VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

APLIKACE PRO SIMULACI ŘÍDICÍHO SYSTÉMU PROFINET

APPLICATION FOR SIMULATION OF PROFINET CONTROLLER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **TOMÁŠ MAZUREK**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. JITKA KRESLÍKOVÁ, CSc.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19793/2016/xmazur07

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Mazurek Tomáš**
Obor: Informační technologie
Téma: **Aplikace pro simulaci řídicího systému PROFINET**
Application for Simulation of PROFINET Controller
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se se standardem Profinet a řídicími systémy PLC SIEMENS SIMATIC S7.
2. Prostudujte principy komunikace mezi řídicím systémem a V/V zařízením.
3. Po dohodě s konzultantem společnosti Siemens CZ navrhněte aplikaci simulující chování řídicího systému.
4. Zvolte vhodné implementační prostředí a vývojové technologie. Implementujte prototyp navrženého systému. Výsledná aplikace bude umožňovat navázání aplikačního vztahu s V/V zařízením a práci s konfiguračními a diagnostickými daty.
5. Použitelnost aplikace demonstrujte na vhodném vzorku dat vybraném po dohodě s vedoucí.
6. Zhodnoťte dosažené výsledky a přínos implementovaného řešení a diskutujte další možnosti rozšíření vytvořené aplikace.

Literatura:

- PIGAN Raimond, METTER Mark. *Automating with PROFINET:* 2nd rev. and enl. edition, 2008. ISBN 978-3-89578-294-7.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů zadání 1 až 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kreslíková Jitka, doc. RNDr., CSc.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce pojednává o vývoji aplikace simulující řídicí systém Profinet. Hlavní funkce takového řídicího systému je navázání aplikačního vztahu se zařízením, které komunikuje pomocí technologie Profinet. Prostřednictvím aplikačního vztahu poté probíhá veškerá výměna informací. Cílem simulace řídicího systému pomocí aplikace, která běží na stolním počítači, je usnadnění testování vstupně výstupních zařízeních, které jsou k systému připojené přes rozhraní Ethernet. Ve výsledné aplikaci lze poté monitorovat a modifikovat komunikaci mnohem snadněji než se skutečným řídicím systémem.

Abstract

This bachelor's thesis deals with development of an application simulating the Profinet controller. The main function of such controller is to establish an application relation with a device, which is communicating via the Profinet technology. Through the application relation then takes place the entire exchange of information. The goal of the controller simulation, run on a desktop computer, is to facilitate the testing of input/output devices connected to the system via Ethernet interface. The resulting application allows to monitor and modify the communication much easier than the real control system.

Klíčová slova

Profinet, testování, řídicí systém, Virtual CPU, průmyslový Ethernet

Keywords

Profinet, testing, controller, Virtual CPU, Industrial Ethernet

Citace

MAZUREK, Tomáš. *Aplikace pro simulaci řídicího systému PROFINET*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Jitka Kreslíková, CSc.

Aplikace pro simulaci řídicího systému PROFINET

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením paní doc. RNDr. Jitky Kreslíkové, CSc. Další informace mi poskytl Ing. Petr Kramný. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Mazurek
17. května 2017

Poděkování

Chtěl bych poděkovat především mojí vedoucí bakalářské práce paní docentce Jitce Kreslíkové za vedení a rady při zpracovávání této práce. Dále taky panu Ing. Petru Kramnému a dalším kolegům ze společnosti Siemens s.r.o. kteří poskytli odborné konzultace.

Obsah

Kapitola 1

Úvod

Automatizace v průmyslové výrobě má v současné době vliv na mnoho důležitých faktorů. Ovlivňuje především ekonomiku, rychlost výroby, množství použitých materiálů a energií, ale také životní podmínky pracovníků, jejich počet, osobní komfort a bezpečnost. Zavádění automatizace do průmyslu je sice poměrně drahý a složitý proces, ale díky značnému zefektivnění výroby má svůj nepostradatelný smysl. V případě nasazení kvalitních a správně fungujících technologií se taková investice navíc velmi rychle vrátí. Nároky kladené na vývoj takových technologií jsou proto velmi vysoké. Jakákoliv kritická chyba, například na již automatizované výrobní lince, může způsobit jak značné finanční ztráty, tak i ohrožení bezpečnosti všech pracovníků a kolaps celé výroby. Z těchto důvodů jsou veškeré technologie pro automatizaci důkladně testovány, než jsou uvedena na trh.

Technologie Profinet je v současné době jedním z předních nástrojů pro průmyslovou automatizaci. Existuje mnoho produktů, které Profinet podporují. Ty jsou zaměřeny na široké odvětví automatizace. Společnost Siemens v Brně vyvíjí a testuje dva takové produkty. Inteligentní napájecí systém PSU8600 a záložní napájení UPS1600. Tyto produkty mohou být připojeny k řídicímu systému, se kterým je navázán komunikační vztah k výměně důležitých informací. V praxi reprezentuje řídicí systém zařízení, nazývané vstupně výstupní kontrolér. Přestože zařízení může fungovat i samostatně bez takového kontroléru, v drtivé většině případů je nějaký typ řídicího systému použit. Bez něj nelze plně využít potenciál zařízení Profinetu.

Aby zařízení splňovalo veškeré normy standardu Profinet, musí být důkladně testováno. To znamená co nejlépe nasimulovat prostředí běžného průmyslového provozu, ve kterém se budou produkty reálně vyskytovat. Z výše zmíněných důvodů se testování proto neobejde bez řídicího systému, který je napojen na testovaný produkt. Jako nejlepší způsob se jeví použít nějaký typ reálného kontroléru, který plní funkci řídicího systému. Příkladem takového kontroléru může být typ označen názvem PLC SIMATIC S7. To ovšem není zdaleka tak snadné, jak se může na první pohled zdát. Počet takových produktů je v testovacích prostorách omezený, a jelikož některé testy vyžadující kontrolér mohou zabrat poměrně dlouhý časový úsek, musely by takové testy na sebe navzájem čekat. Dalším problémem je řízení kontroléru zvenčí. Veškerá nastavení musí být provedena v inženýrském nástroji a poté do kontroléru nahrána. Podobná situace nastává i při pokusu o vyčítání jeho dat. Testy vyžadující takové operace s kontrolérem jsou náročné, jak na samotné provedení, tak i na případnou automatizaci. Proto byl vznesen návrh simulovat potřebné funkce kontroléru počítačovým programem. Cílem této práce je tedy vytvořit aplikaci, běžící na klasickém stolním počítači, která zvládne všechny operace klasického kontroléru, důležité pro testování produktů s technologií Profinet vyvíjených v laboratořích společnosti Siemens.

Práce je rozdělena na několik částí. V kapitole 2 jsou popsány základy standardu Profinet. Následuje kapitola 3, kde jsou vysvětleny nejdůležitější principy síťové komunikace. Tyto dvě kapitoly jsou potřebné pro pochopení fungování vyvíjené aplikace. Její návrh je popsán v kapitole 4. Poslední kapitola 5 pojednává o implementaci jednotlivých částí aplikace Virtual CPU.

Kapitola 2

Profinet

Profinet (PROcess FIeld NET) je komunikační standard založený na průmyslovém Ethernetu, sloužící především pro automatizaci v průmyslové výrobě. Čtyři roky po prvním ohlášení Profinetu v srpnu roku 2000 proběhlo jeho skutečné nasazení. Nabízí veškeré potřeby pro komunikaci přes Ethernet v průmyslové automatizaci. Mimo jiné například komunikaci v reálném čase, správu sítě, přístup z webového rozhraní, možnosti připojení periferií (Profinet IO 2.1) nebo možnost využití modulů (Profinet CBA - Component Based Automation) [6].

Standard lze rozlišit na dvě již zmíněné části. Profinet IO a Profinet CBA. Následující text se bude zabývat převážně variantou Profinet IO, jelikož vyvíjená aplikace využívá právě tento způsob komunikace.

2.1 Profinet IO

Profinet IO poskytuje možnost přímého zapojení distribuovaných zařízení na Ethernetu. Všechna zařízení jsou propojena v jednotné síti a poskytují rozhraní pro vzájemnou komunikaci. V topologii se rozlišují dva hlavní typy zařízení: vstupně výstupní kontrolér (dále jen V/V-kontrolér) a vstupně výstupní zařízení (dále jen V/V-zařízení). Systém je navržen pro rychlou výměnu dat se zpožděním několika milisekund.

Distribuované V/V-zařízení je přiřazeno k programovatelnému V/V-kontroléru. Typ zařízení je přidělen při konfiguraci.

2.2 Typy zařízení

Všechny prvky v síti mají předem určenou roli, podle které se liší jejich fungování. Tyto prvky se dělí do následujících kategorií:

(a) *V/V-supervizor* – inženýrské zařízení, obvykle osobní počítač nebo HMI (Humane Machine Interface), použité pro diagnostiku a uvedení V/V-kontroléru a V/V-zařízení do provozu. Může dočasně zastupovat funkci kontroléru. V/V-supervizor je připojen za běhu a je používán dočasně, obvykle pro hledání chyb nebo pro základní testy. Často je integrován v inženýrském nástroji.

(b) *V/V-kontrolér* – programovatelný kontrolér, obvykle PLC obsahující uživatelský program, který řídí veškerou komunikaci. Minimální Profinet IO konfigurace musí obsa-

hovat alespoň jeden V/V-kontrolér. Do takového kontroléru je pomocí inženýrského nástroje nahrán projekt, který obsahuje veškeré informace o dané konfiguraci.

(c) *V/V-zařízení* – distribuované zařízení, které komunikuje s jedním nebo více V/V-kontroléry. Minimální Profinet IO konfigurace musí obsahovat alespoň jedno V/V-zařízení.

Výsledná aplikace, jejíž návrh a implementace je popsána v této práci, simuluje základní operace V/V-kontroléru. K počítači, na kterém je aplikace spouštěna, musí být proto připojeno alespoň jedno V/V-zařízení.

2.3 Popis zařízení

Každé zařízení Profinetu je popsáno pomocí speciálního souboru, který obsahuje technickou charakteristiku konkrétního zařízení. Ten se nazývá GSD soubor (General Station Description) – značkovací jazyk vycházející z formátu XML. Obsahuje mnoho dat potřebných pro různé inženýrské nástroje, které vyžadují znát veškeré informace o daném zařízení. Mimo jiné obsahuje přístupové body jednotlivých zařízení nazývané DAP (Device Access Point). Každý takový DAP může být rozšířen o moduly. Zařízení lze adresovat několika způsoby:

1. *DAP* – označuje zařízení jako celek.
2. *Jednotlivé sloty* – typicky fyzické rozdělení zařízení na moduly, popřípadě logické části v rámci jednoho modulu.
3. *Jednotlivé subsloty* – každý slot je rozdělen na subsloty podle logických částí modulu.
4. *Indexy* – data jednotlivých subslotů.

V praxi by potom mohlo adresování konkrétní konfigurace (PSU8600 + moduly) vypadat následovně:

1. *DAP* – konkrétní zařízení typu PSU8600.
2. *Jednotlivé sloty* – každý slot reprezentuje jeden připojitelný modul. Například modul rozšiřující počet napěťových výstupů celého zařízení označen "CNX 40A".
3. *Jednotlivé subsloty* – na slotu, kde se nachází modul "CNX 40A", reprezentuje každý subslot jeden napěťový výstup.
4. *Indexy - pro každý* subslot (napěťový výstup) reprezentují indexy konkrétní data výstupu. Příkladem může být konkrétní napětí nebo stav výstupu (zapnuto/vypnuto).

Adresace typu: DAP "MAIN_MODULE_MULTI_40A_LS3", slot 1, subslot 3, index 40 – adresuje data na indexu 40, druhý napěťový výstup (první výstup začíná na subslotu číslo 2) prvního modulu připojeného ke čtyřiceti ampérové variantě zařízení (produktové verze LS3).

V souboru GSD jsou mimo jiné uloženy informace, jaké moduly můžeme připojit na konkrétní DAP. Dále sloty, subsloty a indexy jednotlivých zařízení. Známe-li tedy konfiguraci (konkrétní zařízení s moduly), můžeme pomocí GSD získat kompletní komunikační rozhraní celého zařízení [6].

2.4 Komunikační vztahy

Během aplikačního vztahu je navázáno několik dalších typů vztahu. Tento vztah se nazývá komunikační (CR – Communication relationship) a dělí se do tří typů.

(a) *Record-Data CR* – acyklický přenos dat (záznamů), například pro parametrizaci při startu aplikačního vztahu nebo diagnostiku.

(b) *IO-Data CR* – cyklický přenos dat probíhající v určitém intervalu.

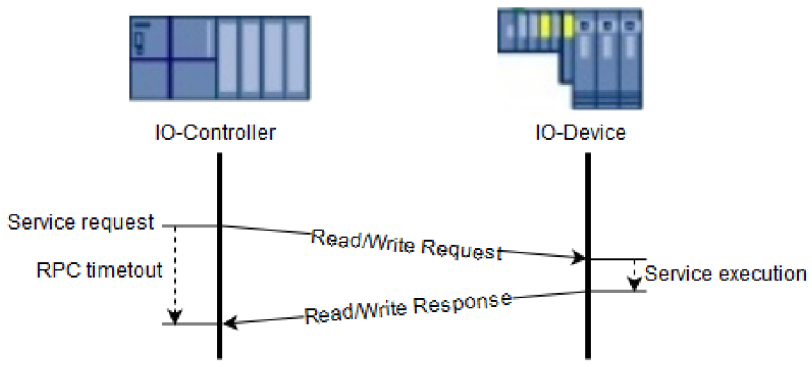
(c) *Alarm CR* – acyklický přenos informace oznamující vzniknutí nějaké nečekané události.

Typické pořadí a průběh navazování těchto komunikačních vztahů je následující. Na začátku při vytváření aplikačního vztahu jsou přeneseny potřebné informace (nastavení týkající se konkrétní konfigurace) pomocí vztahu Record-Data CR. Od určité chvíle po navázání se začnou cyklicky posílat přes IO-Data CR a jsou přenášena po celou dobu navázání aplikačního vztahu. Vznikne-li potřeba ohlásit událost, která nastala, využije se poslední komunikační vztah – Alarm CR [7].

2.4.1 Record Data CR

Record-Data CR je vztah navazující V/V-kontrolér s V/V-zařízením. Přenos probíhá acyklicky pouze v tom okamžiku, ve kterém je takový přenos vyžadován. Tento vztah je použit pro přenos následujících informací:

- Záznamy.
- Data konkrétního zařízení.
- Konfigurační a identifikační data.
- Acyklická vstupně-výstupní data.
- Logovací záznamy zařízení.



Obrázek 2.1: Record Data CR – diagram komunikace [inspirováno z 6]

Tímto způsobem může probíhat čtení dat (Read Request) nebo jejich zápis (Write Request). Každý požadavek reaguje odpovědí.

(a) *Read Request* – požadavek V/V-kontroléru na čtení dat z V/V-zařízení.

(b) *Read Response* – přenos požadovaných dat do V/V-kontroléru.

(c) *Write Request* – přenos dat pro zápis do V/V-zařízení.

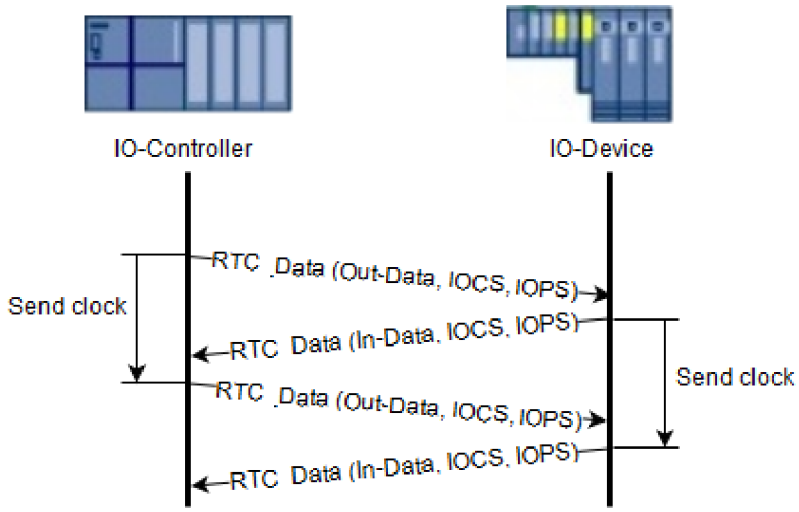
(d) *Write Respone* – potvrzení zápisu V/V-zařízením.

2.4.2 IO-Data CR

Hlavní funkce vztahu je přenos vstupně/výstupních dat (nazývané také cyklická data). Jednotlivá data jsou identifikována pomocí modelu provider/consumer. Informace obsahují data jako: frekvence přenosu, parametry intervalu nebo seznam objektů pro přenos. Počet těchto vztahů je definován v konfiguraci zařízení. V každém případě jsou navázány alespoň dva, které umožňují oboustrannou výměnu dat mezi V/V-kontrolérem (popřípadě V/V-supervisorem) a V/V-zařízením. Cyklicky posílaná data směřují od V/V-zařízení do V/V-kontroléru a naopak v intervalu, který byl předem nakonfigurován. Data nejsou formátována a mají konstantní délku. Maximálně však 240 bajtů. Výpočet intervalu posílání dat (Send clock time) vychází ze vzorce:

$$Sendclocktime = Sendclockfactor * 31.25 mikrosekund$$

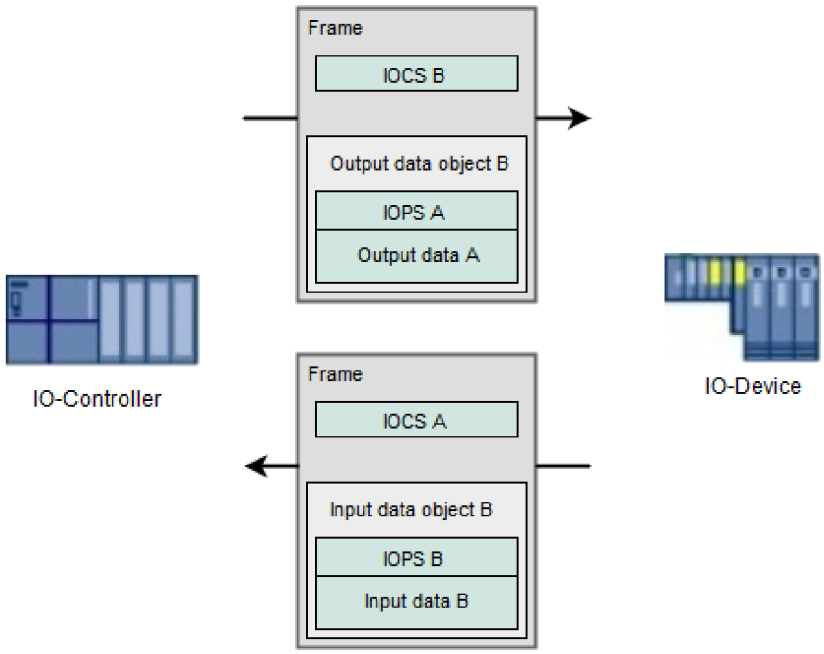
Send clock factor je možno nastavit při navazování aplikačního vztahu a povoluje hodnoty v rozmezí 1 - 128. Hodnota 32 odpovídá frekvenci 1 milisekunda.



Obrázek 2.2: IO-Data CR – diagram komunikace [inspirováno z 6]

Pro identifikaci role jednotlivých zařízení se používá terminologie provider/consumer. Toto rozlišení se liší na základě kontextu výměny dat. Provider dodává data do zařízení typu consumer. Pro výstupní data (z kontroléru do zařízení) přebírá kontrolér roli zvanou provider a zařízení přebírá roli zvanou consumer. Pro vstupní data (ze zařízení do kontroléru) má naopak kontrolér roli consumer a zařízení roli provider. Komunikace zobrazena na obrázku 2.2 začíná po úspěšném navázání aplikačního vztahu. Každá položka IO-Data obsahuje dva atributy, IOPS (provider status) a IOCS (consumer status). Tyto atributy

obsahují informace o kvalitě přenesených dat. IOPS označuje status právě posílaných dat, IOCS označuje status dat, která přišla do zařízení při poslední výměně.



Obrázek 2.3: Struktura cyklických dat [inspirováno z 6]

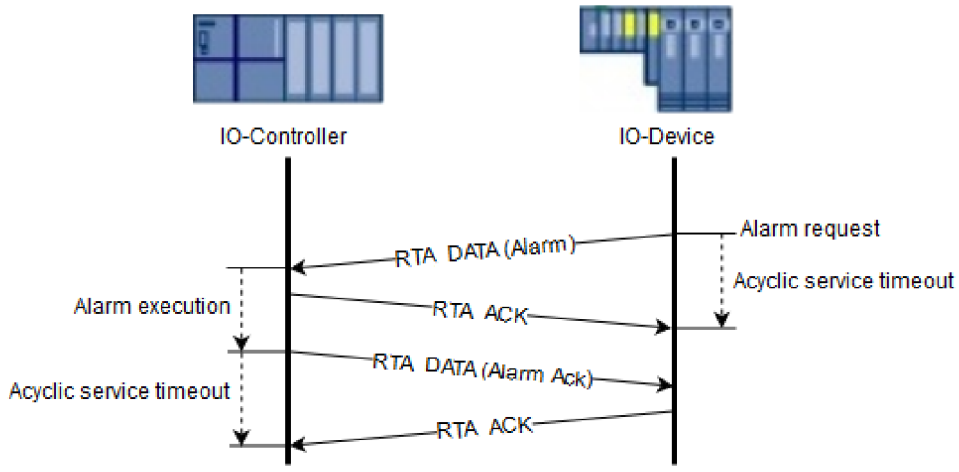
2.4.3 Alarm CR

Alarmy lze chápat, jak už bylo několikrát řečeno, jako ohlášení výskytu události, která není očekávána nebo kterou je potřeba nějakým způsobem ošetřit. Pro přenos takových informací je použit vztah zvaný Alarm CR. Alarm generuje V/V-zařízení a posílá jej V/V-kontroléru. Data ohlašující alarm jsou posílána acyklicky a musí být potvrzena V/V-kontrolérem do doby, která je specifikována protokolem. Lze nastavit, kolik alarmů může být maximálně posláno před samotným potvrzením. Existuje mnoho definovaných typů alarmu. Všechny ale spadají do dělení podle priority.

(a) *Vysoko-prioritní* – musí být zpracován co možná nejrychleji.

(b) *Nízko-prioritní* – nesmí zpomalit zpracování vysoko-prioritního alarmu.

Diagram na obrázku 2.4 zobrazuje vygenerování alarmu na straně V/V-zařízení a jeho potvrzení V/V-kontrolérem.



Obrázek 2.4: Alarm CR – diagram komunikace [inspirováno z 6]

V diagramu jsou rozlišeny následující datagramy:

- *RTA-DATA (Alarm)* – oznámení alarmu z V/V-zařízení do V/V-kontroléru. Zde jsou přenášeny informace jako například identifikace alarmu nebo adresa (slot, sublost, modul ID) vzniklého alarmu.
- *RTA-DATA (Alarm-Ack)* – potvrzení alarmu V/V-kontrolérem na aplikační vrstvě.
- *RTA-ACK* – potvrzení rámce RTA_DATA na protokolové vrstvě.

2.5 Aplikační vztah

Aplikační vztah (AR – Apllication Relationship) je virtuální vztah, který po úspěšném navázání dovoluje výměnu dat mezi dvěma zařízeními na komunikačním kanálu. Ve standardu Profinet jsou definovány dva typy aplikačního vztahu – IOC-AR a implicitní AR. Každý takto navázaný vztah je identifikován pomocí takzvaného ARUUID, což je jednoznačný šestnáctibajtový identifikátor generovaný inženýrským nástrojem. Je generován pro každou konkrétní konfiguraci a liší se pro každé zařízení, se kterým je vztah navázán.

2.5.1 IOC-AR

Tento typ je používán pro výměnu následujících dat:

- Čtení vstupních hodnot (vstupní cyklická data) z V/V-zařízení.
- Zapsání výstupních hodnot (výstupní cyklická data) do V/V-zařízení.
- Přijetí události z V/V-zařízení a reakce na ni. Událostí je myšleno přerušení (alarm) v případě výskytu neočekávaného stavu na V/V-zařízení.
- Čtení acyklických dat z V/V-zařízení.
- Zápis acyklických dat do V/V-zařízení.

2.5.2 Implicitní AR

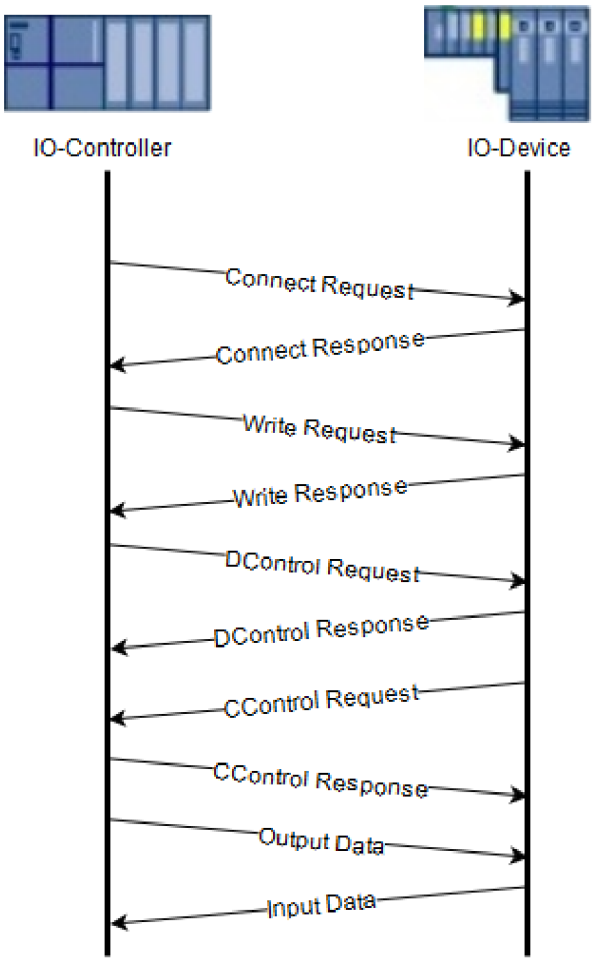
Implicitní AR má vždy nulovou hodnotu ARUUID. Definuje aplikační vztah, mezi kontrolérem/supervisorem a zařízením, který ale podporuje pouze acyklická čtení/zápis dat z/do zařízení. Kontrolér nemusí navázat samotné AR pro tento vztah. Implicitní AR je navázáno vždy a může být kdykoliv použito. Adresování probíhá pomocí slotu, subslotu a indexu [7].

2.5.3 Navázání vztahu

Pro správné navázání vztahu typu IOC-AR je potřeba provést několik předem definovaných kroků. Tyto kroky zajistí jeho úspěšné navázání. Obsahují například požadavek na navázání vztahu, parametrizaci, ukončení parametrizace a potvrzení správného navázání. Poté začíná výměna cyklických dat a je možno zapsání/vyčtení acyklických dat. Sekvence je složena z následujících rámců:

1. *Connect Request* – inicializace navázání vztahu.
2. *Connect Response* – odpověď V/V-zařízení na Connect Request. Poslána pouze v případě, že v požadavku na navázání jsou všechny potřebné informace.
3. *Write Request* – nepovinný přenos parametrů z V/V-kontroléru do V/V-zařízení. Obsahuje všechny informace týkající se konkrétního zařízení, se kterým chceme komunikovat.
4. *Write Response* – odpověď oznamující správné zapsání dat do V/V-zařízení.
5. *DControl Request* – oznámení o úspěšném ukončení parametrizace.
6. *DControl Response* – potvrzení ukončení parametrizace.
7. *CControl Request* – informace z V/V-zařízení o potvrzení správného navázání vztahu.
8. *CControl Response* – potvrzení ze strany V/V-kontroléru. V tomto okamžiku proběhly všechny kroky k vytvoření aplikačního vztahu mezi V/V-kontrolérem a V/V-zařízením.
9. *Input/Output Data* – počátek výměny cyklických dat.

Každé V/V-zařízení musí být možno jednoznačně identifikovat. Identifikace probíhá na základě takzvaného Vendor ID a Device ID. Vendor ID je 16-bitová položka přiřazena výrobcem konkrétního zařízení. V případě společnosti Siemens je hodnota Vendor ID 42. Device ID je šestnáctibitová položka identifikující konkrétní zařízení.



Obrázek 2.5: Connect Request – diagram komunikace [inspirováno z 6]

2.5.4 Ukončení vztahu

Ukončení vztahu může vzniknout na základě úmyslného ukončení V/V-kontrolérem, například není-li už potřeba tento vztah udržovat nebo z důvodu selhání V/V-kontroléru či V/V-zařízení. Informace o tom, kdo vztah ukončil je v hlavičce paketu. Když je aplikační vztah ukončen, jsou také ukončeny všechny komunikační vztahy [7].

2.5.5 Přiřazení jména a IP adresy

Každé V/V-zařízení je v síti identifikováno jak jménem, tak IP adresou (3.3.2). Jméno je přiřazeno ještě před samotným vznikem aplikačního vztahu. Nastavení jména konkrétního zařízení probíhá pomocí V/V-supervisoru pomocí protokolu DCP (Discovery and basic configuration protokol). V/V-supervisor nejdříve vyhledá všechna dostupná zařízení (DCP browse) a jednomu z nich jméno přiřadí. Zařízení poté potvrdí úspěšné přiřazení jména. Přiřazení IP adresy konkrétnímu zařízení probíhá na základě jeho jména. Tato IP adresa V/V-zařízení je nastavena pomocí inženýrského nástroje a nahrána do V/V-kontroléru, který zařízení vyhledá a adresu nastaví. Pro přiřazení adresy může být použit také DHCP server nebo ji lze přiřadit pomocí DCP protokolu V/V-supervisorem.

Kapitola 3

Síťová komunikace

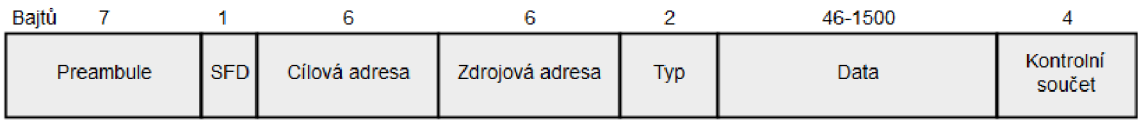
Jak již bylo řečeno, standard Profinet je založen na klasickém Ethernetu, což je technologie používaná pro přenos dat v počítačových sítích. Zařízení Profinetu využívají základní protokoly z rodiny TCP/IP (Transmission Control Protocol) a také UDP/IP (User Datagram Protocol). Pro lepší porozumění fungování Profinetu je důležité znát základy komunikace pomocí těchto protokolů. V následující kapitole není možné pokrýt všechny jejich principy, jsou zde však objasněna ta nejdůležitější fakta pro porozumění fungování zařízení se standardem Profinet, jakožto prvků v síti komunikující pomocí Ethernetu. Kapitola čeprá z [6] kde jsou popsány základy síťové komunikace.

3.1 Ethernetový rámec

Data přenášena na Ethernetu jsou rozdělena na několik menších částí zvané rámce. Každý takový rámec obsahuje všechny potřebné informace pro přenos po síti. Je poslán jako samostatná jednotka o velikost 64 až 1526 bajtů. Základní tvar rámce má hlavičku, data a zakončení. Data jsou složena ze zapouzdřených dat vyšších vrstev. V hlavičce jsou uloženy informace o zdrojové a cílové adrese, podle které je vyhodnoceno, kam je rámec v síti dále poslán. Zakončení rámce obsahuje kontrolní výpočet detekující chyby přenosu. Ethernetovému rámci předchází sekvence bajtů zvaná preambule a také takzvaný jednobajtový oddělovač začátku rámce (SFD - Start Frame Delimiter). Pomocí těchto dat, které mají pevně stanovenou podobu, lze snadno detekovat na straně jakéhokoliv síťového zařízení začátek každého rámce.

Existují dva druhy ethernetového rámce lišící se významem dvoubajtového pole za zdrojovou adresou.

1. *Ethernet II* – dvoubajtové pole před daty značí typ rámce.
2. *IEEE 802.3* – dvoubajtové pole před daty značí délku rámce.



Obrázek 3.1: Struktura ethernetového rámce [převzato z 4]

3.2 Adresování pomocí MAC

Adresování zařízení v síti znamená, že každý prvek v síti musí mít vlastní adresu, na které může být dosažen. Každému ethernetovému rozhraní je přiřazena taková adresa jeho výrobcem, která je pevně stanovená a je unikátní v rámci celého světa. Nazýváme ji MAC adresa (Media Access Control). Je pevně vložena na každou síťovou kartu a identifikuje zařízení v rámci lokální sítě. MAC adresa má 6 bajtů, kde první 3 označují konkrétního výrobce síťové karty a zbylé 3 bajty jsou již volně přiřazeny výrobcem konkrétnímu kusu. To zajišťuje jejich jedinečnost. Identifikace zařízení pomocí MAC adres je jedním ze způsobů, které využívá standard Profinet.

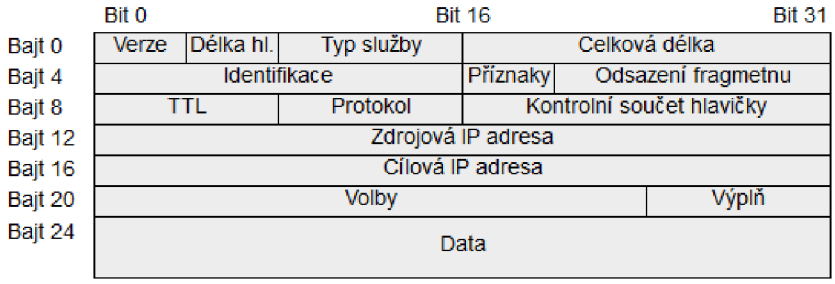
3.3 TCP/IP

TCP/IP může být považováno za kompletní sadu protokolů skládající se ze dvou částí. První z nich TCP (Transmission Control Protocol) řídí samotný přenos dat na lince. Druhá část IP (Internet Protocol) je požadována pro jednoznačné adresování počítačů, popřípadě síťových prvků v síti.

3.3.1 Internet Protocol

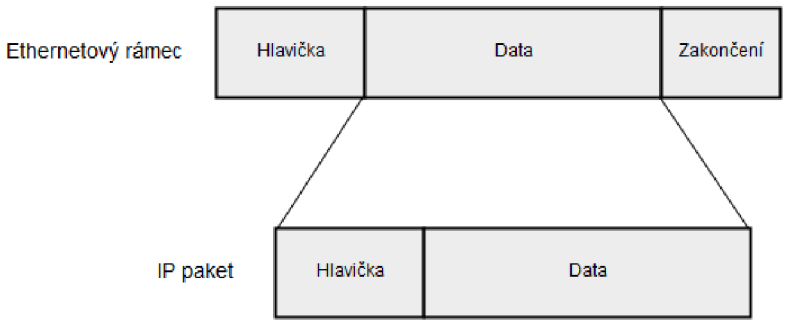
IP protokol se v současné době dělí na dvě verze, IPv4 a IPv6. Druhá zmíněná verze se začala silně využívat až v posledních letech, a to z důvodu nedostatku adres verze IPv4. Taková adresa má totiž pouze čtyři bajty, a proto není dostačují pro adresaci všech zařízení připojených do veřejné počítačové sítě. Musela být proto vytvořena nová verze protokolu předcházející vyčerpání všech adres. Novější verze IPv6 tento problém řeší navýšením počtu bajtů pro adresaci (čtyřnásobně), která by měla vystačit i pro dalekou budoucnost. Existují nástroje, které nedostatek adres řeší (překládání veřejných adres na soukromé, které již nemusí být světově unikátní) a prodlužují tak životnost IP adres verze 4. Také přechod z již zaběhnuté starší verze na novější ve velikých provozech a organizacích není vždy úplně snadný. Z těchto důvodů velká část síťových zařízení stále funguje na IPv4, včetně produktů testovaných v laboratořích společnosti Siemens. Proto bude následující kapitola věnována pouze této variantě.

IPv4 tedy dovoluje adresaci a směrování datových paketů z vysílače k přijímači přes několik různých sítí. Každá stanice, která si přeje komunikovat s jinou stanicí, je identifikována pomocí unikátní IP adresy. Při cestě dat po síti je pomocí cílové adresy rozhodováno, kterému uzlu se paket odešle, dokud nedorazí k cíli. Data, která jsou adresována pomocí IP adresy, jsou nazývána IP datagramy. Ten obsahuje podobně jako ethernetový rámec (3.1) hlavičku následovanou samotnými daty. Hlavička obsahuje množství polí, které zajišťují přenos datagramu po síti. Minimální délka je 20 bajtů a je definován následujícím způsobem.



Obrázek 3.2: Struktura IP paketu [inspirováno z 5]

Takový datagram je zapouzdřen do vyšší vrstvy, což je například ethernetový rámec. Datový blok ethernetového rámce tudíž poté obsahuje ethernetovou hlavičku následovanou celým IP datagramem. Výsledný datagram poté může vypadat následovně:



Obrázek 3.3: Zapouzdření IP paketu do ethernetového rámce [zdroj vlastní]

Uvedený princip je používán v této podobě také pro přenos některých dat mezi Profinet řídicím systémem a vstupně výstupním zařízením.

3.3.2 IP adresy

IP adresy jsou na rozdíl od MAC adres nezávislé na konkrétním hardwaru. Jedna stanice může mít v průběhu času různé IP adresy a jedna IP adresa může patřit různým stanicím, ne však v jednom okamžiku. Adresy se používají pro označení odesílatele a příjemce v každém datovém paketu přenášené IP protokolem. Každá Profinet stanice musí tedy disponovat jedinečnou IP adresou, která je jí přiřazena a pomocí které je stanice adresována.

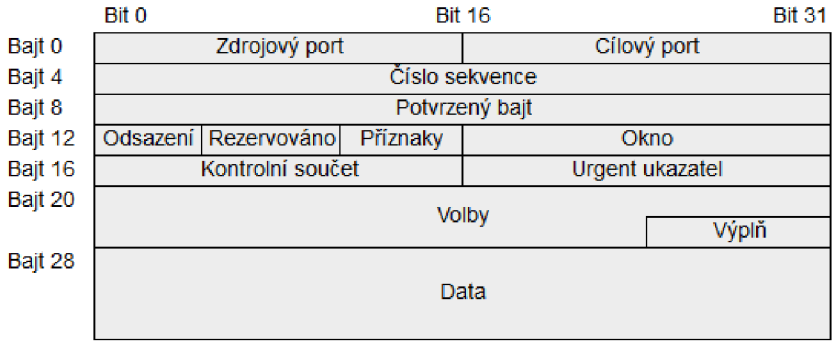
Čtyři bajty, ze kterých se IP adresa skládá, lze rozdělit na dvě logické části. Síťová a hostitelská. Síťová část označuje adresu celé sítě, do které stanice patří, a hostitelská identifikuje již konkrétní stanici. Která část je síťová a která hostitelská, poznáme z takzvané masky sítě. Ta říká, kolik bitů zleva označuje síťovou část. Zbylé bity patří hostitelské části. Díky tomuto rozdělení lze stanice seskupit do skupin, které budou mít stejnou síťovou část IP adresy, avšak jinou hostitelskou. Takové stanice pak patří do jedné sítě. Větší maska tedy znamená menší maximální počet stanic v jedné síti. Všechna zařízení s technologií Profinet, včetně řídicího systému, jsou v praxi seskupovány do stejné sítě, přes kterou spolu komunikují. Přiřazení adresy v Profinet síti ke konkrétnímu zařízení je popsáno v části 2.5.5.

Speciální případ IP adresy je takzvaná broadcast adresa. Rozeznat ji od klasické adresy lze tak, že všechny bity hostitelské části adresy jsou nastaveny na hodnotu 1. Poslání dat

na takovou adresu potom znamená, že cílem jsou všechny stanice v síti, která je uvedena v síťové části adresy. Broadcast adresu využívá v Profinetu například protokol DCP při vyhledávání všech stanic v dané síti. Protokol pošle požadavek na identifikaci na broadcast adresu, všechna zařízení v této síti požadavek obdrží, a pokud stanice podporuje DCP protokol, odpoví na něj příslušnou odpovědí obsahující například název stanice nebo IP adresu. V požadavku je uvedena zdrojová IP adresa, podle které každá stanice zjistí, na jakou adresu odpověď poslat.

3.3.3 TCP

Přenos dat přes Ethernet pomocí IP adres je poměrně nespolehlivý. Datové pakety mohou být ztraceny v důsledku chyby na některém přenosovém médiu nebo přílišným zatížením sítě. Například nemusí dorazit vůbec, mohou dorazit vícekrát než jen jednou nebo v jiném pořadí, než byly původně odeslány. Transportní vrstva, která je postavena nad internetovou vrstvou, může garantovat spolehlivý a kompletní přenos informací mezi odesílatelem a příjemcem. Protokol obsahující taková opatření je nazván TCP (Transmission Control Protocol). Při výměně dat mezi stanicemi jsou data odesílatele potvrzována příjemcem. Jsou zde proto mechanismy jako kontrola chyb, řízení toku dat nebo potvrzování přijetí paketu. TCP navazuje vztah mezi stanicemi pomocí takzvaných portů. Porty označují typ dat přenášených na vyšší vrstvě a rozhodují o tom, jak bude s daty naloženo na straně příjemce. Navázání TCP vztahu je potom adresováno pomocí kombinace IP adresy a portu. Čísla portů jsou přiřazena k určité aplikaci. Například pro službu HTTP je vyhrazen port 80, emailové služby (SMTP) běží na portu 25. Struktura TCP paketu je následující:



Obrázek 3.4: Struktura TCP paketu [inspirováno z 5]

Zařízení Profinetu využívají TCP přenos například pro komunikaci s webovým serverem. Navázání komunikačního vztahu mezi zařízením a řídicím systémem využívá jiný typ přenosu zvaný UDP (3.4).

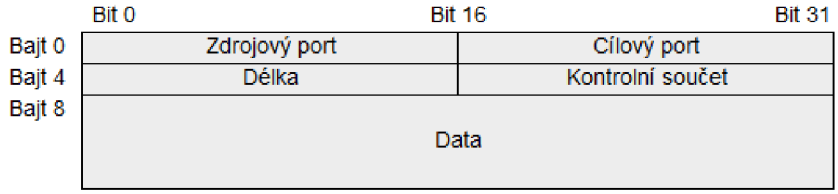
3.4 UDP/IP

Ne vždy je potřeba mít veškerou komunikaci natolik spolehlivou jako je TCP přenos. Situací, kdy není TCP potřeba, je několik:

- Rychlost přenosu je důležitější než jistota spolehlivého přenosu. Zajištění spolehlivosti u TCP zvyšuje režii i počet přenesených informaci. Tím se zpomaluje i samotný přenos.

- Síť je dostatečně zajištěna proti chybám přenosu.
- Spolehlivý přenos dat je dosažen pomocí jiného nástroje. Například pomocí vyšší vrstvy, která má zajištění spolehlivosti ve své vlastní režii.

Byla proto vytvořena minimalistická varianta přenosového protokolu obsahující pouze nejnutnější informace pro přenos dat z jedné stanice do druhé, zvaná UDP (User Datagram Protocol). Přenos UDP protokolem nevytváří žádné spojení mezi stanicemi, jako je tomu u TCP, pouze odešle data na určitou adresu. Příjemce přijatá data přečte, ale už je nemusí (pokud to nevyžaduje vyšší vrstva) nijak potvrdit. Hlavička UDP paketu obsahuje pouze port, délku a volitelnou položku kontrolní součet, jejímž vynecháním lze dosáhnout ještě menší režie přenosu. Díky tomu může být UDP přenos až třikrát rychlejší, než je tomu u TCP. To vše za cenu možné ztráty, duplikací a chyb v pořadí posílaných dat. Právě kvůli vysokým rychlostem přenosu využívá Profinet protokol UDP například pro výměnu dat týkajících se navázání/ukončení aplikačního vztahu popsaného v sekci 2.5, ale také pro všechny komunikační vztahy zmíněné v sekci 2.4. Všechny možné problémy přenosu jsou detekovány na vyšší aplikační vrstvě zapouzdřené právě do UDP.



Obrázek 3.5: Struktura UDP paketu [inspirováno z 5]

3.5 Ostatní protokoly

Zařízení Profinetu využívají několik dalších služeb. Nedůležitější z nich jsou protokoly ARP a ICMP, protokoly nezbytné pro získání potřebných informací pro síťovou komunikaci.

3.5.1 ARP

Protokol se zkratkou ARP (Address Resolution Protocol) má jednu hlavní funkci, a to propojení IP adresy s fyzickou adresou (MAC) nějaké stanice v síti. To je důležité pro přenos IP paketu do odpovídající ethernetové sítě, jelikož stanice jsou zde adresované pomocí MAC adresy. ARP služba funguje automaticky a nemusí být explicitně volána. Každá stanice v síti spravuje tabulku adres nazývanou ARP cache, kde jsou uloženy dvojice IP a MAC adres. Ta je aktualizována automaticky ARP protokolem a nemusí být proto manuálně vytvářena. ARP protokol se stará o získávání požadovaným informací z tabulky, a uživatel tak nemusí přiřazení IP adresy k MAC adrese vůbec řešit. Pokud nějaká dvojice adres není v tabulce obsažena, stanice pošle ARP požadavek na broadcast adresu. Ta obsahuje IP a MAC adresu zdroje a IP adresu cíle. Takový požadavek dostane každá stanice v síti, která vyhodnotí, jestli se shoduje adresa cíle v požadavku s adresou konkrétní stanice. Pokud ano, odešle stanice na zdrojovou adresu požadavku informace o svojí MAC adrese. Stanice, která takovou odpověď přijme, nyní zná požadovanou dvojici MAC a IP adresy a zapíše si je do své ARP tabulky. Existuje také opačná varianta protokolu zvaná RARP, která získává IP adresu pomocí MAC adresy. [8]

3.5.2 ICMP

ICMP protokol (Internet Control Message Protocol) realizuje přenos stavových a chybových zpráv mezi IP síťovými uzly. ICMP nabízí dvě důležité služby pro získání informací na konkrétní síti:

- *Ping* – služba vyšle požadavek zvaný "Echo Request"na konkrétní IP adresu a očekává odpověď zvanou "Echo Reply". Stanice, která obdrží takový požadavek, odešle odpověď na zdrojovou adresu požadavku. Tento mechanismus slouží k určitě diagnostice. Z odpovědi lze zjistit, zda se v síti fyzicky nachází nějaká stanice s danou IP adresou a za jakou dobu přijde z takové stanice odpověď. Stanice v Profinet síti pak může tímto způsobem snadno vyhodnotit například to, zda jsou okolní zařízení zapnutá.
- *Traceroute* – vypisuje všechny uzly, které se nacházejí na cestě datagramu k zadané cílové IP adrese.

Kapitola 4

Aplikace Virtual CPU

Pro testování konkrétních V/V-zařízení je potřeba také V/V-kontrolér, se kterým je vytvořen aplikační vztah. Po navázání vztahu začíná veškerá výměna dat, která musí být monitorována a testována, zda splňuje požadavky definované standardem Profinet, ale také požadavky týkající se samotného zařízení. Testovány musí být informace na všech třech komunikačních kanálech (Record Data CR, IO-Data CR, Alarm CR). Například pomocí vztahu Record Data CR musí být kontrolováno, zda zařízení příjme pouze takové hodnoty, které jsou v platném rozsahu. Rovněž data určená pouze pro čtení nesmí obsahovat nepovolené hodnoty. Testování dat přenášených pomocí tohoto vztahu lze provádět i bez V/V-kontroléru pomocí implicitního vztahu. To už ale neplatí pro IO-Data CR a Alarm CR, jelikož jejich výměna začíná až po úspěšném navázání aplikačního vztahu typu IOC-AR 2.5.1. Existují sice možnosti, jak data monitorovat přímo ve V/V-kontroléru, ale tyto možnosti jsou do jisté míry omezené. Proto je vhodné vytvořit způsob, jak aplikační vztah vytvořit bez V/V-kontroléru a monitorovat data na úrovni aplikace běžící na stolním počítači.

4.1 Připojitelná zařízení

Navrhovaná aplikace, nazvaná "Virtual CPU", má za úkol vytvořit aplikační vztah typu IOC-AR s V/V-zařízením připojeným přes standardní síťové rozhraní. Z důvodu složitosti paketu Connect Request, který obsahuje informace týkající se konkrétního zařízení, bude aplikace zaměřena pouze na dva typy zařízení používané a testované ve společnosti Siemens. Jedná se o následující zařízení rodiny SITOP podporující Profinet:

1. *SITOP UPS 1600* – zařízení chránicí před výpadkem proudu v průmyslové výrobě a zajišťující správné ukončení výroby v případě výpadku. Umožňuje napájení dalších zařízení až 24 volty. Obsahuje dva ethernetové porty, které mohou fungovat jako přepínač. Jsou k dispozici 3 varianty lišící se výstupním proudem – 10 A, 20 A, 40 A. [1]
2. *SITOP PSU 8600* – napájecí zdroj rovněž vybaven dvěma ethernetovými porty. Umožňuje širokou škálu nastavení a diagnostiky potřebné pro řízení automatizované výroby v průmyslu. Aktuálně je možné vybrat si mezi čtyřmi variantami produktu lišícími se počtem výstupů a výstupním proudem. Nabízí také připojení až šesti modulů, které rozšiřují funkcionalitu. K dispozici jsou například moduly pro navýšení počtu výstupů nebo moduly sloužící jako záložní zdroj energie v případě výpadku elektřiny. [2]

Všechny uvedené informace o komunikaci se vstupně výstupním zařízením platí pro všechna zařízení s technologií Profinet. Přestože je aplikace Virtual CPU navržena obecně, bude směrována a testována pouze na uvedených dvou typech zařízení rodiny SITOP. Přímá podpora komunikace s jinými typy zařízení není v momentální době nutná, jelikož společnost Siemens v Brně jiné produkty nevyvíjí ani netestuje. Rozšíření podpory jiných typů zařízení je jedním z možných bodů budoucí práce na aplikaci.

4.2 Požadavky na aplikaci

Společnost Siemens zadala několik hlavních požadavků, které má aplikace splňovat, aby byla přínosem pro usnadnění testování zařízení. Samotná aplikace není určena pro testování validity konkrétních dat, ale má pouze připravit prostředí pro následné vytváření testů. Testovací zařízení bude připojeno ke stolnímu počítači, na kterém bude spuštěna aplikace. Na základě konkrétních požadavků pro testování bude poté možno vypracovat konkrétní testovací skripty, které mohou zpracovávat data získané aplikací Virtual CPU.

4.2.1 Funkcionalita

Aby bylo možné monitorovat a zároveň tedy i testovat všechna potřebná data, musí aplikace navázat validní aplikační vztah typu IOC-AR (2.5.1). Po úspěšném navázání musí začít výměna cyklických dat kanálem zvaným IO-Data CR. Dalším důležitým kritériem je reakce na alarmy. Samozřejmostí je také umožnění acyklického čtení/zápisu dat pomocí vztahu Record Data CR. Tím jsou splněna všechna základní kritéria pro plnohodnotný vztah mezi V/V-kontrolérem a V/V-zařízením. V/V-kontrolér je zde nahrazen aplikací, která jeho chování simuluje.

1. Navázání a ukončení vztahu s konkrétním V/V-zařízením.
2. Monitorování vstupních cyklických dat.
3. Posílání validních cyklických dat z důvodu udržení vztahu.
4. Monitorování přijatých alarmů.
5. Potvrzení přijatých alarmů.
6. Acyklické čtení/zápis dat.

4.3 Návrh

Návrh na implementaci aplikace proběhl ve spolupráci jak s vývojovým, tak testovacím týmem společnosti Siemens. Následující kroky uvedené v této sekci zahrnují všechny části potřebné pro testování zařízení s využitím navrhované aplikace.

4.3.1 Navázání aplikačního vztahu

Pro úspěšné navázání vztahu typu IOC-AR musí být provedeny všechny kroky uvedené v diagramu 2.5. Paket Connect Request obsahuje množství polí týkajících se konkrétního zařízení, se kterým chceme komunikovat. Je potřeba definovat všechna tato pole, jejich datový typ, délku, výchozí hodnoty a rozmístění v paketu. Definování paketu bude probíhat

na základě komunikace mezi skutečným V/V-kontrolérem a V/V-zařízením zachycené pomocí programu Wireshark - nástroj pro zachycení komunikace na síťovém rozhraní. Pole mají proměnlivou délku, proto komponenta, která bude Connect Request vytvářet, musí také počítat aktuální délku jednotlivých položek. Po odeslání paketu je potřeba zachytit odpověď z V/V-zařízení a vyčíst z ní informaci, zda byl Connect Request úspěšný. Dále je potřeba poslat paket DControl Request ze strany V/V-kontroléru, který V/V-zařízení potvrdí paketem DControl Response. Následuje čekání na paket CControl Request, který pošle V/V-zařízení potvrzující zahájení vztahu. V/V-kontrolér musí odeslat paket CControl Response, který potvrdí úspěšné přijetí paketu.

4.3.2 Acyklická data

Po úspěšném navázání aplikačního vztahu bude hlavní smyčka očekávat případný požadavek uživatele zápisu/čtení konfiguračních dat z/do V/V-zařízení 2.1. Ten musí obsahovat adresu cíle zápisu/čtení. V případě zápisu také samotná data. Rovněž je nutné přečíst odpověď takového požadavku. Tím je splněn požadavek, který říká, že aplikace musí umožňovat práci s konfiguračními daty.

4.3.3 Cyklická data

Udržení aplikačního vztahu je podmíněno pravidelnou výměnou cyklických dat 2.2. Tento proces musí běžet na pozadí aplikace, aby bylo možné hlavní smyčku programu využít pro případný zápis acyklických dat. Nabízí se rozdělení programu na vlákna. Komponenta, která bude mít za úkol odpovídat na IO-Data, začne vykonávat svoji funkci ve chvíli, kdy začne V/V-zařízení generovat cyklická data. Data budou monitorována na straně aplikace a bude je tedy možno testovat na správnost podle jejich hodnoty.

4.3.4 Alarmy

Poslední hlavní částí aplikace je monitorování a potvrzení vzniklých alarmů. Aplikace bude monitorovat všechny alarmy vyvolané na straně V/V-zařízení. Pro správnou obsluhu alarmu je potřeba provést kroky uvedené na diagramu 2.4. Stejně jako cyklická data musí být alarmy obslouženy procesem běžícím na pozadí. Vznikne zde tedy další vlákno, které se bude starat o zpracování a potvrzení všech příchozích alarmů. Jeden z požadavků na aplikaci byla také práce s diagnostickými daty. Ten je splněn korektním potvrzování vzniklých alarmů, ale také přijímáním a posíláním cyklických dat popsaných v předešlé sekci.

4.3.5 Uživatelské rozhraní

Uživatelské prostředí bude realizováno pomocí příkazové řádky. Grafické prostředí není v tomto případě nutné, jelikož aplikace je směřována pouze na zaměstnance firmy Siemens, nikoliv běžné uživatele. Ovládání z příkazové řádky také přináší výhodu automatického pouštění aplikace nebo ovládání z jiných nástrojů. Návrh na interakci s uživatelem je následující: Před spuštěním zadá uživatel název zařízení, se kterým chce aplikační vztah navázat. Bude možno definovat konfigurační data, která se do V/V-zařízení zapíšou po vytvoření spojení. Následuje automatická obsluha cyklických dat a alarmů pomocí vytvořených vláken. Pomocí hlavní smyčky bude možné zapisovat acyklická data prostřednictvím příkazové řádky. Do té je možno také vypisovat monitorovaná data, která mohou být využita jinými

testovacími aplikacemi. Při ukončení aplikace bude poslán požadavek na validní ukončení aplikačního vztahu 2.5.4.

4.3.6 Využití pro testování

Externí aplikace (testovací skripty) musí mít přístup ke všem datům zachyceným na lince mezi V/V zařízením a řídicím systémem. Bez toho by aplikace dokázala zastoupit reálný řídící systém, ale nebylo by možno ji využít pro testování zařízení. Aplikace bude mít proto možnost převzít funkce, ve kterých budou zachycená data přístupná. Tyto funkce budou implementovat konkrétní testovací scénáře.

4.4 Výběr technologie

Programovací jazyk, ve kterém bude aplikace implementována, nebyl dopředu stanoven. Většina testů vyvíjených týmem, který testuje produkty rodiny SITOP ve společnosti Siemens, je napsána v programovacím jazyce Python. Proto jsem si vybral tento jazyk i pro implementaci aplikace Virtual CPU. Python jsem zvolil také proto, že je to jazyk, se kterým mám nejvíce zkušeností a vyhovuje mi způsob, jakým v něm lze aplikace implementovat. Další výhodou je možnost použití knihovny Scapy 4.5, která je rovněž napsána v jazyce Python. Vývojových prostředí pro Python je mnoho. Já jsem zvolil program PyCharm od společnosti JetBrains. Je to prostředí, které nabízí širokou škálu nástrojů usnadňujících programování. Obsahuje například nástroje pro krokování kódu, přechod na konkrétní třídy, využití systému pro správu verzí Git, našeptávání možných příkazů a mnoho dalšího. Program PyCharm byl rovněž zvolen z důvodu mé dobré předešlé zkušenosti.

Dále bude potřeba nástroj, kterým lze zachytit a dekódovat data zachycená na určeném síťovém rozhraní. K tomu se nabízí velmi známý a uživateli osvědčený program Wireshark. Ten navíc zná pakety standardu Profinet, takže veškerá data v tomto standardu jsou již dekódována do příslušných polí a lze je snadno vyčíst. Jak již bylo řečeno, bude použita knihovna Scapy pro správu síťové komunikace popsaná v sekci 4.5. Přehled využitých nástrojů a technologií:

- Programovací jazyk Python.
- PyCharm.
- Wireshark.
- Knihovna Scapy.

4.5 Scapy

Pro potřeby výsledné aplikace musíme mít způsob, jak manipulovat s pakety, kódovat a dekódovat jejich data, posílat a přijímat pakety na určeném rozhraní a mnoho dalšího. Vše musí probíhat jak na vrstvě MAC adres, tak na IP vrstvě, protože komunikace mezi V/V-kontrolérem a V/V-zařízením využívá oba způsoby. Nabízí se zde proto známý a velmi často používaný nástroj Scapy. Scapy je knihovna napsaná v programovacím jazyce Python, ve kterém bude rovněž realizována výsledná aplikace pro simulaci kontroléru. Nejen že tento nástroj obsahuje všechnu zmíněnou funkcionalitu, ale také je to nejčastěji používaný

nástroj pro manipulaci síťové komunikace na celém světe. Proto bude využit i pro výslednou aplikaci. [3]

Scapy zvládá všechny základní, ale i pokročilé funkce nutné pro síťovou komunikaci. Zakódování dat podle všech používaných, předem definovaných protokolů jako DNS, DHCP, SNMP a mnoho dalších. Poslání zakódovaných dat na kabel nebo pomocí Wi-fi adaptéru. Dále také: zachytávání dat, automatické spojení dotazu a odpovědi, skenování zatížení sítě nebo přehled statistik přenesených dat. Také umožňuje vytvoření nového protokolu stejným způsobem jako jsou již vytvořeny protokoly DNS, DHCP a další. Tohoto využijeme, jelikož pakety a rámce standardu Profinet knihovna Scapy neobsahuje.

4.6 Zařízení pro testování aplikace

Společnost Siemens zapůjčila pro potřeby vývoje a testování aplikace produkt SITOP UPS 1600 ve dvaceti ampérové variantě. Toto zařízení bude sloužit jako referenční V/V-zařízení. Zařízení je potřeba napájet 24 volty. Byl tedy zapůjčen i napájecí zdroj. Pro testování aplikace s druhým produktem SITOP PSU 8600 je možné libovolně využít prostory společnosti Siemens, ve kterých je dostatek všech variant produktu a připojitelných modulů. Napájení vyžaduje 400 voltů, proto jej není možné zapojit v domácím prostředí.

Kapitola 5

Implementace

Implementace vychází z návrhu uvedeného v kapitole 4.3. Každá sekce popisuje implementaci jednotlivých logických celků, jejichž spojením vzniká výsledná aplikace. Sekce nezacházejí do implementačních detailů, ale poukazují na všechny záležitosti, které musely být při vývoji řešeny.

5.1 Rozvržení programových komponent

Aplikace může být v budoucnu rozšiřována o další funkcionalitu (6.1). Také musí být snadno udržovatelná a upravitelná v případě potřeby. Je proto rozdělena do několika hlavních komponent, kde každá plní svoji konkrétní funkci. Jednotlivé komponenty jsou reprezentovány v jazyce Python pomocí tříd. Vzájemná spolupráce všech komponent zajistí správné fungování aplikace. Program je proto složen z několika souborů:

- `vcpu.py`
- `constants.py`
- `cyclic.py`
- `alarm.py`
- `packet_generator.py`
- `dev_config.py`
- `prof_fields.py`
- `scapy_fields.py`
- `settings.py`

5.1.1 Generování paketů

Nejobsáhlejší třída se nazývá `ProfinetPacketGenerator`. Jak již název napovídá, její úloha je vytvářet všechny potřebné pakety které budou posílány do vstupně výstupního zařízení pomocí síťového rozhraní. Využívá předem definované pole pomocí objektů blíže popsané v kapitole 5.2, které plní konkrétními daty. Data jsou poskládána za sebe v určeném pořadí a ta poté převedena do pole bajtů, které je již připraveno na zapouzdření do vyšší

síťové vrstvy. Pomocné funkce vypočítávají délky proměnných polí a jejich samotná data. Obsaženy jsou také funkce pro výpis již vytvořených polí a funkce pro jejich kontrolu.

5.1.2 Přenos cyklických dat

Aby mohl být splněn požadavek na validní výměnu cyklických dat (2.4.2) mezi vstupně výstupním kontrolérem a zařízením, což je i podmínkou pro udržení komunikačního vztahu, je vytvořena třída s názvem `CycliDataResponder`. Tato komponenta je spuštěna jako samostatný proces, který běží na pozadí hlavní aplikace. Má jednu hlavní funkci, a to poslouchat na určeném síťovém rozhraní veškerou komunikaci, filtrovat pouze rámce (zde je vše přenášeno na vrstvě MAC adres) cyklických dat a na každý takový přijatý rámec validně odpovědět.

5.1.3 Potvrzování alarmů

Obdobně jako komponenta obsluhující cyklická data funguje i komponenta, která potvrzuje přijaté alarmy. Ta je nazvána `AlarmResponder`. Pokud by některý z alarmů, jenž byl vygenerován na vstupně výstupním zařízení, nebyl druhou stranou validně potvrzen, komunikační vztah se ukončí. Proto je vytvořen i další proces běžící nezávisle na hlavním programu. Proces naslouchá na síťovém rozhraní, filtruje rámce alarmů a zahajuje veškeré operace pro potvrzení alarmu (2.4.3).

5.1.4 Aplikační vztah

Samotné navázání vztahu obsluhuje komponenta nazvaná `Vcpu`. Jedná se o hlavní program, který využívá všechny ostatní komponenty a reprezentuje konkrétní komunikační vztah. Třída `Vcpu` využívá komponentu pro generování Profinet paketů (5.1.1), ze které získá potřebná data. Ta poté zabalí do vyšších vrstev. Po zaslání požadavku na vytvoření vztahu jsou odstartovány pomocné procesy pro přenos cyklických dat a potvrzování alarmů. Hlavní smyčka čeká na případný požadavek uživatele na vyčtení nebo zápis konfiguračních dat (2.4.1). Pokud uživatel zadá požadavek na ukončení vztahu, komponenta vykoná operaci pro jeho validní ukončení (2.5.4).

5.1.5 Pomocné komponenty

Neméně důležitou součástí aplikace je několik souborů datového charakteru. Ty převážně neprovádějí výpočty ani nemění výrazným způsobem běh programu, ale pouze nesou potřebné informace pro správné fungování aplikace. Data jsou v jazyku Python uložena v objektech ať už jednoduchých typů (řetězce, čísla) nebo typů složitějších (kombinace jednoduchých).

5.2 Definice polí

Profinet komunikace mezi kontrolérem a vstupně výstupním zařízením obsahuje veliké množství informací potřebné pro správný běh komunikačního vztahu. Ty jsou posílány na klasickém ethernetu. Některé na IP vrstvě, jiné na vrstvě MAC adres. Všechny informace jsou rozděleny do předem definovaných polí, jak je tomu i u jiných známých síťových protokolů. Pro jednotlivá pole musí být předem definována jejich velikost, datový typ, popřípadě výchozí hodnota. Aplikace poté nastaví hodnotu všech potřebných polí, vytvoří

z nich paket (případně rámec), zabalí je do správné síťové vrstvy a pošle jej na specifikované síťové rozhraní. Dodržení správného pořadí posílání paketů zajistí validní navázání a udržení komunikačního vztahu.

5.2.1 Využití knihovny Scapy

Knihovna Scapy má již předem vytvořená pole pro běžně používané protokoly jako SMTP (emailová komunikace), DHCP (přidělování síťových parametrů) nebo FTP (přenos souborů). Profinet protokol je ale knihovně cizí. Je však možné všechna potřebná pole dodefinovat. Přenos na lince je zapouzdřen do několika síťových vrstev. Poslední dvě vrstvy, DCE/RPC (Ditibuted Computing Environment/Remote Procedure Call) a Profinet IO nejsou v knihovně obsaženy, a proto je musíme doplnit. Scapy umožňuje vytvořit vlastní protokol, se kterým může být dále nakládáno jako s již definovanými protokoly. Způsob definice jednotlivých polí, které vrstva obsahuje, probíhá následovně: Nejprve je zapotřebí vybrat správný objekt, který reprezentuje datový typ pole. Udává tedy velikost, způsob kódování (endianita) a datový typ. Některé datové typy, například textové řetězce, mají proměnlivou velikost, kterou je potřeba určit vhodným parametrem. Je nutno nastavit také jméno každé položky a výchozí hodnotu. Tímto způsobem vytvoříme všechna pole, která obsahuje DCE/RPC vrstva. Definice vypadá následovně:

```
ByteField("Version", 4),
ByteEnumField("PacketType", 0, {0: "Request", 2: "Response", 6: "Reject"}),
XByteField("Flags1", FLAGS_1),
XByteField("Flags2", 0x00),
X3BytesField("DataRepresentation", 0x100000),
XByteField("SerialHigh", 0x00),
StrFixedLenField("ObjectUUID", None, length=16),  # unique for each device
StrFixedLenField('InterfacePNIO_UUID', INTERFACE_PNIO_UUID, length=16),
StrFixedLenField("Activity", ACTIVITY, length=16),
LEIntEnumField("ServerBootTime", 0, {0: "Unknown", 1: "Stredni evropa"}),
```

Obrázek 5.1: Definice polí pomocí knihovny Scapy [zdroj vlastní]

5.2.2 Použití vlastního objektu

Problém nastane při pokusu definovat takto také Profinet IO vrstvu. Na rozdíl od DCE/RPC je Profinet část velmi složitá a variabilní. V závislostí na tom, s jakým zařízením pracujeme, se liší velikosti daného paketu pro navázání vztahu. Ta se může pohybovat kolem pár desítek bajtů, ale také může přesáhnout maximální velikosti paketu 1500 bajtů. V takovém případě je nutno paket fragmentovat. Fragmentace probíhá ve většině případů automaticky. Ovšem Profinet tuto fragmentaci řeší svým způsobem. Pokud je Profinet část větší než předem definovaná velikost (v našem případě 1384 bajtů), je rozdělena na několik fragmentů. Informace o fragmentaci jsou uvedeny v DCE/RPC části. Rozdělení musí tedy být prováděno manuálně. Knihovna Scapy tyto operace nepodporuje, a proto nemůže být použita pro vytvoření paketů Profinet IO vrstvy.

Z uvedených důvodů je použit pro vytváření paketů vlastní objekt nazvaný `ProfField`, který obsahuje jméno, délku, datový typ a samotná data. Třída, která má na starost vytváření Profinet paketů, potom tyto objekty skládá v stanoveném pořadí a plní je potřebnými daty.

```
ReleaseReq = [
    ProfField("BlockHeader", 6, None),
    ProfField("Reserved1", 2, 0),
    ProfField("ARUUID", 16, ARUUID),
    ProfField("SessionKey", 2, 2),
    ProfField("Reserved2", 2, 0),
    ProfField("ControlCommand", 2, 4),
    ProfField("ControlBlockProperties", 2, 0)
]
```

Obrázek 5.2: Definice polí pomocí vlastní třídy [zdroj vlastní]

5.3 Datové soubory

Pro definici polí protokolu Profinet jsou využity soubory `prof_fileds.py` pro pole definované vlastní třídou (5.2.2) a `scapy_fields.py` pro pole definované pomocí knihovny Scapy (5.2.1). Další soubor nazvaný `constants.py` obsahuje veškeré pevně nastavené pojmenované hodnoty. Tím se zlepší čitelnost kódu. Všechna data v tomto souboru mají konstantní charakter, není tedy potřeba je měnit. Ovšem v případě potřeby, například při rozšiřování aplikace o další funkcionalitu, mohou být hodnoty upraveny. Změny provedeny v tomto souboru se pak projeví všude, kde jsou používány. Příklad takhle definovaných hodnot může být: číselná identifikace jednotlivých typů paketů, maximální délky fragmentů nebo indexy polí. Soubor `settings.py` obsahuje informace týkající se konkrétního počítače, na kterém aplikace běží, například název síťové karty, na které budeme připojovat vstupně výstupní zařízení. Tento soubor je potřeba upravit při přenesení aplikace na jiný počítač. Poslední použitý soubor nazván `dev_config.py` nese definice připojitelných zařízení. Soubor je blíže popsán v kapitole 5.5.

5.4 Profinet IO pakety

Největší část aplikace tvoří třída `ProfinetPacketGenerator` . Každá její instance reprezentuje data jednoho Profinet požadavku, který lze odeslat (po zapouzdření) do vstupně výstupního zařízení, například požadavek na zahájení aplikačního vztahu nebo požadavek na zápis konfiguračních dat do zařízení. Konstruktor třídy přebírá dva povinné parametry. Jeden z nich je identifikátor určující konkrétní požadavek. Druhý parametr je objekt, nazvaný `dev_config`, definující konkrétní zařízení. Pomocí těchto dvou informací poté třída vytvoří paket konkrétního požadavku s daty konkrétního zařízení. Další nepovinný parametr využívá například požadavek na zápis informací do zařízení, kde parametr určuje samotná data a adresu zápisu. Třída podporuje vytvoření několika požadavků:

(a) *Connect Request* – Navázání aplikačního vztahu. Tento požadavek je složen celkem ze tří paketů, které jsou odeslány do zařízení pro validní navázání vztahu (2.5.3). Jelikož tyto tři pakety jsou poslány vždy ve stejném pořadí, a poslání jen některého z nich nemá význam, je tato sekvence zobecněna jako jeden požadavek nazvaný Connect Request

(b) *Write Request* – Zápis diagnostických dat do zařízení.

(c) *Read Request* – Vyčtení diagnostických dat ze zařízení.

(d) *Release Request* – Ukončení aplikačního vztahu.

Vytvoření konkrétního požadavku využívá množinu definovaných polí pomocí třídy `ProfField` (5.2.2). Množina představuje šablonu pro konkrétní požadavek. Tu třída zkopíruje, zachová výchozí hodnoty, které není potřeba měnit, a přepočítá hodnoty, které mohou být v každém požadavku odlišné. Pro zjišťování proměnlivých polí obsahuje třída `ProfinetPacketGenerator` několik pomocných funkcí, které počítají jejich hodnotu. Jedním z příkladů, kde je potřeba hodnoty polí dopočítat, je vytvoření požadavku na navázání vztahu. V něm jsou uvedeny informace týkající se konkrétního zařízení, se kterým je vztah navazován. Ty jsou uloženy v již zmiňovaném objektu `dev_config`, který třída převezme jako parametr v konstruktoru při vytváření instance. Funkce poté objekt postupně prochází a vytváří podle něj příslušná pole.

Jakmile jsou všechna pole vytvořena, funkce assembly je převede do binární podoby, ve které již mohou být zapouzdřena do vyšších vrstev.

5.5 Předání informací o zařízení

Informace uvedené v GSD souboru (2.3) jsou nezbytné pro navázání komunikačního vztahu. Kontrolér, případně aplikace, která vztah navazuje, musí znát jak samotnou konfiguraci (typy zařízení, se kterým je vztah navazován), tak i informace o samotných zařízeních a připojených modulů. Ty jsou uvedeny právě v GSD. Pro potřeby aplikace není nutno číst celý soubor, který je poměrně obsáhlý a obsahuje i data, jež nejsou pro tyto účely relevantní. Potřebná data každého zařízení budou proto definována v speciálním objektu v jazyce Python. Jelikož aplikace je zaměřena pouze na dva konkrétní typy zařízení (4.1) a několik připojitelných modulů, není proto problém mít jejich technickou charakteristiku uvedenou přímo v aplikaci. Tím získáme výhodu nezávislosti aplikace na GSD souboru. Z takového objektu jsou data čtena komponentou, která vytváří Profinet pakety. V případě budoucího rozšiřování aplikace o podporu dalších zařízení může být vytvořena komponenta, která důležitá data z GSD souboru vyčte a do objektu je zapíše ve stanoveném formátu.

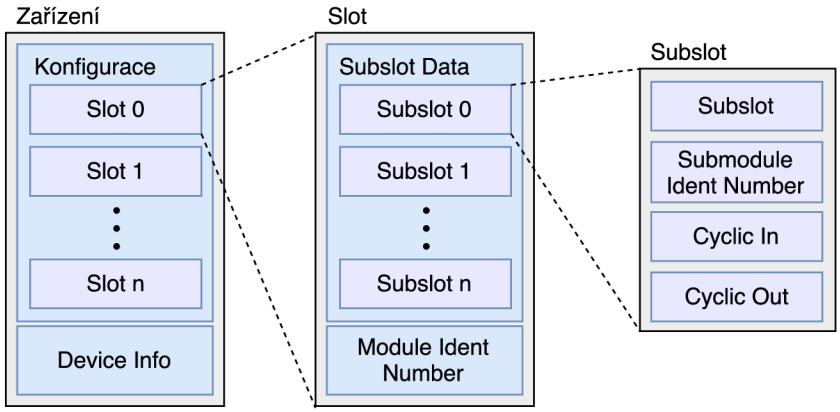
V objektu jsou uložena následující data:

1. V rámci celé konfigurace:
   - *Konfigurace* – použité zařízení a pořadí fyzického zapojení modulů na jednotlivé sloty.
   - *Deivce Info* – identifikátor označující obecný typ zařízení.
2. Pro každý slot:
   - *Subslot data* – definice subslotů na daném slotu.
   - *Module Ident Number* – identifikátor konkrétního modulu.
3. Pro každý subslot:
   - *Subslot* – číslo subslotu.
   - *Submodule Ident Number* – identifikátor jednotlivých subslotů.
   - *Cyclic In* – udává, kolik bajtů dat je cyklicky posíláno ze zařízení do aplikace pro daný subslot.

- *Cyclic Out* – udává, kolik bajtů dat je na straně zařízení přijímáno z kontroléru pro daný subslot.

Aby šlo s objektem definujícím zařízení dobře pracovat, byla vytvořena třída `Slot`, která implementuje několik základních funkcí jako je procházení jednotlivých subslotů, získání jejich délek nebo přístupu ke konkrétním prvkům. Každá instance třídy Slot pak reprezentuje fyzický slot zařízení. Vložení jednotlivých instancí třídy Slot do objektu potom reprezentuje celou konfiguraci.

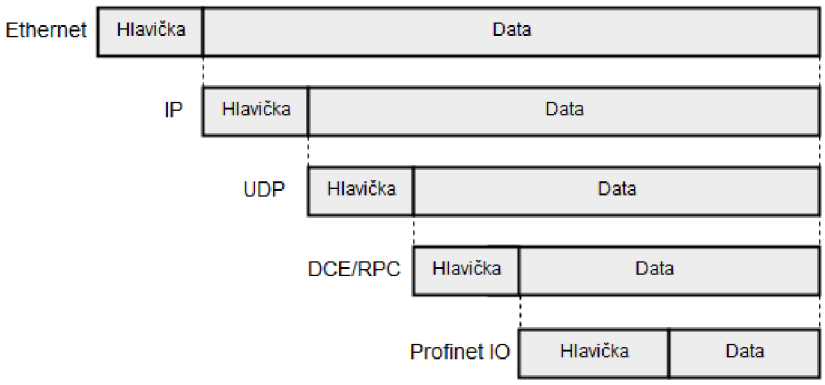
Schéma popisu zařízení pro aplikaci Virtual CPU je zobrazeno na obrázku 5.3.



Obrázek 5.3: Objekt popisující zařízení Profinetu [zdroj vlastní]

5.6 Zapouzdření dat

Komponenta popsána v sekci 5.1.1 generuje binární podobu nejvyšší síťové vrstvy. Data obsahují pouze informace týkající se konkrétního Profinet požadavku, která jsou důležitá pro cílové zařízení, na které data směřují. Aby se ovšem taková data do cílového zařízení dostala, musí být přenesena přes ethernetovou síť. Na té se může nacházet mnoho síťových prvků, které nepodporují Profinet protokol, a takovým datům by tedy nemohla porozumět. Než se data odešlou, musí se proto data "obalit"dalšími daty zajišťujícími síťový přenos. Na přenos většiny dat, které aplikace Virtual CPU používá, je využit protokol UDP 3.4. Mezi UDP vrstvou a Profinet daty se ještě nachází DCE/RPC vrstva. Výsledné zapouzdření dat je znázorněno na obrázku 5.4.



Obrázek 5.4: Zapouzdření Profinet IO paketu do vyšších vrstev [zdroj vlastní]

5.6.1 DCE/RPC

DCE/RPC hlavička obsahuje informace týkající se například fragmentace. Je-li Profinet IO část příliš dlouhá, je potřeba ji fragmentovat a poslat její fragmenty odděleně. Každý takový jeden fragment je pak zapouzdřen do DCE/RPC vrstvy, ve které jsou data potřebná k defragmentaci. Komponenta, která spravuje zapouzdření dat do DCE/RPC vrstvy, nastavuje několik polí.

- *SequenceNum* – pořadové číslo zvyšující se s každým odeslaným datagramem o 1.
- *Opnum* – identifikátor označující typ právě odesílaného požadavku.
- *FragmentLen* – délka Profinet IO dat nebol-li délka fragmentu.
- *FragmentNum* – počet fragmentů Profinet IO dat. V případě, že ke fragmentaci nedošlo, je hodnota vždy 0.
- *Flags* – bitové příznaky týkající se fragmentace.
- *ObjectUUID* – identifikátor zařízení, se kterým probíhá komunikace.

Ostatní hodnoty polí v DCE/RPC vrstvě byly převzaty z referenční odchycené komunikace mezi skutečným řídicím systémem a vstupně výstupním zařízením.

5.6.2 Ethernet, IP, UDP

Další tři vrstvy, do kterých je datagram zapouzdřen, jsou UDP, IP a ethernet. Jedná se o známé standardní protokoly používané mnoha síťovými aplikacemi. Pole těchto protokolů již nemusí být vytvářeno na úrovni aplikace Virtual CPU, jelikož jsou obsaženy v knihovně Scapy, kterou aplikace využívá. Stačí tedy nastavit některé hodnoty polí. Zbylá pole budou mít výchozí hodnotu definovanou knihovnou Scapy. Nastavují se tyto hodnoty:

- *Ethernet* – zdrojová MAC adresa, cílová MAC adresa.
- *IP* – zdrojová IP adresa, cílová IP adresa.
- *UDP* – zdrojový port, cílový port.

Nyní je možné takto zapouzdřená data poslat na síťové rozhraní definované v souboru `settings.py` (5.3), na kterém je vstupně výstupní zařízení připojeno.

5.7 Čtení odpovědi

Všechny požadavky zaslané aplikací na síťové rozhraní vstupně výstupního zařízení Profinetu musí být potvrzeny odpovědí. Tím je zajištěna detekce chyby přenosu, protože UDP komunikace takové chyby nezjistí. Není-li na nějaký požadavek vrácena odpověď, vyskytla se chyba, která mohla nastat z několika důvodů. Buď se požadavek k zařízení vůbec nedostal nebo zařízení požadavek přijalo, ale selhal přenos odpovědi. Dalším důvodem může být vypnuté, popřípadě nefunkční zařízení. Zachycení odpovědi na straně aplikace ještě neznamená, že byl požadavek úspěšně přijat. Z odpovědi musí být vyčtena položka s názvem "Status", která informuje o výsledku požadavku. Chyby musí aplikace detekovat a informovat uživatele o této skutečnosti.

Čtení odpovědí v aplikaci zajišťuje knihovna Scapy. V té je předdefinována podoba odpovědi pomocí objektu stejným způsobem jako na obrázku 5.1. Binární data odpovědi jsou pomocí tohoto objektu dekódována do jednotlivých pojmenovaných polí. Z těch je vyčtena položka "Status", ve které je informace, zda byl požadavek přijat bez chyb. V případě problému je chyba vypsána a program je buď ukončen, nebo pokračuje dále v provádění a pouze informuje uživatele o dané chybě (tisk chybového kódu). O tom, zda chyba celý program ukončí, nebo pouze informuje uživatele, rozhoduje fakt, jak moc je chyba kritická. Například v případě výskytu chyby při navazování vztahu je potřeba program ukončit, jelikož nelze dále pokračovat. Pokud ovšem nastane chyba v zápisu diagnostických dat do zařízení v průběhu již navázaného vztahu, není samotná komunikace ohrožena, a proto není potřeba program ukončit. Funkce pro čtení odpovědi je volána bezprostředně po každém odeslání požadavku, na který je očekávána odpověď. Jelikož každý požadavek má odlišnou odpověď, je funkci předán navíc jeden parametr, který říká, jaký požadavek byl odeslán. Podle parametru jsou binární data odpovědi předána odpovídajícímu objektu, který odpověď dekóduje.

5.8 Předání monitorovaných dat

Aplikace Virtual CPU je vyvíjena převážně z důvodu snadnějšího testování vstupně výstupních zařízení Profinetu. Jelikož samotná aplikace testy přímo neprovádí, ale pouze monitoruje data, která můžou být testována, je potřeba, aby byla tato data poskytnuta externím aplikacím.

Je zapotřebí předat data ze dvou komunikačních kanálů - cyklická data a alarmy. Při spouštění aplikace bude tedy možnost předat aplikaci dvě funkce, každou pro jeden kanál. Jediným parametrem každé funkce budou samotná odchycená data na konkrétním kanálu. Tester tak musí pouze definovat funkci, která bude provádět nějaký konkrétní testovací scénář. Taková funkce bude poté provedena pro každá data odchycena na konkrétním kanálu.

Je nutno říct, že zmíněný způsob nemusí být konečný. Jde pouze o implementaci původního návrhu. Teprve až testování zařízení s aplikací Virtual CPU v praxi ukáže, jaký je nejlepší způsob předání dat. Ten bude s největší pravděpodobností v budoucnu ještě změněn v závislosti na požadavcích vznesených testovacím týmem, který bude s aplikací pracovat.

Kapitola 6

Závěr

V rámci této bakalářské práce byla navržena a implementována aplikace nazvaná Virtual CPU simulující chování řídicího systému standardu Profinet. Pomocí ní je možno nahradit alespoň částečně zařízení, které řídí všechna zařízení s technologií Profinet k němu připojená. Aplikace dokáže navázat aplikační vztah s připojeným zařízením a provádí operace potřebné pro udržení vztahu. Díky simulování činnosti řídicího systému je usnadněno testování zařízení, jelikož pro velkou část testovacích scénářů je potřeba, aby bylo zařízení připojeno na řídicí systém. Data monitorovaná aplikací lze potom využít v konkrétních testech, které kontrolují jejich validitu.

Aplikace Virtual CPU bude v nejbližší době nasazena v laboratořích společnosti Siemens s.r.o. v Brně, kde jsou zařízení Profinetu testována. Data, která poskytuje, budou vyhodnocována testovacím systémem PyTeMat, který je nyní používán pro testování vyvíjených zařízení. Díky nasazení aplikace je možno rozšířit dosavadní testovaní o další scénáře. Také se počítá s dalším vývojem aplikace rozšiřujícím její funkcionalitu.

6.1 Možnosti rozšíření

Aplikace byla vytvořena aby, splňovala konkrétní požadovanou funkcionalitu a mohla být nápomocná pro testování produktů v prostorách společnosti Siemens. Reálný Profinet řídicí systém simuluje pouze ve velmi omezeném rozsahu a nikdy jej přímo nenahradí. Jsou zde ovšem široké možnosti rozšíření, díky kterým se může funkcionalitou reálnému řídícímu systému alespoň přibližovat. Zde jsou uvedeny některé z nich:

- *Podpora všech zařízení Profinetu* – aplikace je psaná obecně, aby fungovala na každém zařízení s technologií Profinet. Je ale potřeba znát jejich technickou charakteristiku uloženou v GSD souboru. Zařízení, pro které byla aplikace vytvářena, mají tuto charakteristiku již uloženou přímo v aplikaci v objektu zvaném `dev_config`, která byla ručně přepsána z GSD souboru. Pro podporu jakéhokoliv jiného zařízení by musel být takovýto objekt dodefinován. To je velice jednoduchý proces, pokud by byla potřeba použít aplikaci pro nějaké konkrétní zařízení, které není definováno. Objekty však není možné vytvořit pro každé existující zařízení. Aby aplikace splňovala podporu všech zařízení, musela by být vytvořena komponenta, která přečte GSD soubor (obsahující definici zařízení) a vygeneruje z něj objekt `dev_config` automaticky.
- *Zápis dat v rámci navázání vztahu* – v GSD souboru se nachází blok, který říká, jaká diagnostická data mají být automaticky zapsána do zařízení v rámci navazování

aplikačního vztahu. V momentální podobě lze taková data zapsat pouze ručně až po jeho navázání. Jakmile bude aplikace číst GSD soubor, může být doplněna i o tuto funkcionalitu.

- *Rychlejší výměna cyklických dat* – výměna je v aplikaci nastavena na nejvyšší možný časový interval 512 milisekund. Vyšší rychlosti nebyly testovány a aplikace je proto ani přímo nepodporuje (přestože lze interval snížit) z důvodu nízkého výkonu počítače, na kterém byla aplikace vyvíjena. V případě potřeby je nutné nižší intervaly otestovat, popřípadě provést optimalizaci aplikace.

- *Grafické uživatelské rozhraní* – grafické rozhraní by nabídlo přehlednější diagnostiku a snadnější ovládání pro méně zkušené uživatele.

Dosažený výsledek práce (aplikace simulující řídicí systém Profinet) ukázal, že je přínosné pracovat na podobných aplikací, které simulují funkce reálného systému. Není potřeba aby simulující nástroj zahrnoval veškerou funkcionalitu reálného systému, ale je potřeba, aby obsahoval takové funkce, které jsou vyžadovány pro konkrétní případ. V tomto případě funkce, potřebné pro testování vstupně výstupního zařízení. Toho se podařilo dosáhnout a díky aplikace Virtual CPU je usnadněno a rozšířeno testování zmíněných zařízení.

Práce volně navazuje na práci kolegy Jana Slováčka, která pojednává o automatizovaném testování pomocí testovacího systému PyTeMat [9].

Literatura

[1] *DC UPS and uninterruptable power supply for the industry*. 2017, [Online; navštíveno 26. 4. 2017].
URL http://w3.siemens.com/mcms/power-supply-sitop/en/dc-ups/Pages/default.aspx

[2] *Product Details - Industry Mall - Siemens WW*. 2017, [Online; navštíveno 24. 4. 2017].
URL https://mall.industry.siemens.com/mall/en/WW/Catalog/Product/6EP3437-8MB00-2CY0

[3] BIONDI, Philippe and the Scapy community: *Welcome to Scapy's documentation!,*. Duben 2010, [Online; navštíveno 29. 12. 2016].
URL http://secdev.org/projects/scapy/doc/

[4] MOORTHY, Vijay: *Gigabit Ethernet*. 1997, [Online; navštíveno 4. 3. 2017].
URL http://www.cse.wustl.edu/~jain/cis788-97/ftp/gigabit_ethernet/

[5] MULLINS, Michael: *Exploring the anatomy of a data packet,*. Červenec 2001, [Online; navštíveno 4. 3. 2017].
URL http://www.techrepublic.com/article/exploring-the-anatomy-of-a-data-packet/

[6] PIGAN, R.; METTER, M.: *Automating with PROFINET*. Publicis Corporate Publishing, 2006, ISBN ISBN: 3-89578-256-4.

[7] POPP, M.: *Industrial Communication with PROFINET*. PROFIBUS&PROFINET International, interní materiály, 2014.

[8] ROUSE, Margaret: *What is Address Resolution Protocol (ARP)?,*. Září 2005, [Online; navštíveno 16. 4. 2017].
URL http://searchnetworking.techtarget.com/definition/Address-Resolution-Protocol-ARP

[9] SLOVÁČEK, J.: *Nástroj pro automatické testování produktů SITOP PSU8600 a SITOP UPS1600,. 2016, bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. 2016-06-15. Vedoucí práce Kreslíková Jitka.*
URL http://www.fit.vutbr.cz/study/DP/BP.php?id=15942

Přílohy

Seznam příloh

Příloha A

Obsah CD

Přiložené CD obsahuje:

- Adresář se zdrojovými kódy této zprávy.
- Elektronická verze této zprávy.
- Zdrojové kódy aplikace.
- Návod na instalaci a používání aplikace.