VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Václav Šnajdr

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

GENERÁTOR DATOVÉHO PROVOZU PRO PRŮMYSLOVÉ PROTOKOLY
TRAFFIC GENERATOR OF INDUSTRIAL PROTOCOLS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Václav Šnajdr**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Zdeněk Martinásek, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: Václav Šnajdr ***ID:*** 195173
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Generátor datového provozu pro průmyslové protokoly

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce je návrh a implementace zásuvného modulu do nástroje J-meter. Modul bude generovat provoz dle průmyslových protokolů IEC61850, IEC60870-5 a DNP3. V teoretické části prostudujte nástroj Jmeter, implementaci zásuvného modulu a výše zmíněné průmyslové protokoly. V praktické části implementujte nejméně dva komplexní zátěžové moduly průmyslových protokolů, které otestujete na experimentálním pracovišti.

DOPORUČENÁ LITERATURA:

[1] HALILI, Emily H. Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites. Packt Publishing Ltd, 2008.

[2] QINCUI, Fu; JIANYUN, Chen. Building experimental platform for intelligent substation by using libiec 61850 and IEDScout. Experimental Technology and Management, 2017, 7: 035.

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Bakalářská práce se zabývá generováním datového provozu průmyslových SCADA protokolů a jejich implementaci do nástroje JMeter. Tento nástroj je možno rozšiřovat pomocí modulů. V teoretické části jsou popsány tři protokoly DNP3, IEC61850 a IEC60870-5. Praktická část je věnována návrhu a implementaci modulu pro DNP3 protokol a částečně popsán postup návrhu u IEC61850 protokolu. Modul DNP3 byl funkčně otestován. Také je tu zmíněn pokus o získání knihovny protokolu TASE.2.

KLÍČOVÁ SLOVA

SCADA, generátor, protokoly, JMeter, modul, DNP3, IEC61850, IEC60870-5, TASE.2

ABSTRACT

This bachelor thesis deals with generating data traffic of industrial SCADA protocols and their implementation into JMeter tool. This tool can be expanded with plugins. Three protocols DNP3, IEC61850 and IEC60870-5 are described in the theoretical part. The practical part is devoted to the design and implementation of the DNP3 protocol module and partly to the design of the IEC61850 protocol. The DNP3 module has been functionally tested. There is also an attempt to obtain the TASE.2 library.

KEYWORDS

SCADA, generator, protocols, JMeter, plugin, DNP3, IEC61850, IEC60870-5, TASE.2

ŠNAJDR, Václav. *Generátor datového provozu pro průmyslové protokoly*. Brno, 2019, 69 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Zdeněk Martinásek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Generátor datového provozu pro průmyslové protokoly“ 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 diplomové práce panu Ing. Zdeňku Martináskovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

podpis autora

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V průmyslovém odvětví, jako je energetika, výroba, technologie budov, ekologie a mnoho dalších, je potřeba sbírat data a dohlížet na správný průběh událostí. To řeší Supervisory Control And Data Acquisition (SCADA) systémy, které mohou komunikovat s okolím prostřednictvím standardizovaných komunikačních protokolů. Systémy nabízejí možnost vzdáleného přístupu a dohledu přes internet. Lze tedy monitorovat technologie na smartphonech nebo tabletech.

Nástroj JMeter umožňuje pomocí modulů sestavit testovací plán, který může reprezentovat jakoukoliv zátěž na zařízení a analyzovat různé komunikace. Jeho možnosti se využijí v této práci pro generování provozu na průmyslových protokolech IEC61850, IEC60870-5 a DNP3.

Cílem bakalářské práce je navrhnout a implementovat přádavný modul do nástroje JMeter, který bude generovat zátěž dvou vybraných protokolů DNP3 a IEC61850. Dílčím cílem je prostudovat výše zmíněné protokoly a nástroj JMeter. V průběhu řešení práce byl také brán v úvahu protokol TASE.2.

1 Průmyslové protokoly SCADA

SCADA (Supervisory Control And Data Acquisition) označuje širokou škálu vzdálených monitorovacích a řídících systémů, která sbírají data. Patří sem obecné řízení procesů, distribuce energie, železniční komunikace, sledování aktiv nebo vodovodní systémy. Systém je vždy tvořen řídící stanicí (Master) a podřízenou stanicí (Slave). Stroje schromažďují, vyměňují a zpracovávají informace. Data získávají ze snímačů a obvykle se přenáší přes Ethernet nebo IP. Prezentace dat je možná přes rozhraní mezi zařízením (systémem) a člověkem (obsluha) neboli HMI (Human-Machine Interface) [1]. Průmyslových protokolů je celá řada, ale tato práce se bude zabývat pouze protokoly DNP3, IEC61850 a IEC60870-5.

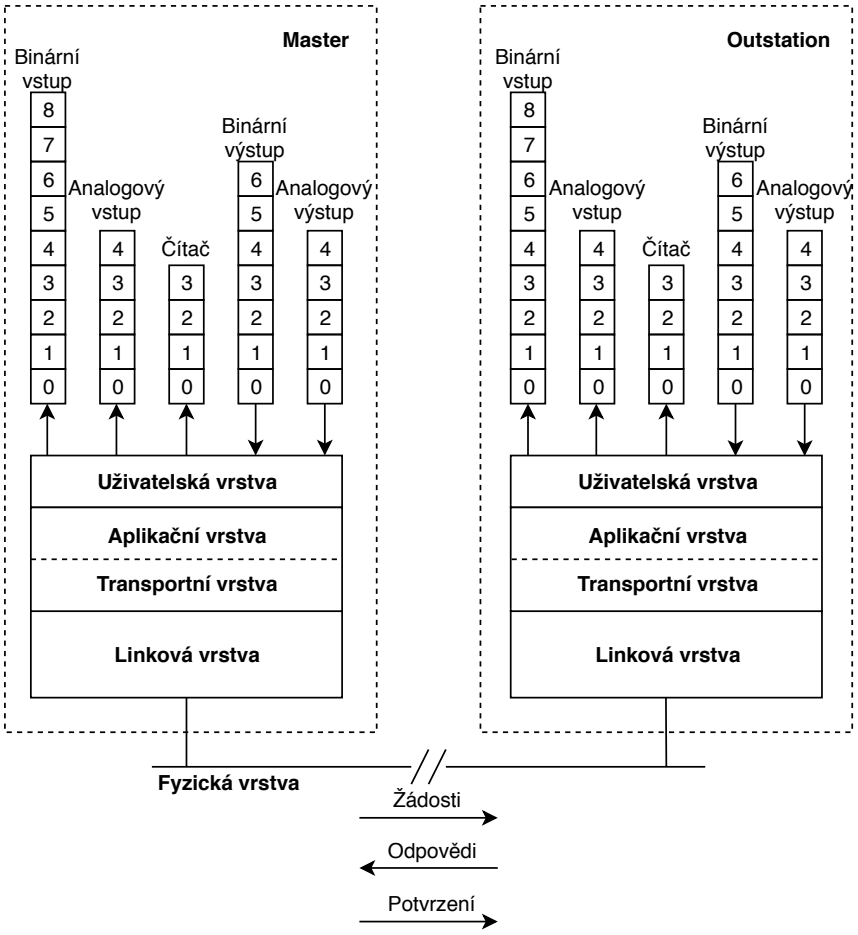
1.1 DNP3 protokol

DNP3 protokol (Distributed Network Protocol) je komunikační protokol používaný v SCADA a vzdálených monitorovacích systémech. Jedná se o otevřený a veřejný protokol, a proto si každý výrobce může vyvinout zařízení DNP3, které je kompatibilní s jinými zařízenímy, a díky tomu je široce používán. Obvykle se jedná o komunikaci mezi řídící hlavní stanicí a podřízenou stanicí RTU (Remote terminal unit), IED (Intelligent Electronic Devices) nebo koncetrátor, která slouží pouze pro sběr dat a kritických stavů a následné předání hlavní stanicí (Master). Vývoj protokolu byl důležitý vzhledem k časovému období včetně potřeby řešení, které by vyhovovalo dnešním požadavkům. Je rozšířen v průmyslových odvětví jako je voda, doprava a ropný a plynárenský průmysl.

V případě topologie s více podřízenými stanicemi probíhá komunikace typicky mezi hlavní stanicí a jednou podřízenou. Master požaduje data z databáze v první podřízené stanici a poté se komunikace přepne na další podřízenou stanici také s požadavkem o data. Takto se Master dotazuje postupně dokola na každou podřízenou stanici. Komunikačním médiem může být telefonní linka, optický kabel nebo rádio. Každá stanice může slyšet zprávy od Mastera, ale je oprávněna reagovat pouze na zprávy adresované jí.

1.1.1 Model datové komunikace

DNP3 je založen na normách IEC (International Electrotechnical Commission), která pracuje na třech vrstvách OSI tzv. modelu EPA (Enhanced Performance Architecture). Model EPA na Obr. 1.1 zobrazuje diagram komunikace Master (hlavní stanice) - Outstation (podřízená stanice) [2]. Každá stanice má uložená data ve své databázi. Master používá hodnoty pro specifické účely např. zobrazování stavů



Obr. 1.1: Datová komunikace EPA modelu.

systému, kontroly uzavřených smyček, upozornění na poplachy atd. Jeho cílem je aktualizovat databázi, čehož dosahuje zasíláním požadavků na vysílací stanici a požáduje vrácení hodnot do databáze. Podřízená stanice reaguje na žádosti a předává svůj obsah databáze.

Datové typy jsou uspořádány do matic. Binární vstupní hodnoty představují stavy fyzických nebo logických zařízeních. Hodnoty v analogovém vstupu představují měřitelné nebo vypočítané odchylky vstupní veličiny. Čítače představují hodnoty počtu, jako jsou kilowatthodiny, které se stále zvyšují. Pokud dosáhnou maxima, přehodnotí se na nulu a začnou počítat znovu.

Kontrolní výstupy jsou uspořádany do pole reprezentujícího fyzické nebo logické zapnutí/vypnutí, body vzestupu a spouštění. Analogové výstupy představují fyzické nebo logické veličiny, které byly žádané.

1.1.2 Uživatelská vrstva

Uživatelská vrstva v hlavním stanici iniciuje přenos dat tím, že zařídí, aby aplikační vrstva odeslala požadavek na podřízenou stanici. Žádost obsahuje pouze funkční kód nebo i více DNP3 objektů, které určují, jaké údaje jsou požadovány.

Uživatelská vrstva podřízené stanice iniciuje odpověď podle toho, jaké údaje požaduje hlavní stanice. Vybere, klasifikuje a prezentuje tato data aplikační vrstvě.

1.1.3 Aplikační vrstva

Zprávy aplikační vrstvy jsou rozděleny na fragmenty, která je určena z vyrovnávací paměti přijímacího zařízení. Pokud je zpráva vetší, vyžaduje více fragmentů. Poskytuje standardizované funkce, datové formáty a postupy pro efektivní přenos hodnot, atributů a řídících příkazů.

Fragment je blok oktetů obsahující informace o požadavku nebo odpovědi přenesené mezi hlavní a podřízenou stanicí. Nese informace, jak bude zpracován nebo zda byl přijat v řádném pařadí.

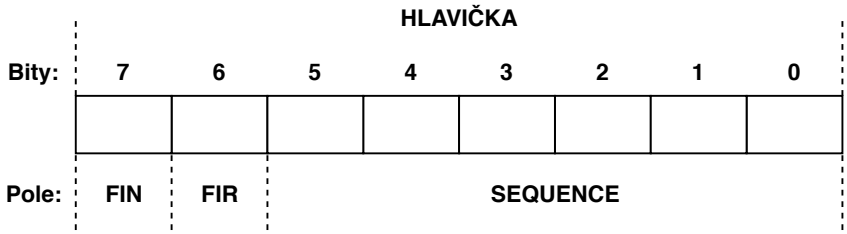
Komunikace s dotazováním na podřízenou stanici se nazývá Polling. Pokud podřízená stanice vyšle odpověď, aniž by obdržela konkrétní požadavek na data, tak se jedná o nevyžádanou zprávu (Unsolicited response). Toto je užitečné v případě, že systém obsahuje mnoho podřízených stanic a hlavní stanice vyžaduje co nejdříve oznámení o změně, než čekat, až se na podřízenou stanici dostane hlavní pollingový cyklus [3].

1.1.4 Transportní vrstva

Po předání zpráv z aplikační vrstvy transportní vrstva řeší segmantaci zpráv, pokud je fragment velký. Při předání rámce z linkové vrstvy tato vrstva poskládá segmenty a předá aplikační vrstvě.

Hlavička transportní vrstvy Obr. 1.2 je první oktet v segmentu a je rozdělena na tři části neboli pole:

- **FIN field** - jedná se o jeden bit. Pokud je nastaven na 1, jedná se o poslední poslaný segment fragmentu,
- **FIR field** - jedná se o jeden bit. Pokud je nastaven na 1, jedná se o první poslaný segment fragmentu,
- **SEQUENCE number field** - jedná se o 6-bit pole. Používá se k ověření, zda poslané segmenty byly přijaty ve správném pořadí a chrání před duplikovanými nebo chybějícími segmenty. Počet se inkrementuje o jedna modulo 64 [3].



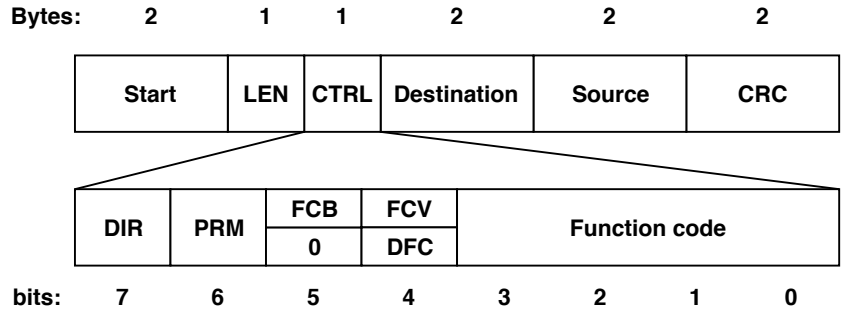
Obr. 1.2: Hlavička transportní vrstvy.

1.1.5 Linková vrstva

Linková vrstva poskytuje propojení rozhraní mezi transportní vrstvou a fyzickou vrstvou. Hlavní účel je adresování stanic, detekce chyb a přidává poslední přenosové oktety specifické pro DNP3 protokol. Tato vrstva přebírá komunikaci ze spodní vrstvy, kde jsou data reprezentována jako souvislý proud, bez ohledu na fyzické komunikační medium. Z transportní vrstvy zakóduje segmenty, vytvoří rámec a vyšle rámec na komunikační kanál. Také řídí tok synchronizace rámců, zpracovává chyby a poskytuje údaje o stavu spojení [3].

Vysvětlení jednotlivých polí hlavičky na Obr.1.3:

- **Start** - začátek DNP3 rámce.
- **LEN** - délka (Length), která udává počet oktetů (5-255) včetně datové části, ale bez CRC bloku.
- **CTRL** - kontrolní blok (Control), který nese informace o rámci. Blíže popsáno v tabulce 1.1.
- **Destination** - adresa cílové stanice.
- **Source** - adresa zdrojové stanice.
- **CRC** - kontrolní součet hlavičky rámce (Cyclic redundancy check).



Obr. 1.3: Hlavička linkové vrstvy.

Bitový prostor v poli **Function code** identifikuje funkci nebo službu spojenou s rámcem datového spojení. Definice hodnot závisí na tom, zda jsou zprávy odesílány z

Tab. 1.1: Význam bitů v CTRL bloku.

DIR = 0	Indikace rámce od Master
DIR = 1	Indikace rámce od Outstation
PRM = 0	Dokončený přenos a výběr funkčního kódu podle Tab. 1.2
PRM = 1	Zahájený přenos a výběr funkčního kódu podle Tab. 1.3
FCB	Bitový součet rámce
FCV = 0	Stav FCB je ignorován
FCV = 1	Stav FCB byl platný
DFC = 0	Vyrovnávací paměť byla připravena
DFC = 1	Vyrovnávací paměť není k dispozici, nebo je stanice zaneprázdněna

Tab. 1.2: Funkční kódy pro PRM = 0.

Funkční kód	Název kódu	Popis služby
0	ACK	Pozitivní potvrzení
1	NACK	Negativní potvrzení
2 - 10	-	Rezervováno
11	LINK_STATUS	Stav spojení
12 - 13	-	Rezervováno
14	-	Zastaralé
15	NOT_SUPPORTED	Služba není podporována

primární stanice do sekundární nebo naopak. Tabulka 1.2 zobrazuje možnosti funkcí dokončeného přenosu a tabulka 1.3 spuštěného přenosu.

1.1.6 Fyzická vrstva

Mnoho implementací používá sériovou komunikaci RS-232 a RS485. Dále je možné použít i optická vlákna nebo paketově orientované sítě TCP/IP, jako je Ethernet[4].

1.2 IEC61850

Průmyslový protokol IEC61850 obsahuje soubor norem, které určují pravidla pro zařízení v rozvodnách a stanovuje požadavky, která jsou z hlediska komunikace kladeny. V rozvodnách se používá spousta protokolů, ale jen soubor norem IEC61850 představuje standardizovanou, jednotnou metodu integrace zařízení a tvorby komunikační sítě. Protokol poskytuje univerzálnost mezi zařízeními a mezi výrobci

Tab. 1.3: Funkční kódy pro PRM = 1.

Funkční kód	Název kódu	Popis služby	FCV bit
0	RESET_LINK_STATES	Reset spojení	0
1	-	Zastaralé	-
2	TEST_LINK_STATES	Testovací funkce pro spojení	1
3	CONFIRMED_USER_DATA	Doručování dat, nutné potvrzení	1
4	UNCONFIRMED_USER_DATA	Doručování dat, potvrzení se nevyžaduje	0
5 - 8	-	Rezervováno	-
9	REQUEST_LINK_STATUS	Stav požadavku spojení	0
10 - 15	-	Rezervováno	-

umožňuje volně vyměňovat informace. Tato IED zařízení zajišťuje ochranu a dohled nad provozem, měření a regulaci [5].

1.2.1 Standardy IEC61850

Přehled standardů je vypsán v následující tabulce 1.4. Jednotlivé standardy definují metody komunikace, komunikační protokol, rozhraní a objektově orientované datové modely pro energetiku nebo elektrizační soustavy. Jsou určeny především pro výrobce IED zařízení, komponent v komunikační síti, testování aplikací a programová vybavení, která musí být v souladu s těmito normami [6][7].

1.2.2 Datový model IEC61850

Datový model představuje jednotlivé funkční celky. Je objektově orientovaný, a tak umožňuje zachovat trvale konzistentní model, který je nezávislý na protokolu nebo reálného přístroje. Skládá se z následujících částí:[6][7]

- **Fyzické zařízení**.
- **Logické zařízení**.
- **Logický uzel**.
- **Datový objekt**.
- **Datový atribut**.

Tab. 1.4: Přehled standardů IEC 61850.

Označení normy	Název
IEC 61850-1	Úvod a přehled
IEC 61850-2	Výklad zvláštních výrazů
IEC 61850-3	Všeobecné požadavky
IEC 61850-4	Systémové a projektové řízení
IEC 61850-5	Požadavky na komunikaci pro funkce a modely zařízení
IEC 61850-6	Konfigurační popisový jazyk pro komunikaci v elektrických stanicích týkající se IED
IEC 61850-7	Základní komunikační struktura pro podřízené stanice a napájecí zařízení
IEC 61850-7-1	Zásady a modely
IEC 61850-7-2	Abstraktní rozhraní pro komunikační služby ACSI
IEC 61850-7-3	Obecné třídy dat
IEC 61850-7-4	Třídy kompatibilních logických uzlů a třídy dat
IEC 61850-8	Mapování specifických komunikačních služeb (SCSM)
IEC 61850-8-1	Mapování do MMS (ISO/IEC 9506 – část 1 a část 2) a do ISO/IEC 8802-3
IEC 61850-9	Mapování specifických komunikačních služeb (SCSM)
IEC 61850-9-1	Přenos vzorkovaných hodnot po sériovém jednosměrném vícebodovém spoji
IEC 61850-10	Zkoušky shody

Fyzické zařízení

Fyzické zařízení představuje ochranný prvek neboli terminál definovaný IP adresou. Chrání před přepětím, nadproudu a dalších elektrických nežádoucích vlivů. Dalším unikátním identifikátorem je název o maximálně deseti znacích. Název fyzického zařízení není standardizován normou IEC 61850, a tak záleží na dohodě s výrobci ochran a samotnými zákazníky.

Logické zařízení

Logické zařízení je v datovém modelu podskupinou fyzických zařízení a v jednom fyzickém zařízení může být definováno několik logických zařízení. Definují skupinu služeb a logických uzlů s podobnými rysy. Účel logických zařízení a jejich název specifikuje výrobce. Podle standardu musí každé logické zařízení obsahovat následující tři logické uzly:

- **LPHD (Physical Device information)** - obecné informace o fyzickém zařízení např. název, stav (aktivní, neaktivní, porucha), reset zařízení atd.
- **LLN0 (Logical Node zero)** - obecné informace o všech logických uzlech např. provozní režim, datové sady a objekty pro zasílání událostí RCB (Report Control Block). Základní informace jsou dědičné i pro ostatní logické uzly ve společném logickém zařízení.
- **LN (Logical Node)** - reprezentuje např. konkrétní ohrannou funkci definovanou podle IEC 61850-7-4.

Logické uzly

Logické uzly obsahují skupiny dat a služeb, které souvisí se specifickou funkcí v elektrizační soustavě. Jedná se o vizualizaci konkrétních zařízení např. výkonový vypínač, odpojovač, zkratovač, apod. a jejich vzájemné blokády, ochranné funkce, funkce pro měření atd.

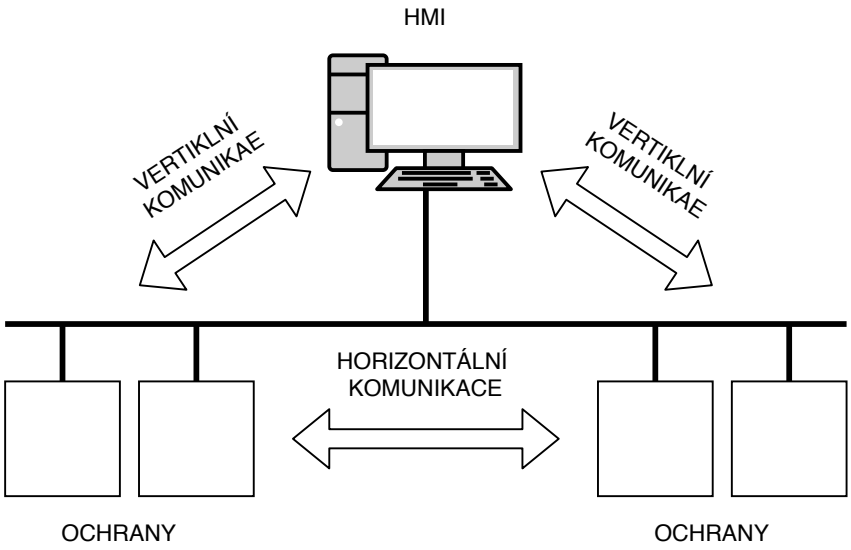
Datový objekt a datový atribut

Datový objekt reprezentuje základní stavební prvek objektově orientovaného modelu IEC 61850. Logický uzel obsahuje několik datových objektů a mohou je přejímat ostatní logické uzly. Součástí datového objektu je datový atribut, kde jeho hodnoty jsou logické stavy - zapnuto, vypnuto nebo nastavení ochranných funkcí, povely, měřené hodnoty atd. Jedná se o nejmenší část modelu.

1.2.3 Vertikální komunikace

Vertikální komunikace je určena pro přenos informací mezi nadřazeným řídícím systémem např. terminál, SCADA klient nebo server a ochranou. Komunikace znázorněna na Obr. 1.4 a poskytuje přístup klient/server, který obsahuje pouze časově nekritická data a služby, a současně může nabízet data více klientům. To se ovšem netýká ovládání primárních prvků, kde z pohledu bezpečnosti je požadován přístup pouze jednomu klientovi. V rámci této komunikace servery nabízí následující služby:[7]

- MMS (Manufacturing Message Specification) služby pro monitoring a sběr dat (stavové hodnoty, události, alarmy, měřené hodnoty apod.),
- dispečerské řídící povely (vyber, proveď, ovládání primárních zařízení, reset signalizačních LED, ...),
- přenos souborů přes FTP (File Transfer Protocol) pro stahování záznamů z poruchových zapisovačů atd.



Obr. 1.4: Komunikace IEC 61850.

1.2.4 Horizontální komunikace

Horizontální komunikace představuje přenos informací pouze na jedné úrovni mezi jednotlivými ochrannými zařízeními. Využívá metodu přístupu klient-klient a umožňuje časově kritickou výměnu dat mezi jednotlivými ochranami. Ukázka komunikace na Obr. 1.4.

Pro časově kritický a spolehlivý přenos informací v horizontální komunikaci podle IEC 61850-7-2 jsou definovány zprávy typu GSE (Generic Substation Event – všeobecná událost rozvodny). Přenášejí se pro určitou skupinu ochranných zařízení pomocí multicastingu [7].

GSE zprávy jsou rozděleny do dvou typů a liší se strukturou a časovou kritičností:

- **GOOSE (Generic Object Oriented Substation Events)** - objektově orientovaná událost rozvodny. Stavová data a proměnné hodnoty v data setech jsou přenášena v určitém časovém intervalu přes místní síť (LAN). Odběratelé těchto dat musí být registrovaní v multicastingu a jejich reakce je závislá na rychlých reakcích ostatních odběratelů. Doba nesmí překročit hranici 4 ms od vzniku události po odeslání zprávy [5].
- **GSSE (Generic Substation State Events)** - generická stavová událost rozvodny. V této zprávě se přenáší pouze stavová data s využitím stavového seznamu. Jedná se o řetězec bitů, nikoliv o datový objekt. Na rozdíl od GOOSE zpráv jsou přenášeny prostřednictvím MMS služby a jejich zpracování a přenos trvá déle [7].

1.3 IEC60870-5

IEC definuje normy IEC60870 pro dálkové, dohledové řízení a sběr dat v oblasti elektrotechniky a automatizace energetických systémů. U nás je zavedená jako ČSN EN 60870. Část 5 poskytuje komunikační profil pro odesílání základních zpráv mezi centrální stanicí a dálkovými podřízenými stanicemi, která jsou napřímo připojená mezi sebou. Je postavený na architektuře EPA, který vychází z modelu ISO/OSI [8].

1.3.1 Normy IEC 60870

Přehled o skupinách norem:[9][10]

- **IEC 60870-1, 60870-2** - zabývají se všeobecnými ustanoveními a provozními podmínkami.
- **IEC 60870-3** - popisuje elektrické charakteristiky rozhraní a podmínky pro zozhraní.
- **IEC 60870-4** - zabývá se s požadavky na vlastnosti. Důležité na provoz systémů dálkového ovládání a zpracování dat.
- **IEC 60870-5** - přenosové protokoly.
- **IEC 60870-6** - týká se protokolů pro dálkové ovládání. Vychází z modelu klient-server. Jde o přenos informací mezi jednotlivými dispečinky.

Pátá část definuje pět dokumentů:

- **IEC 60870-5-1** Formáty přenosového rámce - asynchronní přenos dat s linkovými protokoly, standardy pro kódování, formátování a 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.
- **IEC 60870-5-5** Základní aplikační funkce - standardy pro zajištění interoperability různých zařízení elektrizační soustavy.

Dále pro těchto pět dokumentů jsou postaveny společné normy:

- **IEC 60870-5-101** Společná norma pro základní úkoly dálkového ovládání - popisuje mechanismy, které byly popsány v předchozích odstavcích.

- **IEC 60870-5-102** Přenos integrovaných součtových hodnot v elektrizačních soustavách - pro odborníky v tomto oboru.
- **IEC 60870-5-103** Informační rozhraní ochran - zajišťuje interoperabilitu mezi ochranami a zařízeními řídicího systému podřízené stanice.
- **IEC 60870-5-104** Síťový přístup pro IEC 60870-5-101 používající normalizované transportní profily - stanovuje použití v běžných komunikačních sítích např. Ethernet s TCP/IP [10].

1.3.2 Komunikace pomocí protokolu TCP/IP

Norma IEC 60870-5-104 definuje přenos aplikačních datových jednotek ASDU (Application Service Data Unite) ze společné normy IEC 60870-5-101 a přidává další datové jednotky, které zajistí komunikaci po IP vrstvě.

Zde není aplikována architektura EPA, protože pro komunikaci přes TCP/IP je potřeba použít transportní a síťovou vrstvu. Řídící informace Aplikačního protokolu (ACPI) nabízí mechanizmus pro zjištění začátku a konce datových jednotek ASDU a také slouží pro řízení. Z protokolů ASDU a APCI se vytvoří Jednotka dat Aplikačního protokolu (APDU). Tvorba ASDU a APCI probíhá na aplikační vrstvě. Komunikační přenos mezi systémy je řešen pomocí PDU na jednotlivých vrstvách např. datagramů, packetů nebo rámců [11].

1.3.3 Srovnání IEC 60870-5 s DNP3 a IEC 61850

Norma IEC 60870-5 nachází uplatnění především v elektroenergetice, ale její použití jinde se nedá vyloučit. Podobný protokol s IEC 60870-5 je DNP3. Byl vytvořen v reakci na rostoucí požadavky severoamerického průmyslu. DNP3 se v současnosti používá v mnoha oblastech procesního průmysl [10].

IEC 61850 je z pohledu komunikace vesměs stejný jako IEC 60870-5, ale byly vyvíjeny nezávisle na sobě. Oba definují standard pro přenos dat přes TCP/IP. Nicméně služby a protokoly v reálném čase, informační modely a programovací jazyk SCL nejsou v IEC 60870-5 definovány. IEC 61850 norma by měla do budoucna nahradit komunikaci pomocí IEC 60870-5. Lze říci, že IEC 61850 je modernizací IEC 60870-5 [11].

2 APACHE JMETER

JMeter je nástroj s otevřeným zdrojovým kódem, který dokáže generovat různé zátěžové testy pro služby a zároveň analyzovat výsledky měření. Jelikož má vysoce rozšířitelné jádro, skládá se především z pluginů neboli modulů a díky nim podporuje dané funkce. Testuje webové aplikace, zátěže serveru nebo skupiny serverů, a síťové infrastruktury s různými typy zařízeními. Umožňuje i načítat a testovat mnoho typů protokolů [12].

2.1 Funkce nástroje

Mezi základními prvky pro sestavení testovacího plánu jsou:

- **Samplers** (Vzorkovače) posílají požadavky na server a čekají na odpověď. JMeter obsahuje tyto vzorkovače:
  FTP Request,
  HTTP Request,
  JDBC Request,
  Java object request,
  JMS request,
  JUnit Test request,
  LDAP Request,
  Mail request,
  OS Process request,
  TCP request.
- **Logic Controllers** (Logický kontroler) určuje pořadí a logiku, jak má být sampler zpracován,
- **Listener** (Naslouchač) poskytuje informace o testování pro vykreslení grafů a ukládá je pro další použití,
- **Timer** (Časovač) ovlivňuje časový průběh testování,
- **Assertions** (Tvrzení) pomáhá prokázat správnost odpovědi od testovaného serveru, která se očekává,
- **Pre processor** (Předzpracování) provádí akci před vykonáním požadavku sampleru,
- **Post processor** (Pozpracování) provádí akci po vykonání pořadavku sampleru.

3 Praktická část práce

V praktické části bakalářské práce bude provedeno seznámení s nástrojem, instalace nástroje JMeteru do vývojového prostředí Eclipse a vytvoření modulu pro ukázku, který bude spouštět jednoduchý program. Cílem práce je návrh a implementace dvou modulů do JMeteru, které budou generovat provoz protokolů DNP3 a IEC61850 za pomocí knihoven.

3.1 Instalace JMeteru do vývojového prostředí Eclipse

Nástroj JMeter je vyvíjen v jazyce Java a pro vývoj vlastních modulů se použije vývojové prostředí Eclipse 2018-09 nainstalovaný na operačním systému Windows 10 Home. Pracovat se bude s aktuální verzi JMeteru 5.0. Struktura nástroje umožňuje vytvářet vlastní moduly za pomocí prvků a samplerů. DNP3 a IEC61850 protokoly budou právě jedněmi ze samplerů.

3.1.1 Nastavení prostředí

Z webových stránek[12] je potřeba stáhnout aktuální verzi zdrojových souborů nástroje a extrahovat do adresáře workspace vývojového prostředí Eclipse.

Poté importujeme stažený subory přes File > Import > General > Projects from Folder or Archive

V Project > Properties > Java Compiler > Building > Output folder se vyloučí soubory s koncovkou .metaprops vepsáním ***.metaprops** do Filtered resources, aby nedošlo k jejich sestavení.

Prostředí hlásí, že nemůže nalézt importované balíčky, proto se musí nahradit obsah souboru *.classpath* ve složce projektu za soubor *eclipse.classpath*, kde jsou napsány všechny cesty k balíčkům.

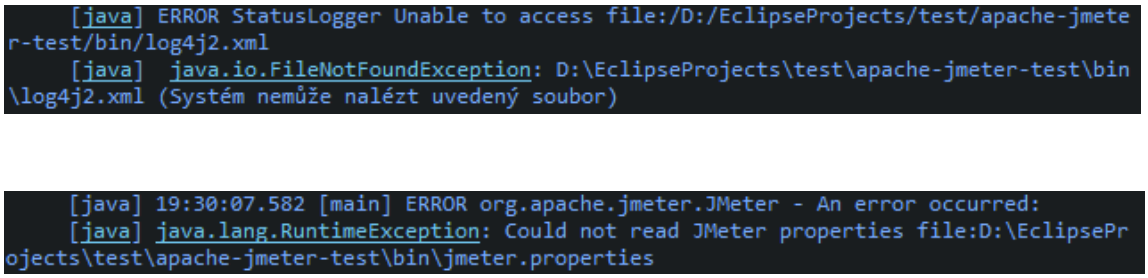
Soubor *build.xml* je program, který všechny svoje části zkompiluje nebo nainstaluje. Otevře se kompilační nástroj **Ant** pomocí Windows > Show View > Other > Ant. Add Buildfiles otevře funkce v souboru.

Spuštěním **download_jars** se stáhnou potřebné soubory typu *.jar*, které jsou nezbytné pro správnou funkci nástroje.

Následně spuštěním funkce **install** [**default**] se vše nainstaluje.

Samotný nástroj JMeter se spustí funkcí **run_gui**.

Po pokusu o spuštění JMeteru se mouhou naskytnou chyby, které jsou zobrazeny na obrázku 3.1. Je totiž potřeba vložit do složky **bin** soubory *log4j2.xml* a *jmeter.properties*, které se smazaly při importování JMeteru do Eclipsu. Soubory jsou k dispozici ve zdrojových nebo binárních souborech na webu. Aby šlo projekt



Obr. 3.1: Nenalezení souboru log4j2.xml a jmeter.properties.

debugovat, je ještě potřeba upravit v hlavní třídě *NewDriver.java*, která se stará o spuštění celého programu, cestu k projektu podle obrázku 3.1 [13].

Výpis 3.1: Správna cesta k projektu.

```
File userDir = new File(System.getProperty("user.dir"));
//tmpDir = userDir.getAbsoluteFile().getParent();
tmpDir = userDir.getAbsolutePath();
```

3.1.2 Vytvoření vlastního modulu

Nový modul se vytvoří z ukázkových příkladů nacházející se ve složce **src/examples**. Každý modul je tvožen ze dvou základních tříd *Sampler.java*, která je důležitá pro vykonání práce modulu, a *SamplerGui.java*, která se stará pomocí grafického rozhraní o vstupní parametry uživatele pro předání do sampleru. Třídy rozšiřují abstraktní třídy, které zajišťují funkčnost a integritu s programem JMeter.

Vložení modulu do JMeteru

Vytvoří se nová složka v adresáři **src/** a pojmenuje se podle názvu protokolu, který se bude realizovat. Zkopírují se obě třídy ukázkového sampleru a vloží do vytvořené složky např. **src/DNP3**. Vhodně se třídy přejmenují. Následně je potřeba automatizovat kompilaci a přesunutí modulu do **apache-jmeter-5.0/lib/ext**. Docílí se tím, že se upraví soubor *build.xml*. Nastaví se relativní cesta k souboru *.jar*:

Výpis 3.2: Nastavení cesty .jar souboru.

```
<findbugs ... >
  ...
  <class location="${dest.jar}/DNP3.jar"/>
</findbugs>
```

Dopsání zdrojové složky a umístění *.class* souborů nového modulu:

Výpis 3.3: Nastavení zdrojové složky.

```
...
<property name="src.DNP3l" value="src/DNP3"/>
...
<property name="build.DNP3" value="build/DNP3"/>
...
```

V další části souboru se vytvoří funkce pro kompilaci, která je ve výpisu A.1 a do sekce **JMeter launch jar** se dopíše část pro vytvoření *DNP.jar* souboru. Kód ve výpisu A.2.

Nakonec v adresáři **src/core/org/apache/jmeter/resources** se otevře soubor *messages.properties* a přidá se na poslední řádek text: **dnp3_modul_title=DNP3**, aby JMeter dohledal, jak se vytvořený modul nazývá [12].

Všechny změny se aktualizují pomocí nabídky File > Refresh v prostředí Eclipse.

Zkouška nového modulu a první spuštění

Pro odzkoušení vytvořeného modulu se spustí aplikace *notepad.exe*. Je zapotřebí ve zdrojovém kódu *DNP3Sampler.java* vytvořit proces, který spustí podle zadané cesty aplikaci:

Výpis 3.4: Spuštění Notepad.exe.

```
...
res.sampleStart(); // Start timing
try {

            // Do something here ...

  String[] command = new String[] {"C:\\Windows\\
notepad.exe"};
  ProcessBuilder pb = new ProcessBuilder(command);
  pb.start();
  ...
} ...
```

Po zkompilování kódu a následném spuštění JMeteru se klikne pravým tlačítkem myši na Test plan > Add > Threads (Users) > Thread Group, následně levým tlačítkem na Thread Group > Add > Sampler > DNP3 a na tlačítko **Start**.

3.2 Zprovoznění OPENDNP3

Program OpenDNP3 je referenční implementace IEEE-1815 (DNP3) pod licencí Apache License. Je napsán v jazyce C++, ale podporuje i vývoj v jazyce Java nebo aplikace .NET. V tomto postupu se použije Java, která je vazbou postavena na normálních souborech Java JAR a nativní sdílenou knihovnou C++ [14].

3.2.1 Sestavení programu a komunikace Windows - Debian

Jako testovací stanice byly vybrány fyzický stroj Windows 10 64bit a virtuální počítač Debian 9 64bit. Na Obr. 3.2 je znázorněn diagram komunikace, která probíhá na ethernetovém rozhraní. Windows 10 má IP adresu 192.168.1.100 a Debian 9 192.168.1.200. Ve Windows 10 se pracuje s vývojovým prostředím Eclipse. U obou stanic je ukázána konfigurace jak Master stanice, tak i podřízené stanice.

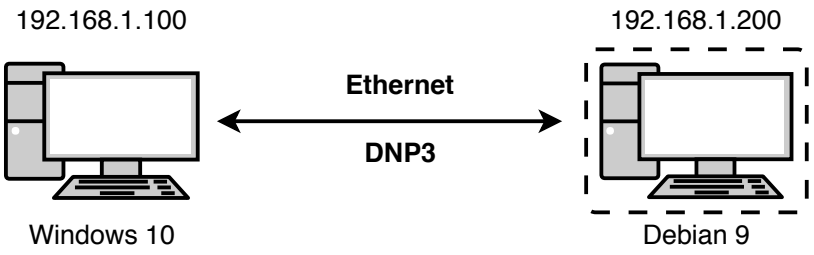
Windows 10 64bit

Bude potřeba nástroj pro buildování CMake, sestavení vytvořeného buildu Visual Studio 2017, který nám vygeneruje knihovnu *.dll* a nástroj Maven k instalaci java knihovny.

Stáhne se OpenDNP3 repozitář z GitHubu verze 2.0.x a extrahuje se do pracovního adresáře
https://github.com/automatak/dnp3

Dále je důležité stáhnout knihovnu ASIO C++ verze 1.10.8
https://github.com/chriskohlhoff/asio/tree/asio-1-10-8

Extrahuje se archiv a obsah složky **asio-asio-1-10-8** se vloží do zdrojového adresáře dnp3-2.0.x **/debs/asio**.



Obr. 3.2: Diagram komunikace mezi zařízeními.

V příkazovém řádku se přejde do pracovního adresáře a vytvoří se nová složka pro sestavení
>cd dnp3-2.0.x
>mkdir build
>cd build

Následně se zadá příkaz pro build knihovny, kde se přepínačem -G zvolí kompilační program tzv. generátor. Důležité je, aby se vybral pro 64 bitovou architekturu, jelikož testovací zařízení jsou 64 bitové. Dále doplňující volbou -DDNP3_JAVA=ON se nastaví podpora pro Javu
>cmake ../ -G "Visual Studio 15 2017 Win64" -DDNP3_JAVA=ON

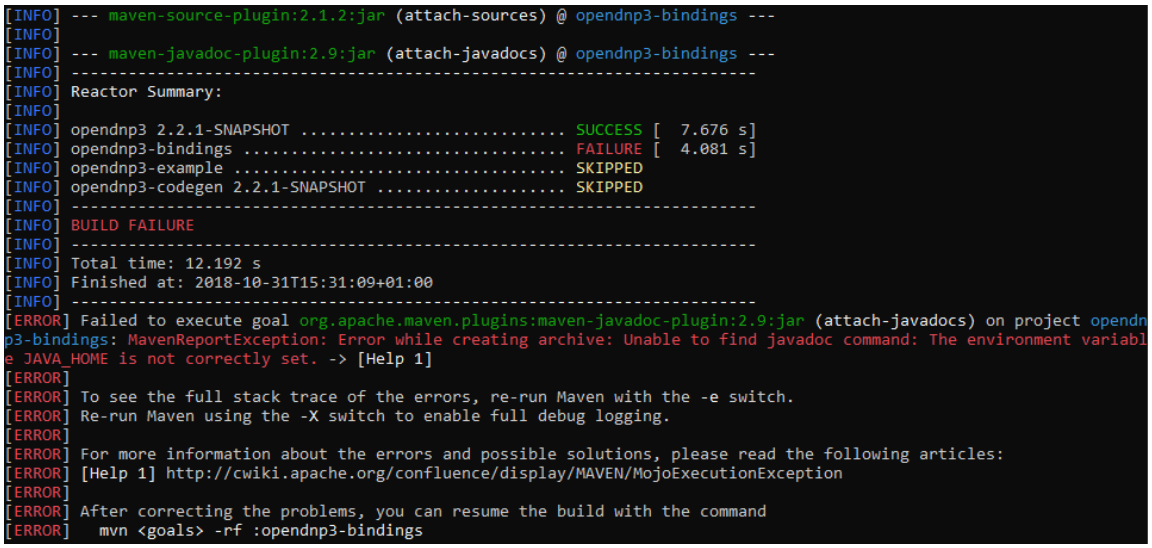
Pokud se naskytnou chyby, je dobré ověřit, zda proměnné prostředí systému Windows zná cesty pro nástroj CMake, domovský adresář Java a knihovny JNI (Java Native Interface) *../Java/jdk-10.0.2/lib*.

Otevře se v **build/** soubor *opendnp3.sln* ve vývojovým prostředí Visual Studio a přes nabídku Sestavit > Sestavit řešení se vtvoří knihovna *opendnp3java.dll* v adresáři **build/Debug**.

Spustí se přikazový řádek a přejde se do složky **java** a provede se lokální build a instalace java knihovny pomocí nástroje Maven do adresáře **java/bindings/target**
>cd java
>mvn install

Může se vyskytnout chyba, která je vidět na Obr. 3.3. Opraví se vložením pár řádků do souboru *pom.xml* v **java/**, které jsou vidět ve výpisu 3.5. Tag **<javadocExecutable>** určuje cestu, kde se nachází *javadoc.exe* a **<additionalparam>** přidá parametr *-Xdoclint:none*, který zakáže upozornění na chybějící komentáře nebo neplatné syntaxe Javadoc či chybějící značky HTML (Hypertext Markup Language) [15].

Vytvořené knihovny *opendnp3java.dll* a *opendnp3-bindings-2.2.1-SNAPSHOT.jar* se pak musí vložit do adresáře budoucího projektu, aby je JVM (Java Virtual Machine) našel.



Obr. 3.3: Chyba při instalaci nástrojem Maven.

Výpis 3.5: Korekce chyby v souboru pom.xml.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-javadoc-plugin</artifactId>
  <version>${maven-javadoc-plugin.version}</version>
  <configuration>
    <!-- Default configuration for all reports -->
    <javadocExecutable>${java.home}/bin/javadoc
</javadocExecutable>
    <additionalparam>-Xdoclint:none</additionalparam>
  </configuration>
  ...
</plugin>
```

Vytvoří se nové projekty v Eclipse. Jeden bude prezentovat řídící stanici a druhý podřízenou stanici. Vhodně se zkopírují zdrojové kódy z ukázkových příkladů v **java-/example/src/main/java/com/automatak/dnp3/example** *MasterDemo.java* a *OustationDemo.java*. Do složek projektů se vloží vytvořené knihovny *opendnp3java.dll* a *opendnp3-bindings-2.2.1-SNAPSHOT.jar*. V nastavení projektu Project > Properties > Java Build Path > Libraries se přidá java knihovna do Classpath > Add JARs...

V kódu se změní vstupní parametry metody *addTCPClient()* objektu *manager* třídy *DNP3Manager*, která přídá TCP klienta. Dále se nastaví podle výpisu 3.6 adresy linkové vrstvy objektu *config*.

Výpis 3.6: Konfigurace MasterDemo.java.

```
...
Channel channel = manager.addTCPClient(
      "client",
      LogMasks.NORMAL | LogMasks.APP_COMMS,
      ChannelRetry.getDefault(),
      "192.168.1.200",
      "192.168.1.100",
      20000,
      PrintingChannelListener.getInstance()
);
MasterStackConfig config = new MasterStackConfig();
config.link.localAddr = 1;
config.link.remoteAddr = 10;
...
```

Podobně se upraví podřízená stanice podle výpisu 3.7.

Výpis 3.7: Konfigurace OustationDemo.java.

```
...
Channel channel = manager.addTCPServer(
      "client",
      LogMasks.NORMAL | LogMasks.APP_COMMS,
      ServerAcceptMode.CloseExisting,
      "192.168.1.100",
      20000,
      PrintingChannelListener.getInstance()
);
OutstationStackConfig config = new OutstationStackConfig(
DatabaseConfig.allValues(5), EventBufferConfig.allTypes(50));

config.linkConfig.localAddr = 10;
config.linkConfig.remoteAddr = 1;
...
```

Debian 9 64bit

Ve virtuálním počítači se stáhne balíček pro sestavení a git
sudo apt-get update
sudo apt-get install build-essential
sudo apt-get install git

Stáhne se knihovna openDNP3 a ASIO C++ verze 1.10.8
cd ∼/
git clone https://github.com/automatak/dnp3.git
git clone -b asio-1-10-8 https://github.com/chriskohlhoff/asio.git

Přesune se **asio** do **dnp3/deps**
mv asio dnp3/deps/

Build, instalace a nastavení dynamického linkování
cd dnp3/
cmake CMakeLists.txt
make
sudo make install
sudo ldconfig

Vytvoří se adresáře, kam se zkopírují ukázkové příklady
mkdir master outstation
cp dnp3/cpp/examples/master/main.cpp master/
cp dnp3/cpp/examples/outstation/main.cpp outstation/

Konfigurace stanice master na Debianu

cd master
nano main.cpp
Úpravy jsou dle výpisu 3.8.

Výpis 3.8: Konfigurace stanice master na Debianu.

```
...
auto channel = manager.AddTCPClient(
      "tcpclient",
      FILTERS,
      ChannelRetry::Default(),
      "192.168.1.100",
      "192.168.1.200",
      20000,
      PrintingChannelListener::Create();
...
```

Kompilace a zapsání linkovacích příkazů pro linker
g++ main.cpp -o master.out -lopendnp3 -lasiopal -lasiodnp3 -lopenpal -lpthread

Konfigurace stanice outstation na Debianu

cd outstation
nano main.cpp
Úpravy podle výpisu 3.9.

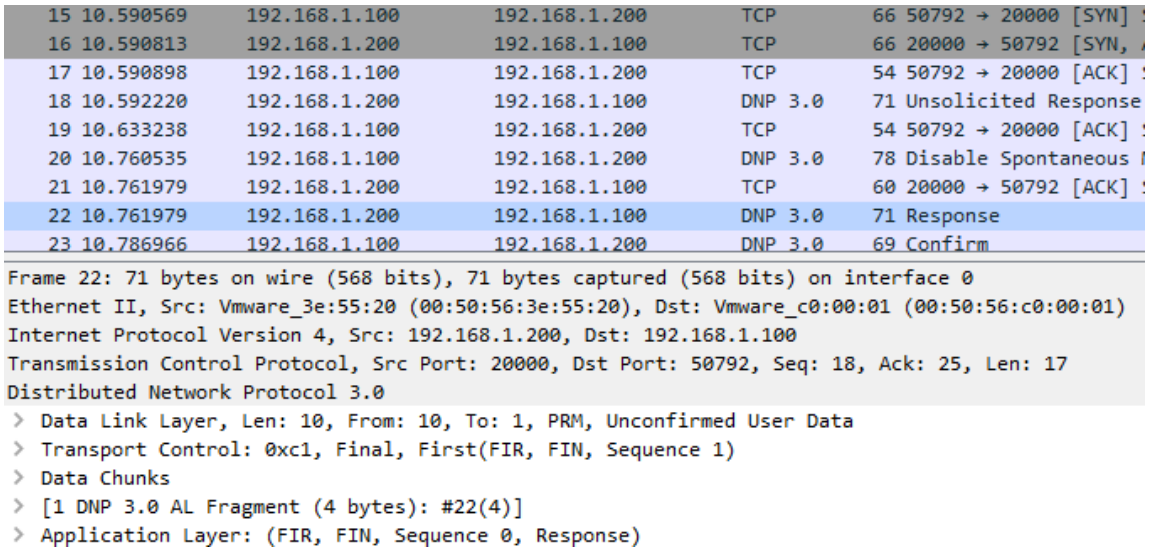
Výpis 3.9: Konfigurace stanice outstation na Debianu.

```
...
auto channel = manager.AddTCPServer(
      "server",
      FILTERS,
      ServerAcceptMode::CloseExisting,
      "192.168.1.200",
      20000,
      PrintingChannelListener::Create());
...
```

Opět kompilace a zapsání linkovacích příkazů pro linker
g++ main.cpp -o outstation.out -lopendnp3 -lasiopal -lasiodnp3 -lopenpal -lpthread

3.2.2 Test komunikace Windows (Master) - Debian (Outstation)

Tímto postupem jsou stanice připravené k testovací komunikaci. Spustí se Master ve Windows a Outstation v Debianu. Ve výpisu 3.10 je vidět vzájemné navázání komunikace. Na Obr. 3.4 je ukázka komunikace pomocí programu Wireshark. Podobně se může otestovat komunikace Debian (Master) a Windows (Outstation).



Obr. 3.4: Ukázka komunikace DNP3 protokolu ve Wireshark.

Výpis 3.10: Navázání komunikace DNP3 protokolu.

```
--MASTER--(Eclipse)
1541334573689 - INFO - manager - Starting thread (0)
Channel state: OPENING
1541334573697 - INFO - client - Connecting to: 192.168.1.200
1541334573700 - INFO - client - Connected to: 192.168.1.200
1541334573701 - INFO - master - Begining task:
Disable Unsolicited
...
Channel state: CLOSED
Channel state: SHUTDOWN

--OUTSTATION--
root@debian:~/outstation# ./outstation.out
ms(1541333484482) INFO    manager - Starting thread (0)
channel state change: OPENING
ms(1541333484547) INFO    server - Listening on:
192.168.1.200:20000
ms(1541333487491) INFO    server - Accepted connection from:
192.168.1.100
channel state change: OPEN
ms(1541333487491) --AL->  outstation - F0 82 80 00
...
channel state change: CLOSED
channel state change: SHUTDOWN
```

3.2.3 Přechod z Windows na systém Linux Debian 9

Předchozí postup byl pro operační systém Windows. Během práce se změnil operační systém na systém Linux s distribucí Debian 9, v kterém bude JMeter spuštěn. V tom případě je potřeba vygenerovat nové knihovny.

Bude potřeba mít nainstalovanou 64-bitovou verzi Javy. Postup pro vygenerování knihoven pro Linux je následující:
Stáhne se knihovna openDNP3 a ASIO C++ verze 1.10.8
cd ∼/
git clone https://github.com/automatak/dnp3.git
git clone -b asio-1-10-8 https://github.com/chriskohlhoff/asio.git

Přesune se **asio** do **dnp3/deps**
mv asio dnp3/deps/

Dále
cd dnp3/
mkdir build
cd build
cmake ../dnp3 -DDNP3_JAVA=ON
make -j
sudo make install

Nyní se může zkopírovat *opendnp3-bindings-2.2.1-SNAPSHOT.jar* knihovnu do nového adresáře v JMeteru. Podobně jako u Windows. Vytvořené *.so* knihovny se naleznou v adresáři **usr/local/lib**. Je možné že bude potřeba přesunout tyto knihovny do **usr/lib**. Seznam vygenerovaných knihoven:

- libasiodnp3.so,
- libasiodnp3.so.2,
- libasiodnp3.so.2.2.1,
- libasiopal.so,
- libasiopal.so.2,
- libasiopal.so.2.2.1,
- libopendnp3.so,
- libopendnp3.so.2,
- libopendnp3.so.2.2.1,
- libopendnp3java.so,
- libopendnp3java.so.2,
- libopendnp3java.so.2.2.1,

- libopenpal.so,
- libopenpal.so.2,
- libopenpal.so.2.2.1.

Nejdůležitější knihovny jsou *libopendnp3java.so*, *libopendnp3java.so.2* a *libopendnp3java.so.2.2.1*. Je ale doporučeno zkopírovat všechny vytvořené do příslušného adresáře, který je popsán výše.

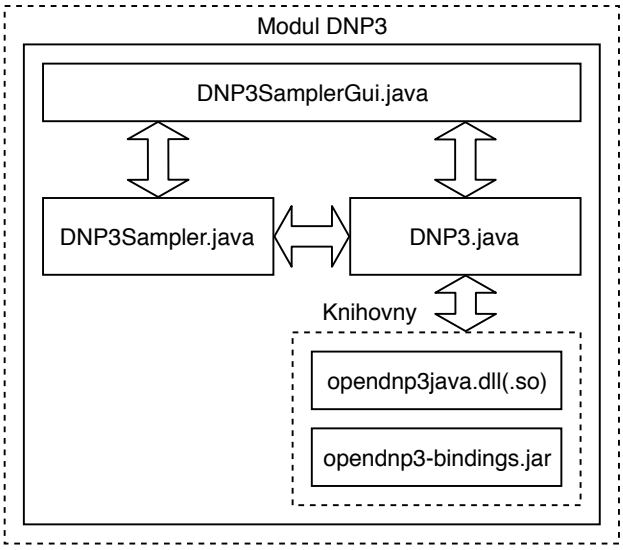
Knihovnu *opendnp3-bindings-2.2.1-SNAPSHOT.jar* je také možné vygenerovat v Linuxu za použití buildovací programu Maven, ale při pokusu o sestavení byla stále zobrazována chyba sestavení. Nejednalo se o chybu jako byla na Obr. 3.3 po příkazu:
mvn install
Tudíš se přešlo na řešení o zkopírovaní již vygenerované knihovny u operačního systému Windows.

3.3 Vlastní návrh a implementace modulu DNP3

V této části se navrhne modul, který bude představovat DNP protokol. V grafické části se vloží požadované parametry, které nastaví komunikaci podle uživatele. Pro ukázku a následný test, se vybraly následující:

- rozsah IP adres podřízených stanic,
- linková adresa hlavní stanice,
- rozsah linkových adres podřízených stanic,
- výběr posílání Unsolicited response (Analogový hodnoty nebo Counter inkrementace),
- časový interval posílání Unsolicited response (nevyžádaných zpráv),
- nastavení portu,
- výběr rozhraní, z které se bude generovat provoz,
- akční tlačítka Apply, Delete a Clear.

Do vytvořeného adresáře v JMeteru, který bude představovat DNP3 modul **apache-jmeter-5.0/src/dnp3** se vytvoří nová třída *DNP.java* a vloží se do ní zdrojové kódy z *OutstationDemo.java*, jelikož se budou vytvářet klienti neboli podřízené stanice. Dále se knihovna *opendnp3java.dll* zkopíruje do adresáře **apache-jmeter-5.0** a *opendnp3-bindings-2.2.1-SNAPSHOT.jar* do **apache-jmeter-5.0/lib**. Po otevření projektu JMeter v Eclipse je potřeba nastavit cestu ke knihovně *opendnp3-bindings-2.2.1-SNAPSHOT.jar*. Na obrázku 3.5 je znázorněno blokové schéma návrhu.



Obr. 3.5: Blokové schéma návrhu DNP3 klienta.

3.3.1 Úprava třídy DNP3.java

V této třídě se změní název statické metody *main()*, která spouští hlavní proces programu, na *startDNP3()*. Později bude volána ze třídy *DNP3Sampler.java*.

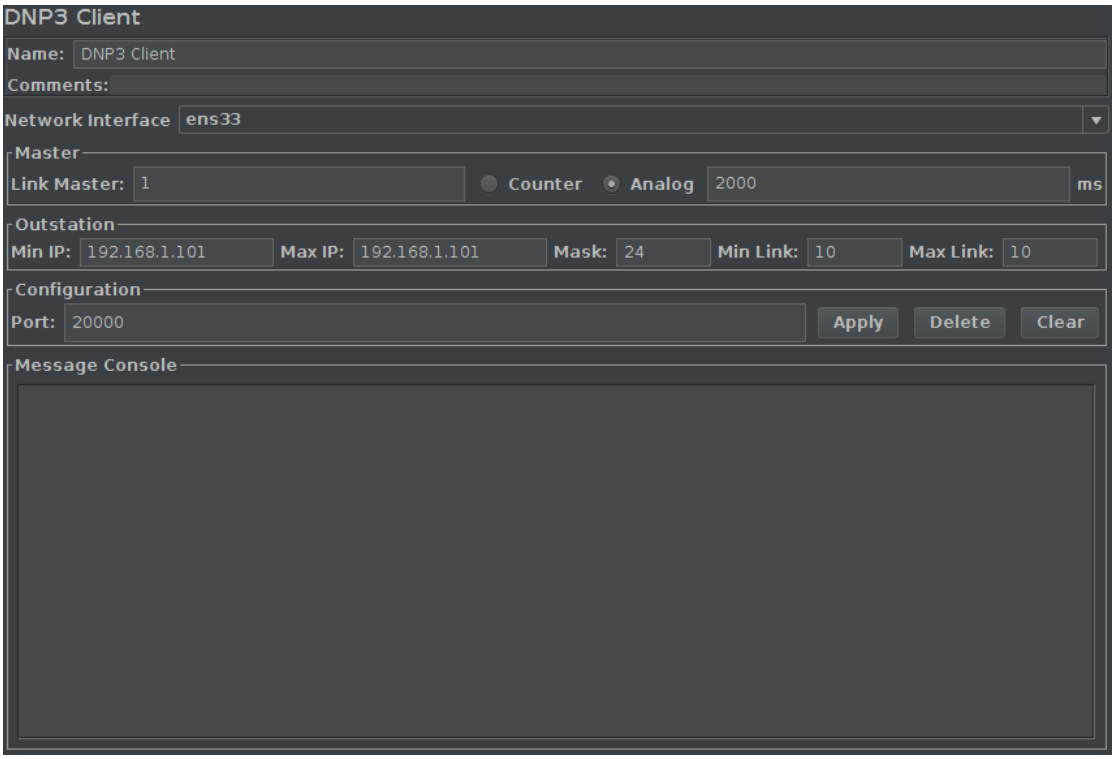
Dopíše se konstruktor pro třídu, v které předá hodnoty *link* (linková adresa klienta), *ip* (IP adresa klienta), *linkMaster* (linková adresa Master stanice), *port* a *time* (časový interval posílání nevyžádaných zpráv). Poté definuje proměnnou *manager*. Ukázka konstruktoru ve výpisu B.1.

Vytvoří se stavová proměnná *command*, uvedena v příloze B.1, která ponese informaci, jaký Unsolicited response zprávy budou posílány nebo zda se mají klienti vypnout. S proměnnou se bude pracovat pomocí *get* a *set* metodami. Poté se stavová proměnná použije v těle *switch*, který pomocí cyklu *while* provádí jednotlivé akce podle toho, jaký stav má proměnná. Tyto akce se provádějí cyklicky se zpožděním, které definuje uživatel v poli pro časový interval posílání Unsolicited response. V části pro akci "analog" byl vytvořen postup pro náhodné vybírání číselných hodnot, které ponese zpráva. V části "counter" se pravidelně inkrementuje čítač. Oba tyto postupy se napsali proto, aby se odlišily zprávy mezi sebou. Ukázka ve výpisu B.1.

Dále je potřeba nastavit klienta, aby mohl posílat nevyžádané zprávy (Unsolicited response) podle výpisu 3.11.

Výpis 3.11: Nastavení nevyžádaných zpráv.

```
// povoli unsolicited zpravy
config.outstationConfig.allowUnsolicited = true;
```



Obr. 3.6: Grafické rozhraní DNP3 klienta.

Vytvoří se statická metoda *setting()*, která bude kontrolovat, zda se nepředaly prázdné proměnné *timeMs*, *linkMaster* a *port*. Pokud řetězce nejsou prázdné, vrátí hodnotu true. Metoda se zavolá ve spouštěcí metodě *startDNP3* a bude v podmínce, která v případě true povolí metodu *run()*. Ukázka kódu ve výpisu B.1.

V poslední řadě je tu metoda *checkEndDNP3()*. V těle metody je podmínka, která čte stavovou proměnnou *quit* ze třídy *DNP3Sampler*. Pokud je proměnná ve stavu true, tak se nastaví proměnná *command* na "quit"a ukončí běžícího klienta.

3.3.2 Třída DNP3SamplerGui.java

Grafická podoba sampleru je rozdělena do několika panelů. První panel na obrázku 3.6 slouží pro zvolení síťového rozhraní viz výpis B.4. Ve druhém panelu se nastaví linková adresa Master stanice a zvolí se jestli klient bude posílat zprávy z čítače nebo generovat analogové hodnoty včetně intervalu opakování. Třetí panel slouží pro zadání rozsahu IP adres a linkových adres klientů.

V předposledním panelu se nastaví port a obsahuje akční tlačítka, která nastaví zadaný rozsah IP adres na rozhraní (tlačítko **Apply**), odstraní přiřazený adresy z rozhraní (tlačítko **Delete**) nebo vyčistí panel **Message Console**, který slouží pro výpis výstupů do konzole. Na Obr. 3.6 je vidět uspořádání jednotlivých komponentů. Všechny panely se přidají v metodě *init()*.

Na každé tlačítko včetně výběru akce (**Counter** nebo **Analog**) je vytvořen *ActionListener* a bude naslouchat na stisknutí. Vše je v metodě *init()* ve výpisu B.3.

Při stisknutí tlačítka **Apply** se zkontroluje metodou *checkEmtyFields()*, jestli nejsou prázdná pole, metodou *deletePreSetOutstations()* vyčistí nastavené předchozí klienty a zaplní pomocný LinkedList *links* pomocí metody *setLinks* linkovými adresami. Předchozí metody jsou uvedeny ve výpisu B.2. Následně se vytvoří script metodou *createIpScript( )*, které předáme minimální a maximální hodnotu IP adresy v dekadickém tvaru, masku, rozhraní, cestu, kde má vytvořit script a řetězec "add " nebo "del ". Řetězec pro přidání adres na rozhraní je "add " a odstranění adres je "del ". Ukázka ve výpisu B.6.

Dekadický tvar adres umožňují metody *getMinIPValue()* a *getMaxIPValue()* ve výpisu B.5.

Tlačítko Delete nejdříve zkontroluje, jestli jsou vyplněná pole a není prázdný LinkedList *links*. Poté vytvoří script s řetězcem "del " pro smazání IP adres, smaže nastavené klienty v paměti a spustí příkazy pro smazání všech vytvořených scriptů.

Panel, který se vytvoří v metodě *messagePanel()*, má tu vlastnost, že výpisy do konzole přesměruje do komponenty **JTextArea**. Části kódu jsou ve výpisu B.4.

Pro předání hodnot mezi grafickým rozhraním a sampleru slouží tři základní metody, které musí obsahovat třída *DNP3SamplerGui.java*. Metoda *configure(TestElement element)* slouží pro nastavení dat do grafického rozhraní. Metoda *createTestElement()* vytváří novou instanci třídy *TestElement* a spouští metodu *modifyTestElement(TestElement te)*, která převede data z grafického rozhraní do třídy *TestElement* [12]. Ukázáno ve výpisu B.2.

Poslední metodou v této třídě je *randomIp()*. Ta vrací náhodně zvolenou IP adresu z rozsahu jednotlivým vláknům, která se zadají při vytváření testovacího plánu JMeteru, jejichž počet se musí shodovat s počtem IP adres v zadaném rozsahu. Metoda je ukázána ve výpisu B.7. Tato metoda se volá ve třídě *DNP3Sampler.java*.

Tato část práce byla komplikovanější, protože se muselo zajistit to, aby každé vlákno mělo svojí IP adresu. Pokud se sešli dvě stejné adresy, psalo to chybu, že adresa je již užívána. Změna portu také nešla, jelikož server (Master) komunikuje vždy na jednom portu. Zkusilo se použít cyklus **for**, který vypisoval a spouštěl postupně podřízené stanice. Toto řešení ale nevedlo k tomu, že se stanice spustí v jednu chvíli současně. Řešením tedy byla metoda *randomIp()* s nápadem, že ke každému vytvořenému vláknu se přiřadí vždy jiná IP adresa. Mohl se místo metody *randomIp()* využít zmíněný cyklus **for**, který bude postupně přiřazovat IP ze seznamu k vláknům. Musela by se i tady ošetřit situace, aby se přiřadila vždy jiná adresa.

3.3.3 Třída DNP3Sampler.java

Poslední úpravy se provedou ve třídě *DNP3Sampler.java* v metodě *sample()*. Pomocným proměnným se přiřadí jednotlivý hodnoty z grafického rozhraní a k náhodné IP adrese se vyčte z množiny *outstations* příslušná linková adresa. Nakonec se vytvoří objekt třídy *DNP3*, předají se mu proměnné do konstruktoru a spustí se metoda *startDNP3()*. Při každém vytvoření klienta (vlákna) se díky metodě *randomIp()* vybere jiná IP adresa, která ještě nebyla zvolena. Opět ukázka ve výpisu 3.12.

Výpis 3.12: Úpravy ve třídě DNP3Sampler.java.

```
...
public SampleResult sample(Entry e) {
  ...
  res.sampleStart();
  try {
    String linkMaster = this.getPropertyAsString(LINKMASTER);
    String port = this.getPropertyAsString(PORT);
    String timeMs = this.getPropertyAsString(TIMEMS);
    String ipOutstation = DNP3SamplerGui.randomIp();
    String linkOutstation = "";
    for(Map.Entry<String, String> entry : outstations.entrySet()) {
      if(entry.getValue().equals(ipOutstation)) {
        linkOutstation = entry.getKey();
        break;
      }
    }
    DNP3 dnp3 = new DNP3(linkOutstation, ipOutstation, linkMaster,
port, timeMs);
    dnp3.startDNP3();
    ...
  }
}
...
```

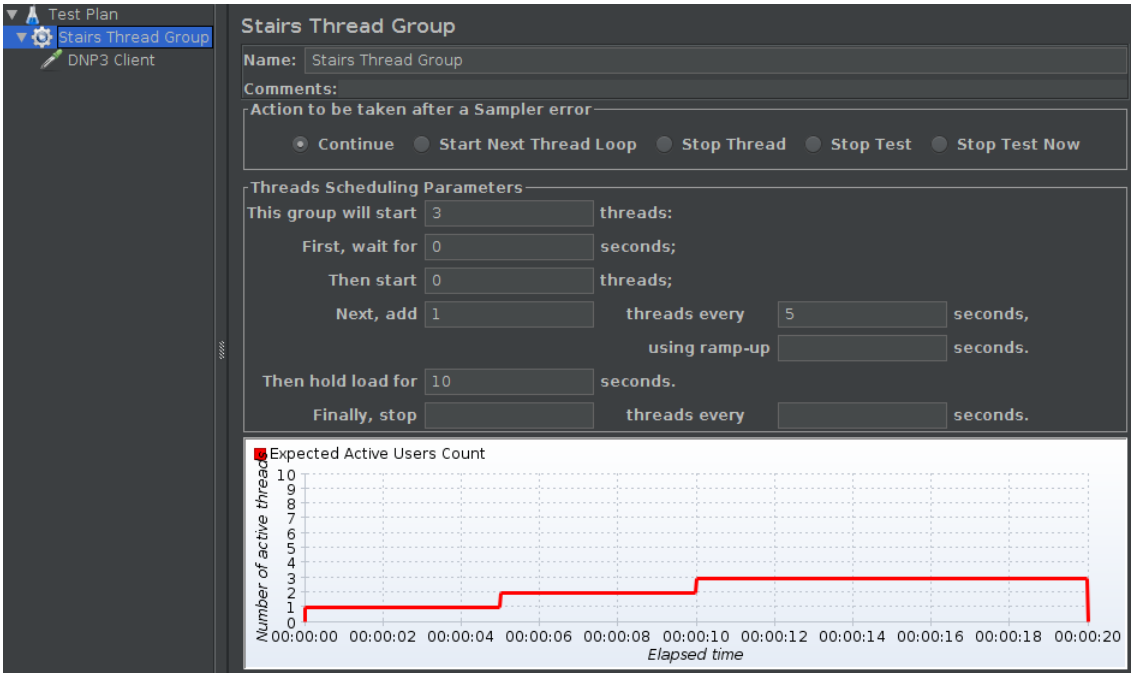
3.3.4 DNP3 Server

Aby se během vývoje modulu testovaly jednotlivé funkčnosti, musel se vytvořit server neboli Master, který bude příjímat nevyžádané zprávy. Byl vytvořen v operačním systému Windows a poté byl hotový server vyexportován jako **Runnable JAR file** a přesunut na druhý systém Debian 9, jiný než v kterém je nástroj JMeter. Jedna ze změn ve zdrojovém kódu ve třídě *DNP3.java* je, že vytváříme objekty z tříd pro Master stanici. Ukázáno ve výpisu C.1. Déle se vytvoří třída *OutstationClass.java*, která slouží pro vytvoření objektů podřízených stanic. Ve třídě *DNP3.java* je **for** cyklem vytvářeno TCP spojení pro jednotlivé podřízené stanice viz výpis C.1. Při

spuštění serveru se zadává IP adresa rozhraní, linková adresa Mastera, port, rozsah podřízených stanic a jejich linkové adresy. Spuštění serveru se provede vepsáním do konzole:
java -jar DNP3Server.jar

3.3.5 Testování implementovaného modulu

První scénář na obrázku 3.7 se provedl se třemi klienty, kteří se spouštěli po pěti vteřinách. Pro vytváření vláken se využil přidaný modul **Stairs Thread Group**. Na Obr. 3.8 je ale vidět, že první dva klienti poslali nevyžádanou zprávu. Je to tím, že každý klient, který se spustí, čeká, až se s ním server spojí. To trvá vždy jinou dobu. V tomto případě se server spojil, když už byl spuštěn i druhý klient. Tato situace je vysvětlena na obrázku 3.10, kde je zobrazen podobný jev. Nevyžádané zprávy se posílají také po pěti vteřinách. Poté co se spustí i třetí klient, posílají se tři nevyžádané zprávy najednou. Viz Obr. 3.9.



Obr. 3.7: Testovací scénář č. 1.

Obrázek 3.10 znázorňuje zpoždění v posílání nevyžádaných zpráv, jelikož server nejprve musí navázat spojení s klienty (značka A). Klient při spuštění začne naslouchat na svém rozhraní (značka L) a po dvou vteřinách, jak je na obrázku vidět, pošle nevyžádanou zprávu (značka S). V případě, že server nestihne navázat spojení, než se pošle zpráva, tak se zpráva nepošle. Ukázáno ve třetím časovém průběhu (.102 - A).

	Time	Source	Destination	Protocol	Length	Info
78	*REF*	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
80	0.003789446	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
82	0.165746005	192.168.1.102	192.168.1.200	DNP 3.0	97	Unsolicited Response
84	0.171138020	192.168.1.200	192.168.1.102	DNP 3.0	81	Confirm
86	4.985495045	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
87	4.990011036	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
89	5.154878911	192.168.1.102	192.168.1.200	DNP 3.0	97	Unsolicited Response
90	5.156765419	192.168.1.200	192.168.1.102	DNP 3.0	81	Confirm

Obr. 3.8: DNP3 Pokus č. 1 - Obr. 1.

	Time	Source	Destination	Protocol	Length	Info
124	*REF*	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
125	0.004030119	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
127	0.168625422	192.168.1.102	192.168.1.200	DNP 3.0	97	Unsolicited Response
128	0.170385053	192.168.1.200	192.168.1.102	DNP 3.0	81	Confirm
130	1.168230659	192.168.1.101	192.168.1.200	DNP 3.0	97	Unsolicited Response
131	1.169812938	192.168.1.200	192.168.1.101	DNP 3.0	81	Confirm
133	5.000313969	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
134	5.001797245	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
136	5.169761830	192.168.1.102	192.168.1.200	DNP 3.0	97	Unsolicited Response
137	5.172618955	192.168.1.200	192.168.1.102	DNP 3.0	81	Confirm
139	6.169421500	192.168.1.101	192.168.1.200	DNP 3.0	97	Unsolicited Response
140	6.172298320	192.168.1.200	192.168.1.101	DNP 3.0	81	Confirm

Obr. 3.9: DNP3 Pokus č. 1 - Obr. 2.

Druhý scénář na Obr. 3.11 byl s dvaceti klienty, kteří se po deseti klientech spouštěli po třiceti vteřinách. Na obrázku 3.12 je vidět prvních 10 klientů v časovém rozmezí přibližně 100ms, kteří poslali zprávu v prvních třiceti sekundách. Po necelé minutě se poslaly zprávy od všech klientů. Na druhém obrázku 3.13 je jich celkem 18 a to proto, že u dvou klientů server nedokázal navázat spojení, i když se o to několikrát pokoušel. Ukázáno ve výpisu 3.13.

Výpis 3.13: Snaha serveru o navázání spojení.

```
1557836850018 - INFO - client - Connecting to: 192.168.1.101
1557836850018 - WARN - client - Error Connecting: Spojení odmítnuto
1557836850149 - INFO - client - Connecting to: 192.168.1.118
1557836850150 - WARN - client - Error Connecting: Spojení odmítnuto
```

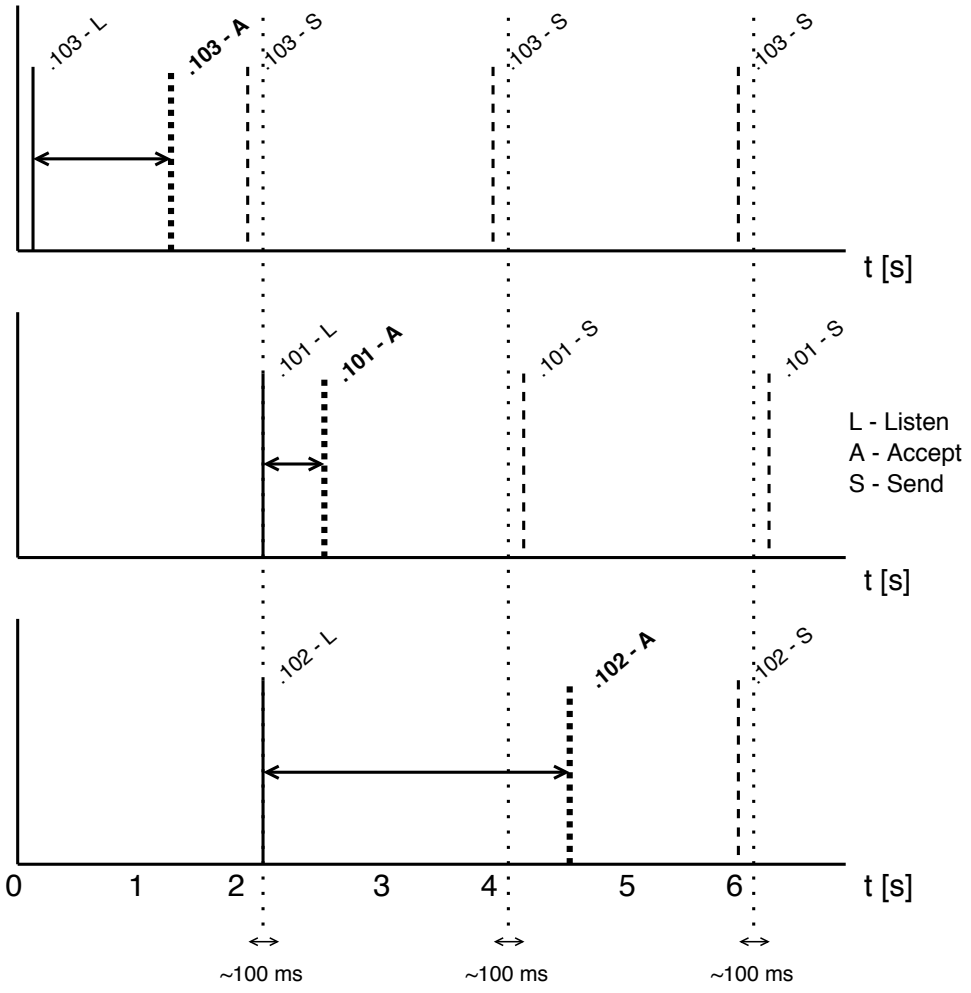
3.4 Vlastní návrh a implementace druhého protokolu

Další kapitola v praktické části se zabývá návrhu druhého protokolu. Jako druhý protokol byl zvolen IEC 61850, u kterého byl problém sestavit knihovnu. Dále tu byl pokus o získání knihovny protokolu TASE.2, ale neúspěšně. Tento protokol je definován ve standardu IEC 60870-6.

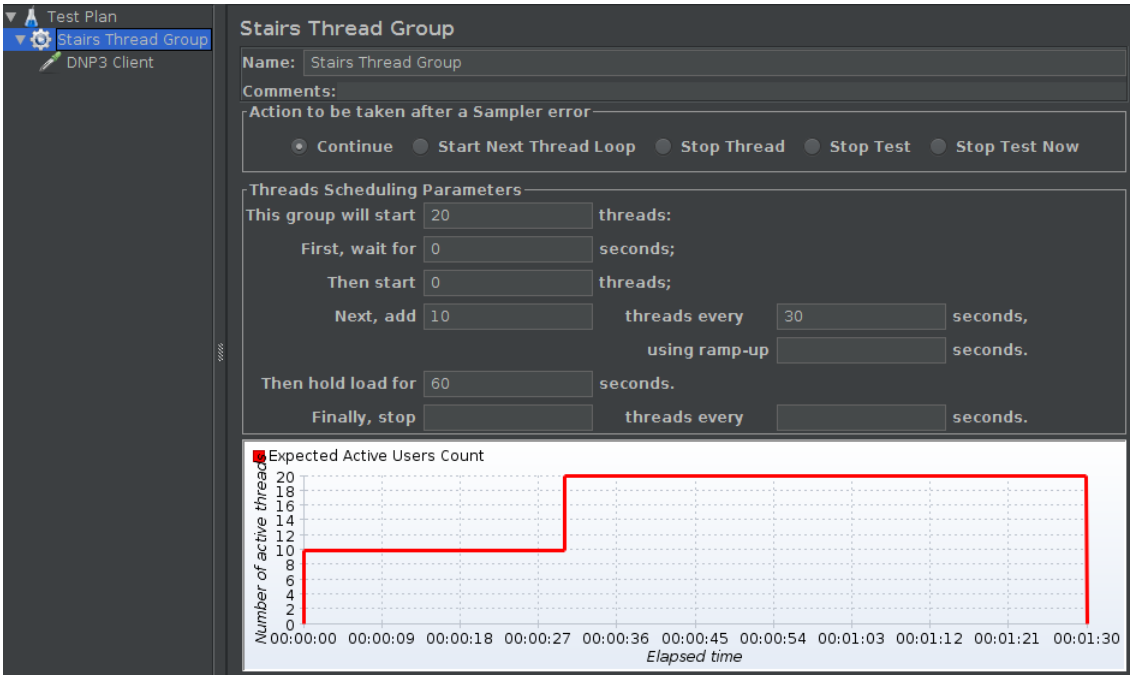
3.4.1 OpenIEC61850

OpenIEC61850 je implementace standardu IEC 61850 s otevřeným zdrojovým kódem napsané v Javě. Podobná a lepší implementace standardu je i v jazyce C pod názvem **libIEC61850** [16]. V této práci se stáhla knihovna verze 1.7.0 ze stránek [17]. Pro sestavení této knihovny je využit nástroj **Gradle**, který právě vypisoval chyby při sestavování. Pro sestavení je potřeva mít stažený nástroj Gradle a poté v adresáři knihovny zadat:
gradle build
Po dlouhém pátrání byla chyba odstraněna a knihovna sestavena. Problém byl v absenci souboru *tools.jar* v používaných verzí. Řešení bylo následující:

- hledání správné verze Javy (funkční verze JDK 1.8.0 201),
- změnit cestu k funkční verzi Javy, jelikož Eclipse používal jinou verzi (JDK 10.0.2).



Obr. 3.10: Potvrzení spojení se serverem.



Obr. 3.11: Testovací scénář č. 2.

	Time	Source	Destination	Protocol	Length	Info
359	28.929189069	192.168.1.119	192.168.1.200	DNP 3.0	97	Unsolicited Response
361	28.930722389	192.168.1.200	192.168.1.119	DNP 3.0	81	Confirm
363	28.936621637	192.168.1.112	192.168.1.200	DNP 3.0	97	Unsolicited Response
365	28.937803267	192.168.1.200	192.168.1.112	DNP 3.0	81	Confirm
366	28.962606319	192.168.1.104	192.168.1.200	DNP 3.0	97	Unsolicited Response
368	28.963818318	192.168.1.200	192.168.1.104	DNP 3.0	81	Confirm
370	28.967501717	192.168.1.115	192.168.1.200	DNP 3.0	97	Unsolicited Response
372	28.970085807	192.168.1.200	192.168.1.115	DNP 3.0	81	Confirm
374	28.970274907	192.168.1.107	192.168.1.200	DNP 3.0	97	Unsolicited Response
376	28.970478717	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
378	28.971395558	192.168.1.200	192.168.1.107	DNP 3.0	81	Confirm
380	28.971881666	192.168.1.120	192.168.1.200	DNP 3.0	97	Unsolicited Response
381	28.972077512	192.168.1.117	192.168.1.200	DNP 3.0	97	Unsolicited Response
382	28.972463499	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
384	28.973819790	192.168.1.200	192.168.1.120	DNP 3.0	81	Confirm
386	28.975572406	192.168.1.200	192.168.1.117	DNP 3.0	81	Confirm
389	29.033516850	192.168.1.109	192.168.1.200	DNP 3.0	97	Unsolicited Response
391	29.034877470	192.168.1.200	192.168.1.109	DNP 3.0	81	Confirm
393	29.036482783	192.168.1.116	192.168.1.200	DNP 3.0	97	Unsolicited Response
395	29.040268366	192.168.1.200	192.168.1.116	DNP 3.0	81	Confirm

Obr. 3.12: DNP3 Pokus č. 2 - Obr. 1.

	Time	Source	Destination	Protocol	Length	Info
610	58.827119348	192.168.1.111	192.168.1.200	DNP 3.0	97	Unsolicited Response
612	58.828381343	192.168.1.200	192.168.1.111	DNP 3.0	81	Confirm
613	58.829584316	192.168.1.108	192.168.1.200	DNP 3.0	97	Unsolicited Response
615	58.830708408	192.168.1.200	192.168.1.108	DNP 3.0	81	Confirm
616	58.831977792	192.168.1.110	192.168.1.200	DNP 3.0	97	Unsolicited Response
617	58.834899848	192.168.1.200	192.168.1.110	DNP 3.0	81	Confirm
619	58.841057222	192.168.1.114	192.168.1.200	DNP 3.0	97	Unsolicited Response
621	58.841731654	192.168.1.102	192.168.1.200	DNP 3.0	97	Unsolicited Response
622	58.841878766	192.168.1.105	192.168.1.200	DNP 3.0	97	Unsolicited Response
625	58.842112623	192.168.1.113	192.168.1.200	DNP 3.0	97	Unsolicited Response
626	58.842220232	192.168.1.106	192.168.1.200	DNP 3.0	97	Unsolicited Response
628	58.845755625	192.168.1.200	192.168.1.114	DNP 3.0	81	Confirm
629	58.846850987	192.168.1.200	192.168.1.102	DNP 3.0	81	Confirm
630	58.847857340	192.168.1.200	192.168.1.105	DNP 3.0	81	Confirm
631	58.848979196	192.168.1.200	192.168.1.113	DNP 3.0	81	Confirm
633	58.850253180	192.168.1.200	192.168.1.106	DNP 3.0	81	Confirm
640	58.923610468	192.168.1.103	192.168.1.200	DNP 3.0	97	Unsolicited Response
641	58.925452581	192.168.1.200	192.168.1.103	DNP 3.0	81	Confirm
643	58.930191127	192.168.1.119	192.168.1.200	DNP 3.0	97	Unsolicited Response
644	58.932388278	192.168.1.117	192.168.1.200	DNP 3.0	97	Unsolicited Response
645	58.937982418	192.168.1.115	192.168.1.200	DNP 3.0	97	Unsolicited Response
646	58.939959046	192.168.1.107	192.168.1.200	DNP 3.0	97	Unsolicited Response
647	58.940112958	192.168.1.200	192.168.1.119	DNP 3.0	81	Confirm
649	58.942075307	192.168.1.112	192.168.1.200	DNP 3.0	97	Unsolicited Response
650	58.943846647	192.168.1.200	192.168.1.117	DNP 3.0	81	Confirm
652	58.944558033	192.168.1.116	192.168.1.200	DNP 3.0	97	Unsolicited Response
653	58.946640104	192.168.1.200	192.168.1.115	DNP 3.0	81	Confirm
655	58.948064644	192.168.1.120	192.168.1.200	DNP 3.0	97	Unsolicited Response
656	58.949658680	192.168.1.200	192.168.1.107	DNP 3.0	81	Confirm
658	58.951177051	192.168.1.104	192.168.1.200	DNP 3.0	97	Unsolicited Response
659	58.956340445	192.168.1.200	192.168.1.112	DNP 3.0	81	Confirm
661	58.957116255	192.168.1.200	192.168.1.116	DNP 3.0	81	Confirm
663	58.964619232	192.168.1.200	192.168.1.120	DNP 3.0	81	Confirm
665	58.965410048	192.168.1.200	192.168.1.104	DNP 3.0	81	Confirm
667	59.024677218	192.168.1.109	192.168.1.200	DNP 3.0	97	Unsolicited Response
668	59.026381295	192.168.1.200	192.168.1.109	DNP 3.0	81	Confirm

Obr. 3.13: DNP3 Pokus č. 2 - Obr. 2.

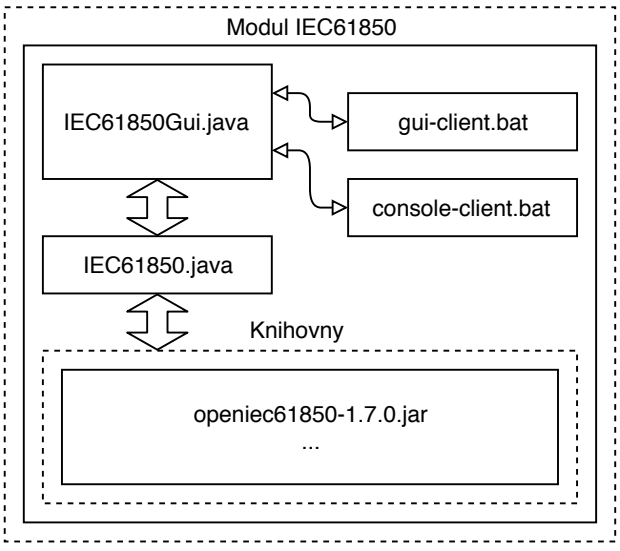
V adresáři **/openiec61850/run-scripts/gui-client** se spustí soubor *openiec61850-gui-client.bat* a tím se zobrazí klient v grafickém režimu. Ukázka klienta na Obr. 3.15.

Následný návrh ukázán na Obr. 3.14 a implementace modulu pro protokol IEC61850 by byl obdobný jako u DNP3 protokolu. Z časových důvodů již nebyl realizován a to zejména z důvodů dlouhého rozjíždění knihovny **openIEC61850**. Je možné zhrnout, že nyní po úspěšném zprovoznění knihovny by implementace modulu neměla být problematická. Bude využívat externí knihovnu přímo v modulu a nebo se teoreticky může externě rovnou spouštět. Na obrázku 3.16 je vidět částečná implementace modulu, který spouští externě klienta protokolu IEC61850.

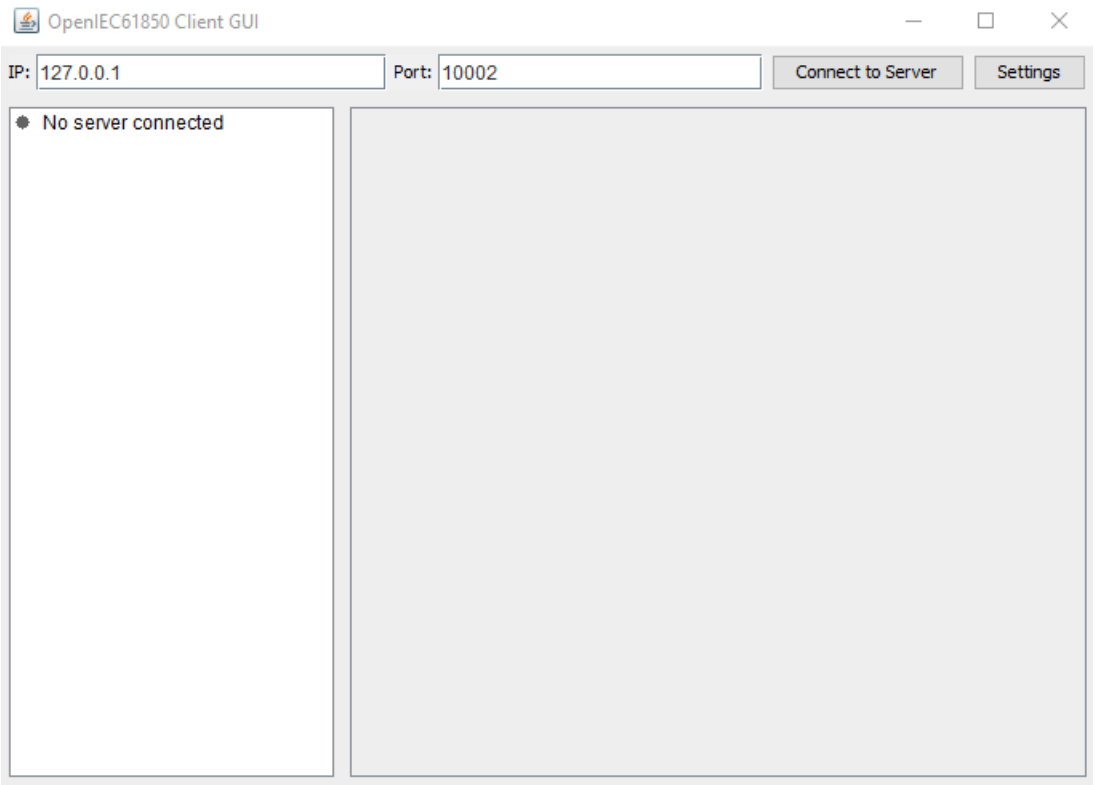
3.4.2 TASE.2

Z důvodu výše popsaného problému byla jako alternativní varianta zkoumán protokol ICCP (Inter Control Center Protocol) s oficiálním názvem TASE.2. Je velice podobný ke standardu IEC 61850, ale používá jinou sadu funkcí MMS a jiný datový model. Knihovna je k dispozici pro C/C++, .NET a Java [18].

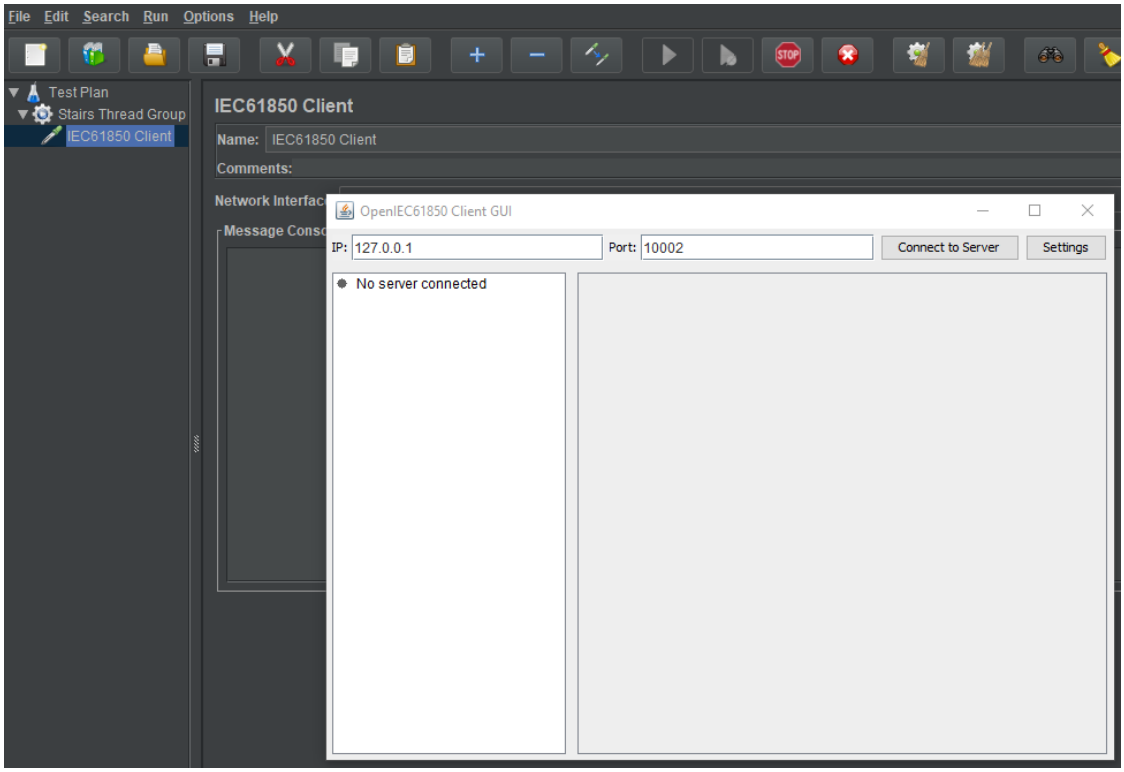
Pro získání této knihovny bylo potřeba zažádat [18]. V odpovědi stálo, že tato knihovna není momentálně k dispozici zdarma, ale lze si ji koupit buď na jeden rok nebo na dobu neurčitou za vyšší cenu.



Obr. 3.14: Blokové schéma návrhu IEC61850 klienta.



Obr. 3.15: Grafický klient protokolu IEC 61850.



Obr. 3.16: Částečná implementace IEC61850 protokolu.

4 Závěr

V teoretické části je popsán systém SCADA, zvolený DNP3 protokol pro řešení bakalářské práce, který je jeden z nejpoužívanějších v monitorovacích zařízeních, druhý zvolený IEC61850 a nakonec třetí protokol IEC60870-5, který do praktické části nebyl vybrán. Protokol DNP3 je vysvětlen, jak pracuje v jedotlivých vrsvách ve tří vrstvém modelu EPA.

V další kapitole v teoretické části je popsán nástroj JMeter, který dokáže simulovat zátěžové testování a analyzovat výsledky. Zde jsou vypsané základní funkce, které nástroj obsahuje.

Praktická část práce je v kapitole tři. Nejprve se popisuje samotná instalace nástroje do vývojového prostředí a tvorba ukázkového nového modulu. Dále je řešeno sestavení programu pro protokol DNP3 pomocí OpenDNP3 repozitáře. Odzkoušená komunikace proběhla v pořádku mezi stanicemi Windows a Debian. Hlavním cílem byl vlastní návrh a implementace modulu pro DNP3 protokol jež je popsán v podkapitole 3.3. Nástroj JMeter byl přesunut na novou stanici Debian Client a byl vytvořen DNP3Server, který se nacházel na druhé stanici Debian a navazoval spojení s podřízenými stanicemi vytvořenými modulem. Po otestování komunikace mezi přídavným modulem nástroje JMeter na Master stanici (DNP3Server), bylo zjištěno, že při ukončení spojení, po zavolání metody *manager.shutdown()*, se z neznámého důvodu neukončí vlákno stanice Master, které bylo vytvořeno objektem *DNP3manager*, a Master se neukončí. Dále bylo zjištěno, že dokud stanice Master nenaváže komunikaci, vypnutí hlavní stanice proběhne bez problému. Zásuvný modul byl tak úspěšně implementován a otestován.

Dalším cílem byl návrh a implementace modulu pro protokol IEC61850 a pomocí repozitáře OpenIEC61850 byly sestaveny knihovny, které přes script spustily grafický režim klienta. Při sestavování a zprovozňování knihovny došlo k několika problémům, které bylo nutné vyřešit a zabraly velké množství času. Konkrétní metody eliminace chyb jsou popsány v podkapitole 3.4.1. Nakonec sestavení proběhlo úspěšně, ale bohužel v rámci semestru již nebylo možné finální implementaci dotáhnout do konce. Návrh modulu by byl obdobný jako u protokolu DNP3 a neměl by být problematicky. Návrh modulu byl realizován a dílčí části zprovozněny, částečně implementovány a vyzkoušeny.

Také tu byl pokus o získání druhého náhradního protokolu TASE.2 za protokol IEC61850. TASE.2 je velice podobný ke standardu IEC61850 a je definován ve standardu IEC 60870-6. Jelikož se nejednalo o knihovnu přístupnou zdarma, tak nebyla získána.

I přes výše popsané problémy s implementací druhého modulu lze konstatovat, že většina definovaných cílů bakalářské práce byla splněna.

Literatura

[1] DSPTelecom, *Remote Monitoring & Control* [online]. 2018, [cit. 9. 11. 2018]. Dostupné z URL: <https://www.dpstele.com/index.php>.

[2] DNP3.org, *DNP3 - Distributed Network Protocol* [online]. 2011, [cit. 9. 11. 2018]. Dostupné z URL: <https://www.dnp.org/Default.aspx>. <https://www.dnp.org/AboutUs/DNP3%20Primer%20Rev%20A.pdf>

[3] IEEE, *1815-2012 - IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3).* 10 Oct. 2012, [cit. 9. 11. 2018]. ISBN 978-0-7381-8829-4.

[4] ABB, *DNP3 Communication Protocol Manual* [online]. May 2014, [cit. 14. 11. 2018]. Document ID: 1MRK 511 301-UUS. Dostupné z URL: <https://library.e.abb.com/public/27dc10ac4ab57d35c1257d940037f7c8/1MRK511301-UUS_-_en_Communication_protocol_manual__DNP__670_series_2.0.pdf>.

[5] FORGUE, Bruno a VLADYKA, Pavel, *IEC 61850: soubor norem pro komunikaci v energetice s velkým potenciálem výhod. Automa* [online]. 3/2010, [cit. 2. 3. 2019]. Dostupné z URL: <http://automa.cz/Aton/FileRepository/pdf_articles/40771.pdf>.

[6] ŠIKULA, J. *Převod standardu IEC 61850 na komunikační protokol ModBus.* [online]. Bakalářská práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. 2016, [cit. 2. 3. 2019]. Dostupné z URL: <https://dspace.vutbr.cz/xmlui/handle/11012/61723>.

[7] STODŮLKA, I. *Model elektrické stanice s komunikačním protokolem IEC 61850.* Diplomová práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. 2012, 97s. [cit. 2. 3. 2019].

[8] MATOUŠEK, P. *Description and analysis of IEC 104 Protocol.* Brno: Vysoké učení technické v Brně, Fakulta informačních technologií. 12/2007, [cit. 11. 5. 2019]. Dostupné z URL: <http://www.fit.vutbr.cz/research/pubs/tr.php?file=%2Fpub%2F11570%2FTR-IEC104.pdf&id=11570>.

[9] PEKÁREK, D. *Popis a testování komunikačních protokolů normy IEC 60870-5-103 a 60870-5-104.* Diplomová práce, Brno: Ústav elektroenergetiky FEKT VUT v Brně. 2017, 72s. [cit. 11. 5. 2019]. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=147028>.

[10] RUDZINSKI, Y., VLADYKA, P. *Komunikační protokoly pro dálkové ovládání IEC/ISO 60870-5. Automa* [online]. 2010, [cit. 11. 5. 2019]. Dostupné z URL: <http://automa.cz/cz/casopis-clanky/komunikacni-protokoly-pro-dalkove-ovladani-iec/iso-60870-5-2010_02_40552_5799/>.

[11] VODEHNAL, S. *Identifikace dostupnosti zařízení v technologických sítích.* Diplomová práce, Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. 2018, 79s. [cit. 11. 5. 2019]. Dostupné z URL: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=171175>.

[12] Apache JMeter, *Apache JMeter™* [online]. 2018, [cit. 19. 10. 2018]. Dostupné z URL: <http://jmeter.apache.org/>.

[13] ŠVEHLÁK, M. *Rozšíření nástroje JMeter* [online]. Brno: Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. 2017, [cit. 19. 10. 2018]. Dostupné z URL: <https://dspace.vutbr.cz/bitstream/handle/11012/65888/Rozsireni_Nastroje_JMeter_xsvehl04.pdf>

[14] Automatak, *OpenDNP3* [online]. 2015, [cit. 2. 11. 2018]. Dostupné z URL: <https://www.automatak.com/opendnp3/docs/guide/current/>.

[15] Oracle, *Java™ Documentation - javadoc* [online]. 2018, [cit. 3. 11. 2018]. Dostupné z URL: <https://docs.oracle.com/javase/8/docs/technotes/tools/unix/javadoc.html>.

[16] OpenMUC, *OpenIEC61850* [online]. 2019, [cit. 18. 5. 2019]. Dostupné z URL: <https://www.openmuc.org/iec-61850/>.

[17] Beanit, *OpenIEC61850* [online]. 2019, [cit. 18. 5. 2019]. Dostupné z URL: <https://www.beanit.com>.

[18] MZ Automation, *ICCP/TASE.2 IEC 60870-6* [online]. 2019, [cit. 18. 5. 2019]. Dostupné z URL: <https://www.mz-automation.de/communication-protocols/iccp-tase-2-iec-60870-6-protocol-library/>.

Seznam symbolů, veličin a zkratek

SCADA	Supervisory Control And Data Acquisition
HMI	Human-Machine Interface
DNP3	Distributed Network Protocol
RTU	Remote terminal unit
IED	Intelligent Electronic Devices
IEC	International Electrotechnical Commission
EPA	Enhanced Performance Architecture
LPHD	Physical Device information
LLN0	Logical Node zero
LN	Logical Node
MMS	Manufacturing Message Specification
FTP	File Transfer Protocol
GSE	Generic Substation Event - vše obecná událost rozvodny
GOOSE	Generic Object Oriented Substation Events - objektově orientovaná událost rozvodny
GSSE	Generic Substation State Events - generická stavová událost rozvodny
ASDU	Application Service Data Unite - Jednotka dat aplikační služby
APCI	Aplication Protocol Control Information - Jednotka určující informace aplikační jednotky
APDU	Application Protocol data Unit
PDU	Protocol Data Unit
HTML	Hypertext Markup Languagel
JVM	Java Virtual Machine
ICCP	Inter Control Center Protocol

Seznam příloh

A Výpisy souboru build.xml

Výpis A.1: Nastavení funkce pro kompilaci.

```
...
<target name="compile-prvniModul" depends="compile-jorphan,
compile-core,compile-components"
  description="
Compile␣components␣specific␣to␣PrvniModul␣sampling.">
  <mkdir dir="${build.prvniModul}"/>
  <javac srcdir="${src.prvniModul}" destdir=
  "${build.prvniModul}" source="${src.java.version}"
  optimize="${optimize}" debug="on" target=
  "${target.java.version}"
  includeAntRuntime="${includeAntRuntime}" deprecation=
  "${deprecation}" encoding="${encoding}">
    <include name="**/*.java"/>
    <classpath>
      <pathelement location="${build.jorphan}"/>
      <pathelement location="${build.core}"/>
      <path refid="classpath"/>
    </classpath>
  </javac>
</target>
<target name="compile-protocols" depends="compile-http,
compile-ftp,compile-jdbc,compile-java,compile-ldap,
compile-mail,compile-tcp,compile-prvniModul"
  description="Compile␣all␣protocol-specific␣components."/>
...
```

Výpis A.2: Nastavení spustitelného .jar souboru.

```
...
<!-- JMeter launch jar -->
...
<!-- DNP3 -->
<jar jarfile="${dest.jar}/DNP3.jar" manifest="${build.dir}
/MANIFEST_BIN.MF">
  <zipfileset file="${resources.meta-inf}/default.notice"
    fullpath="META-INF/NOTICE" />
  <zipfileset file="${resources.meta-inf}/default.license"
    fullpath="META-INF/LICENSE" />
  <fileset dir="${build.DNP3}" includes="**/*.class" />
  <fileset dir="${src.DNP3}" includes="**/*.properties" />
</jar>
...
```

B Výpisy tříd modulu DNP3 Client

Výpis B.1: Úpravy ve třídě DNP3.java.

```
public class DNP3 {
  ...
  // Konstruktor tridy a definice promenne manager
  public DNP3(String link, String ip, String linkMaster,
String port, String time) {
    DNP3.linkMaster = linkMaster;
    DNP3.ipOutstation = ip;
    DNP3.port = port;
    DNP3.linkOutstation = link;
    DNP3.timeMs = time;
    try {
      this.manager = DNP3ManagerFactory.createManager(1,
PrintingLogHandler.getInstance());
    } catch (DNP3Exception e) {...};
  }
  public static void setCommand(String cmd) {...}
  public static String getCommand() {...}

  public void startDNP3() throws Exception {
    ...
      if(setting())
        run(this.manager);
      else return;
    ...
  }

  static void run(DNP3Manager manager) throws Exception {
    ...
    // Create a tcp channel class that will connect to the loopback
    Channel channel = manager.addTCPServer(
      "client",
      LogMasks.NORMAL | LogMasks.APP_COMMS,
      ServerAcceptMode.CloseExisting,
      ipOutstation,
      portTemp,
      PrintingChannelListener.getInstance()
    );
    ...
    config.linkConfig.localAddr = linkOutstationTemp;
    config.linkConfig.remoteAddr = linkMasterTemp;
    // povoli unsolicited zpravy
    config.outstationConfig.allowUnsolicited = true;
    ...
```

```
    long i = 0;
    while (true) {
      if(getCommand() != null) {
        switch(getCommand()) {
        case("quit"):
          ...
        case("analog"):
          OutstationChangeSet setA = new OutstationChangeSet();
          Random r = new Random(i);
          int min = 40;
          int max = 70;
          double a = r.nextInt((max - min) + 1) + min;
          setA.update(new AnalogInput(a, (byte)0x01, 0), 0);
          outstation.apply(setA);
          TimeUnit.MILLISECONDS.sleep(TimeTmp);
          i++;
          break;
        case("counter"):
          OutstationChangeSet setC = new OutstationChangeSet();
          setC.update(new Counter(i,(byte) 0x01, 0), 0);
          outstation.apply(setC);
          TimeUnit.MILLISECONDS.sleep(TimeTmp);
          i++;
          break;
        default:
          System.out.println("Unknown␣command:␣" + getCommand());
          setCommand(null);
          break;
        }
      }
    }
  }
  public static boolean setting() {
    if (linkMaster.equals("") || port.equals("") ||
timeMs.equals("")) {
      System.out.println("Zadejte␣vsechny␣parametry.");
      return false;
      }
    return true;
  }
  protected static void checkEndDNP3() {
    if (DNP3Sampler.quit) {
      setCommand("quit");
    }
  }
}
```

Výpis B.2: Úpravy ve třídě DNP3SamplerGui.java.

```
...
public DNP3SamplerGui() {
  init();
  defaultSetting();
  registerTestStateListener();
}
...
public void configure(TestElement element) {
  super.configure(element);
  selectInt.setSelectedItem(element.getPropertyAsString(
DNP3Sampler.INTERFACE));

  timeMs.setText(element.getPropertyAsString(DNP3Sampler.TIMEMS));
  linkMaster.setText(element.getPropertyAsString(
DNP3Sampler.LINKMASTER));
  minIpOutstation.setText(element.getPropertyAsString(
DNP3Sampler.MINIPOUTSTATION));
  maxIpOutstation.setText(element.getPropertyAsString(
DNP3Sampler.MAXIPOUTSTATION));
  minLinkOutstation.setText(element.getPropertyAsString(
DNP3Sampler.MINLINKOUTSTATION));
  maxLinkOutstation.setText(element.getPropertyAsString(
DNP3Sampler.MAXLINKOUTSTATION));
  mask.setText(element.getPropertyAsString(DNP3Sampler.MASK));
  port.setText(element.getPropertyAsString(DNP3Sampler.PORT));
}
...
public TestElement createTestElement() {
  DNP3Sampler sampler = new DNP3Sampler();
  modifyTestElement(sampler);
  return sampler;
}
...
public void modifyTestElement(TestElement te) {
  te.clear();
  configureTestElement(te);

  Object selectedDev = selectInt.getSelectedItem();
  if (selectedDev != null) {  // pokud se nenajdou zadna rozhrani
    te.setProperty(DNP3Sampler.INTERFACE, selectedDev.toString());
  }
  te.setProperty(DNP3Sampler.TIMEMS, timeMs.getText());
  te.setProperty(DNP3Sampler.MINIPOUTSTATION,
minIpOutstation.getText());
  te.setProperty(DNP3Sampler.MAXIPOUTSTATION,
maxIpOutstation.getText());
```

```
  te.setProperty(DNP3Sampler.MINLINKOUTSTATION,
minLinkOutstation.getText());
  te.setProperty(DNP3Sampler.MAXLINKOUTSTATION,
maxLinkOutstation.getText());
  te.setProperty(DNP3Sampler.MASK, mask.getText());
  te.setProperty(DNP3Sampler.LINKMASTER, linkMaster.getText());
  te.setProperty(DNP3Sampler.PORT, port.getText());
}
...
private void defaultSetting() {
  minIpOutstation.setText("192.168.1.101");
  maxIpOutstation.setText("192.168.1.101");
  minLinkOutstation.setText("10");
  maxLinkOutstation.setText("10");
  linkMaster.setText("1");
  port.setText("20000");
  mask.setText("24");
  timeMs.setText("2000");
  analog.setSelected(true);
}
private void deletePreSetOutstations() {
  if(!links.isEmpty()) {
    links.clear();
  }
  if(!DNP3Sampler.outstations.isEmpty()) {
    DNP3Sampler.outstations.clear();
  }
}
private boolean checkEmtyFields() {
  if (minIpOutstation.getText().isEmpty() ||
maxIpOutstation.getText().isEmpty() ||
      minLinkOutstation.getText().isEmpty() ||
maxLinkOutstation.getText().isEmpty() || mask.getText().isEmpty())
{
    System.out.println("Empty␣fields!");
    return true;
  }
  else return false;
}
private void setLinks(String minLink, String maxLink) {
  int min = Integer.parseInt(minLink);
  int max = Integer.parseInt(maxLink);
  for (int i = min; i < max+1; i++) {
    links.add(String.format("%d", i));
  }
}
private void registerTestStateListener() {
```

```
  TestStateListener testStateListener = new TestStateListener() {
    ...
    public void testStarted() {
      // co se provede po startu testu
      if (counter.isSelected()) {
        DNP3.setCommand("counter");
      }
      if (analog.isSelected()) {
        DNP3.setCommand("analog");
      }
    }
    ...
    public void testEnded() {
      // co se provede po ukonceni testu
      outsIpTaken.clear();
      registerTestStateListener();
    }
  };
  StandardJMeterEngine.register(testStateListener);
}
...
```

Výpis B.3: Metoda init().

```
private void init() {
  ...
  counter.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      analog.setSelected(false);
    }
  });
  analog.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      counter.setSelected(false);
    }
  });
  applyBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      if(checkEmtyFields()) return;
      deletePreSetOutstations();  // vymaze predesle stanice
      // nastavi rozsah linkovych adres
      setLinks(minLinkOutstation.getText(), maxLinkOutstation.
getText());
      // cesta, kam se ma ulozit skript pro pridani adres
      String pathAdd = "/tmp/"+timestamp+"IpAddrAdd.sh";
      // vytvor script IpAddrAdd.sh
      createIpScript(getMinIPValue(minIpOutstation.getText()),
```

```
getMaxIPValue(maxIpOutstation.getText()), mask.getText(),
selectInt.getSelectedItem().toString(), pathAdd, "add ");
      // prikaz ke spusteni scriptu IpAddrAdd.sh
      String[] args1 = new String[] {"/bin/bash","-c",
"pkexec bash "+pathAdd};
      try {
        // spusteni procesu
        Process proc = new ProcessBuilder(args1).start();
        // pockani az se vykona
        proc.waitFor();
        // zobrazeni dialogoveho okna
        JOptionPane.showMessageDialog(new JFrame(),
"Ip adresses added.");
      } catch (Exception ex) {
        System.out.println(ex);
      }
    }
  });
  deleteBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      if(checkEmtyFields()) return;
      if(links.isEmpty()) return;
      String pathAdd = "/tmp/"+timestamp+"IpAddrAdd.sh";
      String pathDel = "/tmp/"+timestamp+"IpAddrDel.sh";

      createIpScript(getMinIPValue(minIpOutstation.getText()),
getMaxIPValue(maxIpOutstation.getText()), mask.getText(),
selectInt.getSelectedItem().toString(), pathDel, "del ");

      deletePreSetOutstations();
      String[] args2 = new String[] {"/bin/bash","-c",
"pkexec bash "+pathDel};
      // prikaz pro vymazani skriptu
      String[] args3 = new String[] {"/bin/bash","-c",
"pkexec rm "+pathDel+" "+pathAdd};

      try {
        // spusti IpAddrDel.sh
        Process proc2 = new ProcessBuilder(args2).start();
        proc2.waitFor();
        // vymaze oba skripty
        Process proc3 = new ProcessBuilder(args3).start();
        proc3.waitFor();
        JOptionPane.showMessageDialog(new JFrame(),
"Ip adresses deleted.");
      } catch (Exception ex2) {
        System.out.println(ex2);
```

```
      }
    }
  });
  clearBtn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      messageArea.setText(null);
    }
  });
}
```

Výpis B.4: Panel síťového rozhraní a Message Console.

```
private JPanel interfPanel() {
  JLabel label = new JLabel("Network␣Interface"); //$NON-NLS-1$
  label.setLabelFor(selectInt);

  try {
    nets = NetworkInterface.getNetworkInterfaces();
  } catch (SocketException e) {
    e.printStackTrace();
  }
  // odfiltrovani neaktivnich rozhrani a loopbacku
  for (NetworkInterface netint : Collections.list(nets)) {
      try {
      if (!netint.isUp()) continue;
      if (netint.isLoopback()) continue;
    } catch (SocketException e1) {
      e1.printStackTrace();
    }
    Enumeration<InetAddress> inetAddresses = netint.
getInetAddresses();
    // vlozeni do LinkedListu devs adresy IPv4
    for (InetAddress ip4 : Collections.list(inetAddresses)) {
      if (ip4 instanceof Inet6Address) continue;
      devs.put(netint.getDisplayName(), ip4.getHostAddress());
    }
  }
  // naplneni adres do rolovaciho vyberu
  if (devs.size() > selectInt.getItemCount()) {
    selectInt.setModel(new DefaultComboBoxModel<String>(
devs.keySet().toArray(new String[devs.size()])));
  }
  JPanel panel = new JPanel(new BorderLayout(5, 0));
  panel.add(label, BorderLayout.WEST);
  panel.add(selectInt, BorderLayout.CENTER);
  return panel;
}
```

```
...
private JPanel messagePanel() {
  JPanel panel = new JPanel();
  messageArea = new JTextArea(18, 75);
  JScrollPane scrollPane = new JScrollPane(messageArea);
  DefaultCaret caret = (DefaultCaret)messageArea.getCaret();
  caret.setUpdatePolicy(DefaultCaret.ALWAYS_UPDATE);
  panel.setBorder(new TitledBorder("Message␣Console"));
  panel.add(scrollPane, BorderLayout.NORTH);
  PrintStream printStream = new PrintStream(new OutputStream() {
    public void write(int b) throws IOException {
      messageArea.append(String.valueOf((char) b));
    }
  });
  System.setOut(printStream);
  System.setErr(printStream);
  return panel;
}
```

Výpis B.5: Metody pro získání IP adresy v decimálním tvaru.

```
private static long getMinIPValue(String minIP) {
  if (minIpOutstation.getText().isEmpty()) {
    return 0;
  }

  String[] parsedMin = minIP.split("\\.");
  int min1 = Integer.parseInt(parsedMin[0]);
  int min2 = Integer.parseInt(parsedMin[1]);
  int min3 = Integer.parseInt(parsedMin[2]);
  int min4 = Integer.parseInt(parsedMin[3]);

  String binaryMin1 = "00000000"+Integer.toBinaryString(min1);
  int diff1 = 8 - Integer.toBinaryString(min1).length();
  binaryMin1 = binaryMin1.substring(8-diff1);

  String binaryMin2 = "00000000"+Integer.toBinaryString(min2);
  int diff2 = 8 - Integer.toBinaryString(min2).length();
  binaryMin2 = binaryMin2.substring(8-diff2);

  String binaryMin3 = "00000000"+Integer.toBinaryString(min3);
  int diff3 = 8 - Integer.toBinaryString(min3).length();
  binaryMin3 = binaryMin3.substring(8-diff3);

  String binaryMin4 = "00000000"+Integer.toBinaryString(min4);
  int diff4 = 8 - Integer.toBinaryString(min4).length();
  binaryMin4 = binaryMin4.substring(8-diff4);
```

```
  // spojeni bitovych retezcu
  String binaryMin = binaryMin1+binaryMin2+binaryMin3+binaryMin4;
  long outMin = Long.parseLong(binaryMin, 2); // zpetny prevod

  return outMin;
}
private static long getMaxIPValue(String maxIP) {
  if (maxIpOutstation.getText().isEmpty()) {
    return 0;
  }

  String[] parsedMax = maxIP.split("\\.");
  int max1 = Integer.parseInt(parsedMax[0]);
  int max2 = Integer.parseInt(parsedMax[1]);
  int max3 = Integer.parseInt(parsedMax[2]);
  int max4 = Integer.parseInt(parsedMax[3]);

  String binaryMax1 = "00000000"+Integer.toBinaryString(max1);
  int diff11 = 8 - Integer.toBinaryString(max1).length();
  binaryMax1 = binaryMax1.substring(8-diff11);

  String binaryMax2 = "00000000"+Integer.toBinaryString(max2);
  int diff22 = 8 - Integer.toBinaryString(max2).length();
  binaryMax2 = binaryMax2.substring(8-diff22);

  String binaryMax3 = "00000000"+Integer.toBinaryString(max3);
  int diff33 = 8 - Integer.toBinaryString(max3).length();
  binaryMax3 = binaryMax3.substring(8-diff33);

  String binaryMax4 = "00000000"+Integer.toBinaryString(max4);
  int diff44 = 8 - Integer.toBinaryString(max4).length();
  binaryMax4 = binaryMax4.substring(8-diff44);

  String binaryMax = binaryMax1+binaryMax2+binaryMax3+binaryMax4;
  long outMax = Long.parseLong(binaryMax, 2);

  return outMax;
}
```

Výpis B.6: Metoda pro vytvoření skriptů.

```
private void createIpScript(long minIPValue, long maxIPValue,
String mask, String interf, String path, String type) {
  try (Writer writer = new BufferedWriter(new OutputStreamWriter(
new FileOutputStream(path), StandardCharsets.UTF_8))) {
    int indexlink = 0;
    for (long i = minIPValue; i < maxIPValue+1; i++) {
      String binary = Long.toBinaryString(i);
      int one = Integer.parseInt(binary.substring(0, 8),2);
      int two = Integer.parseInt(binary.substring(8, 16),2);
      int three = Integer.parseInt(binary.substring(16, 24),2);
      int four = Integer.parseInt(binary.substring(24, 32),2);
      String ip = one+"."+two+"."+three+"."+four;

      writer.write("ip␣addr␣"+type+ "␣"+ip+"/"+mask+"␣dev␣"+interf+
"\n");

      if (type.equals("add␣")) {
        if(DNP3Sampler.outstations.containsValue(ip)) {
        System.out.println("IP␣"+ip+"␣EXIST!");
        return;
      }
      // kontrola stejneho poctu IP a linkovych adres
      if (links.size() == DNP3Sampler.outstations.size()) {
        return;
      }
      DNP3Sampler.outstations.put(links.get(indexlink), ip);
      System.out.println("IP␣ADDED␣"+ip+"␣LINK␣"+links.get(
indexlink));
      }
      else {
        System.out.println("IP␣DELETED␣"+ip+"␣LINK␣"+links.get(
indexlink));
        if (links.size() == indexlink + 1) {
          return;
        }
      }
      indexlink++;
    }
  } catch (IOException exn) {
    System.out.println(exn);
  }
}
```

Výpis B.7: Metoda pro náhodné vybírání IP adresy.

```
public static String randomIp() {
  if (getMaxIPValue(maxIpOutstation.getText()) == 0) {
    return "";
  }
  else {
    if(DNP3Sampler.outstations.size() == outsIpTaken.size()) {
      return "";
    }
    long out = ThreadLocalRandom.current().nextLong(getMinIPValue(
minIpOutstation.getText()), getMaxIPValue(maxIpOutstation.
getText())+1);
    /* Kontrola uz vybranych adres */
    for(Long outIp : outsIpTaken) {
      if(outIp.equals(out)) {
      return randomIp();
      }
    }
    outsIpTaken.add(out);

    String binary = Long. toBinaryString(out);
    int one = Integer . parseInt(binary. substring(0, 8) ,2);
    int two = Integer . parseInt(binary. substring(8, 16) ,2);
    int three = Integer . parseInt(binary. substring(16, 24) ,2);
    int four = Integer . parseInt(binary. substring(24, 32) ,2);
    String ip = one + "." + two + "." + three + "." + four;

    return ip;
  }
}
```

C Výpisy pro DNP3 Server

Výpis C.1: DNP3Server.

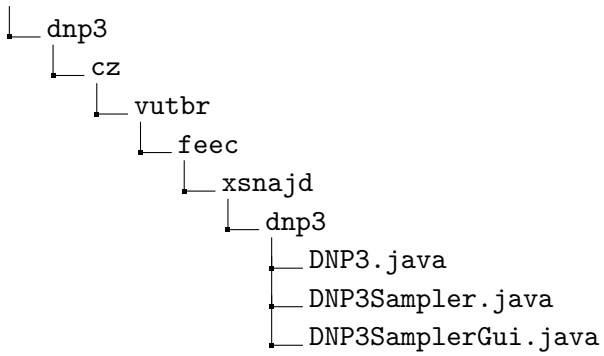
```
...
public static void main(String[] args) {
  InputStreamReader converter = new InputStreamReader(System.in);
  BufferedReader in = new BufferedReader(converter);
  String line = null;
  try {
    System.out.println("Server␣IP:␣");
    line = in.readLine();
    masterIpAddr = line;
    System.out.println("Server␣Link:␣");
    line = in.readLine();
    localAddr = line;
    System.out.println("Port:␣");
    line = in.readLine();
    port = line;
    System.out.println("Outstations");
    System.out.println("Min␣IP:␣");
    line = in.readLine();
    minIP = line;
    System.out.println("Max␣IP:␣");
    line = in.readLine();
    maxIP = line;
    System.out.println("Min␣Link:␣");
    line = in.readLine();
    minLink = line;
    System.out.println("Max␣Link:␣");
    line = in.readLine();
    maxLink = line;
  } catch (IOException e1) {
    e1.printStackTrace();
  }
  if (!setting()) return;
  addOutstations(getMinIPValue(minIP), getMaxIPValue(maxIP),
minLink, maxLink);

  DNP3Manager manager = null;
  try {
    manager = DNP3ManagerFactory.createManager(1,
PrintingLogHandler.getInstance());
  } catch (DNP3Exception e) {
    e.printStackTrace();
  }
  try {
```

```
    run(manager);
  } catch (Exception ex) {
    System.out.println("Exception: " + ex.getMessage());
  } finally {
    manager.shutdown();
  }
}
static void run(DNP3Manager manager) throws Exception {
  int tmpPort = Integer.parseInt(port);
  int tmpLink = Integer.parseInt(localAddr);
  for (OutstationClass o : outList) {
    Channel channel = manager.addTCPClient(
      "client",
      LogMasks.NORMAL | LogMasks.APP_COMMS,
      ChannelRetry.getDefault(),
      o.getIpOutstation(),
      masterIpAddr,
      tmpPort,
      PrintingChannelListener.getInstance()
    );
    MasterStackConfig config = new MasterStackConfig();
    config.link.localAddr = tmpLink;
    config.link.remoteAddr = o.getRemoteAddr();
    config.master.disableUnsolOnStartup = true;

    Master master = channel.addMaster("master", PrintingSOEHandler.
getInstance(), DefaultMasterApplication.getInstance(), config);
    master.enable();
    mastList.add(master);
  }
  ...
}
public static boolean setting() {
  if (minIP.isEmpty() || maxIP.isEmpty() || minLink.isEmpty() ||
maxLink.isEmpty() || port.isEmpty() || masterIpAddr.isEmpty()) {
    System.out.println("Zadejte vsechny parametry.");
    return false;
  }
  return true;
}
private static long getMinIPValue(String minIP) {...}
private static long getMaxIPValue(String maxIP) {...}
private static void addOutstations(long minIPValue, long maxIPValue,
String minLink, String maxLink) {
  int tmpMinLink = Integer.parseInt(minLink);
  int tmpMaxLink = Integer.parseInt(maxLink);
  int j = tmpMinLink;
```

```
  for (long i = minIPValue; i < maxIPValue+1; i++) {
    String binary = Long.toBinaryString(i);
    int one = Integer.parseInt(binary.substring(0, 8),2);
    int two = Integer.parseInt(binary.substring(8, 16),2);
    int three = Integer.parseInt(binary.substring(16, 24),2);
    int four = Integer.parseInt(binary.substring(24, 32),2);
    String ip = one+"."+two+"."+three+"."+four;

    int link = j;
    if (j == tmpMaxLink+1) return;
    j++;
    outList.add( new OutstationClass(link, ip));
  }
}
```

D Obsah přiloženého CD

- `/` ..... kořenový adresář přiloženého CD
  - `jar`
    - `DNP3.jar` ..... externí knihovna pro modul JMeter
  - `lib_opendnp3`
    - `opendnp3-bindings-2.2.1-SNAPSHOT.jar` ..... knihovna DNP3 vygenerována nástrojem Maven
  - `libs_linux` ..... knihovny vygenerované pro Linux
    - `libasiodnp3.so`
    - `libasiodnp3.so.2`
    - `libasiodnp3.so.2.2.1`
    - `libasiopal.so`
    - `libasiopal.so.2`
    - `libasiopal.so.2.2.1`
    - `libopendnp3.so`
    - `libopendnp3.so.2`
    - `libopendnp3.so.2.2.1`
    - `libopendnp3java.so`
    - `libopendnp3java.so.2`
    - `libopendnp3java.so.2.2.1`
    - `libopenpal.so`
    - `libopenpal.so.2`
    - `libopenpal.so.2.2.1`
  - `server_dnp3` ..... zdrojové soubory pro DNP3 Server a spustitelný soubor
    - `src`
      - `cz`
        - `vutbr`
          - `feec`
            - `xsnajd`
              - `dnp3`
                - `DNP3.java`
                - `OutstationClass.java`
    - `DNP3Server.jar`
  - `openiec61850`
    - `libs` ..... sestavené knihovny IEC61850 protokolu
      - `jasn1-iec61850mod-1.10.0.jar`
      - `jcalendar-1.4.jar`
      - `logback-classic-1.2.3.jar`
      - `logback-core-1.2.3.jar`
      - `openiec61850-1.7.0.jar`
      - `slf4j-api-1.7.25.jar`
    - `openiec61850-gui-client` ..... script pro linux
    - `openiec61850-gui-client.bat` ..... spouštěcí script grafického klienta pro Windows
  - `src` ..... zdrojové soubory DNP3 modulu


dnp3
cz
vutbr
feec
xsnajd
dnp3
DNP3.java
DNP3Sampler.java
DNP3SamplerGui.java