka komunikace v programu Wireshark při vysílání řídících příkazů

Source	Destination	Protocol	Length	Info
192.168.1.235	147.230.77.98	104apci	60	<- U (STARTDT act)
147.230.77.98	192.168.1.235	104apci	60	-> U (STARTDT con)
147.230.77.98	192.168.1.235	104asdu	190	-> I (0,0) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (1,0) ASDU=1 M_SP_NA_1 Inrogen IOA[
192.168.1.235	147.230.77.98	104asdu	70	<- I (0,4) ASDU=1 C_CI_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	86	-> I (4,1) ASDU=1 C_CI_NA_1 ActCon IOA=0 | -> I (5,1) ASDU=1 C_CI_NA_1 ActTerm IOA=
192.168.1.235	147.230.77.98	104asdu	76	<- I (1,6) ASDU=1 C_CS_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	76	-> I (6,2) ASDU=1 C_CS_NA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104apci	60	<- S (7)
192.168.1.235	147.230.77.98	104asdu	70	<- I (2,7) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (7,3) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (8,3) ASDU=1 M_SP_NA_1 Inrogen IOA[
192.168.1.235	147.230.77.98	104asdu	69	<- I (3,11) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (4)
192.168.1.235	147.230.77.98	104asdu	78	<- I (4,11) ASDU=1 C_TS_TA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	78	-> I (11,5) ASDU=1 C_TS_TA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104asdu	69	<- I (5,12) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	70	<- I (6,12) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (12,7) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (13,7) ASDU=1 M_SP_NA_1 Inrogen IO
192.168.1.235	147.230.77.98	104apci	60	<- S (16)
192.168.1.235	147.230.77.98	104asdu	70	<- I (7,16) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (16,8) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (17,8) ASDU=1 M_SP_NA_1 Inrogen IO
192.168.1.235	147.230.77.98	104asdu	69	<- I (8,20) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	78	<- I (9,20) ASDU=1 C_TS_TA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	78	-> I (20,10) ASDU=1 C_TS_TA_1 ActCon IOA=0
192.168.1.235	147.230.77.98	104apci	60	<- S (21)
192.168.1.235	147.230.77.98	104asdu	70	<- I (10,21) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (21,11) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (22,11) ASDU=1 M_SP_NA_1 Inrogen
192.168.1.235	147.230.77.98	104asdu	69	<- I (11,25) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (12)
192.168.1.235	147.230.77.98	104asdu	69	<- I (12,25) ASDU=1 C_RD_NA_1 Req IOA=1
147.230.77.98	192.168.1.235	104apci	60	-> S (13)
192.168.1.235	147.230.77.98	104apci	60	<- U (TESTFR act)
147.230.77.98	192.168.1.235	104apci	60	-> U (TESTFR con)
192.168.1.235	147.230.77.98	104asdu	69	<- I (13,25) ASDU=1 C_RD_NA_1 Req IOA=1
192.168.1.235	147.230.77.98	104asdu	70	<- I (14,25) ASDU=1 C_IC_NA_1 Act IOA=0
147.230.77.98	192.168.1.235	104asdu	190	-> I (25,15) ASDU=1 C_IC_NA_1 ActCon IOA=0 | -> I (26,15) ASDU=1 M_SP_NA_1 Inrogen
192.168.1.235	147.230.77.98	104apci	60	<- S (29)

Příloha C – Ukázka rozboru ASDU v programu Wireshark

```
⊞ IEC 60870-5-104-Apci: -> I (1,0)
⊟ IEC 60870-5-104-Asdu: ASDU=1 M_SP_NA_1 Inrogen IOA[2]=2,... 'single-point information'
    TypeId: M_SP_NA_1 (1)
    0... .... = SQ: False
    .000 0010 = NumIx: 2
    ..01 0100 = CauseTx: Inrogen (20)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 1
  ⊞ IOA: 2
  ⊞ IOA: 4
⊟ IEC 60870-5-104-Apci: -> I (2,0)
    START
    ApduLen: 82
    .... ...0 = Type: I (0x00)
    Tx: 2
    Rx: 0
⊟ IEC 60870-5-104-Asdu: ASDU=1 M_ME_NC_1 Inrogen IOA[9]=1,... 'measured value, short floating point number'
    TypeId: M_ME_NC_1 (13)
    0... .... = SQ: False
    .000 1001 = NumIx: 9
    ..01 0100 = CauseTx: Inrogen (20)
    .0.. .... = Negative: False
    0... .... = Test: False
    OA: 0
    Addr: 1
  ⊟ IOA: 1
      IOA: 1
      Value: 42,2489
    ⊟ QDS: 0x00
        .... ...0 = OV: No overflow
        ...0 .... = BL: Not blocked
        ..0. .... = SB: Not Substituted
        .0.. .... = NT: Topical
        0... .... = IV: Valid

Transmission Control Protocol, Src Port: 2404 (2404), Dst Port: 54749 (54749), Seq: 82, Ack: 62, Len: 36
IEC 60870-5-104-Apci: -> I (3,2)
  START
  ApduLen: 18
  .... ...0 = Type: I (0x00)
  Tx: 3
  Rx: 2
IEC 60870-5-104-Asdu: ASDU=1 M_ME_NC_1 Inrogen IOA=1 'measured value, short floating point number'
  TypeId: M_ME_NC_1 (13)
  0... .... = SQ: False
  .000 0001 = NumIx: 1
  ..01 0100 = CauseTx: Inrogen (20)
  .0.. .... = Negative: False
  0... .... = Test: False
  OA: 0
  Addr: 1
⊟ IOA: 1
    IOA: 1
    Value: 22,2
  ⊟ QDS: 0x00
      .... ...0 = OV: No overflow
      ...0 .... = BL: Not blocked
      ..0. .... = SB: Not Substituted
      .0.. .... = NT: Topical
      0... .... = IV: Valid
IEC 60870-5-104-Apci: -> I (4,2)
  START
  ApduLen: 14
  .... ...0 = Type: I (0x00)
  Tx: 4
  Rx: 2
IEC 60870-5-104-Asdu: ASDU=1 C_IC_NA_1 ActTerm IOA=0 'interrogation command'
  TypeId: C_IC_NA_1 (100)
  0... .... = SQ: False
  .000 0001 = NumIx: 1
  ..00 1010 = CauseTx: ActTerm (10)
  .0.. .... = Negative: False
  0... .... = Test: False
  OA: 0
  Addr: 1
⊟ IOA: 0
    IOA: 0
    QOI: Station interrogation (global) (20)
```

Příloha D – Obsah přiloženého CD

- Text bakalářské práce
  - Bakalarska_prace_Michael_Sucharda_2016.pdf
  - Bakalarska_prace_Michael_Sucharda_2016.docx
- Klientská aplikace
  - ClientApp.jar – spustitelný Java soubor klienta
  - Client.bat – Dávkový soubor pro spuštění klientské aplikace
- Serverová aplikace
  - Server.jar – spustitelný Java soubor serveru
  - Server.bat – dávkový soubor pro spuštění serverové aplikace