VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Roman Tomiczek

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

CHYTRÁ DOMÁCNOST – KNIHOVNA PRO SBĚR DAT
SMART HOME - LIBRARY FOR DATA ACQUISITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Roman Tomiczek**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Petr Mlýnek, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Telekomunikační a informační technika**

Ústav telekomunikací

Student: Bc. Roman Tomiczek ***ID:*** 186215
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Chytrá domácnost – knihovna pro sběr dat

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je návrh a vývoj softwarové knihovny pro Smart bránu (Gateway) pro agregaci dat z elektroměrů a zasílání dat do centrální stanice. Cílem je vývoj univerzální knihovny s komunikací DLMS/COSEM pro sběr dat z reálných energetických zařízení (elektroměrů a prvků chytré domácnosti) a následné vizualizace dat. Výstupem je implementace DLMS/COSEM knihovny v C++ a základní ověření v několika scénářích, jak na reálných zařízeních, tak na mikropočítači.

DOPORUČENÁ LITERATURA:

[1] Gurux.DLMS | Gurux for DLMS smart meters. [online]. [cit. 2018-09-13]. Dostupné z: http://www.gurux.fi/Gurux.DLMS

[2] SOOD, J.K., D. FISCHER, J.M. EKLUND a T. BROWN. Developing a communication infrastructure for the smart grid. IEEE Electrical power & energy conference, 2009. Dostupné z: <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5420809>.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: doc. Ing. Petr Mlýnek, Ph.D.

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

ABSTRAKT

Diplomová práce se zabývá vývojem softwarové knihovny pro agregaci dat z elektroměrů a zasílání dat do centrální stanice. Jsou popsány nejvíce používané protokoly v inteligentních sítích. Práce se detailněji zabývá nejdůležitějšími částmi standardu DLMS/COSEM. Nejprve je rozebrán protokol DLMS, který zajišťuje komunikaci mezi klienty a servery. Dále je popsán objektový model COSEM, který specifikuje modelování objektů, pro přístup k měřícím zařízením a systém OBIS, který určuje identifikaci datových objektů. Dále je popsána knihovna Gurux, která se specializuje na inteligentní odečty měřičů. Ve výsledcích práce je popsána realizace a testování softwarové knihovny pro agregaci dat ze serverů protokolem DLMS/COSEM s použitím knihovny GURUX. Na závěr je popsána série testů, které byly provedeny za pomocí této softwarové knihovny.

KLÍČOVÁ SLOVA

DLMS, COSEM, OBIS, chytrá domácnost, knihovna pro sběr dat

ABSTRACT

The diploma thesis deals with the development of a software library for aggregating data from electricity meters and sending data to a central station. The most used protocols in smart grids are described. The work deals in more detail with the most important parts of the DLMS/COSEM standard. The DLMS protocol is described, which ensures communication between clients and servers. Next, the COSEM object model is described, which specifies object modeling, for access to measuring devices, and the OBIS system, which determines the identification of data objects. The Gurux library, which specializes in intelligent meter readings, is also described. The results of the work describe the implementation and testing of a software library for aggregation of data from servers using the DLMS/COSEM protocol using the GURUX library. At the end of the work is described a series of tests that were performed using this software library.

KEYWORDS

DLMS, COSEM, OBIS, Smart Home, library for data collection

TOMICZEK, Roman. *Chytrá domácnost – knihovna pro sběr dat*. Brno, Rok, 65 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: doc. Ing. Petr Mlýnek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Chytrá domácnost – knihovna pro sběr dat“ jsem vypracoval samostatně pod vedením vedoucího diplomové 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové 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 doc. Ing. Petru Mlýnkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Úvod

Advanced Meter Management (AMM) je systém, který spravuje a řídí inteligentní měřiče a jejich komunikační infrastrukturu s rozhraními. Inteligentní měření nabízí různé výhody, protože veřejné služby jsou schopny dosáhnout úspor finančně, provozně a prostřednictvím zvýšené energetické účinnosti. Mezi výhody patří finanční úspory, protože technici nemusí měřiče fyzicky číst, připojovat nebo odpojovat. Fakturace může být založena spíše na skutečném než odhadovaném využití, aby se zvýšila finanční efektivita a přesnost peněžních toků. Lepší řízení nabídky a poptávky po proudu elektřiny do sítě výrazně snižuje výpadky a umožňuje přesné investice do sítě.

V těchto sítích se využívají specifické protokoly, které jsou popsané v první kapitole této práce. Zejména jsou to protokoly ANSI C12.18, C12.19, C12.21, IEC61107 a OSGP.

Tato práce se věnuje blíže protokolu DLMS/COSEM. V další kapitole je popsán DLMS, kde je zdokumentováno jaké zabezpečení obsahuje, dále jeho komunikační model, typ adresace a Application Associations. DLMS může komunikovat buď v síti IP nebo po sériové lince. Tato skutečnost je taky uvedena.

V další kapitole je popsán objektový model COSEM, který specifikuje modelování objektů, pro přístup k měřícím zařízením a systém OBIS, který určuje identifikaci datových objektů na základě šesti hodnotových skupin.

Dále je uvedena knihovna Gurux, která se specializuje na inteligentní odečty měřičů. Tato knihovna implementuje několik různých otevřených komunikačních protokolů a nejčastěji se používá DLMS/COSEM. S komponentou protokolu Gurux DLMS/COSEM lze si vytvořit svůj vlastní systém AMR (Automatic Meter Reading) pro měřiče spotřeby elektřiny, a to prostřednictvím připojení TCP/IP, Terminal nebo sériové komunikace.

Poslední kapitola se zabývá realizací a testováním softwarové knihovny pro agregaci dat ze serverů protokolem DLMS/COSEM s použitím knihovny GURUX. Postupně je popsáno nastavení serverů a vyzkoušení komunikace pomocí programu Gurux Director. Dále tato kapitola obsahuje popis kódu a vývoj softwarové knihovny. Po domluvě s vedoucím práce, testování na reálných zařízeních nebyly provedeny a byly provedeny obsáhlejší testovací scénáře. Na závěr je popsána série testů, které byly provedeny za pomocí této softwarové knihovny. Tyto testy byly nakonec analyzovány za pomocí analyzátoru síťového provozu.

1 Protokoly v energetice

Tato kapitola popisuje různé protokoly, které se využívaly nebo se stále využívají v energetice.

1.1 ANSI C12.18, C12.19, C12.21

ANSI (American National Standards Institute) poskytuje komunikaci na otevřené platformě s měřicím zařízením prostřednictvím optického portu ANSI typu 2. Protokol je zapsán tak, aby odpovídal sedmivrstvému modelu OSI.

1.1.1 ANSI C12.18

ANSI C12.18 je standard, který určuje, jak přenášet data, která jsou definována ANSI C12.19. V této normě byl navržen jazyk PSEM (Protocol Specifications for Electric Metering), aby poskytoval rozhraní mezi měřicím zařízením a jakýmkoli jiným zařízením na komunikačním spojení point-to-point. Protokol tedy podporuje obousměrnou komunikaci. ANSI C12.18 je určen pro komunikaci přes optický port ANSI typu 2. Také specifikuje podrobnosti protokolu nižší úrovně, jako jsou bitová rychlost, schéma detekce chyb a časový limit. Specifikuje také přihlášení a odhlášení relace, čtení a zápis. ANSIC12.18 implementuje 3 vrstvy OSI. Fyzickou, linkovou a aplikační vrstvu.

Konfiguraci měřiče lze provést pomocí služby zápisu. Formát dat je stejný jako ve službě čtení. Standard ANSI C12.18 definuje službu zápisu jako volitelnou. Povolení úrovně přístupu pro dvě předchozí služby lze získat pomocí přihlašovací služby a bezpečnostních služeb. Přihlašovací služba je standardně povinná, zatímco bezpečnostní služba je volitelná. Přihlašovací služba se používá k odeslání veřejného a nešifrovaného identifikačního čísla, zatímco bezpečnostní služba se používá k odeslání hesla. Identifikační číslo i heslo musí odpovídat identickému obsahu tabulky definovanému standardem ANSI C12.19. Výrobce měřiče může definovat konkrétní tabulky ANSI C12.19 jako chráněné heslem, zatímco ostatní mohou být veřejně dostupné a vyžadují pouze přihlášení. Každá struktura datových paketů je popsána tak, že je detekován začátek a konec rámce. Detekce chyb je také implementována na konci paketu pomocí CRC (Cyclic Redundancy Check) [1].

1.1.2 ANSI C12.19

ANSI C12.19 definuje strukturu tabulky pro data, která mají být posílána nejčastěji mezi koncovým zařízením, což je nejčastěji elektroměr a počítačem, který nejčastěji

představuje ruční zařízení se čtečkou. ANSI C12.19 definuje sadu tabulek, které umožní toto zařízení konfigurovat, číst a zapisovat data. Účelem tabulek je definovat struktury pro přenos dat do a z koncových zařízení. Nedefinuje však protokol ani jazyk pro přenos těchto dat. O to se stará související norma ANSI C12.18. Standardní datová struktura C12.19 je definována jako sady tabulek. Tabulky jsou seskupeny do sekcí. Velikost a formát tabulky jsou určeny konfiguračními tabulkami. Data jsou přenášeny čtením nebo zápisem konkrétních tabulek adresovaných číslem [2].

1.1.3 ANSI C12.21

ANSI C12.21 je rozšíření C12.18, podporuje také spojení point-to-point, avšak už ne optickým portem ale bezdrátovou technologii s využitím modemu. ANSI C12.21 také zahrnuje autentizaci [1].

1.2 IEC 61107

IEC 61107 je komunikační protokol, který se používá jako protokol v energetice, a to pro chytré měřiče spotřeby. Protokol byl publikovaný organizací IEC. Protokol je nahrazen normou IEC 62056, avšak pro jeho jednoduchost zůstává používán. Data odesílá ve formátu ASCII pomocí sériového portu. Ke komunikaci se využívá pár vodičů, modulovaný pomocí EIA-485, nebo se využívá optický kanál. Jako vysílač se používá LED dioda a jako přijímač je použita fotodioda [3].

1.3 OSGP

OSGP protokol byl vyvinut OSGP Alliance a následně publikován jako standard ústavem ETSI. Je to jeden z nejpoužívanějších protokolů v energetice, a to pro chytré měřiče spotřeby. Tento protokol je založen na modelu OSI [4].

1.3.1 Aplikační vrstva

Na aplikační vrstvě poskytuje tabulkově orientovaný systém ukládání dat, nejen pro inteligentní měřiče a související data, ale také pro univerzální rozšíření na jiná zařízení inteligentní sítě. Stejně jako u SQL i OSGP podporuje čtení a zápis jednotlivých atributů, více prvků nebo celých tabulek. Jako protokol vyvinutý pro použití v smart grid systémech namísto připojení point-to-point nebo optického připojení, OSGP zahrnuje schopnost pro adaptivní, řízený síťový systém. Tento systém umožňuje každému zařízení OSGP sloužit jako opakovač zpráv, což dále optimalizuje využití šířky pásma opakováním pouze těch paketů, které je třeba opakovat. OSGP

také zahrnuje autentizaci i šifrování pro všechny ústředny, aby byla chráněna integrita a soukromí dat [5].

1.3.2 Síťová vrstva

Pro síťovou vrstvu OSGP používá EN14908-1 s rozšířením pro zabezpečení, ověřování a šifrování. ISO/IEC 14908 je vysoce optimalizována pro efektivní, spolehlivé a škálovatelné řídicí síťové aplikace. Nízká režie ISO / IEC 14908 umožňuje dosáhnout vysokého výkonu bez potřeby vysoké šířky pásma [5].

1.3.3 Fyzická vrstva

Ve fyzické vrstvě OSGP používá ETSI TS 103 908 jako svůj komunikační standard. Zároveň není vázán na konkrétní komunikaci na fyzické vrstvě. Protože vychází z ISO / IEC 14908, který je nezávislý na médiích, OSGP má možnost použití s jakýmkoli současným nebo budoucím fyzickým médiem. Protokol zahrnuje také bezpečnostní opatření omezením přístupu k přenášeným datům a šifrování dat. Protokol je také schopen detekovat špatné provedení měření a zabránění odesílání dat do koncentrátoru. Zde jsou popsány některé bezpečnostní funkce protokolu OSGP:

- RC4 algoritmus – Systém šifrování toku dat, kdy převádí prostý text na šifru. Implementace algoritmu RC4 v OSGP je podobná implementaci používané v WEP.
- Funkce odezvy – OSGP implementuje funkci odezvy, která se využívá při ověřování zprávou.
- Zabezpečené vysílání – Mechanismus používaný k odesílání aktualizací firmwaru.
- Klíče – Protokol používá k šifrování zpráv klíče pro relace a taky hlavní klíč pro identifikaci.

Avšak bezpečnostní funkce obsažené v protokolu OSGP neposkytují celkovou bezpečnost. Implementace algoritmu RC4 obsahuje slabiny zjištěné v systému WEP a je považován za nezabezpečený algoritmus. Funkce hash (digest) má výstup o délce 8 bajtů a je generována procesem v lineárním formátu, což omezuje entropii procesu. Slabost tohoto protokolu umožňuje manipulaci s generovanými odpověďmi. Hlavní klíč používaný k ověření se používá také k zajištění klíčů relace.

Vzhledem k těmto nedostatkům v bezpečnostních opatřeních protokolu se doporučuje, aby byla použita externí bezpečnostní opatření k zajištění komunikace prostřednictvím OSGP, jako je použití šifrovacích nástrojů pro robustní šifrování end-to-end PLC komunikace nebo použití filtrování v PLC komunikaci mezi měřiči a koncentrátorem [5].

2 DLMS

The Device Language Message Specification (DLMS) and Companion Specification for Energy Metering (COSEM) společně tvoří komunikační protokol. Představuje celosvětový standard pro inteligentní měření energie. DLMS/COSEM obsahuje tři hlavní komponenty:

- DLMS – soubor norem vydaných a udržovaným sdružením uživatelů DLMS a začleněných do IEC 62056 podle IEC TC13 WG14.
- COSEM – zahrnuje sadu specifikací transportní a aplikační vrstvy DLMS.
- OBIS – Object Identification Systém, systém pro identifikaci objektů [8].

Asociace uživatelů DLMS definuje čtyři dokumenty, které specifikují DLMS/COSEM. Tyto dokumenty jsou rozděleny do čtyř barevně oddělených knih, takzvaných Coloured Books. Jmenovitě jsou to Green Book, Yellow Book, Blue Book a White Book. Green Book specifikuje protokoly a architekturu, Yellow Book se zabývá testováním, White Book obsahuje slovník pojmů a Blue Book popisuje objektový model COSEM a systém pro identifikaci objektů OBIS.

Inteligentní měřicí systém vyžaduje různé funkce, jako je měření, řízení přístupu, správa a výměna dat v různých komunikačních médiích. Pro flexibilitu a škálovatelnost takového inteligentního systému lze DLMS / COSEM rozdělit na tři části, kterými jsou modelování, tvorba zpráv a transport.

- Modelování – Tento model zahrnuje sadu postupů, ve kterých jsou data navržena a přenášena.
- Tvorba zpráv – Protokol aplikační vrstvy DLMS / COSEM, který určuje zprávy pro přístup k datům modelovaným objekty COSEM.
- Transport – Komunikační profil, který určuje, jak se používá DLMS / COSEM v různých standardizovaných komunikačních médiích. Ty jsou specifikovány v Zelené knize[6].

2.1 Zabezpečení

DLMS/COSEM požaduje bezpečnostní podmínky, kterými jsou autentizace komunikujících stran, řízení přístupových práv pro klienta a ochranu dat COSEM a xDLMS zpráv šifrováním. Existují dva druhy zabezpečení:

- Zabezpečení přístupu se týká práv klienta pro přístup k datům uloženým na daném serveru.
- Zabezpečení přenosu se týká šifrováním aplikované na informace vyměňované mezi serverem a klientem [6].

Prostředky serverů DLMS/COSEM, atributy a metody objektů COSEM, jsou přístupné klientům DLMS/COSEM v rámci Application Associations (AA). Během

založení AA se musí klient a server identifikovat. Server může také vyžadovat, aby se uživatel klienta identifikoval sám. Server může dále vyžadovat, aby se klient autentizoval sám a klient může také požadovat, aby se server autentizoval sám [16].

Mechanismus identifikace uživatele klienta umožňuje serveru rozlišovat mezi různými uživateli na straně klienta a zaznamenávat jejich činnosti přístupem k serveru.

Mechanismy autentizace určují protokol, který mají komunikační jednotky použít k autentizaci během vytváření AA. K dispozici jsou tři různé mechanismy autentizace s různými úrovněmi zabezpečení autentizace [14].

- Zabezpečení bez ověření, neboli zabezpečení nejnižší úrovně. Účelem ověření bez zabezpečení je umožnit klientovi získat některé základní informace ze serveru. Tento mechanismus ověřování nevyžaduje žádnou autentizaci. Klient má přístup k atributům a metodám objektu COSEM v kontextu zabezpečení a přístupových práv převládajících v daném AA.
- Ověřování nízké úrovně zabezpečení (LLS). V tomto případě server vyžaduje, aby se klient autentizoval sám zadáním hesla, které server zná. Heslo je drženo aktuálním objektem „Association SN / LN“, který modeluje AA, které má být vytvořeno. Objekty „Association SN / LN“ poskytují prostředky ke změně zabezpečení. Je-li zadané heslo přijato, může být stanoveno AA.
- Ověřování na vysoké úrovni zabezpečení (HLS). V takovém případě se klient i server musí úspěšně autentizovat, aby vytvořili AA.

Po vytvoření AA lze služby xDLMS použít k přístupu k atributům a metodám COSEM, s výhradou kontextu zabezpečení a přístupových práv [17].

XDLMS APDU mohou být kryptograficky chráněny. Požadovaná ochrana je určena kontextem zabezpečení a přístupovými právy. Pro podporu zabezpečení mezi třetími stranami a servery mezi koncovými stranami mohou takové třetí strany také přistupovat k prostředkům serveru pomocí klienta jako zprostředkovatele. Kromě toho lze data COSEM přenášená xDLMS APDU kryptograficky chránit. Protože jsou tyto bezpečnostní mechanismy aplikovány na úrovni aplikačního procesu/aplikační vrstvy, lze je použít ve všech komunikačních profilech DLMS/COSEM [6].

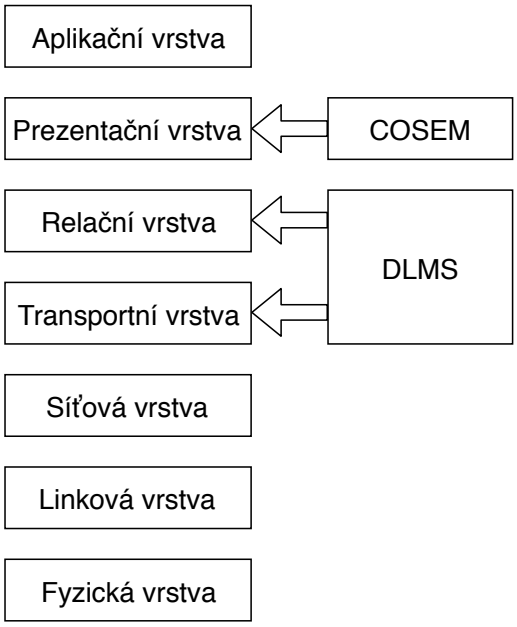
2.2 Komunikační model

DLMS / COSEM používá koncept modelu OSI k modelování výměny informací mezi měřiči a systémy sběru dat. DLMS/COSEM lze zobrazit na modelu OSI/ISO na obrázku 2.1.

Aplikační funkce měřicích zařízení a systémů sběru dat jsou modelovány aplikačními procesy (AP). Komunikace mezi přístupovými body je modelována komunikací mezi aplikačními entitami (AE). Aplikační entita představuje komunikační funkce aplikačních procesů. V aplikačním procesu může být více sad komunikačních funkcí

OSI, takže jeden aplikační proces může být reprezentován více aplikačních entit. Každá aplikační entita však představuje jeden přístupový bod. Aplikační entita obsahuje sadu komunikačních schopností nazývaných prvky aplikačních služeb (ASE). Prvek aplikačních služeb je soudržná sada integrovaných funkcí [13].

Výměna dat mezi systémy sběru dat a měřicími zařízeními je založena na modelu klient/server, kde systémy sběru dat mají roli klienta a měřicí zařízení mají roli serveru. Klient odešle servisní požadavky na server, který odešle servisní odpovědi. Server může navíc iniciovat nevyžádané žádosti o služby, aby informoval klienta o událostech nebo zaslal data za předem nakonfigurovaných podmínek [6].



Obr. 2.1: DLMS/COSEM na modelu OSI.

2.3 Adresace

Pojmenování a adresování jsou nezbytné u komunikačních systémů. Jméno identifikuje komunikující entitu. Adresa určuje, kde lze tuto entitu najít.

Subjekty DLMS/COSEM, včetně klientů, serverů a systémů třetích stran, budou jednoznačně pojmenovány podle svého systémového názvu. Názvy systému musí být trvale přiřazeny. Fyzická zařízení serveru mohou hostit jedno nebo více logických zařízení (LD). Logické zařízení musí být jednoznačně identifikovány svým logickým názvem zařízení (LDN). Logické zařízení hostované stejným fyzickým zařízením sdílejí název systému. Každé fyzické zařízení musí mít odpovídající adresu. Závisí to na komunikačním profilu. Může to být MAC adresa, IP adresa, telefonní číslo nebo jejich kombinace.

Každý klient DLMS/COSEM a každý server, logické zařízení COSEM, je vázán na přístupový bod služby (SAP). SAP leží v podpůrné vrstvě DLMS/COSEM AL. V závislosti na komunikačním profilu SAP může být IP adresa, horní adresa HDLC, adresa LLC atd. Na straně serveru je tato vazba modelována pomocí IC přiřazení „SAP Assignment“. Délka SAP závisí na komunikačním profilu [6].

2.4 Application Associations

Application Associations jsou logická propojení mezi klientem a serverem. Mohou být zřízena na žádost klienta využívajícího služby ACSE AL orientované na spojení nebo mohou být předem stanovena. Mohou být potvrzeny nebo nepotvrzeny. Logické zařízení COSEM může podporovat jednu nebo více AA, každé s jiným klientem. Každé AA určuje kontext, ve kterém probíhá výměna informací.

Potvrzené AA je navrženo klientem a přijato serverem za předpokladu, že:

- uživatel klienta zná server
- aplikační kontext navržený klientem je pro server přijatelný
- mechanismus autentizace navržený klientem je přijatelný pro server a autentizace je úspěšná
- prvky kontextu xDLMS lze úspěšně vyjednat mezi klientem a serverem [6].

Nepotvrzené AA je také navrženo klientem za předpokladu, že jej server přijme. Nedochází k žádnému vyjednávání. Nepotvrzené AA jsou užitečné pro odesílání broadcastových zpráv z klienta na servery.

AA jsou modelovány objekty COSEM „Association SN / LN“, které drží SAP identifikující přidružené partnery, název kontextu aplikace, název mechanismu autentizace a kontext xDLMS. Objekty „Association SN / LN“ také určují specifickou sadu přístupových práv k atributům a metodám objektů COSEM a odkazují na objekt „Security setup“, který obsahuje prvky kontextu zabezpečení. Přístupová práva a kontext zabezpečení se mohou u každého AA lišit.

2.4.1 Služby poskytované pro sdružování aplikací

ACSE se využívá pro účely komunikačních profilů DLMS. Služby poskytované pro vytvoření a uvolnění přidružení aplikací jsou následující:

- COSEM-OPEN Používá se k založení AA podle hodnot parametrů aplikačního kontextu, zabezpečení a dalších parametrů. Je založen na službě ACSE A-ASSOCIATE. AA mohou být stanoveny různými způsoby.
  - Potvrzené AA jsou vytvořeny prostřednictvím výměny zpráv pomocí služby COSEM-OPEN - mezi klientem a serverem pro vyjednávání kontextů. Potvrzené AA lze vytvořit mezi jedním klientem a jedním serverem.

  - Nepotvrzené AA jsou vytvořeny prostřednictvím zaslané zprávy pomocí služby COSEM-OPEN - z klienta na server pomocí parametrů kontextů, které má server podporovat. Mezi klientem a jedním nebo více servery lze vytvořit nepotvrzené AA.
  - Předem stanovená AA. V tomto případě se služba COSEM-OPEN nepoužívá. Klient si musí být vědom kontextů podporovaných serverem. Předem vytvořené AA může být potvrzeno nebo nepotvrzeno.
- COSEM-RELEASE - Služba COSEM-RELEASE se používá k uvolnění AA. Pokud služba proběhne úspěšně, dokončí používání AA bez ztráty informací během přenosu. Předem vytvořené AA nelze uvolnit.
- COSEM-ABORT - Služba COSEM-ABORT způsobuje vynucené uvolnění AA s možnou ztrátou informací při přenosu. Nespoléhá se na službu ACSE A-ABORT [6].

2.5 Aplikační vrstva DLMS

Hlavní komponentou aplikační vrstvy je takzvaný Application Service Object (ASO). Poskytuje služby a aplikační procesy COSEM, a využívá služeb poskytované nižší vrstvou. Aplikační vrstva COSEM obsahuje tři povinné komponenty na straně klienta i na straně serveru.

- Asociační kontrolní servisní prvek, ACSE. Úkolem ACSE je vytvářet, udržovat a uvolňovat přidružení aplikací.
- rozšířený prvek aplikační služby DLMS, xDLMS_ASE. Úkolem xDLMS_ASE je poskytovat služby přenosu dat mezi přístupovými body COSEM.
- kontrolní funkce, CF. Určuje, jak služby ASO budou vyvolávat příslušné služby ACSE, xDLMS ASE a služby podpůrné vrstvy [15].

Na straně klienta je čtvrtý, volitelný prvek, nazvaný SN_MAPPER ASE. Využívá se, když server používá odkazování SN. Poskytuje mapování mezi službami pomocí odkazování LN a SN. DLMS / COSEM AL vykonává také některé funkce prezentační vrstvy OSI:

- Kódování a dekódování ACSE APDU a xDLMS APDU.
- Vytváření a používání dokumentů XML představujících APDU ACSE a xDLMS.
- Použití komprese a dekomprese.
- Použití, ověření a odstranění kryptografického zabezpečení.

2.6 Orientace spojení

DLMS/COSEM vyžaduje pro přenos dat spojení. Tato relace se skládá ze tří fází.

- Nejprve je mezi klientem a serverem AE navázáno spojení na úrovni aplikace, zvané Application Association (AA). Každá vrstva, která musí být spojena, může podporovat jedno nebo více spojení současně.
- Jakmile je stanoveno AA, může dojít k výměně zpráv.
- Na konci výměny dat se uvolní AA a dojde k rozpojení.

Pro účely velmi jednoduchých zařízení jsou povolena také jednosměrná komunikace. Pro multicasting a broadcast jsou předem nastavená AA. U takových AA může komunikační relace zahrnovat pouze fázi výměny zpráv [6].

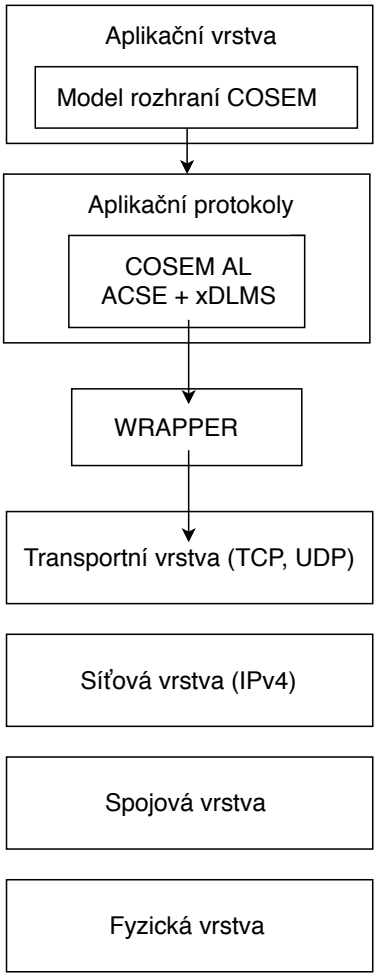
2.7 Transportní vrstva pro sítě IP

Tato kapitola specifikuje transportní vrstvy (TL) pro komunikační profily COSEM pro použití v sítích IPv4. Tyto transportní vrstvy poskytují služby modelu OSI. Transportní vrstva nespojově orientovaná je založena na protokolu User Datagram Protocol (UDP) a transportní vrstva spojově orientovaná je založena na protokolu Transmission Control Protocol (TCP). Přestože hlavní částí transportní vrstvy jsou protokoly UDP a TCP, obsahuje další podvrstvu, zvaná wrapper, viz obr. 2.2. Hlavní funkci Wrapperu je přizpůsobit sadu služeb ve stylu modelu OSI, poskytovanou DLMS/COSEM TL, pro volání funkcí UDP nebo TCP a naopak. Další funkcí wrapperu je schopnost adresování wPort, a také poskytuje informace o délce přenášených údajů. Tato funkce pomáhá rozpoznat příjem kompletního APDU, který může být odeslán a přijat ve více paketech TCP. Na klientské straně může DLMS/COSEM naslouchat na více portech, tedy může hostit několik klientských přístupových bodů nebo serverových logických zařízení. Wrapper tedy může identifikovat tyto přístupové body [6].

2.7.1 TCP

Transportní vrstva COSEM spojově orientovaná je založena na internetovém přenosovém protokolu Transmission Control Protocol neboli TCP. Poskytuje potvrzené doručení dat, detekci chyb, opětovný přenos dat po uplynutí limitu potvrzení. Zabývá se ztracenými, zpožděnými, duplikovanými nebo chybnými datovými pakety. Kromě toho nabízí TCP účinný mechanismus řízení toku a plně duplexní provoz.

TCP jako protokol spojově orientovaný zahrnuje tři fáze. Navázání spojení, výměnu dat a ukončení spojení. Transportní vrstva COSEM poskytuje všechny tři fáze.



Obr. 2.2: Wrapper.

Ve fázi vytvoření připojení jsou služby TCP-CONNECT poskytovány správce procesů TCP připojení. Ve fázi přenosu dat jsou služby TCP-DATA poskytovány aplikační vrstvě COSEM. Ve fázi uzavření připojení jsou služby TCP-DISCONNECT poskytovány znovu správci procesů připojení TCP. Aplikační vrstvě je také poskytovaný služba TCP-ABORT.

Transportní vrstva spojově orientovaná na připojení, založená na protokolu TCP, obsahuje stejnou podvrstvu WRAPPER jako transportní vrstva založený na protokolu UDP [13]. Kromě přizpůsobení služeb ve stylu OSI pro posílání a příjem dat, poskytuje WRAPPER také informace o adresování a délce dat.

2.7.2 Vytvoření transportní vrstvy a TCP spojení

TCP může navázat spojení voláním funkce OPEN. Tuto funkci lze nazvat aktivním nebo pasivním způsobem. Stav připojení TCP pasivní způsobem použitím funkce OPEN převede volající zařízení do stavu LISTEN a čeká na požadavek na připojení

z jakéhokoli vzdáleného TCP a portu. Aktivní volání OPEN způsobí, že TCP naváže připojení ke vzdálenému TCP. Navázání TCP spojení se provádí pomocí tzv. Trojcestného handshake. To je iniciováno jedním TCP volajícím aktivní OPEN a odpovědí dalším TCP, který již byl navázán pasivním OPEN a následně je ve stavu LISTEN.

Tento proces, sestávající ze tří zpráv, naváže spojení TCP a synchronizuje počáteční sekvenční čísla na obou stranách. Tento mechanismus byl pečlivě navržen tak, aby zaručil, že obě strany jsou připraveny k přenosu dat a vědí, že druhá strana je také připravena k přenosu [6].

2.8 HDLC

Tato kapitola specifikuje asynchronní, spojově orientovaný komunikační profil založeny na HDLC. Tento profil podporuje:

- konfigurace point-to-point a point-to-multipoint
- poloduplexní a plně duplexní připojení
- asynchronní přenos start / stop, s 1 start bitem, 8 datovými bity, bez parity, 1 stop bitem [9].

Fyzické zařízení podporuje jeden nebo více komunikačních profilů. V současné době DLMS specifikuje dva profily: 3-vrstvý profil založený na připojení HDLC a profil založený na IP [12] .

HDLC odpovídá vrstvám 2 až 4 OSI modelu. Profil založený na připojení HDLC zahrnuje tři vrstvy. Fyzickou vrstvu (sériové připojení), vrstvu HDLC a aplikační vrstvu. HDLC adresa klienta také nazývaná MAC adresa. Adresa MAC serveru je rozdělena na dvě části: horní část je adresa logického zařízení a dolní část je adresa fyzického zařízení. V některých případech (např. Topologie point-to-point) může být spodní část vynechána. Délka adresy serveru je:

- 1 bajt (pouze horní adresa).
- 2 bajty (1 bajt pro horní adresu a 1 bajt pro dolní adresu).
- 4 bajty (2 bajty pro horní adresu a 2 bajty pro dolní adresu) [10].

Vrstva HDLC je orientována na připojení. To znamená, že vrstvy vrstev HDLC musí nejprve vytvořit logické spojení mezi sebou výměnou a vyjednáváním některých parametrů připojení. Poté si mohou vyměňovat data, pokud je to potřeba, a nakonec připojení ukončit. Datové prvky vyměňované vrstvami HDLC se nazývají HDLC-snímky. Protože se jedná o protokol datového spojení, jsou data organizována do rámců. Snímek je přenášen sítí do cíle, který ověřuje jeho úspěšné přijetí. Je to bitově orientovaný protokol.

Aby byl zajištěn přenos pro spojově i nespojově orientovanou komunikaci, je vrstva datového spojení rozdělena do dvou podvrstev: podvrstva Logical Link Con-

trol (LLC) a podvrstva Media Access Control (MAC). Podvrstva LLC umožňuje kompatibilitu mezi vrstvou nespojově orientovanou a vrstvou orientovanou na spojení. Protože aplikační vrstva COSEM a MAC podvrstva tohoto komunikačního profilu jsou vrstvou orientovanou na připojení, role podvrstev LLC je pouze pro kontrolu směru toku dat. Podvrstva MAC určuje služby a protokoly pro přístup k médiu. V režimu COSEM je vybrán formát formátu HDLC typu 3 [11].

3 COSEM

Informace o objektu jsou uspořádány do atributů. Reprezentují vlastnosti objektu pomocí hodnot atributů. Hodnota atributu může ovlivnit chování objektu. Prvním atributem v kterémkoli objektu je „logical_name“. Je to jedna část identifikace objektu. Objekty, které sdílejí společné vlastnosti, jsou zobecněny jako třída rozhraní s class_id. V rámci určité třídy jsou společné vlastnosti (atributy a metody) popsány jednou pro všechny objekty. Okna třídy rozhraní se nazývají objekty COSEM [18].

Třída rozhraní „Register“ je vytvořena kombinací funkcí nezbytných pro modelování chování generického registru (obsahujícího naměřené nebo statické informace). Obsah registru je identifikován atributem „logical_name“. Logical_name obsahuje identifikátor OBIS. Skutečný obsah registru je nesen svým atributem „value“. Definování konkrétního měřiče znamená definování několika specifických registrů [6].

3.1 COSEM logické zařízení

Logické zařízení COSEM je sada objektů. Každé fyzické zařízení musí obsahovat „Logické zařízení pro správu“. Adresování logických zařízení COSEM je zajištěno adresním schématem nižších vrstev použitého protokolu. Logické zařízení COSEM lze identifikovat podle jedinečného názvu logického zařízení COSEM. Tento název lze získat z instance IC „přiřazení SAP“ nebo z objektu COSEM „COSEM logical device name“. Tento název je definován jako oktetový řetězec až 16 oktetů. První tři oktety jednoznačně identifikují výrobce zařízení. Výrobce odpovídá za zaručení jedinečnosti následujících třinácti oktetů[6].

3.2 Zobrazení asociace logického zařízení

Za účelem přístupu k objektům COSEM na serveru se nejprve vytvoří přidružení aplikací. To charakterizuje kontext, ve kterém budou přidružené aplikace komunikovat. Hlavní části tohoto kontextu jsou:

- kontext aplikace
- ověřovací kontext
- kontext xDLMS

Tyto informace jsou obsaženy ve speciálním objektu COSEM, v objektu „Association“. Jsou definovány dva typy tohoto přidruženého objektu. Jeden pro asociace používající odkazování na krátké názvy („Association SN“) a jeden pro použití odkazování na logické názvy („Association LN“). V závislosti na přidružení mezi klientem a serverem může server udělit různá přístupová práva. Přístupová práva se týkají

sady objektů COSEM, viditelných objektů, ke kterým lze v dané asociaci přistupovat. Kromě toho může být v rámci přidružení také omezen přístup k atributům a metodám těchto objektů COSEM (například určitý typ klienta může číst pouze konkrétní atribut objektu COSEM) [18].

Seznam viditelných objektů COSEM může získat klient přečtením atributu „object_list“ příslušného asociačního objektu. Další informace o přístupových právech (pouze pro čtení, pouze pro zápis, čtení a zápis) k atributům a dostupnosti metod ,v rámci zavedeného přidružení, lze nalézt prostřednictvím specifických atributů (logické jméno) nebo speciální metody poskytované asociačními objekty [6].

3.3 Správa logického zařízení

Logické zařízení pro správu je povinným prvkem každého fyzického zařízení a má vyhrazenou adresu. Musí podporovat přidružení aplikací k veřejnému klientovi s nejnižší úrovní zabezpečení. Jeho úlohou je podporovat odhalení vnitřní struktury fyzického zařízení a podporovat oznamování událostí na serveru. Kromě objektu „Association“, který modeluje asociaci s veřejným klientem, musí logické zařízení pro správu obsahovat objekt „SAP assignment“, který dává jeho SAP a SAP všem ostatním logickým zařízením v rámci fyzického zařízení. Objekt SAP assignment musí být čitelný alespoň veřejným klientem [19]. Pokud je ve fyzickém zařízení pouze jedno logické zařízení, může být objekt „SAP assignment“ vynechán [6].

3.4 Odkazování na objekty COSEM

COSEM objekty související s xDLMS používají pro přístup k atributům a metodám Cosem objektů dvě metody odkazování.

- Odkazování na logické jméno (LN)
- Odkazování na krátké jméno (SN)

V případě odkazování na LN jsou atributy a metody objektů COSEM odkazovány prostřednictvím logického názvu (COSEM_Object_Instance_ID) instance objektu COSEM, do které patří. V případě odkazování na SN jsou atributy a metody objektů COSEM mapovány na proměnné pojmenované DLMS. V souladu s tím jsou specifikovány dva prvky aplikační služby xDLMS. Jeden využívající služby xDLMS s odkazováním na LN a druhý pomocí služeb xDLMS s odkazováním na SN.

Klient ASO vždy používá xDLMS ASE s odkazováním na LN. Server ASO může používat buď xDLMS ASE s odkazováním na LN, nebo xDLMS ASE s odkazováním na SN nebo obojí.

Na straně serveru lze použít buď xDLMS ASE s odkazem na LN, nebo xDLMS ASE s odkazem na SN nebo oba xDLMS ASE. V případě potvrzených AA je referenční metoda vyjednávána během fáze založení AA prostřednictvím kontextu aplikace COSEM. Během doby platnosti stanoveného AA se nemění. Používání služeb LN nebo SN v daném AA je výhradní. V případě nepotvrzených a předem zavedených AA se očekává, že klient AL zná referenční metodu podporovanou serverem. Pokud server používá odkazování LN, jsou služby na obou stranách stejné. Když server používá SN odkazování na Client_SN_Mapper ASE v klientovi mapuje SN odkazování na LN odkazování nebo naopak [6].

3.4.1 xDLMS služby používané klientem s odkazováním na LN

V případě odkazování na LN jsou atributy a metody objektů COSEM odkazovány prostřednictvím identifikátoru instance objektu COSEM, do které patří. Pro tuto metodu odkazování jsou určeny následující doplňkové služby:

- Službu GET používá klient k vyžádání serveru, aby vrátil hodnotu jednoho nebo více atributů.
- Službu SET používá klient k vyžádání serveru, aby nahradil obsah jednoho nebo více atributů.
- Službu ACTION používá klient k vyžádání serveru, aby vyvolal jednu nebo více metod. Vyvolávací metody mohou znamenat odeslání parametrů vyvolání metody a přijetí návratových parametrů.
- ACCESS služba, unifikovaná služba, kterou může klient použít k přístupu k více atributům nebo metodám pomocí jediného dotazu [6].

Tyto služby může klient vyvolat potvrzeným nebo nepotvrzeným způsobem

3.4.2 xDLMS služby používané klientem s odkazováním na SN

V případě odkazování na SN jsou atributy a metody objektů COSEM mapovány na proměnné pojmenované DLMS. Služby xDLMS využívající odkazování SN jsou založeny na službách s proměnným přístupem DLMS.

- Službu Read (čtení) používá klient k vyžádání serveru k vrácení hodnoty jednoho nebo více atributů nebo k vyvolání jedné nebo více metod, pokud se očekávají návratové parametry. Je to potvrzená služba.
- Služba Write (zápis) je používána klientem k vyžádání serveru k nahrazení obsahu jednoho nebo více atributů nebo k vyvolání jedné nebo více metod, pokud se neočekávají žádné návratové parametry.
- Službu UnconfirmedWrite používá klient k vyžádání serveru, aby nahradil obsah jednoho nebo více atributů, nebo k vyvolání jedné nebo více metod, pokud se neočekávají žádné návratové parametry [6].

3.4.3 Nevyžádané služby

Nevyžádané služby jsou iniciovány serverem za předem definovaných podmínek, např. plány, spouštěče nebo události, aby informovaly klienta o hodnotě jednoho nebo více atributů, aniž by o ně klient požádal. Pro podporu oznámení událostí jsou k dispozici následující nevyžádané služby:

- s odkazováním na LN služba EventNotification;
- s odkazováním na SN služba InformationReport.

Pro podporu operace push je k dispozici služba DataNotification. Může být použit jak v aplikačních kontextech pomocí odkazování SN, tak LN [6].

3.5 OBIS

Systém OBIS (Object Identification System) určuje identifikaci datových objektů v komunikaci DLMS / COSEM. Kódy OBIS se používají jako logická jména datových objektů. Asociace uživatelů DLMS definuje a přiděluje kódy OBIS a udržuje seznam platných kódů OBIS [13]. Kódy ID se používají k identifikaci:

- Logických jmen různých instancí tříd rozhraní nebo objektů.
- Dat přenášených komunikačními linkami.
- Dat zobrazená na měřicím zařízení.

Kódy OBIS identifikují datové položky používané v měřicích zařízeních energie v hierarchické struktuře pomocí šesti hodnotových skupin A až F. Všechny tyto hodnoty nemusí být v identifikátoru (např. Skupiny A a B jsou často vynechány). Aby bylo možné rozhodnout, do které skupiny sub-identifikátor patří, jsou skupiny odděleny jedinečnými oddělovači: A-B:C.D.E*F

A Skupina hodnot A definuje médium (typ energie), ke kterému se měření vztahuje.

B Skupina hodnot B definuje číslo kanálu neboli číslo vstupu měřicího zařízení majícího více vstupů pro měření energie stejných nebo různých typů (například v koncentrátorech dat, registračních jednotkách). Takto lze identifikovat data z různých zdrojů. Definice pro tuto skupinu hodnot jsou nezávislé na skupině hodnot A.

C Skupina hodnot C definuje abstraktní nebo fyzické datové položky související s příslušným zdrojem informací. Například proud, napětí, výkon, objem, teplota. Definice závisí na hodnotě skupiny hodnot A.

D Skupina hodnot D definuje typy nebo výsledek zpracování fyzických veličin identifikovaných se skupinami hodnot A a C podle různých specifických algoritmů. Algoritmy mohou dodávat množství energie a poptávky, jakož i další fyzikální veličiny.

E Skupina hodnot E definuje další zpracování nebo klasifikaci množství identifikovaných skupinami hodnot A až D.

F Skupina hodnot F definuje ukládání dat, identifikovaných skupinami hodnot A až E, podle různých fakturačních období [7].

4 GURUX

Knihovna Gurux implementuje několik různých otevřených komunikačních protokolů. Nejčastěji se knihovna Gurux používá pro DLMS/COSEM. S komponentou protokolu Gurux DLMS/COSEM lze si vytvořit svůj vlastní systém AMR (Automatic Meter Reading) pro měřiče spotřeby elektřiny, a to prostřednictvím připojení TCP/IP, Terminal nebo sériové komunikace.

4.1 Open source

Všechny produkty Gurux Open Source jsou licencovány pod licencí GNU General Public License, verze 2 (GPLv2). To znamená, že tyto produkty lze používat zdarma, a to i pro komerční účely za určitých podmínek.

Distribuce softwaru s licencí GPL vyžaduje zveřejnění kompletního zdrojového kódu. Pokud se tedy používá software Gurux, je nutné zpřístupnit zdrojový kód, jinak dojde k porušení licenčních podmínek a tím dojde k porušení autorských práv společnosti Gurux. Tento požadavek platí bez ohledu na to, zda byly modifikovány produkty Gurux. Používání produktu jako takového je zdarma, ale pokud je použita jakákoliv část zdrojového kódu a použije se ve vlastních softwarových produktech, je povinnost uvolnit veškerý zdrojový kód těchto produktů jako Open Source. Pokud ovšem vlastník produktu nechcete uvolnit zdrojové kódy, je potřeba smlouva s duální licencí se společností Gurux [20].

4.2 GURUX software

Gurux poskytuje několik softwarových projektů:

Gurux Device Editor je software pro vytváření šablon měřičů, které odpovídají konkrétní verzi měřiče. Šablony se vytvářejí přidáním správných objektů, tabulek a registrů do šablony. Tato šablona se později používá k přidání měřičů pomocí jiného softwaru Gurux.

Gurux Director je program pro komunikaci s elektroměry. Měřiče jsou přidány do grafického uživatelského rozhraní. Pod každým měřičem je seznam dostupných objektů, ze kterých jde data číst, nebo zapisovat. GuruxDirector je vhodný pro malé množství měřičů, nebo je také vhodný pro použití při začínání s prací s knihovnou Gurux. Díky jeho grafickému rozhraní nejsou zapotřebí žádné programovací dovednosti. Hodnoty všech objektů se ukládají do paměti. Pro komunikaci se zařízením DLMS pomocí GXDLMSDirector musí být fyzické zařízení reprezentováno zařízením v aplikaci. Pro připojení je nutné vybrat výrobce čteného zařízení, dále nastavit typ spojení mezi fyzickým zařízením a aplikací. Director zobrazí všechny nabízená

data a následně se vybere objekty, které chceme číst. Se standardem DLMS/COSEM mohou výrobci používat jinou adresu klienta, adresu serveru a odkazování na logické jméno. Pokud je některý z těchto parametrů nesprávný, měřič neodpovídá.

Gurux AMI je podobný jako Gurux Director, nabízí aplikaci pro čtení měřičů. Lze zde nastavit plánovač a také lze nastavit, které objekty se mají číst z měřičů DLMS. Čtené hodnoty se ukládají do databáze. Gurux AMI UI je snadno použitelný doplněk uživatelského rozhraní pro Gurux Director. Director je aplikace, která může generovat zařízení a ovládat Gurux DLMS AMI [21].

4.3 Navázáni spojení

Knihovna Gurux umožňuje jednodušší komunikaci se zařízeními pomoci protokolu DLMS/COSEM. Inicializace připojení závisí na typu připojení a zařízení. Některá zařízení vyžadují před zahájením komunikace protokolem DLMS handshake protokolem IEC62056-21. První příkaz, který je povinný pro komunikaci DLMS/COSEM je žádost AARE. Tento příkaz je povinný pro všechny typy připojení a typy zařízení. AARE žádost sdělí zařízení, zda je použito ověření a zda je použito odkazování na dlouhé názvy LN, nebo krátké názvy SN. Tuto žádost lze generovat pomocí metody AARQRequest. Jakmile je přijata úplná odpověď, analyzujte ji pomocí metody ParseAAREResponse. Tato metoda nastaví příslušná nastavení pro komponentu GXCOSEM. Pokud vše proběhlo bez problému, spojení je nyní navázáno.

Association View popisuje, jaký typ objektů nabízí měřič, a liší se podle použité autentizace. Tyto data můžou být velmi velká, a za podmínek pomalého přenosového kanálu je vhodné tyto data ukládat, protože se nezmění, dokud nebude aktualizován software měřiče. Struktura dat se liší mezi výrobci a modely zařízení.

Data odpovědi jsou analyzována pomocí metody ParseObjects v komponentě GXCOSEM. Metoda vrací soubor objektů DLMS. Data mohou být následně čteny jako Generic Profile. Jedná se o speciální objekty, které místo hodnoty obsahují řádky a sloupce, takže lze je považovat za tabulku. Měřiče mohou vrátit různé objekty v závislosti na úrovni autentizace. Čtení dat je rozděleno na dvě části. Čtení datových objektů a čtení obecných objektů profilu.

Datové objekty se čtou pomocí metody Read. Data jsou analyzována pomocí metody GetValue, která vrací hodnotu ve formátu, které zařízení poskytlo, např. celé číslo nebo řetězec. Existují dva způsoby, jak číst Obecná data profilu. Zadáním konkrétního objektu nebo zadáním rozsahu. Ve vstupních parametrech je třeba nastavit, kde má být zahájeno čtení dat, pomoci indexu a kolik položek má být přečteno. Čtení dat tímto způsobem je generováno pomocí metod ReadRowsByEntry nebo ReadRowsByRange.

Na závěr je potřeba spojení ukončit odesláním požadavku na odpojení [20].

Data se ne vždy vejdou do jednoho paketu. V tomto případě lze data rozdělit do bloků a jeden blok do rámců. Komponenta Gurux DLMS to zpracovává automaticky. Při přijetí celého rámce se zkontroluje, zda je k dispozici více dat pomocí metody IsMoreData. Pokud je více dat, vygeneruje se nová žádost pro čtení pomocí metody ReceiverReady a zkontrolujte, zda jsou všechna data znovu přijata. Po obdržení celého rámce se zkontroluje, zda nedošlo k chybám [21].

4.3.1 Ověřování pomocí výzvy

Pokud je ověřování na úrovni High nebo vyšší, použije se zabezpečení. Po navázání spojení musí klient odeslat výzvu na server a server musí tuto výzvu přijmout. Po provedení analýzy zprávy AARE je toto ověření povinné. Pokud je vyžadováno ověření, klient odešle výzvu na server a pokud se vše podaří, server vrátí vlastní výzvu, kterou klient zkontroluje. DLMS podporuje tři různé zabezpečení přenosu [21]. Při použití zabezpečení přenosu je každý paket zabezpečen pomocí zabezpečení GMAC. Úroveň zabezpečení je:

- Autentizace
- Šifrování
- Autentizace + Šifrování.

4.4 DLMS server

Gurux poskytuje možnost vytvoření vlastního měřiče kompatibilního s DLMS. Nejprve je nutné definovat vlastní třídu serveru. Lze použít komponentu GXDLMSServer nebo GXDLMSSecureServer a následně implementovat metody pre a post, které jsou vypsané dále v textu. Následně se vytvoří komponenta pro definování rozhraní a spustí se naslouchání na daném rozhraní. Pokud server obdrží nová data, předá je metodě HandleRequest. Při obdržení dat od klienta, vystačí předat data do komponent serveru [20].

- IsTarget – Může existovat několik serverů(měřičů) používajících stejné připojení, například TCP spojení, nebo spojení sériovou linkou. Hlavní funkcí metody IsTarget je kontrola ze strany serveru, zda se klient připojil k serveru a zda je klient akceptován. Pokud klient nemá požadavek na připojení k danému serveru, metoda vrací hodnotu false.
- ValidateAuthentication – Tato metoda se používá k ověření, zda klient má přístup k připojení k serveru. Parametry autentizace informují o použité úrovni autentizace klienta a poskytují hesla klienta. Tato metoda se používá pro ověřování pouze na úrovni autentizace Low.

- Connected – Pokud metoda autentizace proběhne úspěšně lze použit dále metodu pro připojení.
- InvalidConnection – Pokud autentizace proběhne neúspěšně lze použít tuto metodu. Tato metoda může zjistit z jaké IP adresy se klient pokoušel připojit a zda se klient nepokouší připojit metodou Brute Force.
- Disconnected – Metoda se používá pro odpojení DLMS spojení na příkaz klienta, avšak spojení TCP se udržuje stále aktivní.
- PreGet – Metodu PreGet lze použít, pokud hodnoty nejsou v paměti nebo pokud server funguje jako jednotka pro sběr dat (koncentrátor) a hodnoty jsou čteny z jiného měřiče.
- PostGet – Tato metoda se používá po aktualizaci hodnot na serveru. Server pak může ukončit spojení.
- FindObject – Funkce se používá, pokud hledaný objekt na serveru nebyl nalezen v seznamu objektů. Je doporučováno ukládat všechny objekty, které se na serveru nachází, ukládat na seznam objektů. Může se však stát, že na uložení všech objektů na seznam nebude dostatek paměti. Pokud objekt nebyl nalezen, vrátí se klientovi chybová hláška.
- GetAttributeAccess – Metoda je volána ke kontrol, zda klient má oprávnění přistupovat k atributům objektu COSEM. Pokud klient nemá oprávnění pro daný objekt, pošle se klientovi chybová hláška o zamítnutí přístupu k danému objektu.
- PreRead – Metoda se volá před pokusem o čtení objektu COSEM. PreRead lze použít, pokud hodnoty nejsou v paměti nebo pokud server funguje jako jednotka pro sběr dat (koncentrátor) a hodnoty jsou čteny z jiného měřiče.
- PreWrite – Před pokusem o zápis do objektu COSEM se volá PreWrite. Server zde může před aktualizací například ověřit data.
- PostWrite – PostWrite je volán po zápisu hodnot do objektů. Server může uložit novou hodnotu do souboru nebo do databáze.
- GetMethodAccess – Metoda se používá ke kontrole, zda klient má oprávnění pracovat s objektem COSEM. Pokud klient nemá oprávnění pro daný objekt, obdrží chybovou hlášku o odepření přístupu.
- PreAction – Před pokusem o práci objektu COSEM se volá PreAction. Zde může server předat danou akci jinému zařízení.
- PostAction – Po provedení akce se volá PostAction. Zde může server zpracovat operace se soubory nebo s databází.

4.5 Dohoda mezi klientem a serverem

Klient a server může využívat různé funkce a metody pro práci s daty. Avšak ne všechny funkce a metody daný server nebo klient podporuje. Takže po připojení klienta k serveru, klient dává návrh služeb, které chce použít. Server zkontroluje, které služby podporuje, a vrací dohodnutou shodu. Na základě takto poslaných hodnot klient se dozví, jaké funkce jsou podporovány. Všechny služby nejsou nutně podporovány klientem ani serverem, jelikož ve standardu DLMS neexistují žádná omezení týkající se minimálních funkcí, které musí server podporovat. Taktéž podpora některých funkcí může záviset na úrovni použité autentizace. Zde jsou vypsány funkce, které mohou být vyžadovány [20].

- Obecné zabezpečení – Obecné zabezpečení podporuje server a používá se k zabezpečenému spojení mezi klientem a serverem pomocí symetrického nebo asymetrického šifrování.
- Obecný blokový přenos – Tento mechanismus se používá pro přenos dat, která jsou delší než maximální povolená délka PDU.
- Služba čtení (Read) – Čtení hodnot je podporováno serverem a používá odkazování na krátké názvy a slouží pro čtení dat ze serveru.
- Služba zápisu (Write) – Zápis dat stejně jako čtení dat používá odkazování na krátké názvy a slouží pro ukládáni dat na server.
- Služba nepotvrzeného zápisu (Un-confirmed write) – Nepotvrzený zápis se používá s odkazem na krátké názvy k zápisu hodnot na server, avšak bez potvrzení úspěšného zápisu, který by server posílal.
- Služba atributu 0 s podporou zápisu – Je sada atributů podporována serverem a slouží pro zápis všech COSEM objektů v jedné zprávě.
- Služba atributu 0 s podporou čtení – Je sada atributů podporována serverem a slouží pro čtení všech COSEM objektů v jedné zprávě.
- Služba priority správy – Prioritní správa používá odkazování na logické názvy a využívá se pro zpracování naléhavých zpráv.
- Blokový přenos pro požadavek čtení – Přenos bloku dat s požadavkem na čtení dat z měřiče, se používá pro data, která se nevejdou do jedné PDU bloku.
- Blokový přenos pro požadavek na zápis dat – Přenos bloku dat s požadavkem zápisu dat na server se používá pro data, která se nevejdou do jedné PDU. Příklad aktualizace obrazu (Firmware) může trvat více než jednu PDU a pak je použit blokový přenos.
- Služba mnohonásobného odkazování – Při čtení nebo zápisu několika objektů s jedním požadavkem se používá tato služba. Příklad použití například čtení dat ze seznamu požadavků.
- Služba informačních zpráv – Informační zprávy se používají s odkazováním na

krátké názvy.

- Služba přístupu – Použitím této funkce může klient provézt několik požadavků pro čtení, zápis nebo jiný příkaz v jednom příkazu.
- Služba parametrizovaného přístupu – Parametrizovaný přístup se používá s odkazováním na krátké názvy, pokud je Generic Profile čten podle rozsahu nebo záznamu.
- Služba selektivního přístupu – Selektivní přístup se používá s odkazováním na logické názvy, pokud je Generic Profile načten podle rozsahu nebo záznamu.
- Funkce Get – Funkce Get se používá s odkazem na logické názvy ke čtení hodnoty ze serveru.
- Funkce Set – Sada se používá s odkazem na logické názvy k zápisu hodnot na server.
- Služba notifikace událostí – Server používá oznámení událostí s odkazem na logický název k odesílání událostí pro daný cíl.

4.6 Push zprávy

Pomocí zpráv Push lze nakonfigurovat elektronický měřič energie prostřednictvím DLMS. Obvykle klient odešle žádost a měřič jako server tuto žádost přijme a pošle klientovi odpověď. S funkcí Push měřič odešle zprávu bez dotazu. Klient neposílá ani potvrzení, takže neexistuje žádný důkaz, že zpráva je skutečně předána. Rychlost předávání informací Push závisí na rychlosti zpracování přístupového bodu, nebo koncentrátoru dat, protože za jedním přístupovým bodem mohou být tisíce serverů. Komponenta Gurux.DLMS podporuje zprávy DLMS Push. Zprávy push jsou konfigurovány pomocí třídy GXDLMSPushSetup. Tyto zprávy lze použít k odeslání vybraného obsahu (PushObjectList) ze serveru na danou adresu. Pomocí ServiceType lze vybrat druh přenosového kanálu. Zprávu push lze spustit pomocí metody Activate nebo naplánovat čas odeslání. U zpráv push je jeden problém a to, že nepopisuje obsah zprávy push. Posílá pouze hodnoty dat [20].

4.7 Zabezpečení

Existují celkem tři metody, jak zašifrovat přenos dat

- AES-GCM-128
- ECDH-ECDSAAES-GCM-128SHA-256
- ECDH-ECDSAAES-GCM-256SHA-384

Symetrická kryptografie – AES-GCM-128 Tento způsob šifrování je nejpoužívanější pro zajištění spojení mezi klientem a serverem. V symetrické kryptografii jsou ode-

sílaná data zabezpečena pomocí šifrování GMAC. Pro správné šifrování dat jsou potřebné následující informace:

- Used secure level - Použitá úroveň zabezpečení
- System title - Název systému : tento parametr má hodnotu osmi oktetů, které identifikují server. První tři oktety obsahují ID výrobce a zbývajících pět oktetů obsahuje sériové číslo serveru (elektroměru).
- Block cipher key - Blok šifrovacího klíče :tento parametr má hodnotu 16 oktetů a definuje tajný šifrovací klíč. Tento parametr je potřebné získat od výrobce měřiče a každý výrobce má svůj vlastní klíč. Knihovny Gurux používají výchozí standardní klíč DLMS: 000102030405060708090A0B0C0D0E0F.
- Authentication key - Ověřovací klíč : Ověřovací klíč je hodnota 16 oktetů, která se používá v AAD (Additional Authentication Data). Tyto informace jsou potřebné od výrobce měřiče a každý výrobce má svůj vlastní klíč.
- Invocation Counter - Počítadlo vyvolání nebo počítadlo snímků je hodnota čítače, která se zvyšuje při každém šifrování zprávy. Zvýšení počitadla vyvolání způsobuje, že zprávy jsou šifrovány odlišně, dokonce i když obsah zprávy je stejný. Počitadlo vyvolání se nepoužívá nešifrované zprávy. Hodnota čítače vyvolání se obvykle ukládá do datového objektu. Pokud čítač vyvolání není správný, měřič zprávu nepřijme. Na měřiči může být několik čítačů vyvolání. Pro každé zabezpečené přidružení obvykle existuje jeden čítač vyvolání. U některých měřičů může existovat vlastní čítač vyvolání pro připojení optického portu a vlastní čítač vyvolání pro připojení TCP / IP. Před spuštěním šifrovaného připojení musí klient navázat normální připojení (obvykle se to provádí bez ověření) a vyčíst počítadlo vyvolání z měřiče. Po přečtení je připojení hodnoty čítače ukončeno.

Pro dešifrování je pouze nutný Block cipher key. Used secure level tento parametr udává úroveň zabezpečení:

- Žádné
- Autentizace
- Šifrování
- Šifrování a autentizace

Asymetrická kryptografie (kryptografie veřejného klíče) Další dvě metody používají metodu asymetrické kryptografie. To znamená že obě strany komunikace mají dva klíče. Veřejný a soukromý klíč. Existují dvě možnosti navázání asymetrického kryptografického připojení k měřiči. Metoda dohoda na klíč nebo použití metody Diffie-Hellman. Pro připojení k měřiči jsou zapotřebí následující informace. Použitá úroveň zabezpečení, použitá bezpečnostní sada, autentizační klíč, soukromý a veřejný klíč pro klienta, veřejný klíč serveru (měřič), název systému klienta a název systému serveru.

Klient nejprve musí poslat certifikát serveru. Server ovšem může mít pouze jeden certifikát obdržený klientem. Pokud tedy na server přistupuje víc klientů musí pokaždé aktualizovat certifikát. Poté musí server vygenerovat veřejný a soukromý klíč.

Při použití komunikace pomoci Diffie-Hellman je nejprve potřeba vytvořit klíč pro dohodu a podpisový klíč a aktualizovat je na server. Po navázání spojení jsou dočasné klíče generovány a sdíleny tajně. To znamená, že šifrovací klíč se u každého připojení liší. To se provádí pokaždé po zprávě AARQ / AARE [20].

4.8 Připojení k serveru pomocí dynamických IP adres.

Pro připojení k serveru pomocí připojení GPRS, musí mít server statickou IP adresu (SIM karta). Pokud má server dynamickou IP adresu, musí tedy on zahájit komunikaci. Server tedy může zasílat zprávy které jsou předem naplánovány podle seznamu. Nejde tedy kdykoliv přistupovat k serveru. Ve veřejných sítích GPRS jsou spojení vždy navázána z modemu na GPRS a nikdy z GPRS na modem. Není možné, aby klientská aplikace otevřela připojení.

- Statická IP adresa: Server nebo čtecí systém může zahájit připojení.
- Dynamická IP adresa: Server vždy zahájí připojení.

Když servery používají dynamické IP adresy nebo brána firewall blokuje přístup k měřiči, je možné, že se měřič naplánuje na připojení k serveru pomocí připojení TCP / IP v daném časovém intervalu. Všechny severy nepodporují připojení podle rozvrhu v daném časovém intervalu.

Připojení klienta na server je následující. Klientská aplikace jako server naslouchá na daném portu a čeká připojení TCP / IP od měřičů (serverů). Když je navázáno nové připojení TCP / IP, začne se chovat jako klient DLMS. Hlavní myšlenkou je naslouchat příchozí spojení. Po navázání komunikace. připojíme komunikaci ze serveru (klientské aplikace) na klienta. Tak se použije se připojení, které navázal server.

Pro čtení více serverů současně, je nutné pro každý server vytvořit vlastní vlákno [20].

5 Výsledky práce

Tato kapitola se zabývá realizací a testováním softwarové knihovny pro agregaci dat ze serverů protokolem DLMS/COSEM. Postupně je zde popsáno nastavení serverů, a vyzkoušení komunikace pomocí programu Gurux Director. Dále tato kapitola obsahuje popis kódu a vývoj softwarové knihovny. Na závěr je sepsána sada testů, které byly provedeny.

Během testování se posílaly OBIS kódy, které byly předem rozděleny do scénářů. Softwarová knihovna pro agregaci dat poskytuje pět možných scénářů.

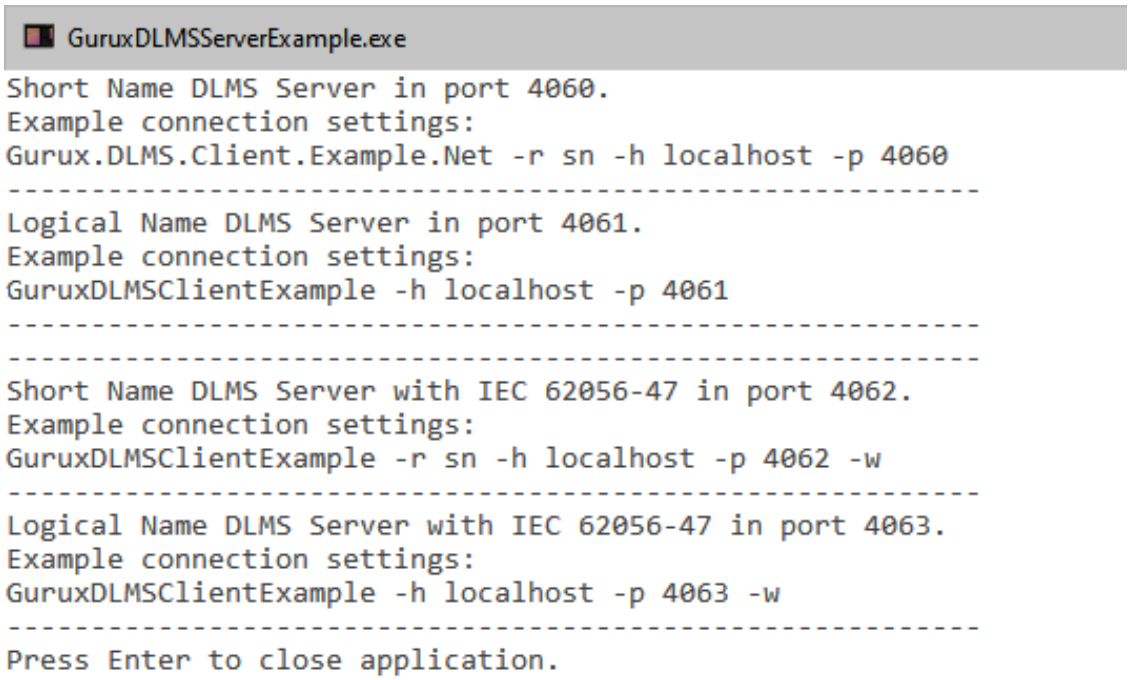
1. Scénář pro čtení jednoho OBIS kódu, který obsahuje jednu hodnotu.
2. Scénář pro čtení jednoho OBIS kódu, který obsahuje více hodnot, celkem devět. Tento OBIS kód sloužil pro testování objemových rozdílů OBIS kódů a je popsán níže v kapitole „Objemové rozdíly OBIS kódů“.
3. Scénář pro čtení deseti OBIS kódů dotazující se na napětí 5.1.
4. Scénář pro čtení 64 OBIS kódů.
5. Scénář pro čtení všech OBIS kódů, které se nacházejí na serveru.

Výsledný projekt, který zahrnuje DLMS/COSEM server a softwarovou knihovnu pro agregaci dat na straně klienta je dostupný na webové službě GitHub: *https://github.com/xtomic04/GuruxDP.git*.

5.1 Gurux server

Před samotným vytvářením klientské aplikace bylo nutné nastavit server. Knihovna Gurux nabízí čtyři druhy serverů 5.1, které se liší svým zabezpečením. Po výběru jednoho serveru je nutné nastavit, na jakých portech má tento server naslouchat. Ve výchozím nastavení je tento port nastaven na hodnotu 4060. Druhým povinným parametrem je určení, na jaké IP adrese má server běžet. Pokud se neurčí žádná IP adresa, server ve výchozím nastavení běží na IP adrese 127.0.0.1, což je lokální smyčka počítače a zároveň běží na IP adrese na kterou je nastaveno síťové rozhraní počítače. Ne server lze přiřadit požadované OBIS kódy a také jejich hodnoty.

Přidání nových OBIS kódů se provádí implementací do zdrojového kódu serveru. Je důležité definovat jaký typ objektu má být nový OBIS kód. OBIS kódy, které se budou používat pro testování, mají datový typ *CGXDLMSData*. To znamená, že obsahují dvě hodnoty. První hodnota u všech objektů je logický název, neboli hodnota OBIS kódů. Druhá hodnota je samotná hodnota objektu. Zde se ukládá hodnota napětí.



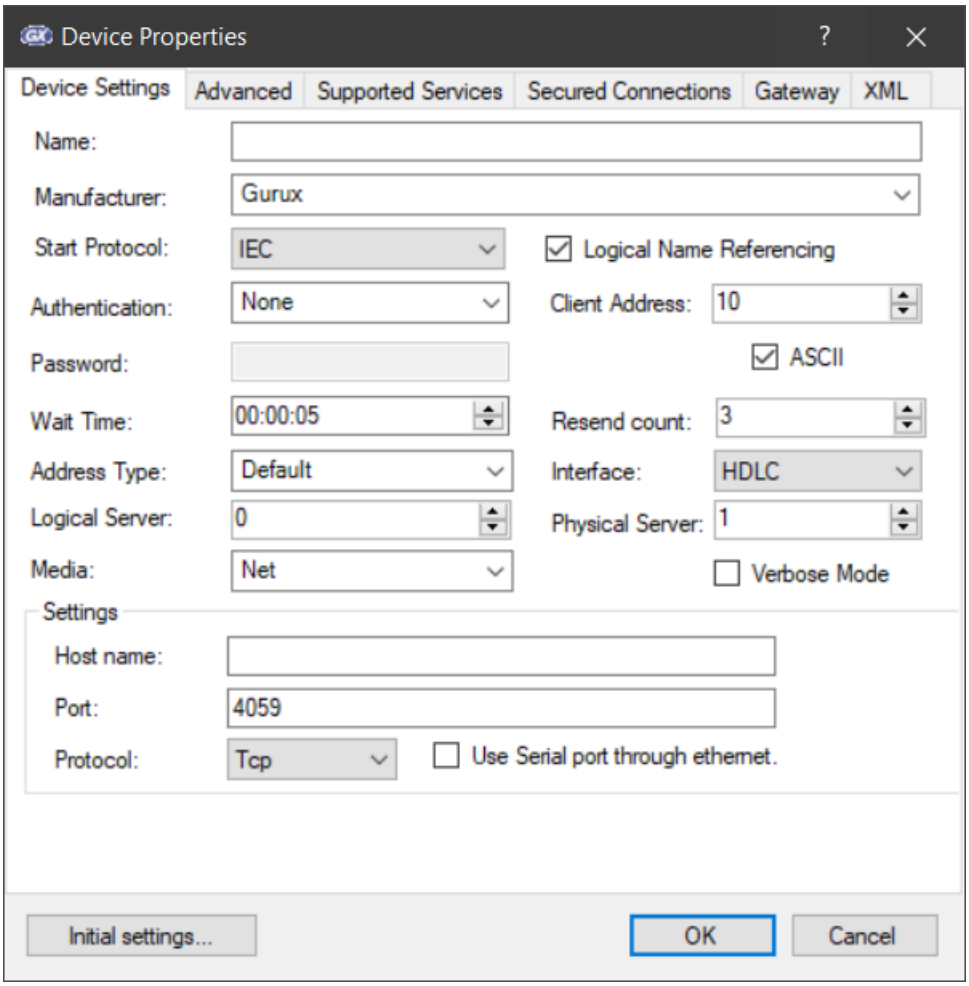
Obr. 5.1: Gurux server DLMS/COSEM.

5.2 Gurux Director

Pro počáteční vyzkoušení, zda je server správně nastaven, a zda je možná komunikace s ním, byl použit Gurux Director. Tento program je právě vhodný pro počáteční práci s knihovnami GURUX. Jeho grafické uživatelské rozhraní nevyžaduje po uživateli žádné programátorské schopnosti. Pro vyzkoušení přenosu je nutné přidat zařízení. Pro přidání zařízení je nejprve potřeba nastavit parametry zařízení, viz obr. 5.2. Gurux Director umožňuje nastavit výrobce, použitý protokol (IEC nebo DLMS), typ rozhraní (HDLC, WRAPPER, MBUS), režim autentizace, typ média (Net, Seriál, Terminal), hostname, port, protokol transportní vrstvy (TCP, UDP), atd. Gurux Director také umožní uložit do souboru data načtené ze serveru pro pozdější analýzu. Po úspěšném připojení zobrazí seznam načtených OBIS kódů a umožní přečtení jejich hodnot. Pokud je možné se připojit na vytvořený server, je jasné, že běží v pořádku a lze zahájit vytvoření vlastní klientské aplikace.

5.3 Softwarová knihovna pro agregaci dat

Jako základ pro vývoj softwarové knihovny pro agregaci dat ze serverů DLMS/COSEM byla použita knihovna GuruxDLMSClient. Vytvořená aplikace slouží pro sběr dat DLMS/COSEM podle různých předem definovaných scénářů. Pro úspěšné připojení k serveru je nutné nastavit správnou IP adresu a také port, na kterém server



Obr. 5.2: Gurux Director.

běží. Aplikace se spouští z příkazového řádku. Jako povinný argument aplikace je nutné zadat ip adresu serveru. Port je nastaven ve výchozím režimu na hodnotu 4060. Po spuštění, aplikace vyžaduje zvolit v minutách celkovou délku testu sběru dat ze serveru. Následně je třeba zadat v jakých intervalech se má aplikace dotazovat na server. Hodnotu se zadává ve vteřinách. Jako poslední je třeba vybrat jeden z předem definovaných scénářů, které jsou popsány na počátku této kapitoly. Každý scénář obsahuje jiný počet a jiné OBIS kódy. Seznam OBIS kódů, které obsahuje první scénář je vypsán v tabulce 5.1. Tyto kódy se nacházejí i v třetím scénáři. Čtvrtý scénář se dotazuje na předem definovaných 64 OBIS kódů.

Po následném spuštění aplikace vypisuje do konzole hodnoty OBIS kódů, viz obr. 5.3. Na závěr aplikace vypíše do konzole počet všech provedených spojení na server a celkový počet neúspěšných spojení. Během testu sběru dat, pokud dojde k neúspěšnému spojení, vypíše se do konzole chybová hláška. Taktéž pokud se aplikace dotazuje na OBIS kód, který se nevyskytuje na serveru, aplikace informuje vypsáním hlášky do konzole. Během testu se veškerá síťová komunikace zachytávala

OBIS kód	Popis
1-0:1.8.0.255	Činná energie +A
1-0:1.8.1.255	Činná energie +A tarif T1
1-0:1.8.2.255	Činná energie +A tarif T2
1-0:1.8.3.255	Činná energie +A tarif T3
1-0:1.8.4.255	Činná energie +A tarif T4
1-0:2.8.0.255	Činná energie -A
1-0:2.8.1.255	Činná energie -A tarif T1
1-0:2.8.2.255	Činná energie -A tarif T2
1-0:2.8.3.255	Činná energie -A tarif T3
1-0:2.8.4.255	Činná energie -A tarif T4

Tab. 5.1: Seznam OBIS kódů v 1. scénáři

programem Wireshark. Záznam komunikace se uložil do souboru a bude sloužit pro pozdější analýzu.

5.4 Popis zdrojového kódu klientské aplikace

Knihovna na začátku definuje a nastavuje všechny potřebné parametry, které jsou nutné pro správné fungování. Postupně se definuje úroveň logů, které aplikace během spuštění vypisuje do konzole. Dále nastavuje adresu klienta a serveru, úroveň autentizace, která je ve výchozím nastavení vypnutá. Následně se nastaví typ rozhraní pro použití wrapperu a na závěr nastaví výchozí hodnoty pro port, která se rovná hodnotě 4060 a pro adresu, která je ve výchozím nastavení definována na lokální smyčku (localhost). Aplikace umožňuje při spuštění využít vstupních parametrů.

- -h – IP adresa nebo hostname (ve výchozím nastavení je nastaven localhost)
- -p – číslo portu (výchozí hodnota 4060)
- -S – číslo sériového portu
- -i – použití protokolu IEC
- -a – nastavení autentizace (ve výchozím nastavení autentizace není)
- -P – heslo pro autentizaci
- -c – adresa klienta (ve výchozím nastavení 16)
- -s – adresa server (ve výchozím nastavení 1)
- -n - adresa serveru jako sériové číslo
- -r – použití krátkých názvů nebo logických názvů
- -w – použití wrapru nebo HDLC
- -t – úroveň logování do konzole
- -g – seznam OBIS kódů pro čtení ze serverů

Obr. 5.3: Výpis OBIS kódů z konzole.

Aplikace tedy následně zjišťuje, které vstupní parametry byly použity a nastavuje jejich hodnoty. Do této části, kód poskytuje firma Gurux. Následně je kód vytvořený pro specifický účel této práce pro sběr dat a testování.

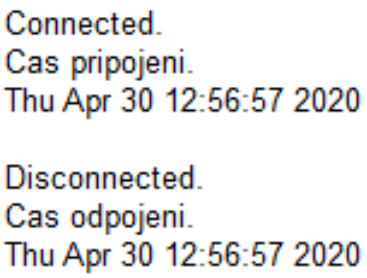
Aplikace následně vyžaduje nastavení testování od uživatele. Je nutné nastavit počet serverů na které se má aplikace během testování připojit. Následně je třeba definovat celkovou dobu testování v minutách a v jakých intervalech se má aplikace dotazovat na servery. Jako poslední parametr je nutné vybrat jeden z předem definovaných scénářů, které se liší v počtu OBIS kódů, které se v dotazu posílají na server.

Aplikace následně se pokusí vytvořit spojení. Nejprve se ovšem musí vyjednat zabezpečení klienta. Tato funkce vyžaduje následující parametry. UseLogicalNameReferencing, adresa klienta, adresa serveru, použitá autentizace, heslo pokud ho vyžaduje autentizace a použité rozhraní (wrapper nebo HDLC). Pokud se požaduje šifrovaný přenos je potřeba definovat parametry SystemTitle, BlockCipherKey a AuthenticationKey. Tato část je detailněji popsána v kapitole zabezpečení komunikace. Aplikace znovu zkontroluje hodnotu portu a adresy a pokud jsou tyto hodnoty va-

lidní, provede spojení. Dále se spojení inicializuje a provede se zobrazení asociace. Pokud se všechno provede bez problému, může se přikročit k dotazování OBIS kódů na server. Po úspěšném čtení dat se spojení ukončí a aplikace čeká nastavenou dobu uživatelem, dokud neprovede další spojení a čtení dat. Na závěr po skončení testu se vypíše do konzole celkový provedený počet spojení a počet neúspěšných spojení.

5.5 Vyhodnocení příjmu dat

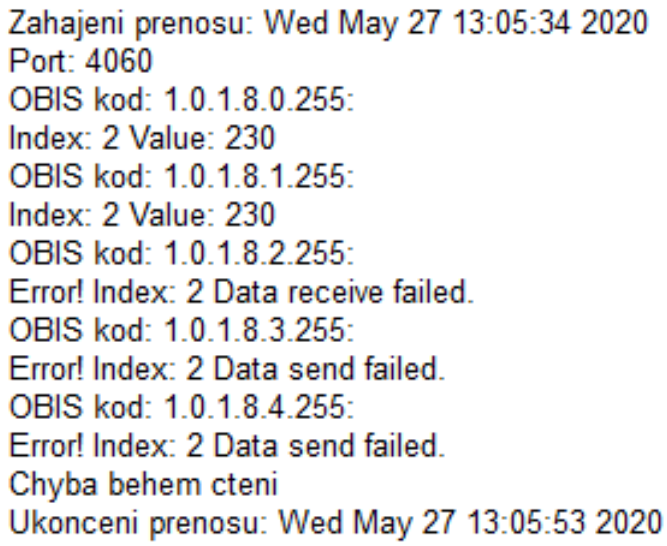
Aplikace během testování je vybavena ukládáním logů pro pozdější analýzu příjmu dat. Je stěžejní pro správné vyhodnocení dat vzít v úvahu kolik bylo pokusů o spojení mezi klientem a serverem a kolik spojení opravdu proběhlo. Při absenci těchto logů, by mohlo dojít ke zkreslení výsledku testování. Například při testování přenosu datového objemu jednotlivých scénářů je důležité zaznamenávat, kolik bylo celkem provedených úspěšných přenosů, jelikož vyskytnou se během testů neúspěšné pokusy o připojení a data se ze serveru nepošlou, datový objem bude logicky menší. Každý server ukládá své logy do samostatného souboru. Při každém úspěšném spojení vypíše status připojeno *(Connected)* nebo odpojeno *(Disconnected)*. Zároveň při změně statusu zaznamená aktuální datum a čas pro lepší pozdější analýzu. Server tedy loguje pouze úspěšné provedené spojení. Příklad logu je na obrázku 5.4



Obr. 5.4: Log serveru.

Logování na straně klienta neboli aplikace pro sběr dat je obdobná. Aplikace vytváří samostatné soubory s logy pro jednotlivé servery. Názvy souborů mají tvar dataExport[port]. Mezi závorky port se ukládá číslo portu, ke kterému se aplikace snaží připojit. Při úspěšném připojení se do logů ukládá aktuální datum a čas připojení a odpojení. Následně při úspěšném připojení aplikace ukládá hodnotu OBIS kódů. Při neúspěšném připojení aplikace ukládá hlášku o neúspěšném pokusu o připojení k serveru. Pokud dojde k přerušení komunikace během čtení dat, do logu se ukládá kdy k chybě došlo 5.5.

Na straně klienta se vytváří také log, který ukládá všechny připojení do jednoho souboru. Na konci tohoto souboru jsou pak informace o průběhu celého testu. Jsou



Obr. 5.5: Log klienta při neúspěšném čtení dat.

zaznamenány celkové počty pokusů o spojení, předpokládány počet OBIS kódů, počet přijatých OBIS kódů, úspěšnost příjmu dat, vypočtena chybovost přenosu a počet neúspěšných spojení.

Aplikace jak na straně klienta, tak serveru zobrazuje zpětnou vazbu také do konzole. Na straně serveru se tyto informace neliší od logů, které se ukládají. Při každém úspěšném spojení vypíše status připojeno nebo odpojeno. Zároveň při změně statusu zaznamená aktuální datum a čas, viz obr. 5.6.

```
Logical Name DLMS Server with IEC 62056-47 in port 4060.
Example connection settings:
GuruxDLMSClientExample -h localhost -p 4060 -w
------------------------------------------------------------
Press Enter to close application.
Connected.
Cas pripojeni.
Wed May 20 09:51:51 2020

Disconnected.
Cas odpojeni.
Wed May 20 09:51:52 2020
```

Obr. 5.6: Výpis logu serveru do konzole.

Na straně klienta se do konzole vypisuje více informací, než které se ukládají do logů, viz obr. 5.7. Aplikace na začátku spojení zobrazí hlášku *"Zahajeni prenosu spojeni"* a vypíše aktuální datum a čas. Následně informuje, na jaké číslo portu se snaží

připojit. Pokud se aplikaci podaří úspěšně navázat spojení informuje dále úspěšně provedenou inicializaci spojení a GETAssociationView. Následně vypíše hodnotu OBIS kódu, kterého hodnotu chce číst. Pokud OBIS kód se nachází na serveru, vypíše jeho hodnotu. Pokud se OBIS kód na serveru nenachází, vypíše hlášku „OBIS kod neexistuje".

```
Zahajeni prenosu spojeni.
Wed May 20 09:51:50 2020
_____________________________________________________________
Port: 4060
_____________________________________________________________
InitializeConnection
GetAssociationView


OBIS kod: 1.0.1.8.0.255:
Index: 2 Value: 230

Ukonceni prenosu.
Wed May 20 09:51:52 2020


Celkovy pocet spojeni: 1
Predpokladany pocet OBIS kodu: 1
Pocet prijatych OBIS kodu: 1
Uspesnost 1/1
Chybovost: 0,000
Celkovy pocet neuspesnych navazani spojeni se servrem: 0
```

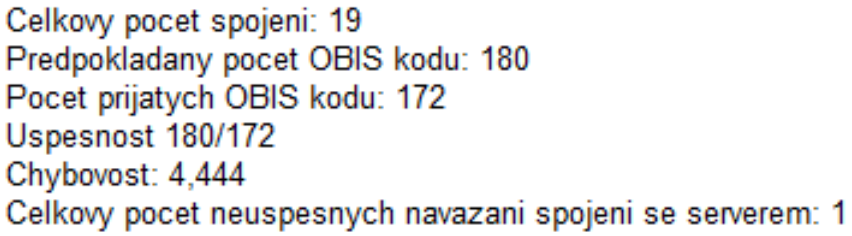
Obr. 5.7: Výpis logu klienta do konzole po úspěšném provedeném spojení.

Pokud se ovšem spojení nepodaří navázat do konzole se vypíše hláška „Connect failed Invalid parameter." Následně selžou i další funkce pro Inicializaci spojení, která vypíše i chybový kód a také funkce pro AARQ požadavek vypíše chybový kód.

V případě že se spojení přeruší po jeho navázání může se stát, že některé hodnoty OBIS kódu se poslaly. V takovém případě jejich hodnoty se vypíšou do konzole. A hodnoty OBIS kódů, které se už nestihly poslat, zastoupily chybové hlášky, které oznamují chybný přenos. Na konci funkce GetReadOUt vypíše chybový kód a také funkce ReleaseRequest vypíše chybový kód.

Po zakončení přenesu se objeví hláška *„Ukonceni prenosu"* a znovu je zaznamenán čas a datum pro zjištění, jak dlouho celé spojení trvalo. Po skončení tetování do konzole se vypíše celkový počet pokusů o spojení a celkový počet spojení, které se nepovedlo navázat.

Na obrázku 5.8 jsou zobrazeny finální statistiky měření, během kterého došlo k chybě. Celkem bylo provedeno 19 pokusů o připojení, z toho 18 pokusů bylo úspěšných. Během těchto 18 pokusů mělo přijít celkem 180 OBIS kódů. Z výsledných statistik je patrné, že bylo přijato pouze 172. To znamená, že došlo k chybě během čtení dat. Výsledná chybovost tedy činí 4,4%.



Obr. 5.8: Výpis logu klienta po skončení testování s chybou.

5.6 Spouštění serverů

Pro testování je potřeba spustit několik serverů. Pro jednodušší spouštění byly vytvořeny dva dávkové soubory (.bat). První dávkový soubor *pripojeni.bat* využívá nástroj plink pro připojení pomocí příkazového řádku.

```
plink -batch -ssh pi@192.168.10.208 -pw raspberry
cd /home/guruxServer/GURUX/GuruxDLMSServerExample/bin/;
./gurux.dlms.server.bin -p %1
```

Následující příkaz spustí program plink, následně parametr *-batch* deaktivuje všechny interaktivní výzvy. Bez tohoto parametry by bylo nutné každý server postupně spustit. Následně parametr *-ssh pi@10.34.20.14* se připojí pomocí SSH protokolu ve tvaru login@ipAdresa. Parametr *-pw* označuje heslo pro přihlášení. Poslední část příkazu označuje cestu ke spustitelnému souboru pro server. Na konci příkazu je %1, které slouží pro vstupní parametr celého příkazu. Tento vstupní parametr mění číslo portu, na kterém server běží.

Druhý dávkový soubor *serverLoopPi.bat* slouží pro nastavení počtu serverů.

```
FOR /L %%i IN (4060,1,4061) DO (
start /B pripojeni.bat %%i
timeout 1
)
```

Tento dávkový soubor obsahuje smyčku for. Smyčka začíná na čísle 4060, což označuje také číslo portu prvního severu. Následně inkrementuje číslo o jeden. Ve smyčce se příkaz start, který spouští první dávkový soubor *plink.bat*. Parametr */B* slouží pro spuštění aplikace bez vytváření nového okna. Bez tohoto parametru by se otevřelo tolik nových oken, kolik by se spustilo serverů. Parametr *%%i* slouží pro

nastavení portu pro server. Na závěr souboru je příkaz *timeout 1*, který slouží pro spuštění serverů po jedné vteřině, aby nedošlo k zahlcení.

5.7 Testování

Bylo uděláno několik testů. Během měření klient a server vždy běžely na jiném fyzickém zařízení. Byly použity dva počítače, které byly spojeny ethernetovým kabelem, nebo počítač a mikropočítač RaspberryPi. Celá komunikace se zachytávala Wiresharkem pro následnou analýzu. Cílem pozdější analýzy bylo nalezení odpovídající hodnoty napětí. Také bylo nutné zjistit, kolik dat se nachází na jednotlivých vrstvách modelu OSI. Zde je vypsán seznam provedených testů:

- Testování na mikropočítači RaspberryPi, emulujícího elektroměr.
- Maximální možný počet spuštěných serverů na RaspberryPi, emulující několik elektroměrů (například agregace dat na koncentrátoru).
- Analýza zachycené síťové komunikace.
- Rozpady spojení TCP.
- Testování a analýza objemu přenesených dat.
- Objemové rozdíly OBIS kódů.
- Objem dat a čas přenosu jednotlivých scénářů.
- Test zatížení linky.
- Zabezpečená komunikace šifrováním.

5.7.1 Testování na Raspberry Pi

V rámci testování bylo nutné zprovoznit server, která představuje měřák na mikropočítači Raspberry Pi. Na začátku bylo nutné zkontrolovat konektivitu mezi Raspberry Pi a počítačem. Na Raspberry Pi příkazem ifconfig se zjistilo nastavení síťového adaptéru, viz obr. 5.9.

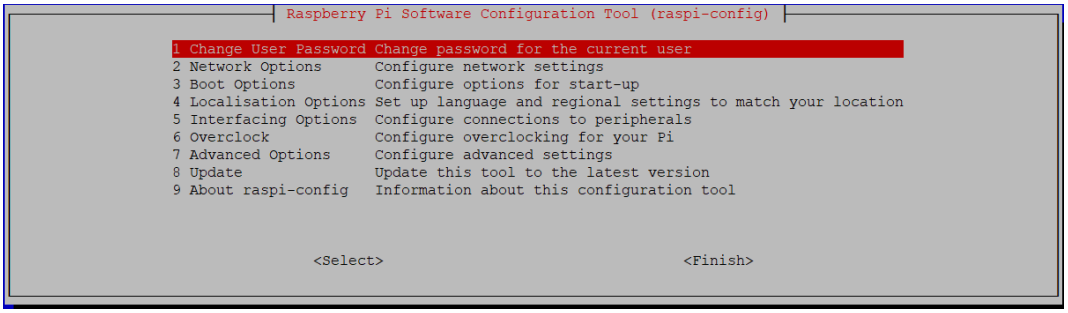
Následně bylo nutné nastavit parametry síťového adaptéru na počítači, aby byly ve stejné síti. Jednalo se o parametry IP adresa a maska sítě. Aby bylo možné vzdáleně přistupovat z jiného počítače do příkazové řádky bylo potřeba povolit na mikropočítači SSH protokol. Příkazy *sudo raspi-config* jsme se dostali do konfiguračního nástroje (modrá obrazovka s možnostmi v šedém poli uprostřed), kde je možně nastavovat základní parametry mikropočítače, viz obr. 5.10.

Dále *Interfacing oprions/SSH/Enable/YES*. Protokol SSH je v tomto bodě na Raspberry Pi povolen a lze přistupovat z jiného počítače. Na počítači lze pro vzdálený přístup použit příkazovou řádku nebo program Putty. V programu Putty je potřeba zadat IP adresu Raspberry PI a zvolit protokol SSH. Po připojení konzole

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.34.20.14  netmask 255.255.255.252  broadcast 10.34.20.15
        inet6 fe80::ba27:ebff:fe28:bb1b  prefixlen 64  scopeid 0x20<link>
        ether b8:27:eb:28:bb:1b  txqueuelen 1000  (Ethernet)
        RX packets 66  bytes 6673 (6.5 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 64  bytes 7609 (7.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 19  bytes 1462 (1.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 19  bytes 1462 (1.4 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Obr. 5.9: Natavení síťového adaptéru na RaspberryPi.

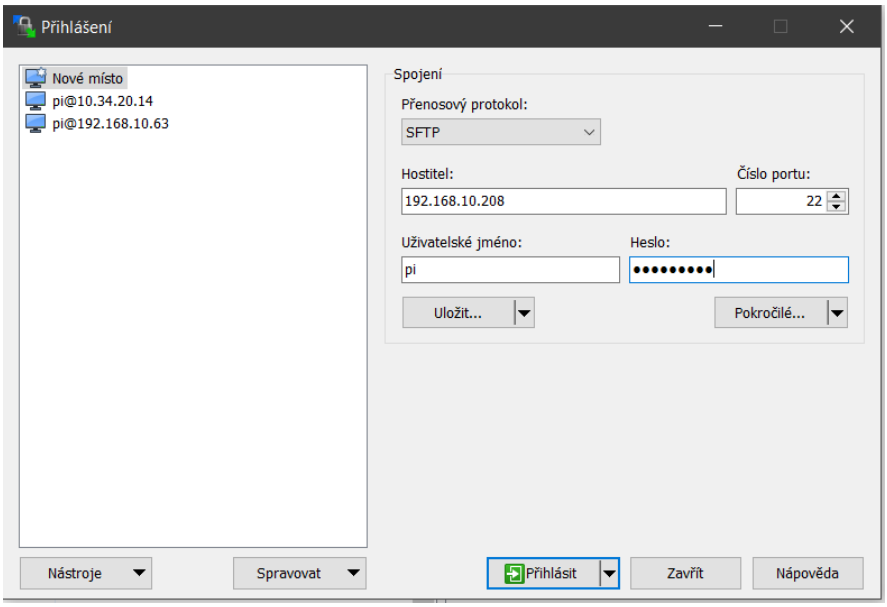


Obr. 5.10: Menu základního nastavení RaspberryPi.

vyzve uživatele k zadání uživatelského jména (login) a hesla. Ve výchozím nastavení je uživatelské jméno *PI* a heslo *Raspberry*. Komunikace je tedy zajištěna a lze přikročit k nahrání serveru na mikropočítač.

Pro přenos souborů byl použit program WinSCP. Pro připojeni je třeba nastavit parametry Hostitel (ip adresa mikropočítače), a pro přihlášení Uživatelské jméno a heslo, viz obr. 5.11.

Po nakopírování souborů je třeba projekt se serverem zkompilovat. Gurux už obsahuje soubor *makefile*, který obsahuje instrukce pro kompilování. Pro zkompilování projektu je třeba ve složce *development* vytvořit další dvě složky *lib* a *obj*. Nyní lze příkazem *make* spustit kompilaci projektu. Po úspěšné kompilaci se vytvoří soubor gurux.dlms.server.bin, který je spustitelný. Následně příkazem *./gurux.dlms.server.bin* lze spustit server. Za příkaz *./gurux.dlms.server.bin* lze přidat vstupní parametr *-p* a číslo, které definuje, na jakém portu bude server naslouchat.



Obr. 5.11: Nastavení připojení pro přenos souborů.

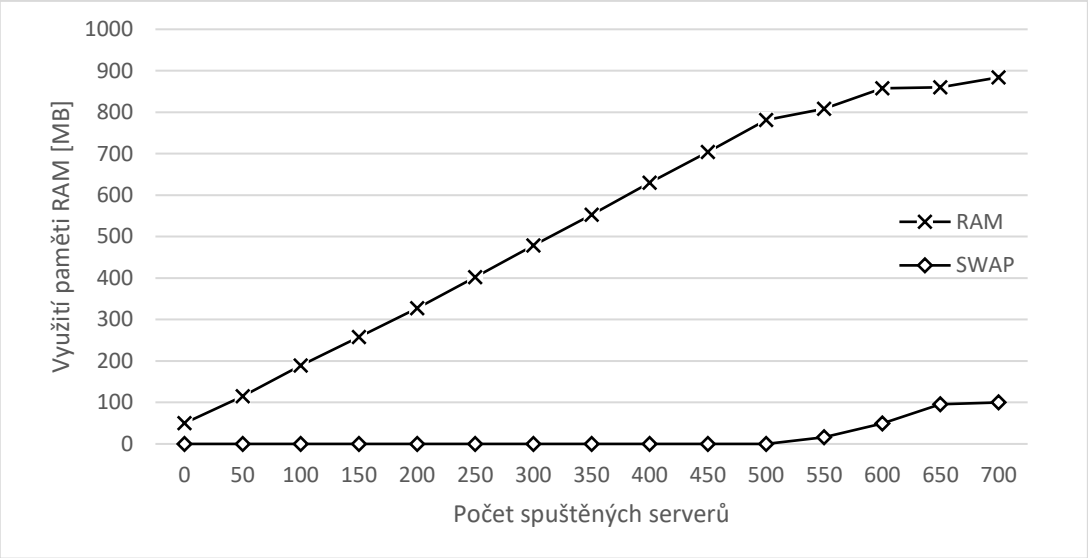
5.7.2 Maximální počet serverů

Každý server samozřejmě vyžaduje nějaké hardwarové nároky. V dalším testu se zkoušelo maximální počet serverů, které je schopen mikropočítač spustit. Test probíhal postupným spouštěním padesáti serverů, a pokaždé se kontrolovalo využití Procesoru a operační paměti. Každý server se spouštěl na samostatném vlákně. Nejprve se zkontroloval stav využití hardwaru před spuštěním serverů. Tato hodnota byla referenční hodnotou pro další porovnání. Stav využití hardwaru se kontroloval příkazem *htop*. Příkaz *htop* je především pro interaktivní prohlížení procesů, avšak poskytuje také vizuální informace o stavu procesoru, swap paměti a operační paměti.

Mikropočítač před spuštěním severů využíval 50 MB operační paměti a necelé 1% z kapacity procesoru. Po spuštění padesáti serverů stouplo využití operační paměti o 65 MB na 115 MB a využití procesoru stouplo na 1,2 %. Tak test pokračoval po maximální využití hardwaru mikropočítače.

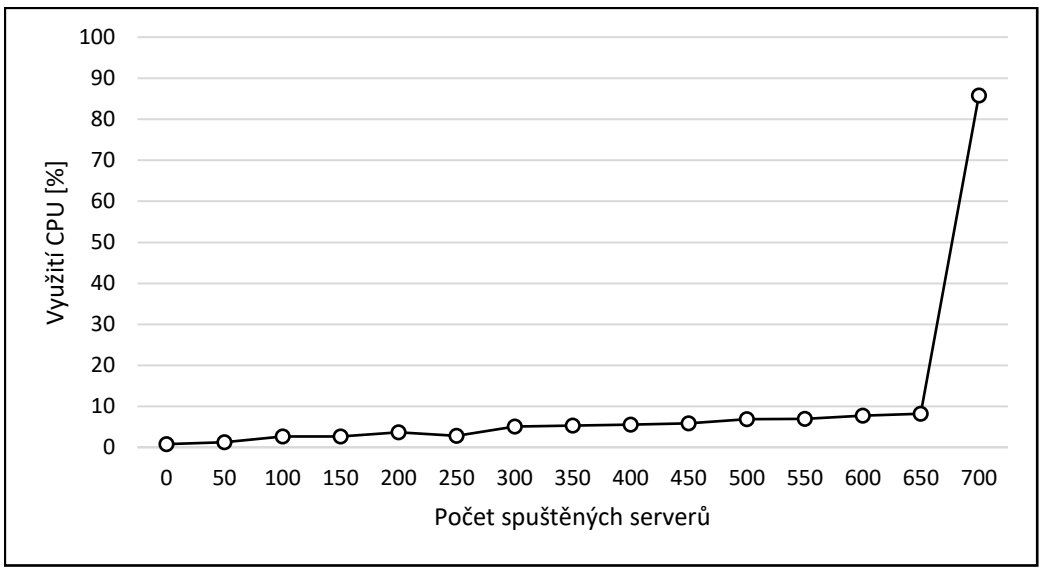
Z obrázku 5.12 je patrné, že využití operační paměti v závislosti na počtu spuštěných serverů do počtu 500 je lineární. Využití operační paměti vždy stoupo v rozmezí od 65 MB do 77 MB. Dalším zvýšením počtu serveru, mikropočítač začíná využívat odkládacího prostoru (Swapu). Velikost této paměti je 100 MB a mikropočítač ho začíná využít při 550 serverech pro uvolnění operační paměti. Při 650 serverech mikropočítač využívá 96 MB/100 MB z odkládací paměti a 860 MB z operační paměti. Při dalším zvýšení počtu serverů dojde už k zahlcení což je patrné z grafu využití procesoru, viz obr. 5.13.

Co se týče využití procesoru mikropočítače, jsou zde nároky serverů mnohem



Obr. 5.12: Závislost využití operační paměti na počtu spuštěných serverů.

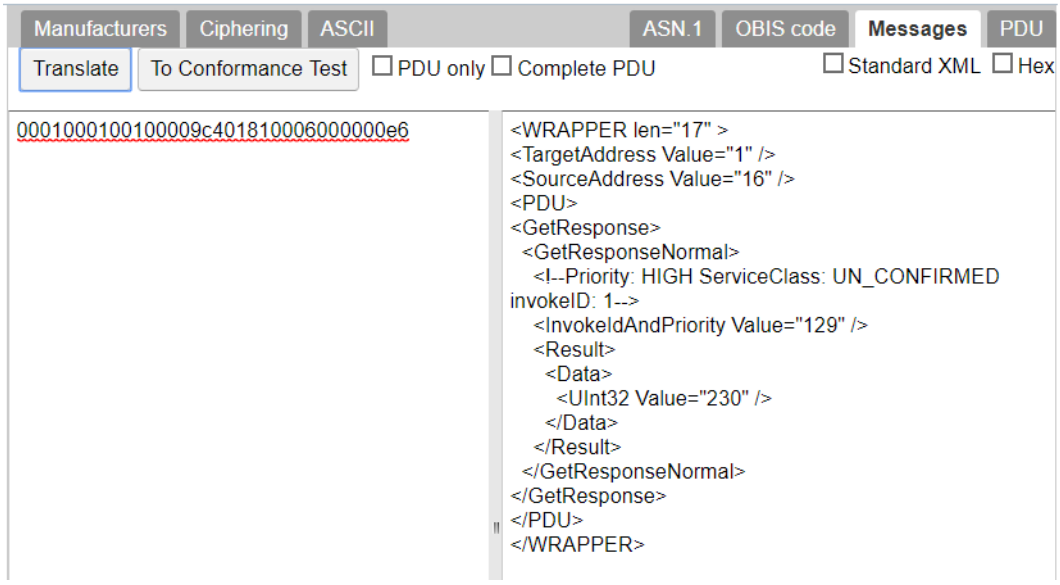
menší. Využití procesoru při každém zvýšení počtu serverů stoupl pouze v jednotkách procent až do momentu, kdy došlo k zahlcení pamětí SWAP a operační paměti. Využití procesoru mikropočítače lze pozorovat na obrázku 5.13. Jako maximální počet spuštěných serverů lze uvažovat hodnotu 650 serverů.



Obr. 5.13: Závislost využití CPU na počtu spuštěných serverů.

5.7.3 Analýza komunikace

Po detailní analýze paketů se podařilo odpovídající hodnotu napětí zjistit. Nalezená hodnota napětí byla ve Wiresharku v aplikační vrstvě v hexadecimální tvaru. Pro dekódování byl použit nástroj GuruxDLMSTranslator. Tento nástroj byl vyvinut za účelem překladu dat do formátu XML, který je lépe čitelný pro uživatele. Výsledné napětí je vidět na obrázku 5.14.



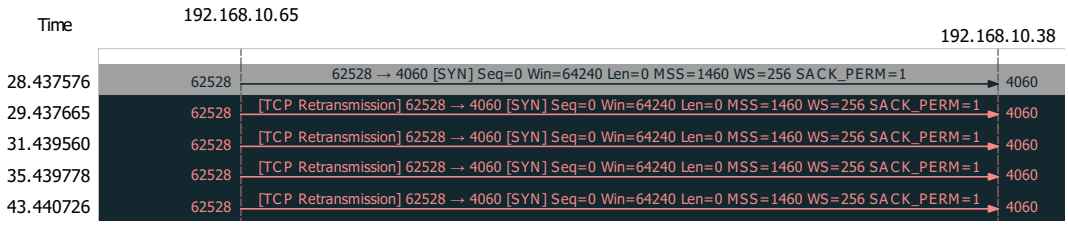
Obr. 5.14: Gurux DLMS Translator.

5.7.4 Rozpady spojení

Během testování může dojít ke dvěma možnostem, kdy bude server nedostupný. V prvním případě je server nedostupný už před zahájením spojení.

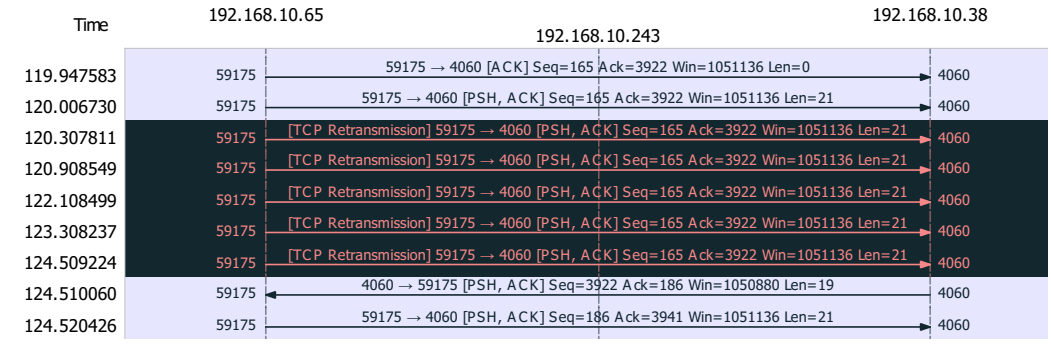
V simulaci tohoto scénáře byl během komunikace odpojen server, což simuluje nedostupnost serveru. Během zachytávání komunikace je patrné, jak protokol TCP se snaží navázat spojení. Ze zachycené komunikace analyzátorem Wireshark je patrné, jak klient posílá na server TCP paket s nastaveným příznakem (flags) SYN, pro navázání spojení. Jelikož klient neobdržel TCP paket s nastaveným příznakem SYN, ACK opakuje tento přenos. Protokol TCP při prvním nepovedeném připojení opakovaně posílá TCP Retransmission, a to celkem čtyřikrát v předem definovaných intervalech. První TCP Retransmission se posílá po jedné vteřině po nepovedeném pokusu. Tato hodnota jedné vteřiny je hodnota TCP Retransmission Timeout (RTO). Další TCP Retransmission se pošle v intervalu RTO*2, to znamená za další dvě vteřiny od předchozího paketu. Další interval se znovu zdvojnásobí od předchozího paketu. TCP Retransmission se tedy posílají v intervalech 1 s, 2 s, 4 s, 8 s,

viz obr. 5.15. Když ani na pátý pokus se spojení nedaří navázat, aplikace tedy čeká 16 vteřin, aplikace už neposílá TCP Retransmission a vyhodnotí tento pokus jako neúspěšný a pokračuje v testu.



Obr. 5.15: Neúspěšný pokus o spojení.

Druhá možnost, kdy může dojít k výpadku spojení, je po navázání TCP spojení, během čtení dat. Pokud klient znovu neobdrží odpověď od serveru, posílá TCP Retransmission čtyři krát. Zde se ovšem liší hodnota RTO. V předchozím případě se tato hodnota rovnala jedné vteřiny. V tomto případě se hodnota RTO rovná 0,3 vteřiny. TCP Retransmission se tedy posílají v intervalech 0,3 s, 0,6 s, 1,2 s a 2,4 s, viz obr. 5.16. Pokud klient ani za další interval neobdrží odpověď, tedy za 4,8 vteřiny, ukončí přenos. Do konzole se vypíše chybová hláška. První chybová hláška *recv failed 10060* informuje o tom, že připojení vypršelo. Pokus o připojení selhal, protože připojená strana po určité době neodpověděla správně nebo navázané připojení selhalo, protože připojený hostitel neodpověděl. Další chybová hláška *send failed 10054* informuje o tom, že klient resetoval spojení, z důvodu násilného uzavření spojení. Na závěr se do konzole zobrazí další dvě chybové hlášky. První *GetReadOut failed* informuje o tom, že došlo k chybě během čtení dat. A další chybová hláška *ReleaseReauest failed* informuje o tom, že spojení nebylo řádně ukončeno.

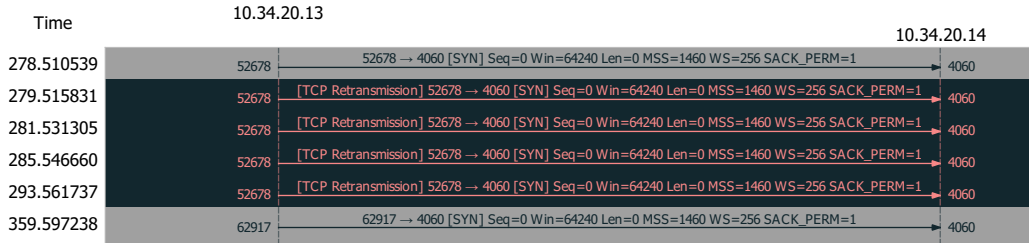


Obr. 5.16: TCP Retransmission během čtení dat.

Pokud ovšem do pátého pokusu se spojení znovu naváže, v takovém případě aplikace vyčká a po obnovení spojení dokončí přenos bez ztráty dat.

Jelikož se rozpady spojení chovají jinak, pokud server běží na platformě Windows nebo na platformě Linux, bude popsán i rozpad spojení, pokud server běží na mikropočítači Raspberry Pi. Počítač a RaspberryPi byly přímo spojeny ethernetovým kabelem.

Nejprve byl vyzkoušený test pro připojení k serveru, který není spuštěn. klient posílá na server TCP paket s nastaveným příznakem (flags) SYN, pro navázání spojení. Jelikož klient neobdržel TCP paket s nastaveným příznakem SYN, ACK opakuje tento přenos. Zde se navázání spojení nijak neliší od předchozího testu. TCP Retransmission se posílají celkem čtyři krát s počátečním intervalem TCP Retransmission Timeout (RTO) jedna vteřina, viz obr. 5.17.

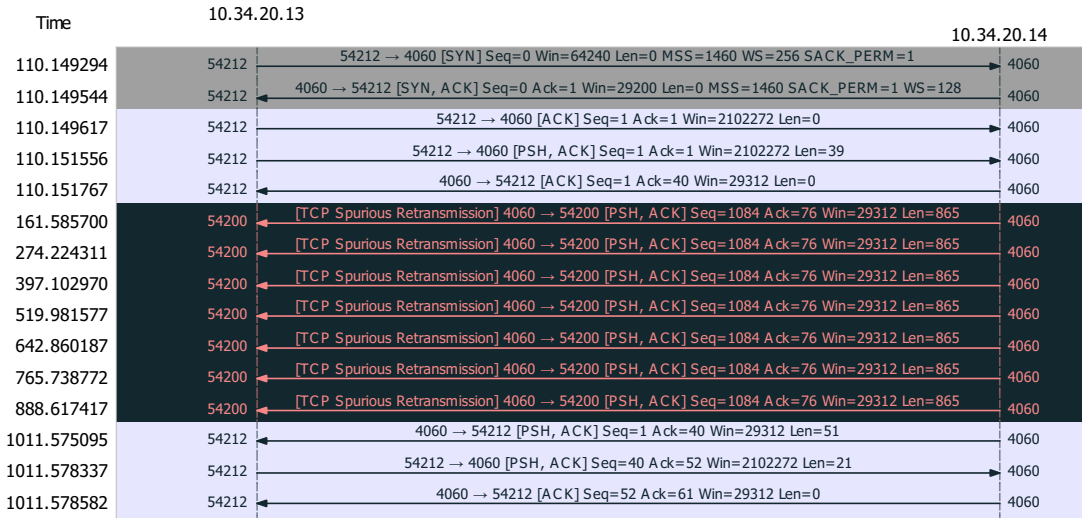


Obr. 5.17: TCP Retransmission během čtení dat.

Pokud dojde k rozpadu spojení během čtení, aplikace v předchozím testu na platformě Windows čekala 9,3 vteřiny, pokud se spojení opět nenaváže, jinak měření ukončila. Pokud ovšem běží server na mikropočítači Raspberry Pi a dojde k rozpadu spojení během čtení, aplikace okamžitě ukončí spojení TCP a pokračuje s vypsáním chybových hlášek. Ze zachycené komunikace analyzátorem Wireshark je patrné, že aplikace ze strany klienta pošle jenom jeden paket TCP Retransmission v intervalu 0,3 s. Aplikace vypíše chybovou hlášku *recv failed 10061*, která informuje o nedostupnosti síti. Došlo k pokusu o poslání soketu do nedosažitelné sítě. To obvykle znamená, že místní software nezná žádnou cestu k dosažení vzdáleného hostitele. Následně vypíše další chybové hlášky, které jsou stejné jako v předchozím testu, tedy chyba *send failed 10054* informuje o tom, že klient resetoval spojení, z důvodu násilného uzavření spojení. *GetReadOut failed* informuje o tom, že došlo k chybě během čtení dat. A další chybová hláška *ReleaseReauest failed* informuje o tom, že spojení nebylo řádně ukončeno.

Pokud spojení bude opět obnoveno, aplikace na straně klienta se pokusí provést další spojení. Ze zachycené síťové komunikace lze vyčíst, že klient a server navázal spojení pomocí trojcestného Handshake. Následně klient pošle žádost o inicializaci spojení. Zde ovšem nastává problém. Klient už uzavřel TCP spojení a chce pokračovat v novém spojení, ale server nedostal informaci o tom, že se první TCP spojení ukončilo, a tak se ho snaží udržet. Server tedy posílá pakety TCP Spurious Re-

transmission. Toto spojení je snaží udržet 15minut, viz obr. 5.18. Tento časový úsek je dán hodnotou *tcp_retries2*. Tato hodnota ovlivňuje časový limit připojení TCP, když opakované přenosy RTO zůstávají nepotvrzeny. Výchozí hodnota je 15 opakování a poskytuje hypotetický časový limit 924,6 sekund. Po uplynutí této doby, první spojení TCP server ukončí, naváže druhé TCP spojení, na které klient čeká a přenos dále v pořádku pokračuje.



Obr. 5.18: TCP Retransmission během čtení dat.

5.7.5 Testování a analýza přenášeného objemu dat

V dalším testu bylo za úkol zjistit zatížení linky při dotazu na server na jeden OBIS kód, který obsahoval hodnotu napětí. Analýza provozu se opět provedla programem Wireshark. Po navázání TCP spojení došlo celkem k pěti výměnám informací na každé straně klient-server. Pokud se nepočítá navázání a ukončení TCP spojení, tak celá konverzace obsahovala dvanáct paketů. Těchto dvanáct paketů zabralo celkem 2123 bytů. Pokud do konverzace započítáme i navázání a ukončení TCP spojení, tak celé spojení obsahuje 18 paketů, a tyto pakety obsahují celkem 3131 bytů.

Dále byly provedeny dlouhodobé testy. Každý test byl spuštěn po dobu jedné hodiny a aplikace se dotazovala na server v intervalu jedné minuty. Celkem bylo provedeno šest testů:

1. Dotazování na 1 server a 1 OBIS kód.
2. Dotazování na 1 server 10 OBIS kódů.
3. Dotazování na 5 serverů a 1 OBIS kód.
4. Dotazování na 5 serverů a 10 OBIS kódů.
5. Dotazování na 1 server a 1 OBIS kód (bez ukončení spojení).
6. Dotazování na 1 server 10 OBIS kódů (bez ukončení spojení).

	1 OBIS kód		10 OBIS kódů	
	Tx (B)	Rx (B)	Tx (B)	Rx (B)
1 server	41k	158k	81k	185k
1 sever (bez ukončení)	25k	133k	67k	172k
5 serverů	207k	747k	413k	942k

Tab. 5.2: Objem dat

První čtyři testy bylo možné provést softwarovou knihovnou pro agregaci dat bez zásahu do zdrojových kódů. Testy byly provedeny postupně a veškerá komunikace byla zachytávána analyzátorem síťového provozu Wiresharkem. Zachycené pakety po skončení testování byly použity pro analýzu dat, zejména jeho objemu.

Pro pátý a šestý test bylo nutné změnit zdrojový kód. Jelikož norma DLMS vyžaduje po provedeném dotazu vždy ukončit navázané spojení. Cílem těchto dvou testů bylo pokusit se zmenšit datový objem přenosu dat. Jelikož navázání a ukončení spojení TCP je následující.

Po otevření nového připojení TCP odešle klient paket SYN na server a aktualizuje svůj stav na SYN-SENT. Server poté zašle odpověď SYN-ACK klientovi, který změní svůj stav připojení na SYN-RECEIVED. Klient následně odpoví pomocí ACK a připojení je zřízeno. Nyní jsou klient a server připraveni přenášet data.

Pro ukončení spojení klient odešle FIN paket na server a aktualizuje svůj stav na FIN-WAIT-1. Server obdrží od klienta požadavek na ukončení a odpoví ACK. Po odpovědi bude server ve stavu CLOSE-WAIT. Jakmile klient obdrží odpověď ze serveru, přejde do stavu FIN-WAIT-2. Takže zatímco připojení je ukončeno z pohledu klienta, server musí také ukončit jeho připojení. K tomu dochází ihned poté, co server odešle poslední ACK. Následně server pošle paket FIN a klient odpoví paketem ACK. Je zde veliká režie přenosu. Celkem se pro navázání spojení a ukončení pošle 7 paketů.

Záměrem testu bylo, že při prvním dotazu se na navázalo spojení, a to se udržovalo během celé hodiny, až do skončení testu. Po poslání posledního dotazu se spojení ukončilo. Ve výsledku došlo k úspoře dat, jelikož nebylo nutné posílat pakety pro navázání spojení a jeho ukončení při každém novém dotazu na server. Nevýhodou těchto dvou testů je, že se blokuje komunikační kanál i když jím neprochází žádná data.

Z tabulky 5.2 je patrné, že test bez ukončení spojení snížil objem dat při přenosu. Pro menší objem dat je tento rozdíl procentuálně největší. Ve srovnání s testem při dotazu na jeden server a jeden OBIS kód. Pro poslaná data tento rozdíl činí 39% a pro přijata data rozdíl činí 15%. Při navýšení počtu OBIS kódů se procentuální rozdíl snížil. Pro poslaná data je rozdíl 17% a pro přijatá data rozdíl činí 7%.

5.7.6 Objemové rozdíly OBIS kódů

Jak již bylo řečeno výše v teoretické části OBIS kódy identifikují datové objekty v komunikaci DLMS/COSEM. Kódy OBIS se používají jako logická jména datových objektů. Tyto datové objekty však nejsou všechny stejné. Například OBIS kód 1.0.1.8.0.255, která se v aplikaci pro shromažďování dat používá pro dotaz na server pro jeden OBIS kód, obsahuje pouze jednu hodnotu. Existují však OBIS kódy, definující datové objekty, které obsahují více hodnot. V tomto testu bude porovnán datový objem různě velkých datových objektů, které definuje pouze jeden OBIS kód.

První OBIS kód, který bude obsahovat pouze jednu hodnotu, bude OBIS kód 1.0.1.8.0.255. Tento OBIS kód obsahuje hodnotu činné energie A+. Pomocí analyzátoru síťové komunikace Wireshark, byl zachycen přenos dat. Celkem bylo přeneseno i s navázáním a ukončením přenosu 18 paketů TCP. Přenos dat obsahoval tedy 12 paketů TCP. Těchto dvanáct paketů zabralo celkem 2123 bytů. Pokud do konverzace započítáme i navázání a ukončení TCP spojení, tak celé spojení obsahuje 18 paketů, a tyto pakety obsahují celkem 3131 bytů.

Druhý OBIS kód, který obsahuje více hodnot má hodnotu 0.0.1.0.0.255. Tento OBIS kód slouží k uložení aktuálního času a obsahuje celkem devět hodnot:

1. Logical Name – Název objektu.
2. Čas – Aktuální datum a čas.
3. Časové pásmo.
4. Status.
5. Datum začátku letního času.
6. Datum konce letního času.
7. Odchylka – Při použití letního času odchylka v minutách.
8. Povolení letního času.
9. ClockBase – Definuje, odkud pocházejí informace o času.

Znovu pomocí analyzátoru síťové komunikace Wireshark, byl zachycen přenos dat. Celkem bylo přeneseno i s navázáním a ukončením přenosu 32 paketů TCP. Přenos dat obsahoval tedy 26 paketů TCP. Těchto 26 paketů zabralo celkem 2402 bytů. Pokud do konverzace započítáme i navázání a ukončení TCP spojení, tak celé spojení obsahuje 32 paketů, a tyto pakety obsahují celkem 4166 bytů.

5.7.7 Objem dat a čas přenosu scénářů

Tento test se detailněji zaměří na datový objem a čas přenosu jednotlivých scénářů. Budou porovnány čtyři scénáře.

1. Dotaz na 1 OBIS kód, který obsahuje jednu hodnotu.
2. Dotaz na 1 OBIS kód, který obsahuje devět hodnot.
3. Dotaz na 10 OBIS kódů, které obsahují jednu hodnotu.

4. Dotaz na 64 OBIS kódu, ale na serveru se jich bude nacházet 11.
5. Dotaz na 64 OBIS kódu.
6. Dotaz na všechny OBIS kódy, které se nacházejí na serveru.

Jednotlivé scénáře byly zachytávány síťovým analyzátorem Wireshark. V první části se test zaměřoval na datový objem jednotlivých scénářů. Tyto hodnoty jsou důležité, jelikož Gurux umožňuje posílání dat po síti v GPRS/LTE připojení. A jelikož se platí za množství poslaných dat, je důležité zjistit kolik jednotlivé scénáře vyžadují dat.

Z tabulky 5.3 ze sloupce Celkové množství dat je zajímavé porovnání mezí dotazováním na jeden OBIS kód, který obsahuje jiný počet hodnot. I když se aplikace dotazuje pouze na jeden OBIS kód je rozdíl v objemu dat dvojnásobný. Taky zajímavé porovnání v objemu dat mezi dotazem na 10 OBIS kódů a 64 OBIS kódů. Zde se počet OBIS kódu výrazně liší, ale rozdíl v objemu dat je pouze 5 %. Zde je důležité si uvědomit jakým způsobem se aplikace dotazuje na jednotlivé OBIS kódy.

V prvním případě, kdy se aplikace dotazuje na jeden OBIS kód o různém počtu hodnot je rozdíl v objemu dat 50 %. Aplikace se totiž dotazuje na různé hodnoty postupně. To znamená, že v případě, kdy má OBIS kód jednu hodnotu, klient posílá jeden dotaz a server vrací jednu odpověď. Ale v případě, kdy má OBIS kód více hodnot, v tomto konkrétním případě devět hodnot, jelikož je to OBIS kód 0.0.1.0.0.255, který slouží k uložení aktuálního času a obsahuje celkem devět hodnot. Klient posílá postupně devět dotazů a server vrací postupně devět odpovědí. Zde se tedy nakumuluje rozdíl v datovém objemu.

V druhém případě, kdy se aplikace dotazuje na různý počet OBIS kódů není rozdíl mezi 10 OBIS kódy a 64 OBIS kódy v objemu dat tak markantní, pouze 5 %. Totiž scénář, který obsahuje zmíněných 64 OBIS kódů, tak na serveru se jich nachází v tomto případě pouze jedenáct. Klient ale neposílá 64 dotazů na server ale jenom jedenáct. Jelikož po navázání spojení, ještě před čtením dat obdrží klient funkcí *Association View* jaký typ objektů nabízí server. Klient tedy ví, jaké OBIS kódy server vlastní. Klient se tedy dotazuje pouze na OBIS kódy, od kterých ví, že dostane od serveru odpověď. Takže v porovnání mezi scénáři s 10 OBIS kódy a 64 OBIS kódy se počet dotazu na server liší minimálně.

Datový objem tedy závisí na několika faktorech. První je velikost jednotlivých OBIS kódů. Je důležité si uvědomit, kolik jeden OBIS kód obsahuje hodnot. Druhým faktorem je, jaké OBIS kódy server obsahuje.

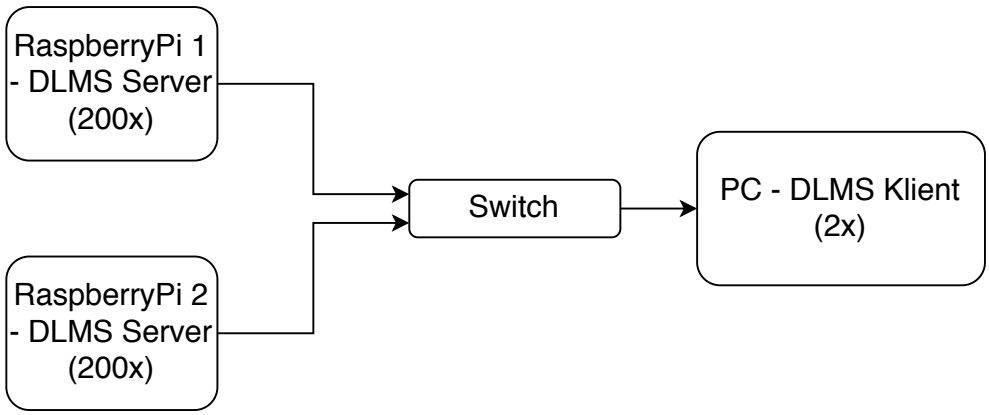
V druhé části testu, budou porovnávány doby trvání přenosu jednotlivých scénářů. Čas přenosu dat závisí pouze na počtu OBIS kódů a počtu hodnot v jednotlivých OBIS kódech ve scénáři. Aplikace postupně obsluhuje OBIS kódy, které se nachází ve scénáři. Z tabulky ze sloupce čas přenosu je patrné, že čas jednotlivých scénářů postupně roste, jelikož scénáře jsou seřazeny od nejmenšího po největší.

	Tx (B)	Rx (B)	Celková data (B)	Čas přenosu (s)
1 OBIS kód (1 hodnota)	744	2503	3247	0.1213
1 OBIS kód (9 hodnot)	1342	4965	6307	0.2458
10 OBIS kódů	1482	3205	4687	0.1869
64 OBIS (11 na serveru)	1672	3286	4958	0.3649
64 OBIS (vše na serveru)	5542	8991	14533	0.6973
Všechny OBIS kódy	13459	20768	34227	2.1256

Tab. 5.3: Objem dat a čas přenosu

5.7.8 Zatížení linky

Tento test se zaměřoval na maximální zatížení linky. Zapojení hardwaru je zobrazeno na obrázku 5.19. Byly zapotřebí dva mikropočítače Rapsberry Pi. Každý měl svoji IP adresu a na každém mikropočítači běželo současně 200 DLMS serverů emulující 200 elektroměrů. Oba mikropočítače byly zapojeny do přepínače, který byl následně připojen do počítače. Na počítači byli současně spuštěni dva klienti, jelikož jeden klient je schopen se připojit pouze na jednu IP adresu.



Obr. 5.19: Zapojení.

Následně z počítače byly spuštěny testy, které trvaly pět minut a dotazovaly se na servery v intervalu jedné vteřiny. Během testu každý klient se připojil na server 2000 krát. Cílem bylo pokusit se zjistit maximální bitovou rychlost přenosu dat na linkové vrstvě. Síťová komunikace byla zaznamenávána analyzátorem síťového provozu Wiresharkem.

Zaznamenané výsledky jsou zobrazeny v tabulce 5.4. Výsledky v rychlosti přenesených dat se zjišťoval pomocí analyzátoru síťového provozu Wiresharkem a jeho funkcí. Tento program umožňuje zjištění přenosové rychlosti na jednotlivé zařízení. Rychlost odeslaných dat se pohyboval v rozmezí od 36 kb/s do 41,1 kb/s. Rychlost

přijatých dat se pohybovala v rozmezí od 122 kb/s do 138,5 kb/s.Následně byly provedeny testy, kdy se klient dotazoval postupně jenom na jeden server. A byly zjištěny podobné rychlosti přenosu na linkové vrstvě.

	Tx (bit/s)	Rx (bit/s)
RaspberryPi 1	41,1k	138,5k
RaspberryPi 2	36,4k	122,8k

Tab. 5.4: Rychlost přenosu dat

5.7.9 Zabezpečení komunikace

Ve výchozím nastavení pro přenos dat není nastavené žádná autentizace nebo šifrování. Gurux nabízí tři úrovně zabezpečení. Autentizace, šifrování a jejich kombinace. Cílem následujícího testu bylo pokusit se data při přenosu zašifrovat a pokusit se je analyzovat.

Pro zašifrování dat existují tři metody, ale Gurux pro jazyk ANSI C++ zatím zprostředkoval pouze zabezpečení symetrickou šifrou AES-GCM-128. Pro zjištění nastavení všech správných parametrů byl nejdřív využit Gurux Director. Po úspěšném navázáni spojení v Gurux Director bylo možné začít s implementací kódu v softwarové knihovně pro agregaci dat. Bylo nutné nastavit úroveň zabezpečení na šifrování a následně definovat parametry *SystemTitle*, *BlockCipherKey* a *AuthenticationKey*. Pro správnou funkčnost je potřeba tyto hodnoty nastavit stejně i na straně serveru.

Výpis 5.1: Příklad implementace nastavení šifrování.

```
1  cl.GetCiphering()->SetSecurity(DLMS_SECURITY_ENCRYPTION);
2  std::string hex = "4D4D4D0000BC614E";
3  GXHelpers::HexToBytes(hex , tmp);
4  cl.GetCiphering()->SetSystemTitle(tmp);
5  hex = "000102030405060708090A0B0C0D0E0F";
6  GXHelpers::HexToBytes(hex, tmp);
7  cl.GetCiphering()->SetBlockCipherKey(tmp);
8  hex = "D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF";
9  GXHelpers::HexToBytes(hex, tmp);
10 cl.GetCiphering()->SetAuthenticationKey(tmp);
```

Následně byl proveden hodinový test přenosu dat s intervalem jedné minuty. Aplikace se dotazovala na jeden server na jednu hodnotu OBIS kódu. Následnou analýzou zachycených dat byl zjištěn objem přenesených dat.

Došlo ke zjištění, že zašifrovaná data mají větší datový objem než data bez jakéhokoliv zabezpečení. Srovnání objemů dat je patrné v tabulce 5.5. Rozdíl v zaslaných

	1 OBIS kód		10 OBIS kód;	
	Tx (B)	Rx (B)	Tx (B)	Rx (B)
1 server bez zabezpečení	41k	149k	81k	185k
1 server s šifrováním	45k	153k	90k	190,5k

Tab. 5.5: Objem dat

datech na server mezi přenosem bez zabezpečení a zabezpečeným přenosem je 9 %. Rozdíl v přijatých datech ze server mezi přenosem bez zabezpečení a zabezpečeným přenosem je 2,5 %.

Byl také proveden test odpojení serveru během čtení dat. Zda šifrování nemá vliv na obnovení komunikace po jeho přerušení a zda bude aplikace pokračovat ve čtení dat po obnovení komunikace. Po analýze zachycení síťového provozu bylo zjištěno, že po odpojení linky aplikace posílá TCP Retransmission s hodnotou RTO 0,3 vteřiny. Pokud dojde k obnovení komunikace dokud nevyprší TCP TimeOut, aplikace pokračovala ve čtení dat bez chyby.

Pro zjištění, zda byla data zašifrována byl použit nástroj GuruxDLMSTranslator. Po přeložení zachycených dat nebylo možné zjistit hodnotu dat na které se dotazuje klient server ani jaké odpovědi obdržel klient.

Závěr

Tato semestrální práce se zabývala protokoly využívanými pro Smart Metering. První kapitola popisuje protokoly ANSI C12.18, C12.19, C12.21, IEC61107 a OSGP. Dále se práce více zaměřuje na protokol DLMS/COSEM. Je detailněji popsána každá hlavní komponenta DLMS/COSEM.

Nejprve je popsán protokol DLMS, který zajišťuje komunikaci mezi klienty a servery. Je zdokumentováno jaký používá komunikační model, jaké zabezpečení a typ adresace. A jak je řešen přenos pomocí TCP/IP.

Dále se práce zabývá objektovým modelem COSEM, který specifikuje modelování objektů, pro přístup k měřícím zařízením. Model COSEM používá objektově orientovaný přístup.

Následně je popsána knihovna Gurux, která se specializuje na inteligentní odečty měřičů. Je popsán, jaký software poskytuje, jak navazuje spojení, jaké možnosti umožňuje pro vývoj DLMS serverů a klientů a jaké obsahuje zabezpečení.

V praktické části je popsán vývoj softwarové knihovny pro agregaci dat ze serverů DLMS/COSEM s použitím knihovny GURUX. Po domluvě s vedoucím práce, testování na reálných zařízeních nebyly provedeny a byly provedeny obsáhlejší testovací scénáře. Nejprve je popsán výběr vhodného serveru pro testování a ověření komunikace pomocí programu GuruxDirector. Dále je popsán vývoj softwarové knihovny, práce také obsahuje popis kódu. Dále je popsána série měření, které byly provedeny za pomocí této softwarové knihovny. Měření se zaměřovaly na komunikaci mezi klientem a serverem. Testy byly provedeny mezi dvěma počítači nebo mezi počítačem a mikropočítačem RaspberryPi. Byly popsány testy na mikropočítači RaspberryPi, maximální možný počet spuštěných serverů na RaspberryPi, analýza zachycené síťové komunikace, rozpady spojení TCP, testování a analýza objemu přenesených dat, objemové rozdíly OBIS kódů, objem dat a čas přenosu jednotlivých scénářů, test zatížení linky a test zabezpečené komunikace šifrováním.. Všechny testy byly nakonec analyzovány za pomocí analyzátoru síťového provozu.

Literatura

[1] *2007 Power Systems Conference: (PSC) Advanced metering, protection, control, communication, and distributed resources : 13th - 16th March, 2007, Clemson University, Clemson, South Carolina*. 1. Piscataway, NJ: IEEE, c2007. ISBN 978-1-4244-0854-2.

[2] Implementing an Advanced Meter Reading infrastructure using a Z-Wave compliant Wireless Sensor Network. *1* [online]. 2011, 6 [cit. 2019-12-07]. ISSN 978-989-95055-7-5. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6028316

[3] *Smart Grid: Communication-Enabled Intelligence for the Electric Power Grid*. 2014. Wiley-IEEE Press, 2014. ISBN 978-1-118-82021-6.

[4] ETSI OSGP. *Https://www.etsi.org* [online]. Sophia Antipolis Cedex - FRANCE: 2012 [cit. 2019-12-07]. Dostupné z: https://www.etsi.org

[5] *OSGP* [online]. 2019 [cit. 2019-12-07]. Dostupné z: https://osgp.org/en

[6] *DLMS/COSEM Architecture and Protocols* [online]. 2019, **2019**(1) [cit. 2019-12-10]. Dostupné z: https://www.dlms.com/files/Green_Book_Edition_9-Excerpt.pdf

[7] *COSEM Interface Classes and OBIS Object Identification System* [online]. 2019, [cit. 2019-12-10]. Dostupné z: https://www.dlms.com/files/Blue-Book-Ed-122-Excerpt.pdf

[8] *Comparison of the communication protocols DLMS/COSEM*, SML and IEC 61850 for smart metering applications, S. Feuerhahn, M. Zillgith, C. Wittwer and C. Wietfeld, Brussels, 2011, pp. 410-415, doi: 10.1109/SmartGridComm.2011.6102357.

[9] High-level Data Link Control (HDLC). *Https://www.tutorialspoint.com* [online]. 2019 [cit. 2019-12-10]. Dostupné z: https://www.tutorialspoint.com/high-level-data-link-control-hdlc

[10] The DLMS communication. *Icube.ch* [online]. route de Botyre 26 1966 Ayent Switzerland, 2019 [cit. 2019-12-10]. Dostupné z: https://icube.ch/DLMSSurvivalKit/dsk1.html

[11] *2013 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON*

2013): Krabi, Thailand, 15-17 May 2013. Piscataway, NJ: IEEE, [2013]. ISBN 978-1-4799-0546-1.

[12] *Implementing DLMS/COSEM in smart meters*, 2011 8th International Conference on the European Energy Market (EEM), Zagreb, G. Štruklec and J. Maršić, 2011, pp. 747-752, doi: 10.1109/EEM.2011.5953109.

[13] *A Survey of Communication Protocols for Automatic Meter Reading Applications*,"in IEEE Communications Surveys and Tutorials, vol. 13, no. 2, pp. 168-182, T. Khalifa, K. Naik and A. Nayak, Second Quarter 2011, doi: 10.1109/SURV.2011.041110.00058.

[14] *DLMS / COSEM Protocol Security Evaluation.* Weith, Loren. , (2014).

[15] *Towards visual smart metering exploiting wM-Bus and DLMS/COSEM*", G. Di Leo, C. Liguori, V. Paciello, A. Pietrosanto and P. Sommella„Shenzhen, 2015, pp. 1-6, doi: 10.1109/CIVEMSA.2015.7158602.

[16] *EFuzz: A Fuzzer for DLMS/COSEM Electricity Meters.* Henrique Dantas, Zekeriya Erkin, Christian Doerr, Raymond Hallie, and Gerrit van der Bij. Association for Computing Machinery, New York, NY, USA, 31–38. DOI:https://doi.org/10.1145/2667190.2667194

[17] *Analysis of security features in DLMS/COSEM: Vulnerabilities and countermeasures* N. Luring, D. Szameitat, S. Hoffmann and G. Bumiller,Washington, DC, 2018, pp. 1-5, doi: 10.1109/ISGT.2018.8403340.

[18] *DLMS/COSEM in The Internet of Things: Key Applications and Protocols* ,Olivier Hersent; David Boswarthick; Omar Elloumi, Wiley, 2012, pp.179-192, doi: 10.1002/9781119958352.ch11.

[19] *Energy Informatics: 4th D-A-CH Conference*, EI 2015, Karlsruhe, Germany, November 12-13, 2015, Proceedings

[20] *Www.gurux.fi* [online]. [cit. 2019-12-10]. Dostupné z: https://www.gurux.fi

[21] Data Quality Study of AMR Systems. *Http://www.diva-portal.se* [online]. 2015, 2015 [cit. 2019-12-10]. Dostupné z: http://www.diva-portal.se/smash/get/diva2:883106/FULLTEXT01.pdf

Seznam symbolů, veličin a zkratek

AA	Application Associations
AMM	Advanced Meter Management
AARE	Application Associations Response
AARQ	Application Associations Request
ACSE	Association Control Service Element
AE	Application Entity
AMR	Automatic Meter Reading
AP	Application Process
APDU	Application Protocol Data Unit
ASE	Application Service Elements
COSEM	Companion Specification for Energy Metering
CRC	Cyclic Redundancy Check
DLMS	Device Language Message Specification
GPRS	General Packet Radio Service
HDLC	High-level Data Link Control
HSL	High Level Security
IP	Internet Protocol
LD	Logical Device
LLC	Logical Link Control
LLS	Low Level Security
LN	Logical Name
LTE	Long-Term Evolution
MAC	Media Access Control
OBIS	Object Identification Systém
OSGP	Open Smart Grid Protocol
PDU	Protocol Data Unit
PLC	Programmable Logic Controller
PSEM	Protocol Specifications for Electric Metering
SAP	Service Access Point
SN	Short Name
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WEP	Wired Equivalent Privacy
XDLMS	Extended DLMS
XML	eXtensible Markup Language

Seznam příloh

A Jednotlivé scénáře a jejich OBIS kódy

OBIS kód	Název
1-0:1.8.0.255	Činná energie +A

Tab. A.1: 1. scénář

OBIS kód	Název
0-0:1.0.0.255	Čas

Tab. A.2: 2. scénář

OBIS kód	Název
1-0:1.8.0.255	Činná energie +A
1-0:1.8.1.255	Činná energie +A tarif T1
1-0:1.8.2.255	Činná energie +A tarif T2
1-0:1.8.3.255	Činná energie +A tarif T3
1-0:1.8.4.255	Činná energie +A tarif T4
1-0:2.8.0.255	Činná energie -A
1-0:2.8.1.255	Činná energie -A tarif T1
1-0:2.8.2.255	Činná energie -A tarif T2
1-0:2.8.3.255	Činná energie -A tarif T3
1-0:2.8.4.255	Činná energie -A tarif T4

Tab. A.3: 3. scénář

OBIS kód	Název
0-0:96.1.1.255	Identifikátor zařízení
0-0:96.13.0.255	Textová zpráva
0-0:96.13.1.255	Textová zpráva
0-0:96.3.10.255	Indikátor stavu odpojovače/limiteru
0-1:96.3.10.255	Indikátor stavu blokovacího relé 1
0-2:96.3.10.255	Indikátor stavu blokovacího relé 2
0-0:96.14.0.255	Indikátor aktuálního tarifu
0-0:96.3.10.255	Indikace směru toku energie v jednotlivých fázích
1-0:1.7.0.255	Aktuální výkon odběru s rozlišením na 1 W
1-0:2.7.0.255	Aktuální výkon dodávky do sítě s rozlišením na 1 W
1-0:1.8.0.255	Činná energie +A

1-0:1.8.1.255	Činná energie +A tarif T1
1-0:1.8.2.255	Činná energie +A tarif T2
1-0:1.8.3.255	Činná energie +A tarif T3
1-0:1.8.4.255	Činná energie +A tarif T4
1-0:2.8.0.255	Činná energie -A
1-0:2.8.1.255	Činná energie -A tarif T1
1-0:2.8.2.255	Činná energie -A tarif T2
1-0:2.8.3.255	Činná energie -A tarif T3
1-0:2.8.4.255	Činná energie -A tarif T4
1-0:1.8.139.255	Činná energie +A po dobu působení mag. pole
1-0:3.8.0.255	Jalová energie +Q
1-0:3.8.1.255	Jalová energie +Q tarif T1
1-0:3.8.2.255	Jalová energie +Q tarif T2
1-0:3.8.3.255	Jalová energie +Q tarif T3
1-0:3.8.4.255	Jalová energie +Q tarif T4
1-0:4.8.0.255	Jalová energie -Q
1-0:4.8.1.255	Jalová energie -Q tarif T1
1-0:4.8.2.255	Jalová energie -Q tarif T2
1-0:4.8.3.255	Jalová energie -Q tarif T3
1-0:4.8.0.255	Jalová energie -Q tarif T4
1-0:1.7.0.255	Aktuální výkon odběru s rozlišením na 1 W
1-0:21.7.0.255	Okamžitý činný výkon v L1
1-0:41.7.0.255	Okamžitý činný výkon v L2
1-0:61.7.0.255	Okamžitý činný výkon v L3
1-0:2.7.0.255	Aktuální výkon dodávky do sítě s rozlišením na 1 W
1-0:22.7.0.255	Okamžitý činný výkon v L1
1-0:42.7.0.255	Okamžitý činný výkon v L2
1-0:62.7.0.255	Okamžitý činný výkon v L3
0-0:0.2.2.255	Název aktuální TOU tabulky pro řízení tarifů
0-0:13.0.0.255	Funkční možnosti TOU (Time of Usage)
0-0:13.0.1.255	Odpojovač kalendáře aktivit (TOU)
1-0:1.6.0.255	Střední hodnota výkonu +A za 15 minut, hodnota +A za měsíc
1-0:2.6.0.255	Střední hodnota výkonu +A za 15 minut, hodnota +A za měsíc
1-0:8.8.0.255	Profil denních hodnot registru jalové dodávky Rc- (90 dnů)
1-0:7.8.0.255	Profil denních hodnot registru jalové dodávky Ri- (90 dnů)
1-0:6.8.0.255	Profil denních hodnot registru jalové spotřeby Rc+ (90 dnů)
1-0:5.8.0.255	Profil denních hodnot registru jalové spotřeby Ri+ (90 dnů)

1-0:1.7.0.130	Nastavení výrobny na 0 %
1-0:1.7.0.133	Nastavení výrobny na 33 %
1-0:1.7.0.166	Nastavení výrobny na 66 %
1-0:1.7.0.200	Nastavení výrobny na 100 %
1-0:1.7.0.131	Potvrzení o nastavení výrobny na 0 %
1-0:1.7.0.134	Potvrzení o nastavení výrobny na 33 %
1-0:1.7.0.167	Potvrzení o nastavení výrobny na66 %
1-0:1.7.0.201	Potvrzení o nastavení výrobny na 100 %
1-0:2.7.0.130	Nastavení spotřeby na 0 %
1-0:2.7.0.133	Nastavení spotřeby na 33 %
1-0:2.7.0.166	Nastavení spotřeby na 66 %
1-0:2.7.0.200	Nastavení spotřeby na 100 %
1-0:2.7.0.131	Potvrzení EMS nastaveném rozsahu regulace spotřeby na 0 %
1-0:2.7.0.134	Potvrzení EMS nastaveném rozsahu regulace spotřeby na 33 %
1-0:2.7.0.167	Potvrzení EMS nastaveném rozsahu regulace spotřeby na 66 %
1-0:2.7.0.201	Potvrzení EMS nastaveném rozsahu regulace spotřeby na 100 %

Tab. A.4: 4. scénář