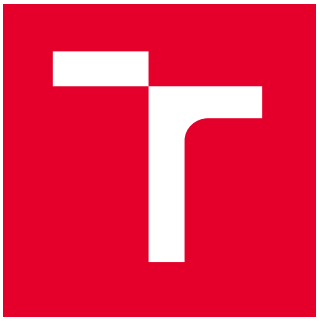


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

IMPLEMENTACE PROTOKOLU ETHERNETIP V PROGRAMOVACÍM JAZYCE JAVA

JAVA IMPLEMENTATION OF ETNERNETIP PROTOCOL

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **Tomáš Čišecký**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Kryštof Zeman, Ph.D.**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Telekomunikační a informační systémy**

Ústav telekomunikací

Student: Tomáš Čišecký ***ID:*** 230791

Ročník: 3 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Implementace protokolu EthernetIP v programovacím jazyce Java

POKYNY PRO VYPRACOVÁNÍ:

Cílem bakalářské práce bude seznámení se s protokolem EthernetIP. V teoretické části práce bude proveden stručný popis historie protokolu a detailní popis jeho struktury. V praktické části bude poté samotný protokol implementován v programovacím jazyce Java. Nejdříve jako samostatně běžící program a následně bude začleněn do systému OpenMUC. Výsledkem práce tedy budou dvě samostatně běžící aplikace.

DOPORUČENÁ LITERATURA:

[1] EtherNet/IP TM – CIP on Ethernet Technology [online]. 2023 [cit. 2023-09-11]. Dostupné z https://www.odva.org/wp-content/uploads/2021/05/PUB00138R7_Tech-Series-EtherNetIP.pdf

[2] openMUC [online]. 2023 [cit. 2023-09-11]. Dostupné z: https://www.openmuc.org/

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 3.6.2025

Vedoucí práce: Ing. Kryštof Zeman, Ph.D.

prof. Ing. Jiří Mišurec, CSc.
předseda rady studijního programu

ABSTRAKT

Bakalárska práca je zameraná na detailný teoretický rozbor priemyselného protokolu Ethernet/IP, jeho štruktúry, návrh a realizáciu aplikácie komunikujúcej prostredníctvom tohto protokolu a jej integráciu do open-source frameworku OpenMUC. V práci sú vysvetlené spôsoby komunikácie, formát správ a funkcionalita protokolu. Vytvorená užívateľská aplikácia je implementovaná v programovacom jazyku Java.

KĽÚČOVÉ SLOVÁ

CIP, Ethernet, Ethernet/IP enkapsulácia, hlavička enkapsulácie, explicitná komunikácia, implicitná komunikácia, Java, OpenMUC, TCP/IP, UDP

ABSTRACT

The bachelor's thesis is focused on detailed theoretical analysis of the industrial Ethernet/IP protocol, it's structure, design and initial implementation of an application communicating via this protocol and its integration into the open-source framework OpenMUC. The thesis describes the communication methods, message format and protocol functionality. The created user application is implemented in Java programming language.

KEYWORDS

CIP, Ethernet, Ethernet/IP, encapsulation, encapsulation header, explicit messaging, implicit messaging, Java, OpenMUC, TCP/IP, UDP

ČIŠECKÝ, Tomáš. *Implementace protokolu EthernetIP v programovacím jazyce Java*. Bakalárska práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2025. Vedúci práce: Ing. Kryštof Zeman, Ph.D.

Vyhlásenie autora o pôvodnosti diela

Meno a priezvisko autora:	Tomáš Čišecký
VUT ID autora:	230791
Typ práce:	Bakalárska práca
Akademický rok:	2024/25
Téma záverečnej práce:	Implementace protokolu EthernetIP v programovacím jazyce Java

Vyhlasujem, že svoju záverečnú prácu som vypracoval samostatne pod vedením vedúcej/cého záverečnej práce, s využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej záverečnej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto záverečnej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákonníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podpisuje iba v tlačenej verzii.

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce panovi Ing. Kryštof Zeman, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Práca svojím obsahom a tématikou spadá do oblasti sieťových technológií a priemyselnej automatizácie. Cieľom tejto práce je detailný teoretický rozbor štruktúry protokolu Ethernet/IP a popis jeho histórie. Praktickým výstupom je realizácia komunikácie prostredníctvom protokolu Ethernet/IP. V prvom prípade ako samostatne bežiaca GUI aplikácia, v druhom ako súčasť frameworku OpenMUC.

Práca je rozdelená do piatich teoretických kapitol a jednej kapitoly, ktorá popisuje praktickú časť.

Prvá kapitola sa zameriava na Ethernet, ktorý je využívaný v Ethernet/IP ako fyzická a linková vrstva. Zahŕňa históriu vývoja štandardu, typy používaných prenosových médií, komunikačné metódy a kategórie Ethernetu. Kapitola tiež popisuje štruktúru a vysvetľuje rozdiel medzi rámcami IEEE 802.3 a Ethernet V2.

V druhej kapitole je vysvetlená sada TCP/IP, ktorú Ethernet/IP využíva v rámci sieťovej a transportnej vrstvy. Obsahuje popis štruktúry TCP/IP modelu a jeho porovnanie s modelom OSI, rozbor protokolov IP, TCP, UDP a typy adresovania.

Tretia kapitola sa zaoberá popisom histórie a vzniku Ethernet/IP, jeho vlastnosťami a umiestnením v rámci OSI modelu. Popisuje štruktúru rámca Ethernet/IP a zoznam príkazov protokolu.

Obsahom štvrtej kapitoly je popis komunikácie v Ethernet/IP, vysvetlenie explicitnej a implicitnej komunikácie, popis ich formátu správy a rozdielov medzi nimi. V závere kapitoly sú charakterizované typy EtherNet/IP zariadení.

Piata kapitola sa zaoberá protokolom CIP. Popisuje jeho základné vlastnosti a využitie. Vysvetľuje princip objektovo-orientovanej reprezentácie v CIP. Charakterizuje základné objekty potrebné pre Ethernet/IP komunikáciu a základné všeobecné komunikačné služby, ktoré CIP poskytuje.

Šiesta kapitola predstavuje prakticku časť práce. Jej prvá časť sa venuje vytvorenej GUI aplikácii – zahŕňa podrobný návrh aplikácie, popis jednotlivých komponentov, a vývojového diagramu aplikácie. Súčasťou je dokumentácia komunikácie, vrátane výstupov aplikácie a analýz z programu Wireshark. Druhá časť kapitoly sa zameriava na popis systému OpenMUC a integráciu aplikácie v rámci tohto frameworku. Vysvetľuje základnú štruktúru, spôsob spracovania požiadaviek a reakcii v rámci OpenMUC backendu.

1 Ethernet

Ethernet je názov pre súrhn technológií poskytujúcich špecifikácie fyzickej a linkovej vrstvy na kontrolu prístupu k zdieľanému sieťovému médiu. Stal sa dominantou technológiou pre kabelové lokálne siete (LAN). Okrem prepojenia počítačov sa používa aj na pripojenie dátových úložísk, zariadení spotrebnej elektroniky (televízne prijímače, herné konzoly) a tiež ako drôtové rozhranie pre prístupové body WiFi a zariadenia pre prístup k internetu. Ethernet poskytuje mnoho výhod, oproti iným LAN technológiam – jednoduchú inštaláciu a riadenie, flexibilitu a škálovateľnosť, interoperabilitu medzi rôznymi predajcami a nižšie finančné náklady na inštaláciu [4].

1.1 Vývoj

Pôvodná verzia bola vyvinutá v 70. rokoch 20. storočia. Táto technológia bola štandardizovaná ako Ethernet version 1 konzorciom spoločností DEC, Intel a Xerox (DIX), pričom neskôr bola zdokonalená na Ethernet 2. v 80. rokoch Institute of Electrical and Electronics Engineers (IEEE) definovala formálny štandard pre Ethernet, známy ako IEEE 802.3 CSMA/CD. Obe verzie Ethernetu (Ethernet II a IEEE 802.3 CSMA/CD) sú prakticky totožné, nie však kompletne identické. Hlavným rozdielom je formát linkového rámca na úrovni MAC. Vetva Ethernetu II sa od roku 1982 už nevyvíja a všetky nové verzie „Ethernetu“ patria do vývojovej vetvy IEEE 802.3 [23].

1.2 Prenosové média

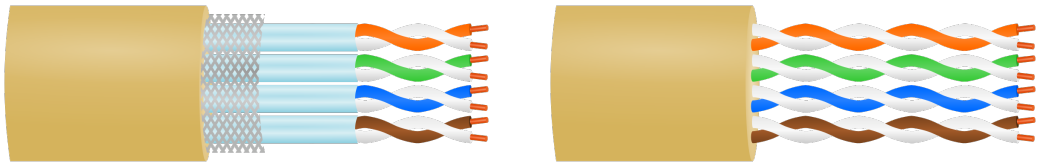
Ethernet môže byť implementovaný pomocou troch typov kabeláže [4]:

- **Koaxiálny kábel**, skrátene koax, pozostáva z jedného medeného vodiča obklopeného izoláciou, kovovým štítom a plastovým plášťom. Štít chráni pred elektromagnetickým rušením (EMI), ktoré by mohlo spôsobiť útlm – zníženie sily a kvality signálu. EMI generujú rôzne zdroje ako výbojky, mikrovlnné rúry, mobilné telefóny či rádioprijímače. Koax sa bežne využíva na nasadenie káblovej televízie.
- **Krútená dvojlinka** pozostáva z dvoch alebo štyroch párov medených vodičov obklopených plastovou izoláciou. Vodiče v rámci paru sú omotané okolo seba navzájom kvôli redukcii presluchu – formy EMI, ktorá vzniká, keď signál z jedného vodiča preniká alebo interferuje so signálom na druhom vodiči. Krútená dvojlinka môže byť buď **netienená** alebo **tienená**, pričom existuje niekoľko prevedení tienenej, napr. FTP, S/FTP, F/FTP a ďalšie, ktoré sa líšia

spôsobom tienenia. Tienená je odolnejšia voči EMI, avšak všetky typy krútenej dvojlinky tŕpia vyššou stratou signálu ako koaxiálny kábel [4], [27].
Existuje viacero kategórií krútenej dvojlinky, odlišujúcich sa počtom zákrut na palec jednotlivých párov:

Tab. 1.1: Prehľad jednotlivých kategórií krútenej dvojlinky [2]

Typ	**Počet párov**	**vysielacia frekvencia**	**vysielacia rýchlosť**	**Max. trasa**	**Uplatnenie**
Cat1	1	400 kHz	1 Mbps	100 M	Telefónne linky
Cat2	2	4 MHz	4 Mbps	100 M	Token Ring
Cat3	3	16 MHz	10 Mbps	100 M	10Base-T Ethernet
Cat4	4	20 MHz	16 Mbps	100M	Token Ring, 10Base-T Ethernet
Cat5	4	100 MHz	100 Mbps	100 M	100Base-T Ethernet
Cat5e	4	100 MHz	1000 Mbps	100 M	1000Base-T Ethernet
Cat6	4	250 MHz	1000 Mbps	100 M	1GBase-T Ethernet
Cat6a	4	500 MHz	10 Gbps	100 M	10GBase-T Ethernet
Cat7	4	600 MHz	10 Gbps	100 M	10GBase-T Ethernet
Cat7a	4	1000 MHz	10 Gbps	100 M	10GBase-T Ethernet
Cat8	4	2000 MHz	40 Gbps	100 M	10GBase-T Ethernet

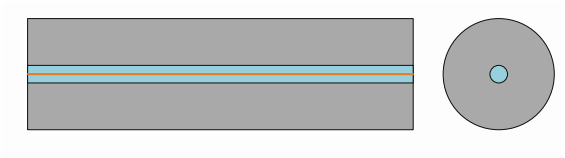


(a) Typ S/FTP (b) Typ UTP

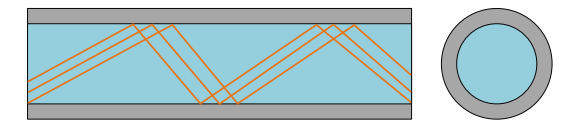
Obr. 1.1: Rozdiel medzi S/FTP a UTP [5], [6]

- **Optické vlákno** používa na prenos signálu svetlo. Ethernet podporuje dva typy vlákien:
  - Jednovidové vlákno – pozostáva z veľmi malého jadra umožňujúceho prechod len jedného lúča alebo módu svetla. Týmto sa výrazne znižuje útlm a rozptyl svetelného signálu, čo podporuje vysokú šírku pásma na dlhé vzdialenosti (v desiatkach kilometrov) [4].

  - Viacvidové vlákno – pozostáva z väčšieho jádra, ktoré umožňuje prechod viacerým módom svetla. Vzniká tak väčší rozptyl svetla, preto podporuje kratšie vzdialenosti [4].



(a) Jednovidové vlákno

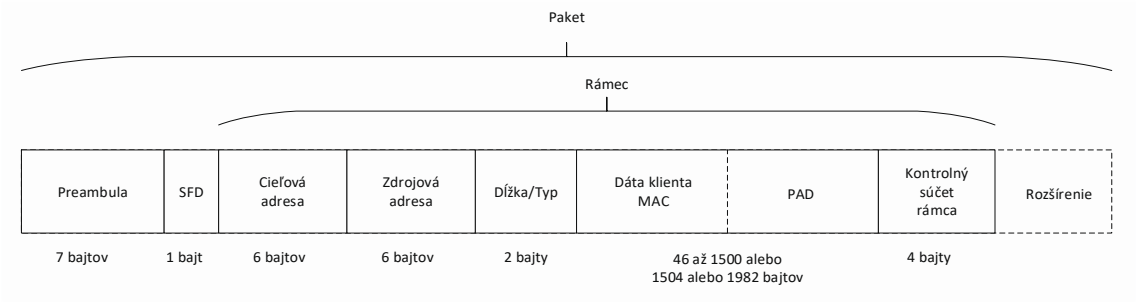


(b) Viacvidové vlákno

Obr. 1.2: Porovnanie optických vlákien [14]

1.3 Štruktúra ethernetového rámca

IEEE 802.3 definuje rámec o minimálnej veľkosti 64 bajtov. Menšie rámce sú považované za neúplné a sú zahodené. Rámec IEEE 802.3 pozostáva z následujúcich častí [12]:



Obr. 1.3: Štruktúra ethenetového rámca podľa IEEE 802.3-2022 štandardu [12]

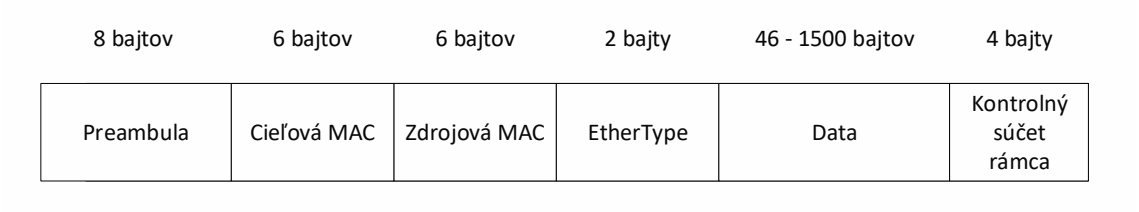
- **Preambula** – séria 56 bitov alternujúcich 1 a 0, ktorá sychronizuje komunikáciu v Ethernetovej sieti.
- **Začiatok rámca (SFD)** – sekvencia 10101011. Následuje hneď za preambulou. Po tejto skvencii hneď následuje MAC rámec.
- **Cieľová adresa MAC** – identifikuje, komu má byť rámec odoslaný. Môže to byť jeden užívateľ(unicast), skupina(multicast), alebo všetci(broadcast).
- **Zdrojová adresa MAC** – MAC adresa zdrojového sieťového rozhrania. Prvých 24 bitov adresy určuje výrobcu sieťového rozhrania, zvyšných 24 bitov je unikátnym identifikátorom hostiteľa.
- **Dĺžka/typ** – 16 bitové pole, ktoré má jeden z dvoch významov, v závislosti od numerickej hodnoty:
  - **hodnota $\leq$ 1500dec** – pole udáva počet bajtov MAC dát klienta,
  - **hodnota $\geq$ 1536dec** – pole udáva EtherType MAC klienta protokolu.

  Interpretácie ako dĺžka alebo typ sa navzájom vylučujú.
- **Dáta klienta MAC** – pole dlhé minimálne 46 bajtov, maximálna dĺžka sa môže líšiť v závislosti od použitia:
  - 1500 bajtov -pre základné rámce,
  - 1504 bajtov -pre rámce so značkou vLAN (Q-tagged rámce),
  - 1982 bajtov – rozšírenie pre tzv. „envelope frames“, ktoré umožňujú pridanie dodatočných prípon potrebných pre protokoly vyšších vrstiev.
- **Kontrolný súčet rámca** – Obsahuje 4 bajtovú hodnotu CRC(Cycle Redundancy Check). CRC je algoritmus na kontrolu integrity dát.
- **Rozšírenie** – obsahuje sekvenciu rozšírujúcich bitov, ktoré sa líšia od dátových.

  Obsah nie je zahrnutý do FCS výpočtov [4], [12].

V praxi sa však dnes využíva prevážne **Ethernet v2 rámec**, ktorý je najpoužívanejším rámcom v LAN sieťach. V súčastnosti väčšina TCP/IP aplikácií prenáša pakety prostredníctvom rámcov Ethernet 2. Ethernet v2 rámec sa odlišuje typom a usporiadaním niektorých polí. **EtherType** – pole popisujúce ktoré protokoly sú

priamo zapúzdrené na linkovej vrstve (napríklad IPv4 alebo IPv6). Význam ostatných polí je podobný ako u rámca IEEE 802.3 [10].



Obr. 1.4: Štruktúra rámca Ethernet v2 [10]

1.4 Komunikačné metódy

Ethernet bol pôvodne vyvinutý na podporu prostredia so zdieľaným médiom. To umožnilo dvom alebo viacerým užívateľom používať rovnaké fyzické sieťové médium. Sú dva spôsoby komunikácie na zdieľanom fyzickom médiu: **Half-Duplex** a **Full-Duplex** komunikácia [4].

1.4.1 Half-Duplex komunikácia

Zariadenie môže vysielať alebo prijímať, ale nie obe súčasne. Ethernet využíva Carrier Sense Multiple access with Collision Detect (CSMA/CD) na kontrolu prístupu k médiu. Funguje to následovne [4]:

Zariadenie, ktoré chce vysielať monitoruje fyzickú linku kvôli detekcii prenosu nosného signálu. Ak je linka voľná a uplynula medzirámcová doba, začne vysielať. Ak dôjde k vyslielaniu dvoch staníc súčasne, vzniká kolízia. Vtedy obé zariadenia pošlú 32bitovú sekvenciu rušenia aby boli ostatné zariadenia informované o kolízii. Chybné rámce sú zahodené a obe zariadenia čakajú náhodnú stanovenú dobu, než znovu prepošlú svoje rámce, kvôli zníženiu pravdepodobnosti vzniku ďalšej kolízie. To je zabezpečené pomocou spätného časovača. Kolízia musí byť zachytená ešte pred dokončením odoslania rámca. K tomu slúži tzv. časový slot. Kolízia, ktorá vznikne po uplynutí tohto času sa nazýva neskorá kolízia. Použitie half-duplexu môže byť výhodné z hľadiska ceny a nenáročnej inštalácie. Používa sa najmä v drsnejších prostrediach v ktorých je zvýšená pravdepodobnosť poškodenia. Medzi jeho nevýhody patrí nízka priepustnosť, nekompatibilita a vysoká odozva [4], [15].

1.4.2 Full-Duplex komunikácia

Full-duplex Ethernet podporuje komunikáciu v rovnakom čase poskytovaním oddelených ciest pre vysielanie a prijímanie. Týmto je efektívne zdvojnásobená prie-

pustnosť sieťového rozhrania. Full-duplex Ethernet bol formalizovaný v rámci IEEE 802.3x [15].

Výhody:

- zariadenia môžu vysielať a prijímať zároveň,
- nepoužíva CSMA/CD, keďže sa nepredpokladá vznik kolízii na funkčnej full-duplexnej linke,
- umožňuje rýchlejší prenos dát.

Nevýhody:

- vyššia spotreba energie,
- cena – potrebné sofistikovanejšie zariadenia, lepšia kabeláž a ďalší doplnkový hardvér,
- náchylnosť na chyby – v prípade nespravného spravovania, pri simulátnom prenose môže dochádzať k strate dát či opakovaným prenosom.

Je ideálnym riešením pre komplexné siete, v ktorých sú vysoká rýchlosť a efektivita žiadúce. Full-duplex je v dnešnej dobre prakticky štandardom v moderných ethernetových LAN sieťach [4], [15].

1.5 Kategórie Ethernetu

Pôvodný 802.3 štandard sa vyvíjal v čase, tým pádom podporoval vyššie prenosové rýchlosti, väčšie vzdialenosti a novšie hardvérové technológie. Hlavné kategórie boli organizované na základe ich rýchlosti [4], [12], [24]:

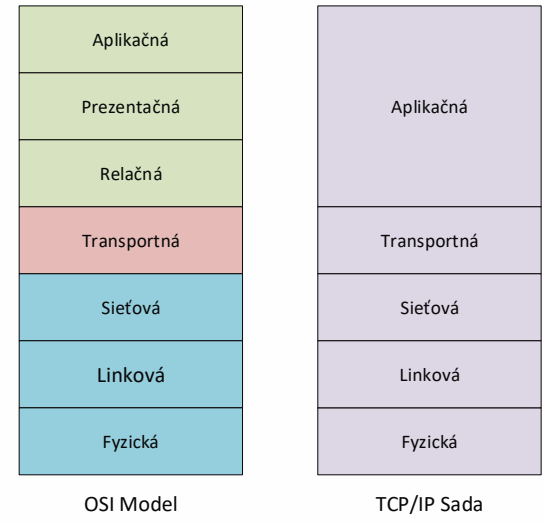
- **Ethernet** (10 Mbps) – pôvodná varianta, podporoval koax, krútenú dvojlinku a optické vlákno
- **Fast Ethernet** (100 Mbps) rýchlejšia, vylepšená verzia pôvodného Ethernetu, definovaná štandardom IEEE 802.3u. Podporuje krútenú dvojlinku a optické vlákno.
- **Gigabit Ethernet** – pôvodne bol definovaný len pre optické vlákna (IEEE 802.3z), neskôr doplnená varianta pre krútenú dvojlinku (802.3ab).
- **10 Gigabit Ethernet** – posledná širšie dostupná verzia. Definovaná bola v roku 2003 ako IEEE 802.3ae. Ako prenosové médium používa hlavne optické vlákno a full-duplex komunikáciu.
- **Vyššie rýchlosti** (40, 100, 200, 400, 800 Gbit/s, 1,6 Tbit/s) – zatiaľ nie sú príliš rozšírené, alebo sú vo vývoji.

2 Sada TCP/IP

Transmission Control Protocol/Internet Protocol (TCP/IP) je sada protokolov, ktoré špecifikujú komunikačné štandardy medzi počítačmi a detailné konvencie smerovania a prepájania sietí. Sada zabezpečuje end-to-end komunikáciu, špecifikuje, akým spôsobom budú dáta paketizované, adresované, vysielané, presmerované a prijaté. Najvýznamnejšími protokolmi tejto sady sú **TCP, UDP** a **IP** [11], [19].

2.1 Štruktúra

Pôvodná verzia bola definovaná ako štvorvrstvá softvérová reprezentácia nad hadrvérom. Tento model sieťovej komunikácie predchádza OSI modelu, preto sa vrstvy oboch modelov úplne nezhodujú. V dnešnej dobe sa však už na TCP/IP pozerá ako päť-vrstvý model s vrsvami podobnými tým z OSI modelu. TCP/IP model pozostáva z následujúcich vrstiev [8]:



Obr. 2.1: Porovnanie OSI a TCP/IP [8]

- **Fyzická** – TCP/IP nedefinuje žiadny konkrétny protokol na fyzickej vrstve, podporuje však všetky štandardné a proprietárne protokoly. Na tejto úrovni prebieha komunikácia medzi dvomi uzlami – buď medzi počítačom a smerovačom, alebo medzi dvoma počítačmi. Jednotkou komunikácie je jeden bit. Po nadviazaní spojenia medzi dvoma uzlami prúdi medzi nimi prúd bitov.

- **Linková** – ani na tejto vrstve TCP/IP nedefinuje špecifické protokoly, ale podporuje všetky štandardné a proprietárne. Na tejto úrovni je komunikácia

stále medzi dvomi uzlami. Jednotkou komunikácie je paket nazývaný rámec. Rámec zapúzdruje dáta, ktoré obdržal od sieťovej vrstvy s pridanou hlavičkou a niekedy koncovým doplnkom. Hlavička obsahuje okrem iných komunikačných informácií aj zdrojovú a cieľovú adresu rámca.
Cieľová adresa je dôležitá na určenie správneho príjemcu rámca, nakoľko k zdroju môže byť pripojených viacero uzlov.
Zdrojová adresa je potrebná na prípadnú odozvu alebo potvrdenie, ktoré môžu vyžadovať niektoré protokoly.

- **Sieťová** – na sieťovej vrstve TCP/IP podporuje Inetrnet Protocol (IP). IP je prenosový mechanizmus používaný protokolmi sady TCP/IP. IP prenáša dáta v paketoch zvaných datagramy. Každý datagram sa prepravuje samostatne. Datagramy môžu cestovať naprieč rôznymi trasami, môžu byť doručené v nesprávnom poradí alebo byť duplikované.

- **Transportná** – jej úlohou je doručenie celej správy, ktorá môže byť vo forme segmentu, užívateľského datagramu alebo paketu v závislosti od použitého protokolu. Tradične bola transportná vrstva reprezentovaná v TCP/IP prostredníctvom dvoch protokolov **User Datagram Protocol (UDP)** a **Transmission Control Protocol (TCP)**. V posledných rokoch bol predstavený nový protokol s názvom Stream Control Transmission Protocol (SCTP).

- **Aplikačná** – v TCP/IP modeli predstavuje kombináciu reláčnej, prezentačnej a aplikačnej vrstvy z modelu OSI. Umožňuje užívateľovi prístup k privátnemu internetu alebo globálnemu Internetu. Definuje množstvo protokolov poskytujúcich rôzne služby ako elektronická pošta, prenos súborov, prístup k World Wide Web (www) a podobne.

2.2 Internet Protocol

IP je primárny protokol sieťovej vrstvy a základ celej TCP/IP sady. Jeho úlohou je doručovanie paketov od zdrojového užívateľa k cieľovému, identifikovateľným na základe IP adries v hlavičkách paketov. Okrem smerovania medzi sieťami poskytuje hlásenia o chybách, ako aj fragmentáciu a opätovné zostavenie informačných jednotiek zvaných datagramy na prenos cez siete s rôznymi maximálnymi dĺžkami sieťových jednotiek. Prvá významná verzia Internet Protocol version 4 (IPv4) je dominantným protokolom v Internete. Jeho následovníkom je novšia verzia IPv6, ktorej nasadenie sa zvyšuje približne od roku 2006 [8], [19].

IP adresovanie zahŕňa priraďovanie IP adries a súvisiacich parametrov užívateľským rozhraniam. Adresný priestor je rodelený na podsiete zahŕňajúce sieťové prefixy. IP smerovanie je vykonávané všetkými užívateľmi, ako aj smerovačmi, ktorých primárnou úlohou je prenášať pakety cez hranice sietí. Smerovače medzi sebou komunikujú prostredníctvom špeciálne navrhnutých smerovacích protokolov, buď vonkajších alebo vnútorných, čo závisí od topológie siete.
Existujú štyri typy adresovacích metód v IP [8], [19]:

- **Unicast** – doručenie správy jednému konkrétnemu uzlu pomocou one-to-one spojenia.
- **Broadcast** – doručenie správy všetkým uzlom v sieti pomocou one-to-all spojenia.
- **Multicast** – doručenie správy skupine uzlov, ktorá má o ňu záujem prostredníctvom spojenia one-to-many-of-many alebo many-to-many-of-many.
- **Anycast** – doručenie správy ktorémukoľvek užívateľovi, typicky tomu, ktorý je najbližšie k zdroju prostredníctvom spojenia one-to-one-of-many.

2.3 Transmission Control Protocol

Vznikol v pôvodnej iplementácií sieťovej architektúry, kde dopĺňal IP. Odtiaľ pochádza názov celej sady ako TCP/IP. TCP je spojovo-orientovaný prenosový protokol, definovaný na transportnej vrstve. Zabezpečuje spoľahlivý, odolný voči chybám a zoradený prenos prúdu bajtov (oktetov) medzi aplikáciami bežiacimi na zariadeniach komunikujúcich prostredníctvom IP siete. Keďže ide o spojovo-orientovaný protokol, musí odosielateľ pred vysielaním nadviazať spojenie na základe dohodnutých pravidiel. Toho je možné dosiahnuť pomocou **three-way-handshake** procedúry. TCP je optimálny na prenosy vyžadujúce presné a konzistentné doručenie, avšak nie je vhodný na časovo-kritické prenosy, nakoľko pri ňom môže dochádzať k veľkým časovým oneskoreniam spôsobeným preposielaním neúspešných správ. TCP používa mnoho internetových aplikácií ako World Wide Web (www), email, File Transfer Protocol a iné [8], [19].

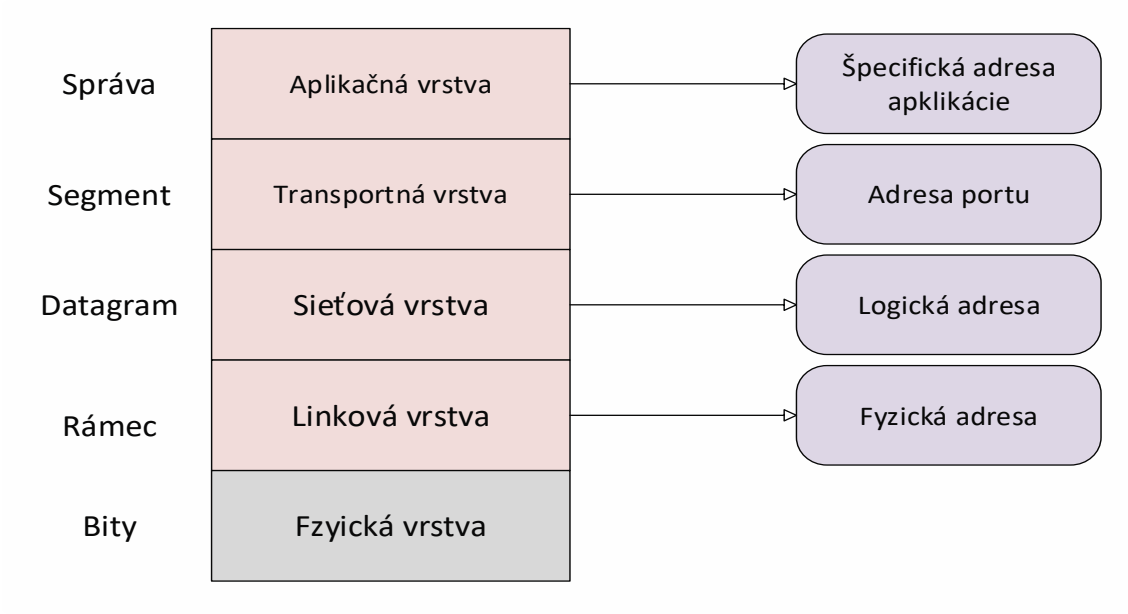
2.4 User Datagram Protocol

UDP je na správy orientovaný protokol definovaný na transportnej vrstve. Pri posielaní správ nevytvára (a neoveruje) spojenie s druhou stranou, ani si neoveruje, čo odoslal. Poskytuje kontrolné súčty (checksum) pre integritu dát a čísla portov na

adresovanie rôznych funkcií zdroja a cieľa datagramu. UDP je vhodnejší skôr pre časovo-kritické aplikácie [8], [13], [19].

2.5 Adresovanie

V TCP/IP sa používajú 4 úrovne adresovania: fyzická adresa, logická adresa, adresa portu a špecifická adresa aplikácie. Každá adresa je priradená jednej vrstve TCP/IP architektúry, ako je vidieť na obrázku [8]:



Obr. 2.2: Adresovanie v TCP/IP [8]

- **Fyzická adresa** – adresa uzla definovaná jeho LAN alebo WAN sieťou, veľkosť a formát závisí od siete. Príkladom je MAC adresa.
- **Logická adresa** – 32bitová adresa, jednoznačne definuje zariadenie pripojené k internetu. Do tejto kategórie patria IPv4 a IPv6 adresy.
- **Adresa portu** – 16 bitov, slúži na identifikáciu konkrétneho procesu v rámci komunikácie, reprezentovaná ako jediné číslo (napríklad 80).
- **Špecifická adresa aplikácie** – navrhnuté špeciálne pre danú aplikáciu (napríklad e-mailová adresa, URL). Počas prenosu sú zmenené na zodpovedajúce portové a logické adresy. Dĺžka nie je pevne definovaná.

3 Základy EtherNet/IP

EtherNet/IP bol predstavený v roku 2001 a dnes je najrozvinutejším, overeným a kompletne priemyselným riešením siete Ethernet, dostupným pre priemyselnú automatizáciu. Jeho popularita rastie naďalej, nakoľko užívateľov lákajú výhody otvorených technológií a internetu. EtherNet/IP patrí do skupiny sietí, ktoré implementujú CIP v rámci svojich vyšších vrstiev [21].

3.1 Historický kontext

Záver 20. storočia bol bohatý z hľadiska industrializácie a zrýchleného napredovania vývoja moderných technológií. Expanzia automobilového, farmaceutického či potravinárskeho priemyslu spôsobovala zvýšený dopyt po automatizácii a riadení výrobných procesov. V tom čase bolo v priemysle obvyklé, že každý výrobca mal svoj vlastný protokol na komunikáciu medzi zariadeniami, v dôsledku čoho vzniklo viacero štandardov, ako sú **Modbus [16], Profibus [17], ControlNet [22] či DeviceNet [25]**.
Každý z týchto protokolov mal svoje vlastné špecifiká, čo spôsobovalo nekompatibilitu medzi sebou navzájom a vyžadovalo špecifický hardvér. V dôsledku toho začali vznikať požiadavky na skvalitnenie komunikačných služieb medzi zariadeniami [9].

3.2 Cieľové vlastnosti

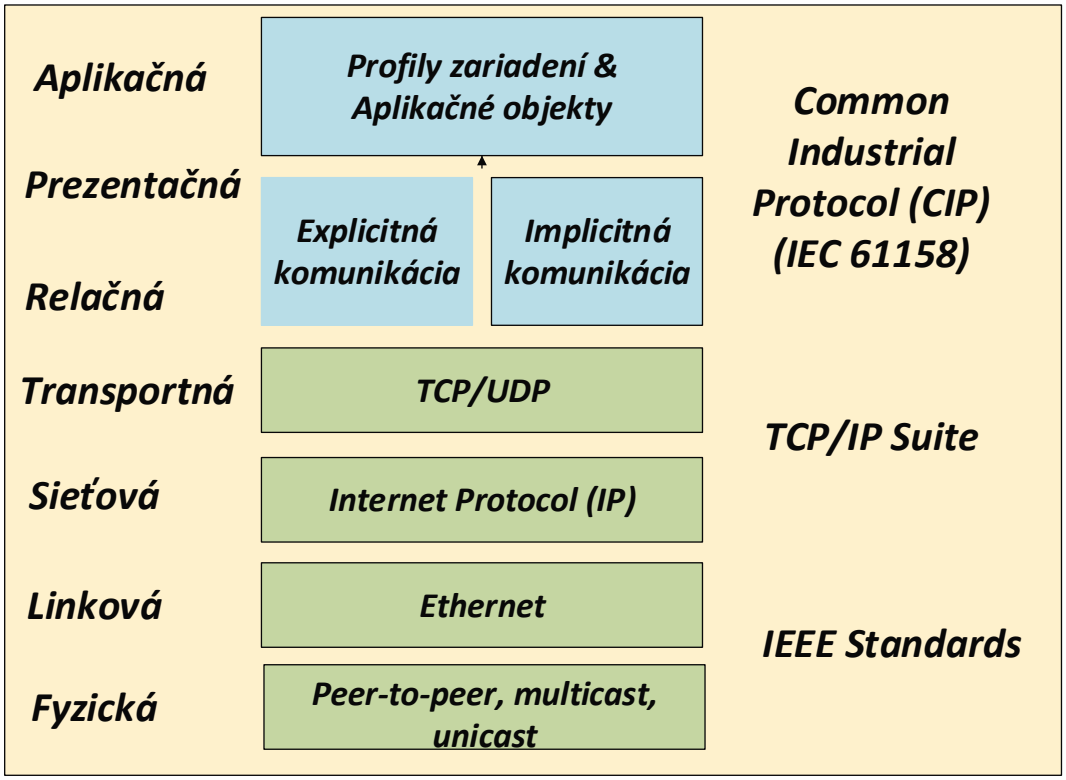
Cieľom bolo vytvoriť protokol, ktorý by spĺňal následovné kritéria [9]:

- **zjednotená infraštruktúra** -komunikácia prostredníctvom jednej sieťovej infraštruktúry. Najlepším adeptom bol v tom čase Ethernet, ktorý už bol ako štandard využívaný v kancelárskych a podnikových sieťach.
- **interoperabilita** -vlastnosť, ktorá zaisťuje, že zariadenia od rôznych výrobcov a s rôznymi technickými parametrami budú navzájom dokázať komunikovať, zároveň zahŕňa flexibilitu a škálovateľnosť,
- **komunikácia v reálnom čase s nízkou latenciou** – čo bolo dôležité najmä pri aplikáciach na monitorovanie alebo riadiacich systémoch,
- **náklady** – zjednotenie pod spoločný komunikačný štandard by eliminovalo potrebu vytvárať mnoho separátnych sietí, čo by bolo výhodné z finančného hľadiska.

Odpoveďou na tieto požiadavky bol vznik **CIP (Common Industrial Protocol) [9].**

3.3 Štruktúra

EtherNet/IP, podobne ako iné typy CIP sietí (siete komunikujúce prostredníctvom CIP) vychádza z Open Systems Interconnection (OSI) modelu,ktorý definuje framework na implementáciu sieťových protokolov v siedmych vrstvách: **fyzická, linková, sieťová, transportná, relačná, prezentačná a aplikačná**. Siete,ktoré následujú tento model definujú kompletnú sadu sieťovej funkcionality od fyzickej implementácie až po vrstvu užívateľského rozhrania. Protokol EtherNet/IP implementuje CIP od relačnej vrstvy vyššie. Časť „IP“ v názve EtherNet/IP odkazuje na Industrial Protocol. Keďže Ethernet/IP používa štandardný Ethernet a sadu protokolov TCP/IP, zaručuje kompatibilitu a koexistenciu s inými protokolmi a aplikáciami [19], [20].

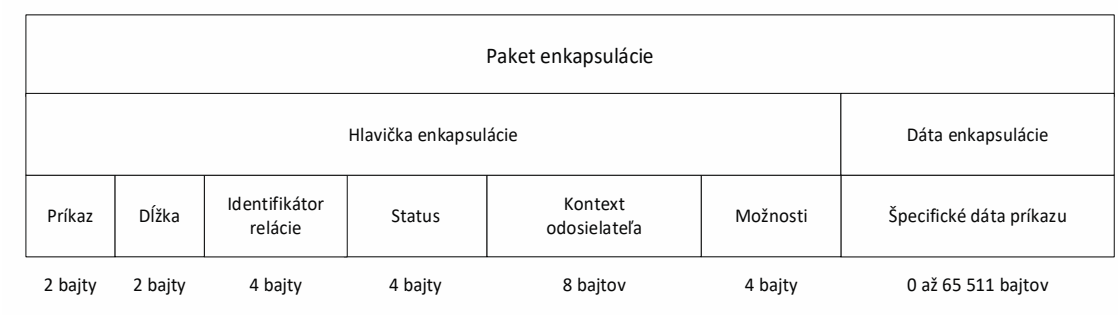


Obr. 3.1: Súvislosť Ethernet/IP s OSI Modelom [19]

Ethernet/IP používa CIP ako aplikáčnu vrstvu a može využívať všetky CIP funkcie a služby potrebné na komunikáciu medzi zariadeniami [26].

3.4 Enkapsulácia v Ethernet/IP

Všetky zapúzdrené správy posielané prostredníctvom TCP alebo UDP na port 0xAF12 pozostavajú z pevne definovanej hlavičky dĺžky 24 bajtov, za ktorou nasleduje voliteľná dátová časť. Maximálna dĺžka správy enkapsulácie (vrátane hlavičky) je 65535 bajtov. Skladá sa z nasledujúcich častí [18], [26]:



Obr. 3.2: Štruktúra paketu protokolu Ethernet/IP [26]

- **Príkaz** – určuje, akú funkciu má cieľové zariadenie vykonať. Používané príkazy sú:

Tab. 3.1: Zoznam príkazov enkapsulácie [18]

Hex. kód	**Meno**	**Popis**
0x0000	NOP	Žiadna operácia (No Operation) používa TCP
0x0004	ListServices	Zoznam podporovaných služieb, používa TCP i UDP
0x063	ListIdentity	Základné informácie o dstupných zariadeniach, používa TCP i UDP
0x0064	ListInterfaces	Informácie o sieťových rozhraniach, používa TCP i UDP
0x0065	RegisterSession	Vytvorenie relácie, používa TCP
0x0066	UnRegisterSession	Ukončenie relácie používa TCP
0x006F	SendRRData	Prenos dát v explicitnej nespojovej komunikácii, používa len TCP
0x0070	SendUnitData	Prenos dát v spojovej explicitnej komunikácii, používa TCP
0x0072	IndicateStatus	Oznamenie stavu odosielateľovi, používa TCP
0x0073	Cancel	Zastavenie spracovania požiadavky, používa TCP

Zvyšné kódové označenia mimo týchto príkazov (napríklad 0x0001 až 0x0003) sú rezervované pre historické účely (RA). Kódové značenia od 0x00C8 po 0xFFFF sú rezervované pre budúce rozšírenia špecifikácie.

- **Dĺžka** – pole špecifikujúce veľkosť dátovej položky správy v bajtoch. Ak správa neobsahuje dáta, pole je nastavené na 0. Celková dĺžka správy musí odpovedať súčtu čísla uvedeného v tomto poli a 24bajtovej veľkosti hlavičky enkapsulácie.
- **Identifikátor relácie** – generuje prijímateľ požiadavku RegisterSession. Odosielateľ ju potom používa vo všetkých paketoch enkapsulácie – odoslaných prostredníctvom príkazov uvedných v tabuľke 3.1.
- **Status** – indikuje, či prijímateľ vykonal požadovaný príkaz enkapsulácie. Nula v odpovedi indikuje úspešné vykonanie príkazu. Ak prijímateľ dostane požiadavok s nenulovou hodnotou, ignoruje ho. kódové hodnoty statusu (okrem rezervovaných hodnôt) sú:

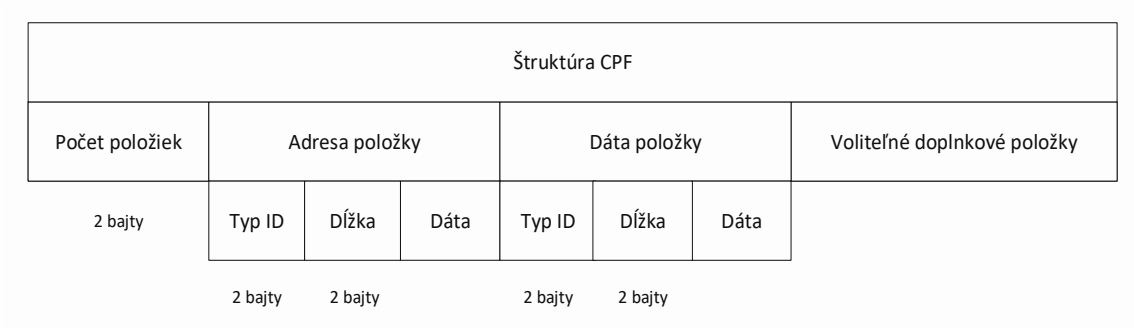
Tab. 3.2: Chybové kódy [18]

Hex. kód	**Popis**
0x0000	Úspech
0x0001	Odosielateľ poslal nesprávny alebo nepodporovaný príkaz
0x0002	Nedostatočná pamäť na spracovanie príkazu na strane príjemcu. Nejde o aplikačnú chybu, nastáva len v prípadoch, keď enkapsulačná vrstva nemôže získať potrebné pamäťové zdroje.
0x0003	Nesprávne formatované dáta v dátovej časti správy enkapsulácie.
0x0064	Odosielateľ použil nesprávny identifikátor relácie
0x0065	Príjemca obdržal správu nesprávnej veľkosti
0x0069	Nepodporovaná revízia protokolu enkapsulácie

- **Kontext odosielateľa** – Odosielateľ príkazu priradí hodnotu do poľa hlavičky. Prijímateľ vráti túto hodnotu v odpovedi bez úpravy. Príkazy, ktoré neočakavajú odpoveď môžu toto pole ignorovať.
- **Možnosti** – odosielateľ paketu enkapsulácie nastaví možnosti na nulu. Príjemca overí či je položka nulová. Ak nie je, paket zahodí. Účelom tohto poľa je poskytnúť bity na modifikáciu významu príkazov enkapsulácie.
- **Špecifické dáta príkazu** – štruktúra tohto poľa závisí od kódu príkazu. Väčšina príkazov používa buď pevne definovanú čtruktúru, alebo CPF (Common Packet Format). CPF umožňuje príkazom štruktúrovať ich pole špecifických dát príkazu rozšíriteľným spôsobom.

3.5 Štruktúra CPF

Common Packet Format (CPF) je štruktúra, ktorá poskytuje spôsob usporiadania dátového poľa enkapsulácie pre príkazy, ktoré toto pole špecifikujú. Ak to definícia príkazu požaduje, CPF umožňuje zabalenie viacerých položiek do jednej štruktúry enkapsulácie, ako je vidieť na následovnom obrázku [26]:



Obr. 3.3: Štruktúra CPF [26]

4 Komunikácia v EtherNet/IP

Aplikačná vrstva CIP definuje sadu aplikačných objektov a profilov zariadení ktorá definuje spoločné rozhrania a správania. Komunikačné služby CIP tiež umožňujú end-to-end komunikáciu medzi zariadeniami v rôznych CIP sieťach. EtherNet/IP mapuje komunikačné služby na Ethernet a TCP/IP, čo zabezpečuje interoperabilitu medzi zariadeniami od rôznych predajcov v Ethernete, ako aj v iných CIP sieťach [21].

4.1 Príncip

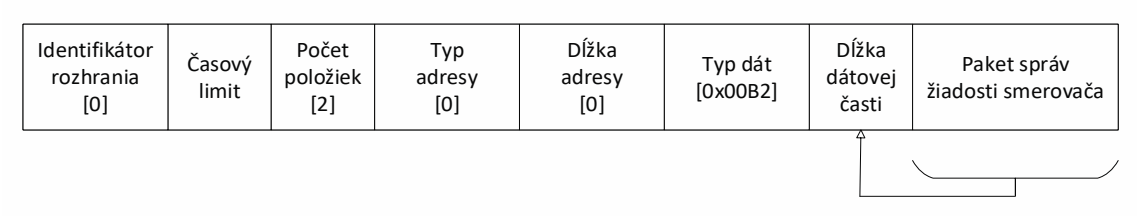
Pred začiatkom komunikácie je nutné, aby sa vytvorila logická relácia „session“ poslaním správy **Register Session**. Odosielateľ posiela požiadavku na vytvorenie relácie. Cieľ pošle odpoveď s prislušným identifikátorom relácie. v rámci tejto relácie potom prebieha výmena správ. Na ukončenie komunikácie sa odosiela **Unregister Session**,ktorá uzatvára reláciu. Príjemca po prijatí tejto správy iniciuje ukončenie TCP/IP spojenia. EtherNet/IP definuje dva hlavné typy komunikácie: **explicitnú** a **implicitnú**. Všetky spojenia v Ethernet/IP sú vytvorené prostredníctvom správy **UCMM Forward Open** [21].

4.2 Explicitná komunikácia

Explicitná komunikácia vo všeobecnosti funguje na klient/server príncipe. Tento typ sa používa na časovo nekritické dáta, bežne na informácie. Explicitné správy obsahujú popis ich významu, takže prenos je menej efektívny, avšak veľmi flexibilný. V EtherNet/IP explicitné správy používajú TCP a na pripojenie port 0xAF12hex (44818dec). Explicitné správy môžu byť posielané buď spojovo (connected) alebo nespojovo (unconnected) [26].

- **Spojová explicitná komunikácia** – vyžaduje vytvorenie spojenia pred posielaním správ. Tým sa zabezpečí, že všetky zdroje potrebné na riadenie spojenia sú rezervované na tento účel po dobu trvania spojenia. To umožňuje časovo rýchlejšiu odozvu na požiadavky. Toto je veľmi užitočné v aplikáciach, ktoré požadujú pravidelné explicitné požiadavky.
- **Nespojová explicitná komunikácia**- tento mechanizmus využíva veľmi limitované množstvo zdrojov v uzloch, ktoré môžu byť niekedy výrazne zaťažené. Preto by sa tento typ mal používať iba v prípadoch, kedy aplikácia vyžaduje nepravidelné a zriedkavé intervaly požiadaviek. Nevytvára sa priame spojenie medzi dvoma komunikačnými uzlami (klient a server).

Explicitné správy sú v Ethernet/IP posielané s TCP/IP hlavičkou a používajú príkaz zapúzdrenia SendRRData (nespojová) alebo SendUnitData (spojová). Plná enkapsulácia UCMM požiadavky, ktorá vychádza z CPF štruktúry vyzerá napríklad následovne: [26]:



Obr. 4.1: Enkapsulácia UCMM požiadavky [26]

Paket správ žiadosti smerovača obsahuje CIP správu, ktorá môže byť požiadavkom alebo odpoveďou. CIP definuje štandardný dátový formát na doručovanie dát z a do objektu Message Router. Tento formát sa v rámci CIP využíva na rôznych miestach, vrátane služby Unconnected Send a dátových štruktúr UCMM väčšiny CIP sietí [26].

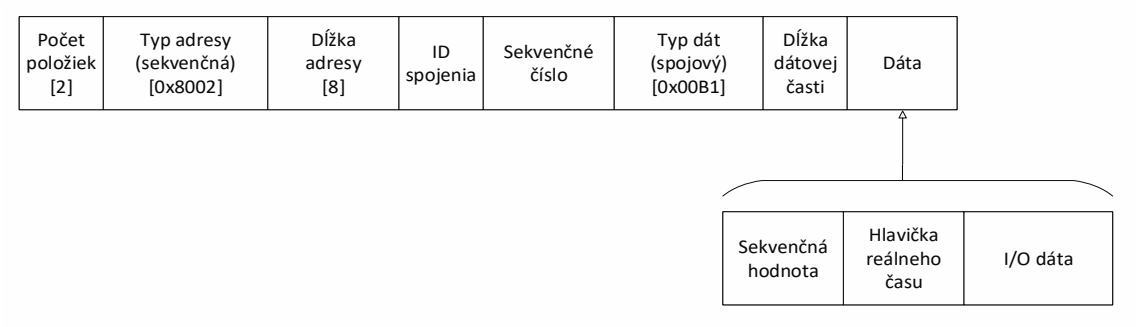
4.3 Implicitná komunikácia

Implicitná komunikácia často býva označovaná ako „I/O“ a vo všeobecnosti je časovo kritická. Typicky sa používa na prenos real-time dát, kde je kladený dôraz na rýchlosť a nízku latenciu. Implicitné správy obsahujú veľmi málo informácií o ich význame, teda prenos je viac efektívny, ale zároveň je menej flexibilný ako pri správach explicitných. Vysielané dáta sú rýchlo interpretované. Pri implicitnej komunikácii sa vytvorí spojenie (CIP spojenie) medzi dvoma zariadeniami a následne sú produkované implicitné správy podľa vopred preddefinovaného spúšťacieho mechanizmu, typicky na určenej frekvencii paketov. Obe zariadenia sa dohodnú na dátovom formáte,ktorý budú používať. V implicitnej komunikácii EtherNet/IP používa UDP a môže byť multicast alebo unicast [21].

Spojenia sú vytvárané použitím služby Forward Open Request objektu Connection Manager Object. Forward Open Request obsahuje všetky parametre pripojenia, vrátane prenosovej triedy, spúšťača produkcie, časových informácií, elektronického kľúča a ID pripojenia. Implicitné správy môžu využiť CIP komunikačný model Producer/Consumer. V tomto modeli vysielajúce zariadenie (producer) prenáša dáta raz, bez ohľadu na počet príjemcov (consumers). Všetky zainteresované zariadenia obdržia rovnaké dáta. V EtherNet/IP sú produkované dáta identifikované pomocou IP multicast adresy a ID CIP pripojenia. Model Producer/Consumer prináša

väčšiu efektivitu siete keď viacerí spotrebitelia potrebujú prijímať rovnaké dáta od poskytovateľa. V I/O pripojeniach keď sa vytvorí spojenie, nie je overovanie požiadavka/odpoveď, dáta sú jednoducho poskytované a spotrebované v intervaloch určených Production Triggerom, ktorý bol nastavený pri vytváraní spojenia [21].

I/O správy v Ethernet/IP sú posielané s UDP/IP hlavičkou. Nepoužívajú hlavičku enkapsulácie, avšak správa stále dodržiava formát CPF. Na prenos I/O správ sa používa port 0x08AE (2222dec). Formát I/O správy môže vyzerať napríklad následovne: [26]:



Obr. 4.2: Formát I/O správy [26]

Pole s dátami obsahuje I/O dáta s 16bitovým prefixom sekvenčnej hodnoty pre paket. Stav Run/Idle môže byť indikovaný prostredníctvom hlavičky reálneho času, prípadne poslaním paketu s I/O dátami (Run), alebo bez I/O dát (Idle) [26].

4.4 Typy EtherNet/IP zariadení

Existuje niekoľko definovaných klasifikácií zariadení na základe ich správania a typov podporovaných EtherNet/IP komunikácií, ktoré podporujú, zoradených od najjednoduchších po najkomplexnejšie [21]:

- **Explicit Message Server** – server explicitných správ odpovedá na komunikáciu typu požiadavok/odpoveď spustenú klientom explicitných správ. Príkladom takéhoto servera je čítačka čiarových kódov.
- **Explicit Message Client** – klient explicitných správ inicializuje požiadavok/odpoveď komunikáciu s ďalšími zariadeniami. Rýchlosť správ a požiadavky na latenciu zvyčajne nie sú príliš náročné. Príkladmi explicitných klientov sú HMI zariadenia, programovacie nástroje alebo aplikácie založené na Linuxe, ktoré zbierajú dáta z kontrolných zariadení.
- **I/O adapter** – prijíma požiadavky na implicitné komunikačné spojenia od I/O skenera, následne poskytuje svoje I/O dáta podľa požiadovanej frekvencie.

I/O adaptér môže byť jednoduché digitálne vstupné zariadenie, alebo niečo komplexnejšie, ako napríklad pneumatický ventilačný systém.

- **I/O Scanner** – inicializuje implicitnú komunikáciu so zariadeniami I/O adaptéra. Skener je typicky najkomplexnejší typ EtherNet/IP zariadenia nakoľko musí riešiť záležitosti ako ktoré spojenie konfigurovať a ako konfigurovať zariadenie adaptéru. Skener typicky podporuje incializáciu explicitných správ. Príkladom I/O skenera je programovateľný kontrolér.

4.5 ODVA

Open DeviceNet Vendors Association je medzinárodná asociácia združujúca členov najvýznamnejších svetových automatizačných spoločností. Spoločne ODVA a jej členovia podporujú sieťové technológie založené na CIP. ODVA riadi vývoj týchto otvorených technológií a pomáha priemyselným spoločnostiam a užívateľom CIP sietí prostredníctvom svojich aktivít v oblasti vývoja štandardov, certifikácie, vzdelávania dodávateľov a zvýšovania povedomia v priemysle. ODVA ponúka testovanie zhody ako súčasť svojich certifikačných aktivít na zaistenie, že produkty postavené na jej špecifikáciách fungujú v systémoch s viacerými dodávateľmi [21].

5 Common Industrial Protocol

Common Industrial Protocol (CIP) je na médiu nezávislý, spojovo-založený, objektovo-orientovaný protokol navrhnutý pre aplikácie na automatizáciu. Zahŕňa komplexný súbor komunikačných služieb potrebných pre aplikácie na automatizáciu: ovládanie, bezpečnosť, synchronizácia, pohyb, konfigurácia a informácie. Umožňuje užívateľom integrovať tieto aplikácie v rámci podnikových sietí Ethernet a internetu [1].

5.1 Základné vlastnosti

Podporovaný stovkami zariadení po svete a ako protokol skutočne nezávislý na médiu CIP poskytuje užívateľom zjednotenú komunikačnú architektúru po celom podniku. Taktiež im poskytuje benefity z mnohých výhod otvorených sietí, pričom chráni ich ivestície do automatizácie pri budúcich modernizáciach [21].
CIP prináša: [21]:

- koherentnú integráciu riadenia I/O operácií, konfigurácie zariadení a zberu dát,
- bezproblémový tok informácií naprieč množstvom sietí,
- schopnosť implementácie multi-vrstvových sietí bez zvýšených nákladov a zložitosti spojených s mostami a proxy servermi,
- minimalizáciu investicií do systémového inžinierstva, inštalácie a spustenia prevádzky,
- slobodu vybrať si najlepšie z ponúkaných produktov, so zárukou konkurenčných cien a nízkych nákladov na integráciu.

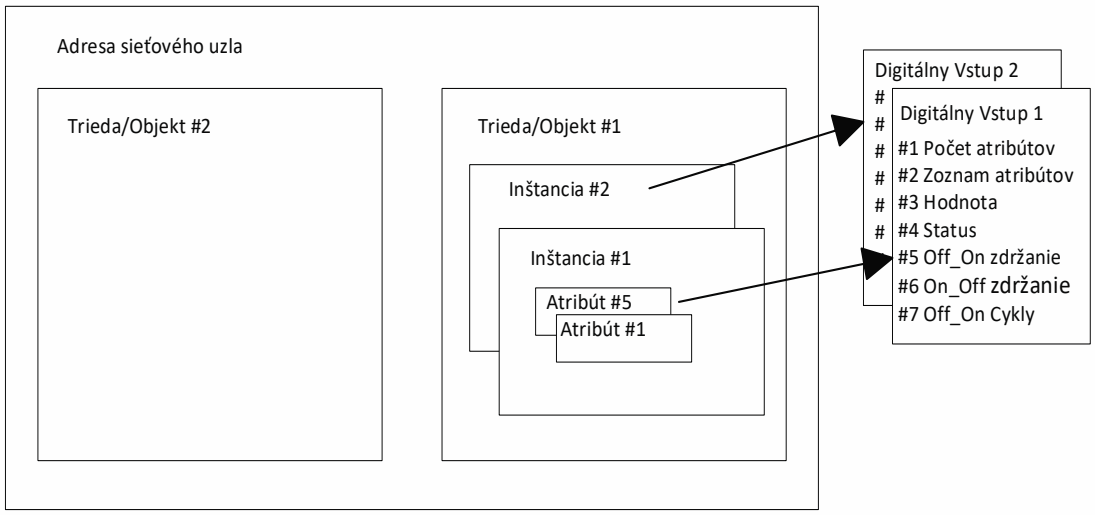
5.2 Obejktovo-orientovaná reprezentácia

V rámci aplikačnej vrstvy CIP sú zariadenia reprezentované prostredníctvom objektového modelu. Objekty v rámci zariadení sú skupiny súvisiacich dát a funkcionalít spojených s týmito dátami. Pri popise CIP protokolu sa používajú následujúce pojmy [1], [7], [21]:

- **Objekt** – abstraktná reprezentácia konkrétnej komponenty v rámci produktu.
- **Trieda** – sada objektov, ktoré reprezentujú rovnaký druh systémovej komponenty. Všetky objekty triedy majú identickú štruktúru a správanie, môžu sa však líšiť v hodnotách atribútov.
- **Inštancia** – popis konkrétny a reálny výskyt objektu.
- **Atribút** - popis navonok viditeľnej vlastnosti alebo črty objektu. Typicky poskytujú informácie o stave alebo riadia prevádzku objektu.

- **Inštancovať** – vytvoriť inštanciu objektu so všetkými atribútmi inicializovanými na nulu, pokiaľ nie sú v definícii objektu definované predvolené hodnoty.
- **Správanie** – popisuje správanie objektu. Je reakciou na rôzne udalosti, ktoré objekt rozpozná (napríklad prijatie požiadavky služby, detekcia interných chýb alebo uplynutie časovačov).
- **Služba** – funkcia podporovaná triedou alebo objektom. CIP definuje sadu všeobecných služieb, ktoré poskytuje na definovanie tried objektov a služieb špecifikovaných výrobcom.
- **Komunikačné objekty** – odkazuje na triedu bojektov, ktoré riadia a zabezpečujú výmenu implicitných (I/O) a explicitných správ počas behu systému.
- **Aplikačné objekty** – odkazuje na triedu objektov, ktoré implementujú vlastnosti špecifické pre daný produkt.

CIP priamo nešpecifikuje, ako sú dátové objekty implementované, ale skôr uvádza, ktoré dátové hodnoty alebo atribúty musia byť podporované a sprístupnené iným CIP zariadeniam. Následujúci obrázok znázorňuje spôsob, akým CIP aplikuje objektovo-orientovaný model [1], [7], [21].



Obr. 5.1: Príncip objektovo-orientovaného pohľadu v CIP [21]

5.3 Typy CIP objektov

CIP vyžaduje určité objekty na popis zariadenia, jeho funkcionality, komunikácie a jednoznačnej identifikácie. Toto sú tri typy objektov definované CIP [21]:

- **Povinné objekty** – musia obsahovať všetky CIP zariadenia. Tieto objekty zahŕňajú Identifikačný objekt, Objekt správ smerovača a sieťové špecifické objekty.

- **Aplikačné objekty** – popisujú spôsob zapúzdrenia dát zariadením. Tieto objekty sú špecifické pre typ zariadenia a funkciu.
- **Objekty špecifikované výrobcom** -popisujú služby špecifické pre konkrétneho poskytovateľa. Sú voliteľné a nie sú uvedené v preddefinovanom profile zariadenia.

5.4 Všeobecné služby CIP

V tejto práci boli implementované nasledujúce služby explicitnej komunikácie [1]:

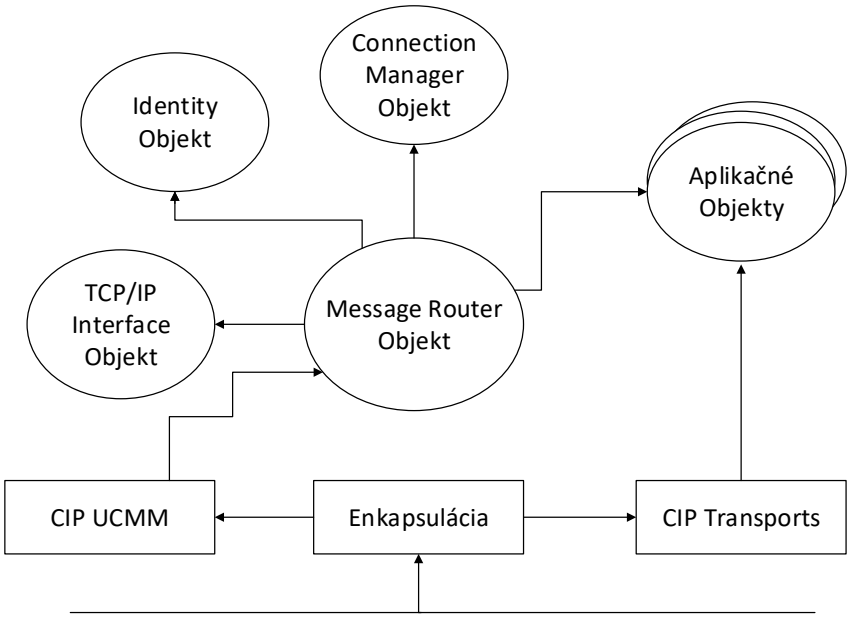
- **Get_Attributes_All – (0x01)** – vracia obsah inštančných alebo triedných atribútov definovaných v definícii objektu. Ak je úspešne doručená požiadavka, v odpovedi sú poskytnuté atribúty. V prípade chyby je odpoveďou chybové hlásenie. Zariadenie musí zahrnúť iba implementované atribúty.
- **Get_Attribute_List – (0x03)** – vracia obsah vybraných atribútov, ktoré boli uvedené v požiadavke. Klient musí poskytnúť zoznam atribútov zadanním ich ID.
- **Reset – (0x05)** – vyvolá Reset zadaného objektu alebo triedy. Zvyčajne spôsobí prechod do defaultného stavu. Ak by Reset spôsobil, že objekt prejde do stavu, v ktorom nebude schopný odpovedať, musí poskytnúť odpoveď pred vykonaním Resetu. V prípade chyby zariadenie odpovie chybovým hlasením. V opačnom prípade poskytne potvrdenie o úspešnom vykonaní služby. Sú tri oficiálne definované typy Resetu:
  - 0 – podobné cyklu vypnutia/zapnutia zariadenia, je to defaultný stav,
  - 1 – návrat do továreňských nastavení,
  - 2 – reset typu „out-of-box“, komunikačné nastavenia ostávajú zachované.
- **Get_Attribute_Single – (0x0E)** – vráti obsah jedného atribútu špecifikovaného v požiadavke. Ak nastane chyba, odpoveď bude typu Error Response. V opačnom prípade sa vráti úspešná odpoveď s údajmi atribútu.
- **Set_Attribute_Single – (0x01)** – nastavuje novú hodnotu atribútu. Pred zápisom je nutná validácia hodnoty – kontrola typu, rozsahu a prístupového pravidla. Ak je validácia úspešná, prebehne zápis a zariadenie vráti úspešnú odpoveď na Set_Attribute_Single. Ak nastala chyba, je poslaná odpoveď typu Error Response.

5.5 Implementácia objektov v Ethernet/IP

Každé EtherNet/IP zariadenie by malo implementovať minimálne prvú inštanciu každého z nasledujúcich objektov [18]:

- Identity (0x01)
- Message Router (0x02)
- Connection Manager (0x06)
- TCP/IP Interface (0xF5)

Ak je použitý Ethernet ako médium, zodpovedajúcim linkovým objektom musí byť Ethernet Link objekt. V prípade použitia iného média musí výrobca špecifikovať linkový objekt. Hoci nemá kód triedy objektu, každé zariadenie musí tiež implementovať Unconnected Message Manager (UCMM). UCMM zodpovedá za spracovanie nespojových explicitných správ. To zahŕňa vytváranie explicitnej komunikácie i zostavenie I/O spojení. Pri explicitnej komunikácii UCMM zabezpečuje preposlanie správ objektu Message Router (0x02), ktorý ich rozposiela príslušným objektom [18].



Obr. 5.2: Základný objektový model EtherNet/IP zariadenia [18]

5.5.1 Identity (0x01)

Poskytuje identifikačné a základné informácie o zariadení. Musí byť súčasťou všetkých CIP produktov. Objekt Identity obsahuje nasledujúcich 7 povinných atribútov [1]:

Tab. 5.1: Základné atribúty objektu Identity

ID	**Pravidlo prístupu**	**Názov atribútu**	**Dátový typ**	**Popis**
1	Get	Vendor ID	UINT	Identifikácia výrobcu podľa čísla
2	Get	Device Type	UINT	Typ zariadenia podľa klasifikácie
3	Get	Product Code	UINT	Kód konkrétneho produktu od daného výrobcu
4	Get	Revision Major Revision Minor Revision	STRUCT of: USINT USINT	Verzia zariadenia, ktoré Identity reprezentuje Hlavná verzia Vedľajšia verzia
5	Get	Status	WORD	Celkový stav zariadenia
6	Get	Serial Number	UDINT	Sériové číslo zariadenia
7	Get	Product Name	SHORT STRING	Názov zariadenia čitateľný pre človeka

Okrem uvedených povinných atribútov Identity môže obsahovať aj voliteľné atribúty ako: Štát zariadenia, aktívny jazyk, geografická lokácia a ďalšie. Ich kompletný zoznam je poskytovaný v: [1]. Z dostupných implementovaných služieb Identity podporuje Get_Attribute_Single, Get_Attributes_All, Get_Attribute_List a Reset. Ak sú dostupné voliteľné atribúty s pravidlom prístupu Set, poskytuje aj Set_Attribute_Single službu [1].

5.5.2 Message Router 0x02

Poskytuje komunikačný bod, prostredníctvom ktorého môže klient adresovať službu ktorejkoľvek triede objektu alebo inštancii nachádzajúcej sa vo fyzickom zariadení. Message Router definuje 4 voliteľné inštančné atribúty, uvedené v tabuľke 5.2. Z implementovnaných všeobecných služieb Message Router Object podporuje služby Get_Attribute_Single a Get_Attributes_All [1].

Tab. 5.2: Inštančné atribúty objektu Message Router [1]

ID	Pravidlo prístupu	Názov	Typ	Popis
1	Get	Object list	STRUCT	Zoznam podporovaných objektov v zariadení. Obsahuje podatribúty.
		Number	UINT	Počet podporovaných tried objektov.
		Classes	ARRAY of UINT	Pole s číslami podporovaných tried objektov.
2	Get	Number Available	UINT	Maximálny počet spojení podporovaných zariadením.
3	Get	Number Active	UINT	Aktuálny počet využívaných spojení.
4	Get	Active Connections	ARRAY of UINT	Zoznam ID aktívnych spojení.

5.5.3 Connection Manager (0x06)

Trieda Connection Manager alokuje a spravuje interné zdroje spojené s I/O a explicitnými spojeniami. Konkrétna inštancia vytvorená touto triedou sa označuje ako Connection Instance alebo Connection Object. Z dostupných všeobecných služieb CIP podporuje Get_Attributes_All, Get_Attribute_Single a Set_Attribute_Single. Špecifické pre objekt sú však tieto služby, ktoré slúžia na vytvorenie (a ukočenie) spojenia s koncovým zariadením [1]:

- **Forward_Open (0x54)** – vytvára jedno alebo dve spojenia, v závislosti od požiadavky. Tieto dve spojenia sa označujú ako O–>T (Originator to Target) a T–>0 (Target to Originator). Ďalšími parametrami služby sú napríklad identifikátory spojení, sériové čísla, požadované intervaly prenosu (RPI) a cesta k cieľovému zariadeniu.
- **Large_Forward_Open (0x5B)** – líši sa od Forward_Open len maximálnou veľkosťou pripojenia, ktoré môže nadviazať.
- **Forward_Close (0x4E)** – slúži na ukočenie nadviazaného spojenia. Požiadavka Forward_Close spôsobí dealokáciu všetkých zdrojov vo všetkých uzloch spojenia.

Connection Manager má mnoho voliteľných inštančných atribútov, ich kompletný zoznam je definovaný v [1]. Následujúca tabuľka popisuje niektoré z nich:

Tab. 5.3: Atribúty objektu Connection Manager [1]

ID	pravidlo prístupu	Názov	Popis
1	Set	Open Requests	Počet prijatých položiek na službu Forward Open (FO)
2	Set	Open Format Rejects	Počet požiadaviek FO, zamietnutých pre chybný formát
3	Set	Open Resource Rejects	Počet požiadaviek FO, zamietnutých pre nedostatok prostriedkov
4	Set	Open Other Rejects	Počet požiadaviek FO, zamietnutých z iných dôvodov
5	Set	Close Requests	Počet požiadaviek na Forward Close

5.5.4 Ethernet Link (0xF6)

Slúži na sledovanie stavových informácií a počítadiel typických pre komunikačné rozhranie typu Ethernet 802.3. Každé zariadenie musí podporovať presne jednu inštanciu tohto objektu pre každé zo svojich IEEE 802.3 komunikačných rozhraní. Väčšina atribútov objektu je voliteľná, povinné sú uvedené v tabuľke 5.4. Ethernet Link objekt podporuje nasledujúce všeobecné služby [18]:

- Get_Attribute_All,
- Get_Attribute_Single,
- Set_Attribute_Single.

Tab. 5.4: Povinné inštančné atibúty objektu Ethernet Link [18]

ID	Pravidlo prístupu	Názov	Typ	Popis
1	Get	Interface Speed	UDINT	Aktuálne používaná rýchlosť rozhrania v MBps
2	Get	Interface Flags	DWORD	Stavové príznaky rozhrania (bitová mapa)
3	Get	Physical Address	ARRAY OF 6 USINTs	MAC adresa sieťového rozhrania (6 bajtov)

5.5.5 TCP/IP Interface (0xF5)

Slúži na konfiguráciu TCP/IP sieťového rozhrania zariadenia. Možno pomocou neho nastaviť napríklad IP adresu zariadenia, sieťovú masku a adresu východzej brány. Zariadenie musí obsahovať presne jednu inštanciu tohto objektu pre každé sieťové rozhranie podporujúce komunikáciu TCP/IP. Objekt definuje 6 povinných atribútov, uvedených v tabuľke 5.5. Zvyšné atribúty sú implementované v závislosti od konfigurácie zariadenia. TCP/IP podporuje všeobecné služby [18]:

- Get_Attribute_All,
- Get_Attribute_Single,
- Set_Attribute_Single,
- Set_Attribute_All.

Tab. 5.5: Inštančné atribúty objektu TCP/IP Interface [18]

ID	**Pravidlo prístupu**	**Názov**	**Typ**	**Popis**
1	Get	Status	DWORD	Stav sieťového rozhrania
2	Get	Configuration Capability	DWORD	Bitové príznaky schopností rozhrania
3	Set	Configuration Control	DWORD	Bitové príznaky konfigurácie rozhrania
4	Get	Physical Link Object Path Size Path	STRUCT UINT padded EPATH	Cesta k fyzickému link objektu Dlžka cesty Logické segmenty idetifikujúce link objekt
5	Set	Physical Link Object Interface Configuration IP Address Network Mask Name Server Name Server 2 Domain Name	STRUCT STRUCT UDINT UDINT UDINT UDINT STRING	Cesta k fyzickému link objektu Konfigurácia TCP/IP siete IP adresa zariadenia Adresa východzej brány Primárny server Sekundárny server Doména
6	Set	Host Name	STRING	Meno užívateľa

6 Výsledky

Praktická realizácia práce pozostávala z dvoch častí. V prvej časti bola vytvorená GUI aplikácia v Jave, ktorá slúži ako klient a poskytuje pripojenie k serveru. V druhej časti bola funkčná logika aplikácie integrovaná do prostredia OpenMUC, výsledkom čoho vznikol driver poskytujúci komunikačné služby EtherNet/IP prostredníctvom vytvorených kanálov.

6.1 Základná štruktúra GUI aplikácie

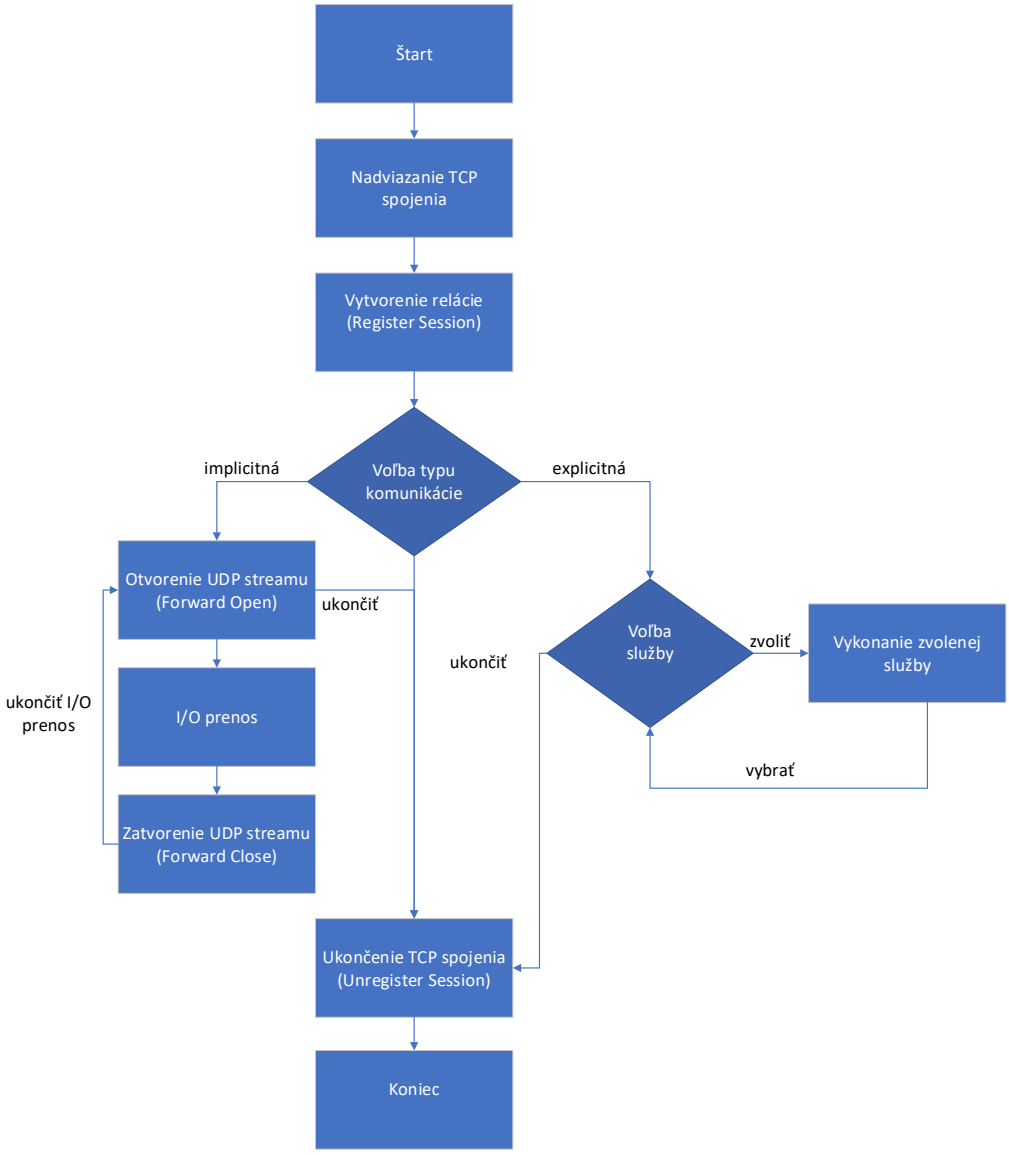
Aplikácia klienta pozostáva z troch implementačných častí:

- **Grafické užívateľske rozhranie** – vytvorené pomocou knižnice Swing v Jave. Sprostredkuje užívateľovi grafickú komponentu zloženú z tlačidiel a textového poľa, pomocou ktorej dokáže nadviazať spojenie s koncovým zariadením a zároveň sledovať výstupy komunikácie. Na prvej úrovni sú tlačidlá potrebné na korektné zadanie IP adresy. Iné znaky nie sú povolené. Definíciu a štrukúru jednotlivých grafických prvkov aplikácie riadi trieda `ENIP_form`.
- **Komunikačná a logická vrstva** – zahŕňa metódy a konštrukcie na tvorbu požiadaviek jednotlivých služieb, spracovanie odpovedí, parsovanie hodnôt zadaných užívateľom a ďalšie funkcionality. Napríklad metódy `SendRRData_set()`, `Get_Attribute_Single_set()`, `response_reading()` a ďalšie.
- **Objektovo-orientovaná dátová štruktúra** – triedy a konštrukcie ktoré zariaďujú definície nových CIP objektov, formátu atribútov, zostavenie segmentov či základnej štruktúry paketu. Patria sem napríklad `CIP_object_library`, `CIP_attribute_format`, `Encapsulation_packet` a ďalšie.

6.2 Návrh

Následujúci obrázok popisuje základnú funkcionalitu aplikácie a tok jednotlivých operácií. Architektúra aplikácie bola navrhnutá tak, aby obsahovala dve základné vetvy – pre implicitnú i explicitnú komunikáciu. Obsahuje dva hlavné rozhodovacie body. Prvý nastane po úspešnom vytvorení relácie, kedy užívateľ volí typ komunikácie, prípadne môže ukončiť spojenie. Druhým rozhodovacím bodom je voľba konkrétnej explicitnej služby. Opäť je možnosťou aj ukončenie spojenia. Po vykonaní zvolenej služby sa opakuje výber služby. Toto prebieha cyklicky, pokým užívateľ neukončí spojenie, alebo nedôjde k inej systémovej chybe, ktorá spôsobi prerušenie spojenia. Vo vetve s implicitnou komunikáciou užívateľ nastaví parametre pre I/O

spojenie a po potvrdení sa spustí UDP stream na porte 2222. Po ukončení I/O prenosu nasleduje ukončenie behu aplikácie.



Obr. 6.1: Vývojový diagram GUI aplikácie

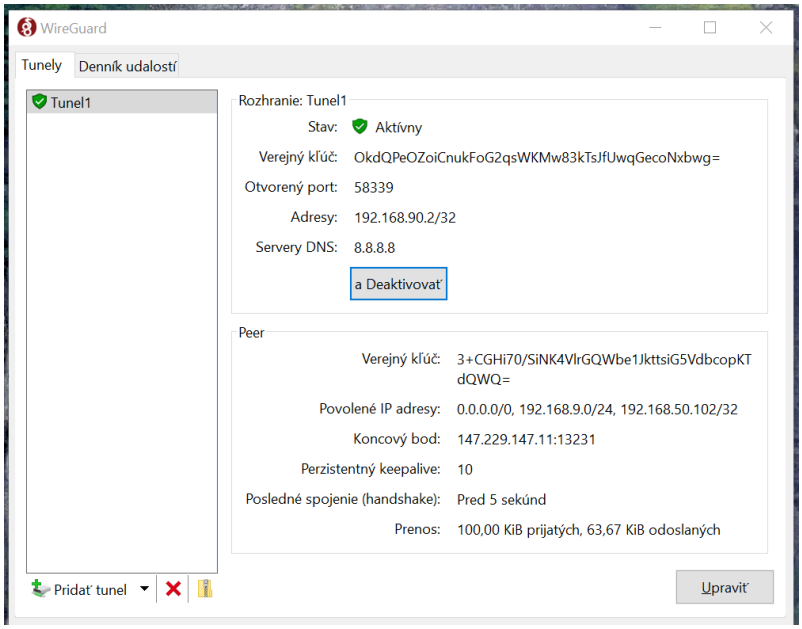
6.3 Koncové zariadenie

Ako koncové reálne zariadenie bol v práci použitý SIMATIC S7-1500, na ktorý sa pripájalo prostredníctvom IP tunelu realizovaného pomocou nástroja Wireguard. Zariadenie malo však limitované možnosti komunikácie – poskytovalo prakticky len podporu služieb Get_Attribute_Single a Get_Attributes_All. Z CIP objektov boli prakticky a komunikačne dostupné len Identity a Connection Manager, aj tie však s obmedzenými možnosťami.

Z toho dôvodu boli ako ďalšie varianty na testovanie funkcionality použité verejne dostupné knižnice OpENer [1] a CIPster [2]. Tieto knižnice boli spúšťané vo VM s nainštalovanou distribúciou Linux Debian. Poskytovali vlastnosti a funkcionality jednoduchého EtherNet/IP servera.

OpENer slúžil predovšetkým na explicitnú komunikáciu. Okrem Identity a Message Routera implementuje aj TCP/IP a Ethernet Link objekty. Podporuje všetky Get* služby dostupné v implementácii a službu Set_Attribute_Single. Nepodporuje implicitnú komunikáciu.

CIPster bol väčšinou využitý na testovanie funkcionality I/O komunikácie. Vďaka podpore služby Forward Open bolo možné pomocou neho realizovať implicitnú komunikáciu. Z hľadiska explicitnej komunikácie je jeho funkcionalita obmedzená.



Obr. 6.2: Vytvorený tunel pomocou programu Wireguard poskytuje spojenie s koncovým zariadením

[1] [knižnica OpENer] `https://github.com/EIPStackGroup/OpENer`
[2] [knižnica CIPster] `https://github.com/liftoff-sr/CIPster`

6.4 Kódové riešenie

GUI aplikácia je z hľadiska zdrojového kódu rozčlenená do nasledujúcich tried:

- `Main` – základná a spúšťacia trieda aplikácie. Tvorí kompletnú logiku aplikácie, zabezpečuje nadviazanie a ukončenie spojenia. Všeobecne je možné ju rozdeliť do nasledujúcich logických úsekov:
  - **GUI logika** – metódy a listenery charakterizujúce správanie aplikácie pri manipulácii s tlačidlami. (Napríklad `useForwardOpen()`).
  - **Tvorba a prevádzka spojení** – konštrukcie na nadviazanie TCP a UDP spojení, vytvorenie a prevádzka relácie, validácia IP adresy a ďalšie. Patria sem metódy ako `TCP_establish()` alebo `SendRRData_set()`.
  - **CIP komunikácia** – programové komponenty, vykonávajúce spracovanie požiadavkov a odpovedí služieb CIP. Túto kategóriu tvoria napríklad všetky metódy čítania atribútov zo zariadenia.
  - **Parsery a konvertory** – prevádzajú a analyzujú rôzne typy poskytnutých dátových typov a štruktúr na iné typy. Typickým príkladom je metóda `attribute_interpret()` ktorá poskytuje CIP dáta v čitateľnej podobe ďalším metódam na spracovanie.
  - **Generátory** – vytvárajú štruktúry a segmenty dát, potrebné na prevádzku komunikácie. Takýto generátor je napríklad metóda `CPF_build()`, ktorá zapúzdruje CIP segment do CPF formátu.
  - **Interpretér chýbových hlásení** – metóda `generalStatus_control()`, ktorá poskytuje chýbové kódy pri zlyhaní vykonávania CIP služieb.

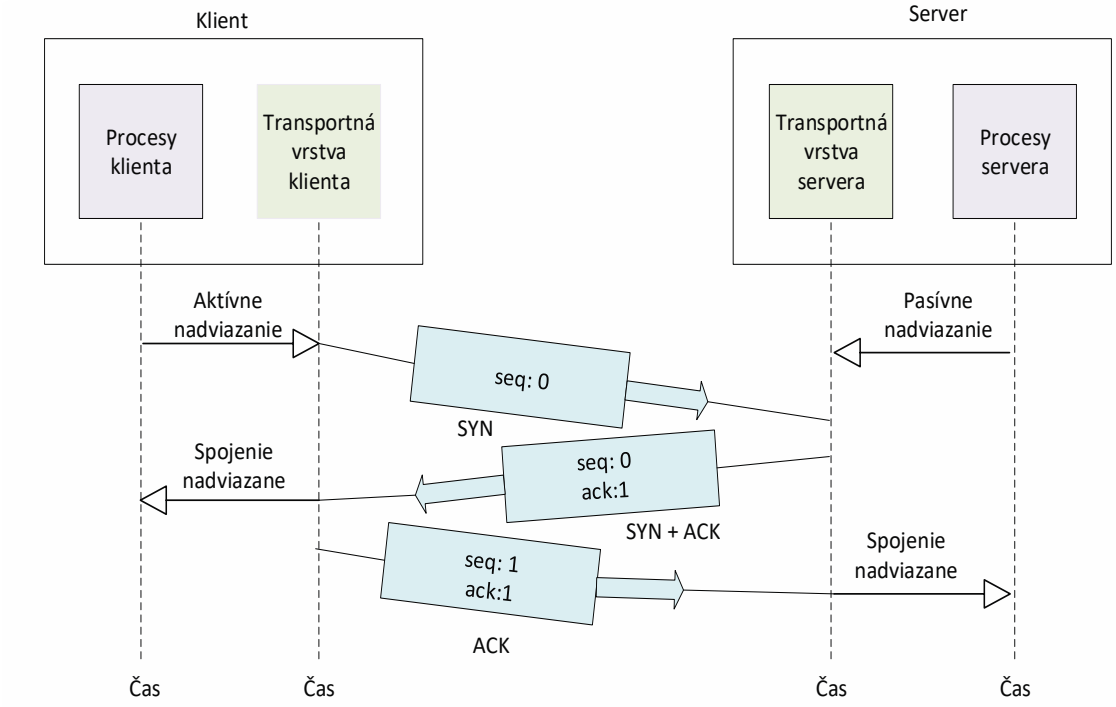
- `ENIP_form` – poskytuje funkcionality a komponenty grafického užívateľského rozhrania aplikácie.
- `CIP_object_library` – obsahuje register (knižnicu) jednotlivých objektov. Je realizovaná použitím dátovej štruktúry statická mapa `HashMap<>`. Každý objekt má pridelený zoznam atribútov, tak ako je stanovené špecifikáciami [1] a [18]. Obsahuje metódy `getObjectProfile()` a `hasObjectProfile()` na prístup k objektom.
- `Encapsulation_packet` – zostavenie paketu enkapsulácie, prevod na pole bajtov a poskytuje metódy na nastavenie dôležitých polí paketu.
- `CIP_object_format` – definuje štruktúru konkrétneho objektu.
- `CIP_segment_parameters` – generuje CIP segment cesty pre požiadavky na základe zadaných parametrov.
- `CIP_attribute_format` – špecifikuje vlastnosti konkrétneho atribútu .

6.5 Priebeh aplikácie

Komunikáciu iniciuje klient. Po spustení aplikácie sa v poskytnutom GUI zobrazí výzva na zadanie IP adresy, na ktorej je dostupné koncové zariadenie, komunikujúce prostredníctvom protokolu EtherNet/IP. Prvým krokom komunikácie je teda nadviazanie TCP spojenia. K tomu sa používa three-way handshake algoritmus.

6.5.1 Vytvorenie TCP spojenia

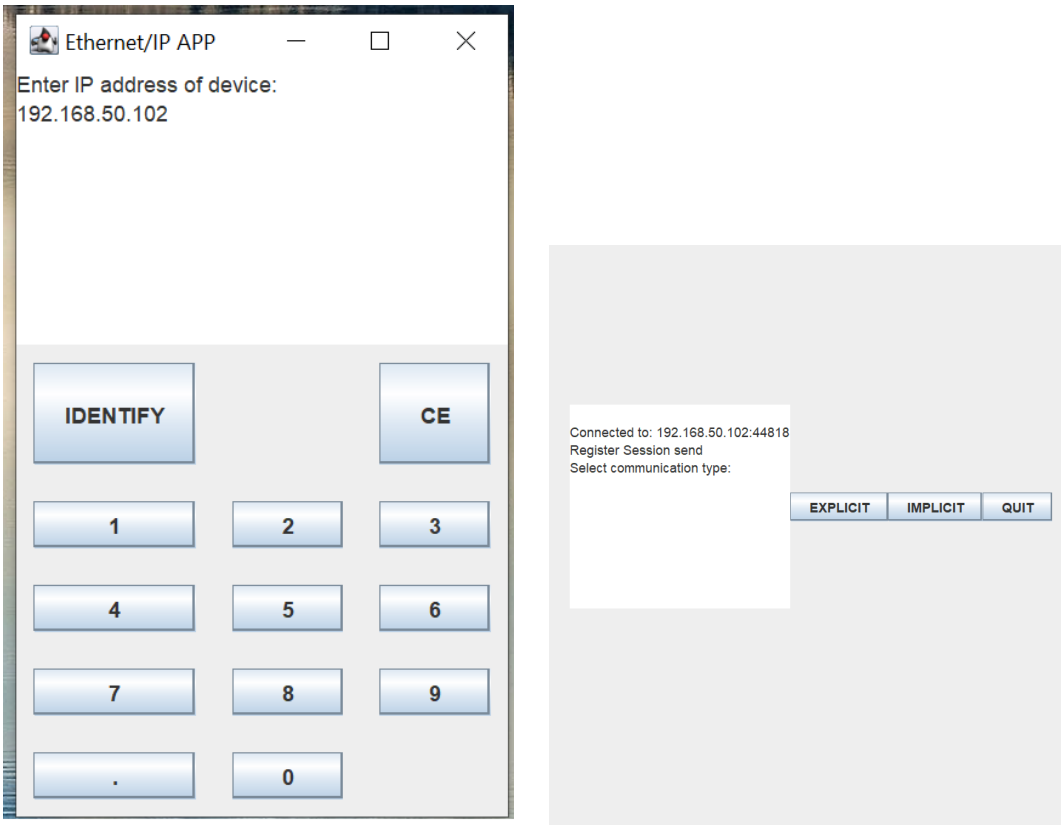
Klient ako prvý odosiela správu SYN. Sekvenčné číslo je nastavené na 0. Jedná sa teda o aktívne nadviazanie spojenia. Server (koncové zariadenie) odpovedá správou s príznakmi SYN a ACK. Paket obsahuje sekvenčné číslo servera (seq:0) a potvrdenie prijatia SYN paketu od klienta (ack:1). To signalizuje súhlas s vytvorením spojenia. Klient odpovedá správou ACK čím potvrdzuje prijatie správy SYN+ACK od serveru (číselné hodnoty sú seq:1, ack:1). Týmto je spojenie nadviazané a komunikácia môže začať. V aplikácii je vytvorenie TCP spojenia zaistené prostredníctvom metódy `TCP_establish()`.



Obr. 6.3: Nadviazanie TCP spojenia - three way handshaking [8]

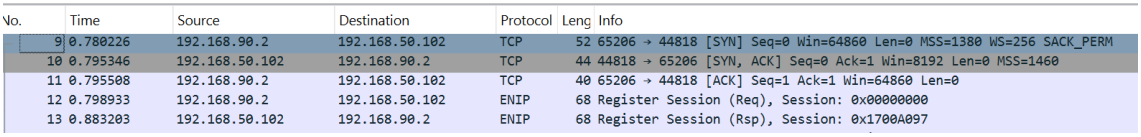
6.5.2 Register Session

Následuje nadviazanie spojenia na úrovni EtherNet/IP. K tomu je potrebné vytvoriť reláciu, čo sa vykonáva prostredníctvom Register Session správy. Ako prvý posiela klient požiadavku Register Session. Dĺžka správy musí byť nastavená na 4B, identifikátor relácie musí byť 0. Server v prípade validnej správy požiadavky v odpovedi poskytne konkrétny identifikátor relácie. Ten sa bude používať vo všetkých nasledujúcich správach až do ukončenia relácie. V aplikácii je táto časť riešená metódou `Register_Session_set()`, ktorej úlohou je zostavenie paketu požiadavky v korektnom formáte, jeho odoslanie a extrahovanie identifikátora relácie z odpovede zo servera. Po úspešnom vytvorení relácie sa aktualizujú GUI tlačidlá a užívateľ je vyzvaný na voľbu typu komunikácie.



(a) Vstup aplikácie (b) Výstup po zadaní IP adresy

Obr. 6.4: Zobrazenie úspešneho Register Session



No.	Time	Source	Destination	Protocol	Leng	Info
9	0.780226	192.168.90.2	192.168.50.102	TCP	52	65206 → 44818 [SYN] Seq=0 Win=64860 Len=0 MSS=1380 WS=256 SACK_PERM
10	0.795346	192.168.50.102	192.168.90.2	TCP	44	44818 → 65206 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460
11	0.795508	192.168.90.2	192.168.50.102	TCP	40	65206 → 44818 [ACK] Seq=1 Ack=1 Win=64860 Len=0
12	0.798933	192.168.90.2	192.168.50.102	ENIP	68	Register Session (Req), Session: 0x00000000
13	0.883203	192.168.50.102	192.168.90.2	ENIP	68	Register Session (Rsp), Session: 0x1700A097

Obr. 6.5: Zachytenie nadviazania spojenia vo Wiresharku

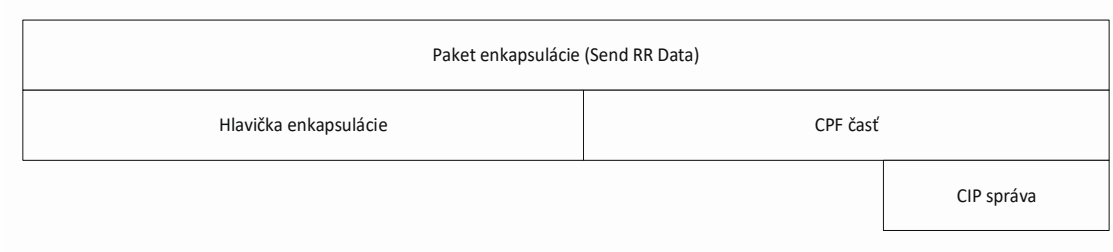
6.5.3 Priebeh EtherNet/IP komunikácie

V tomto stave má užívateľ tri možnosti:

- EXPLICIT – po stlačení má užívateľ možnosť výberu konkrétnej služby explicitnej komunikácie.
- IMPLICIT – umožnuje nastavenie parametrov potrebných pre implicitnú komunikáciu a spustenie Forward Open, čím sa aktivuje I/O prenos.
- QUIT – ukončenie komunikácie a aplikácie samotnej.

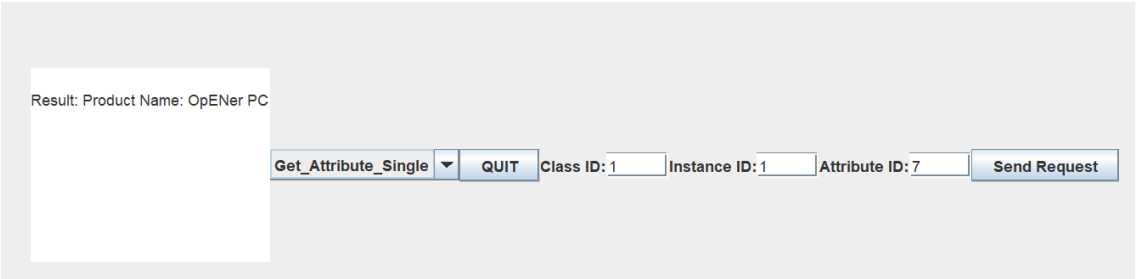
Priebeh explicitnej komunikácie

V rámci explicitnej komunikácie boli implementované všeobecné služby vysvetlené v kapitole 5.4 Každá z týchto služieb vyžaduje zadanie určitých parametrov – identifikátorov triedy, inštancie a atribútu. O tom, ktoré parametre treba zadať, je užívateľ informovaný v GUI. Pri výbere konkrétnej služby sú v GUI zobrazené potrebné tlačidlá a informácie na zadanie služby. To zabezpečuje metóda `useServiceSelect()`. Po zadaní potrebných parametrov sa volá metóda `useSendRequest()`, ktorá spracuje vstupné hodnoty a poskytne ich v potrebnej forme do `SendRRData_set()`, ktorá je hlavným uzlom explicitnej komunikácie. Je v nej zostavený paket správy požiadavky SendRRData, obsahujúci segment s volanou CIP službou. Jednotlivé CIP služby sú spracované prostredníctvom dvoch typov metód. Po spracovaní CIP segmentu je pridelená časť s CPF segmentom a vzniká tak zapúzdrený Ethernet/IP paket, volajúci konkrétny CIP objekt zariadenia. Paket je následne odoslaný. Po prijatí odpovede metóda volá na základe zvolenej služby metódu CIP služby, ktorá spracuje dáta poskytnuté zo zariadenia. Po výpise/zápise hodnoty sa cyklus opakuje - užívateľ môže opäť zvoliť z vybraných služieb, prípadne ukončiť komunikáciu tlačidlom QUIT.

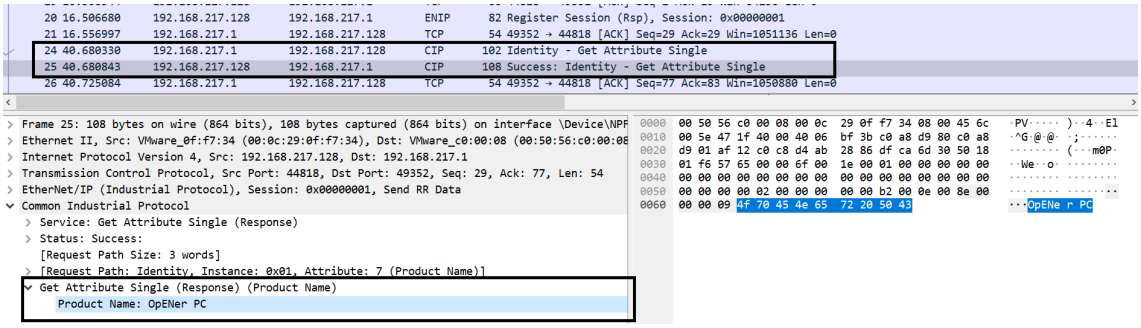


Obr. 6.6: Približná štruktúra vytvoreného Ethernet/IP paketu

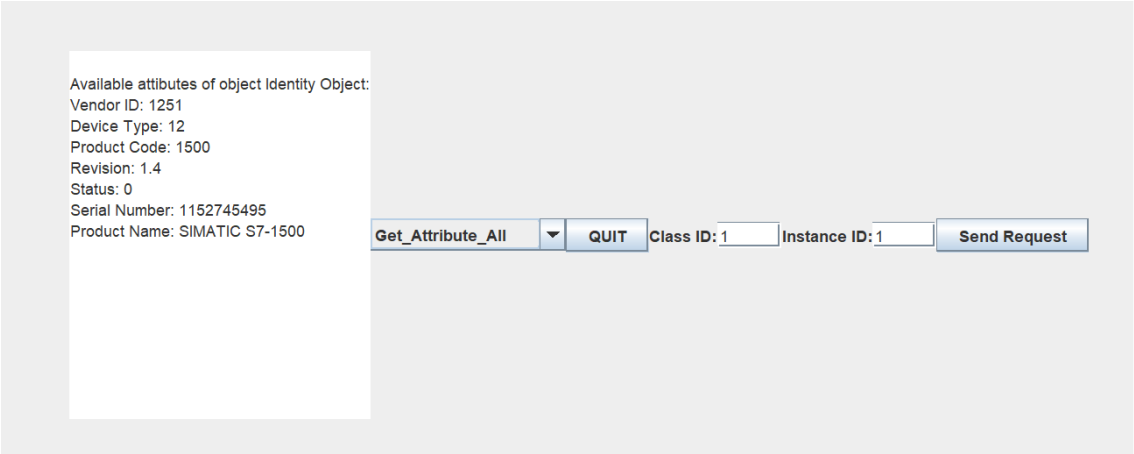
Následujúce obrázky ilustrujú výstupy aplikácie po volaní explicitných služieb a výmenu správ medzi klientom a serverom zobrazenú vo Wiresharku. Vyznačené oblasti na obrázkoch poukazujú na úspešný priebeh služieb.



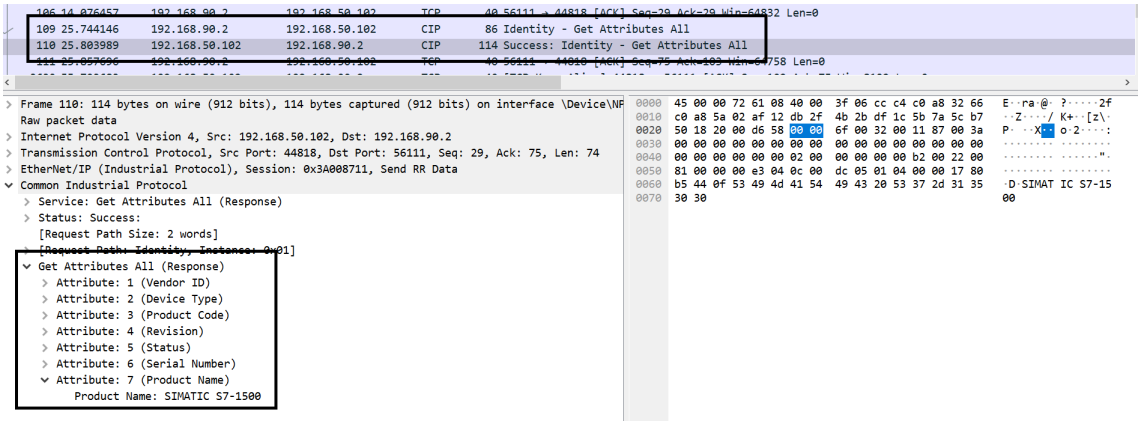
Obr. 6.7: Úspešné volanie služby Get_Attribute_Single



Obr. 6.8: Zobrazenie služby Get_Attribute_Single vo Wiresharku



Obr. 6.9: Úspešné volanie služby Get_Attributes_All



Obr. 6.10: Zobrazenie služby Get_Attribute_Al vo Wiresharku

Priebeh implicitnej komunikácie

Po zvolení implicitnej komunikácie zo strany klienta sa v GUI zobrazí skupina parametrov týkajúcich sa I/O spojenia. Tieto parametre – napríklad identifikátory vstupného, výstupného a konfiguračného bodu spojenia, sú predvyplnené hodnotami, potrebnými pre validné nadviazanie I/O spojenia s knižnicou CIPster, ale je možné ich upraviť v prípade potreby. Po potvrdení nastavenia stlačením tlačidla Forward Open sa začína proces vytvárania implicitného spojenia. Ešte pred odoslaním požiadavky na Forward Open sa na strane klienta inicializuje UDP soket, ktorý slúži na príjem dát zo servera. Tento krok je potrebný, pretože po úspešnej odpovedi na Forward Open môže začať server okamžite vysielať I/O dáta prostredníctvom UDP portu 2222 a ak by nebol soket už otvorený, tieto pakety by boli stratené.

V ďalšom kroku je zostavená Forward Open požiadavka obsahujúca potrebné informácie týkajúce sa vytváraného spojenia. Po jej spracovaní server nastaví ID

spojenia, ktoré aktivuje a začne vysielať UDP datagramy na port 2222.

Na strane klienta beží paralelne listener, ktorý prijíma a spracováva UDP pakety v reálnom čase.

Prerušenie implicitného spojenia je možné tlačidlom Forward Close, ktoré vyvolá odoslanie príslušnej požiadavky na ukončenie spojenia. Server následne deaktivuje spojenie a zastaví prenos UDP paketov. Zároveň aplikácia deaktivuje svoj soket. Neukončí sa však celá komunikácia – klient môže znovu iniciovať spojenie opätovným stlačením Forward Open. Následujúci obrázok ilustruje priebeh celej komunikácie:

6	0.004426	192.168.217.128	192.168.217.1	ENIP	82 Register Session (Rsp), Session: 0x00000008
7	0.055545	192.168.217.1	192.168.217.128	TCP	54 50848 → 44818 [ACK] Seq=29 Ack=29 Win=1051136 Len=0
8	2.291467	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0x00000000, SEQ=0000000000
9	2.384789	192.168.217.1	192.168.217.128	CIP CM	144 Connection Manager - Forward Open
10	2.385851	192.168.217.128	192.168.217.1	CIP CM	124 Success: Connection Manager - Forward Open
11	2.389596	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000001, T->O
12	2.431599	192.168.217.1	192.168.217.128	TCP	54 50848 → 44818 [ACK] Seq=119 Ack=99 Win=1050880 Len=0
13	2.802666	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000001, O->T
14	2.889554	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000002, T->O
15	3.299434	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000002, O->T
16	3.390927	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000003, T->O
17	3.797597	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000003, O->T
18	3.889842	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000004, T->O
19	4.293711	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000004, O->T
20	4.390180	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000005, T->O
21	4.795701	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000005, O->T
22	4.889451	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000006, T->O
23	5.292079	192.168.217.1	192.168.217.128	CIP I/O	66 Connection: ID=0xC0DE001F, SEQ=0000000006, O->T
24	5.389095	192.168.217.128	192.168.217.1	CIP I/O	190 Connection: ID=0xBE3F0004, SEQ=0000000007, T->O
25	5.539425	192.168.217.1	192.168.217.128	CIP CM	120 Connection Manager - Forward Close
26	5.541185	192.168.217.128	192.168.217.1	CIP CM	108 Success: Connection Manager - Forward Close
27	5.586150	192.168.217.1	192.168.217.128	TCP	54 50848 → 44818 [ACK] Seq=185 Ack=153 Win=1050880 Len=0
28	8.106449	192.168.217.1	192.168.217.128	ENIP	78 Unregister Session (Req), Session: 0x00000008

Obr. 6.11: Priebeh I/O prenosu

6.5.4 Unregister Session

Po stlačení tlačidla QUIT je vyvolaná metóda `Unregister_Session_set()` ktorá zostaví zapúzdrený Ethernet/IP paket typu Unregister Session požiadavka, a jeho odoslanie na server. Týmto sa ukončí EtherNet/IP relácia. Server ako odpoveď iniciuje ukončenie TCP spojenia poslaním paketu s príznakmi FIN, ACK. Následne klient potvrdí žiadosť poslaním paketu ACK a napokon sám posiela paket FIN, ACK. Server posiela potvrdenie ACK. Týmto je TCP spojenie a celá komunikácia korektne ukončená.

192.168.217.1	192.168.217.128	ENIP	78 Unregister Session (Req), Session: 0x00000009
192.168.217.128	192.168.217.1	TCP	60 44818 → 50646 [FIN, ACK] Seq=29 Ack=53 Win=64256 Len=0
192.168.217.1	192.168.217.128	TCP	54 50646 → 44818 [ACK] Seq=53 Ack=30 Win=1051136 Len=0
192.168.217.1	192.168.217.128	TCP	54 50646 → 44818 [FIN, ACK] Seq=53 Ack=30 Win=1051136 Len=0
192.168.217.128	192.168.217.1	TCP	60 44818 → 50646 [ACK] Seq=30 Ack=54 Win=64256 Len=0

Obr. 6.12: Ukončenie relácie a TCP spojenia

6.6 Integrácia do systému OpenMUC

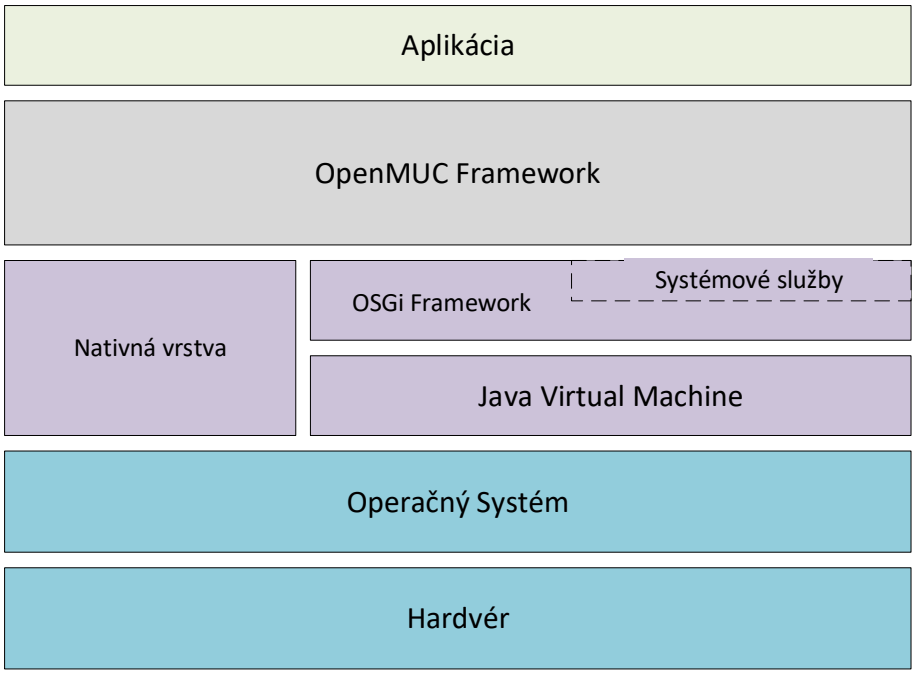
V druhej časti práce bola vytvorená aplikácia integrovaná do OpenMUC.

6.6.1 Štruktúra OpenMUC

OpenMUC je open-source framework založený na jazykoch Java a OSGi, určený pre vývoj aplikácií v oblasti monitorovania, zberu a spracovania priemyselných dát v priemyselných, energetických a IoT systémoch. OpenMUC poskytuje užívateľovi napríklad následujúce výhody [3]:

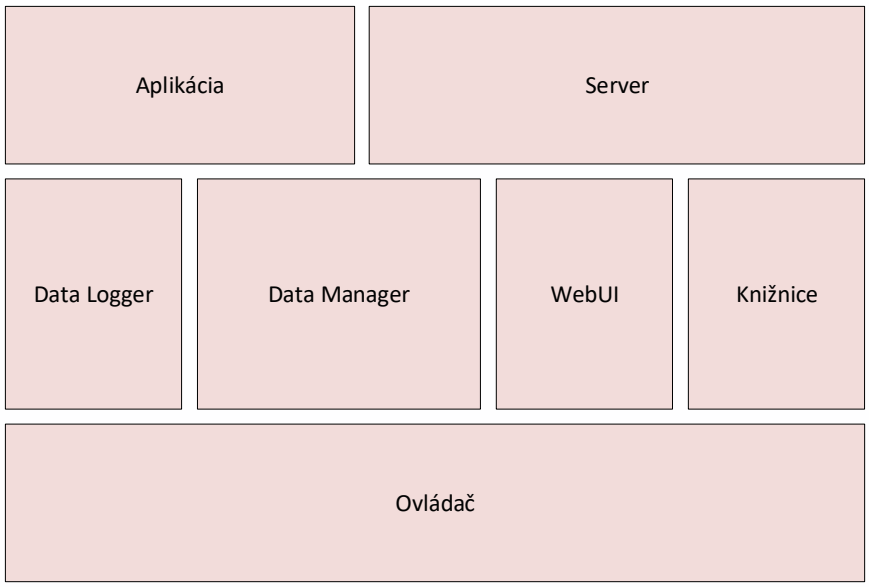
- **Jednoduchý vývoj aplikácií** – vývojár sa môže sústrediť na logiku aplikácie a nemusí riešiť detaily technológie.
- **Jednoduchá konfigurácia** – všetky komunikačné a dátové parametre možno dynamicky konfigurovať prostredníctvom centrálneho súboru, frameworku alebo webového rozhrania.
- **Komunikačná podpora** – podporuje množstvo známych komunikačných protokolov, ako napríklad SNMP alebo CSV, pričom nové ovládače protokolov je možné pridať pomocou rozhrania pluginu.
- **Záznam dát** – umožňuje zaznamenávanie dát v rôznych formátoch. Nové dátové záznamníky môžu byť pridané prostredníctvom pluginu.
- **Webové rozhranie** – pohodlné užívateľské rozhranie na konfiguráciu a vizualizáciu. Existujúce aplikácie WebUI:
  - nástroj prístupu ku kanálom,
  - konfigurátor kanálov,
  - exportér údajov,
  - prehliadač médií.

Framework OpenMUC beží v prostredí OSGi, ktoré je prevádzkované prostredníctvom virtuálneho stroja Java . Základný operačný systém a hardvér môže byť ľubovoľný, ak dokáže spustiť virtuálny stroj Java 8 [3].



Obr. 6.13: Softvérové vrstvy OpenMUC [3]

OpenMUC pozostáva z rôznych sofvérových modulov implementovaných ako OSGi balíky, ktoré bežia v prosredí OSGi. Následujúci obrázok znázorňuje moduly, ktoré tvoria OpenMUC:



Obr. 6.14: OpenMUC moduly [3]

Všetky moduly, okrem data managera sú voliteľné. Funkcionality jednotlivých modulov sú [3]:

- **Data Manager** – sprostredkúva interkaciu medzi aplikáciami, zariadeniami a záznamníkmi (loggermi).
- **Ovládač** – zabezpečuje výmenu dát so zariadením.
- **Data Logger** – ukladanie vzorových dát.
- **Aplikácia** – možnosť tvorby vlastnej aplikácie na spracovanie vzorkovaných dát a ovládanie zariadení.
- **Server** – prístup k dátam a konfigurácii cez REST API pre vzdialené aplikácie.
- **Web UI** – konfigurácia, vizualizácia vzorkovaných dát a export záznamov.
- **Knižnice** – poskytujú pomocné triedy pre viaceré časti frameworku.

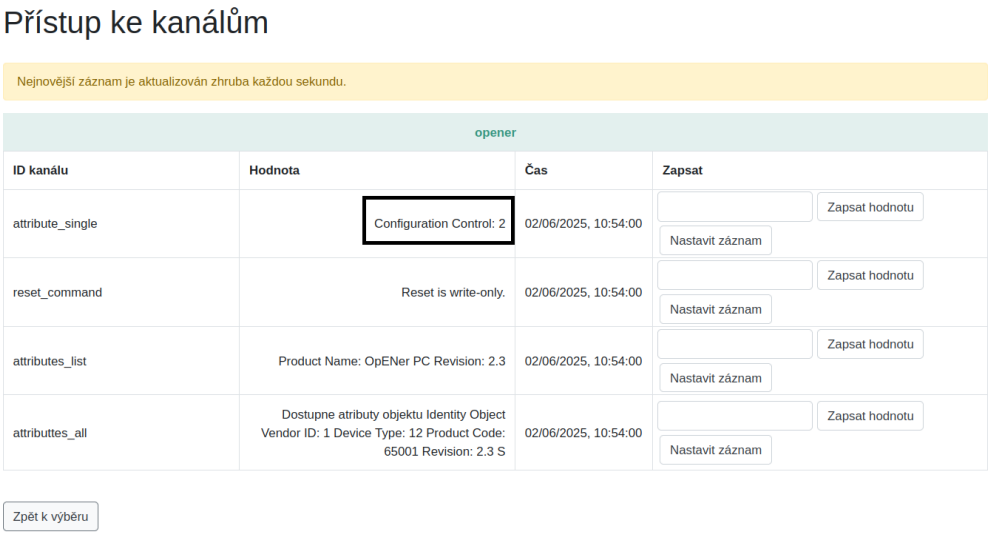
6.6.2 Registrácia ovládača

Vývoj ovládača pre EtherNet/IP (enip) prebiehal vo virtuálnom stroji s operačným systémom Linux Debian a predinštalovanými potrebnými komponentmi. Logika samotného protokolu bola prevzatá z už vytvorenej GUI aplikácie a prispôsobená na použitie v systéme OpenMUC. V rámci implementácie boli doplnené nasledujúce Java triedy:

- `EnipChannelAddress` – nastavenie adresy kanála ovladača,
- `EnipConnection` – hlavná komunikačná trieda, zabezpečuje vytvorenie spojenia TCP, EtherNet/IP relácie, obsluhu komunikácie a jej ukončenie,
- `EnipDeviceAddress` – nastavenie adresy koncového zariadenia,
- `EnipDeviceSettings` – nastavenie parametrov pre komunikáciu so zariadením,
- `EnipDeviceScanSettings` – obmedzenie skenovnania zariadenia
- `EnipDriver` – definuje ovládač enip,
- `MessageHandler` – zdrojový kód triedy `Main` z pôvodnej GUI aplikácie, upravený pre potreby ovládača.

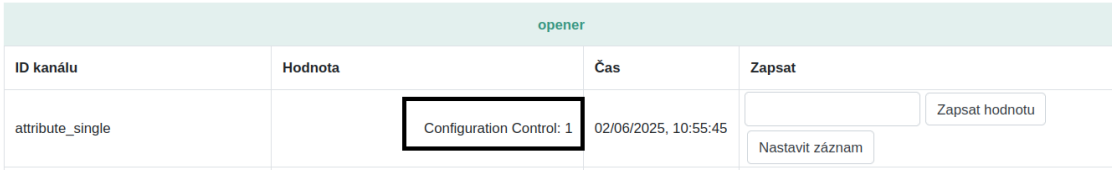
6.6.3 Implementácia do OpenMUC

Po implementovaní potrebných súborov a spustení OpenMUC sa zaregistrujú dostupné ovládače, medzi ktorými bude v prípade úspešného predošlého buildu aj novo vytvorený ovládač enip. Po prihlásení do Web UI v prehliadači sa zobrazia dostupné zariadenia a ich ovládače. Dostupné kanály a ich výstupy možno zobraziť pomocou nástroja prístupu ku kanálom. Pre ovládač enip boli vytvorené 4 kanály - každý pre jednu z dostupných implementovaných explicitných služieb. Služba Set_Attribute_Single je však implementovaná tak, že sprostredkúva zápis hodnoty do kanálu, ktorý zobrazuje jeden konkrétny atribút (vykonáva službu Get_Attribute_Single). Následujúce obrázky ilustrujú funkcionalitu jednotlivých služieb a priebeh služby Set_Attribute_Single [3]:



Obr. 6.15: Zobraznie výstupov kanálov

Obr. 6.16: Zápis novej hodnoty atribútu



Obr. 6.17: Zobrazenie novej hodnoty atribútu

Zápis novej hodnoty teda prebehol úspešne. Je však nutné novú hodnotu zapísať dvakrát za sebou, aby došlo k aktualizácii. Kanál attributes_all však nezobrazuje kompletný zoznam atribútov, ale len časť z nich. To je pravdepodobne spôsobené obmedzenou veľkosťou STRING v rámci Web UI. Pri reprezentácii hodnoty kanálu pomocou BYTE_ARRAY sa však zobrazí kompletný zoznam.

Záver

Cieľom bakalárskej práce bol detailný teoretický rozbor štruktúry protokolu Ethernet/IP a popis jeho histórie. Praktickým výstupom bola realizácia samostatnej Java aplikácie, ktorá umožňuje komunikáciu prostredníctvom tohto protokolu a následne jej integrácia do systému OpenMUC.

V teoretickej časti práce boli postupne vysvetlené všetky kľúčové aspekty protokolu EtherNet/IP – od fyzickej vrstvy (typy kabeláže), až po popis implementácie CIP objektov a ich implementáciu v prostredí priemyselných sietí.

V praktickej časti bola vytvorená aplikácia klienta, ktorá dokáže korektne nadviazať TCP spojenie so zariadením dostupným na zadanej IP adrese, iniciovať EtherNet/IP komunikáciu poslaním požiadavky Register Session a vykonať sériu CIP služieb. Podporované sú služby na čítanie hodnôt atribútov (Get_Attribute_Single, Get_Attributes_All a Get_Attribute_List) a služba Set_Attribute_Single na zápis novej hodnoty atribútu. V rámci realizácie implicitnej komunikácie sú dostupné služby Forward_Open a Forward_Close. Aplikácia zároveň interpretuje chybové stavy v prípade negatívnej odpovede zo zariadenia a umožňuje korektné ukončenie komunikácie pomocou služby Unregister Session.

Pri testovaní na reálnom zariadení SIMATIC S7-1500 vznikali nekompatibility pri niektorých CIP správach. To mohlo byť spôsobené tým, že zariadenie nie je primárne určené pre EtherNet/IP siete, aj keď protokol podporuje.

Implementácia do prostredia OpenMUC bola realizovaná registráciou ovládača, ktorý umožňuje čítanie a zápis hodnôt atribútov pomocou poskytnutých kanálov. Pri zápise hodnoty do kanála je nutné novú hodnotu zadať dvakrát, pretože pri prvom pokuse dochádza k „zamrznutiu“ kanálov. Hodnota je však v prípade, že sa jedná o nastaviteľný atribút a správny kanál úspešne zapísaná.

Zmyslom tejto práce bolo vytvoriť základnú platformu klienta na komunikáciu v sieťach založených na protokole EtherNet/IP, ktorá slúži ako východisko pre budúce rozšírenie o ďalšie CIP služby a podporu nových typov objektov.

Literatúra

[1] *Common Industrial Protocol (CIP™)*. 3.3. Open DeviceNet Vendor Association, Inc, 2007. Dostupné z: https://www.odva.org/subscriptions-services/specifications/.

[2] *GCABLING ELECTRONIC*. 2023. Dostupné z: https://www.gcabling.com/what-is-twisted-pair-cable-and-types/.

[3] *OpenMUC*. C2025. Dostupné z: https://www.openmuc.org/.

[4] Balchunas, A. *Cisco CCNA Study Guide*. v2.71. c2014. Dostupné z: https://www.routeralley.com/completed/ccna_studyguide.pdf.

[5] Bosma, A. *S-FTP twisted pair cable shielding*. November 2021. Retrieved from Wikimedia Commons. Dostupné z: https://commons.wikimedia.org/wiki/File:S-FTP_twisted_pair_cable_shielding.svg.

[6] Bosma, A. *U-UTP twisted pair cable shielding*. November 2021. Retrieved from Wikimedia Commons. Dostupné z: https://commons.wikimedia.org/wiki/File:U-UTP_twisted_pair_cable_shielding.svg.

[7] Brooks, P. Ethernet/IP-industrial protocol. In: *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*. 2001, sv. 2, s. 505–514 vol.2. DOI: 10.1109/ETFA.2001.997725.

[8] Forouzan, B. A. *TCP/IP protocol suite*. 4th ed. Boston: McGraw-Hill Higher Education, 2010. ISBN 978-0-07-337604-2.

[9] Galloway, B. a Hancke, G. P. Introduction to Industrial Control Networks. *IEEE Communications Surveys & Tutorials*. 2013, sv. 15, č. 2, s. 860–880. DOI: 10.1109/SURV.2012.071812.00124.

[10] HUAWEI. *Ethernet II Frame*. C2024. Dostupné z: https://support.huawei.com/enterprise/en/doc/EDOC1100174721/ea0a043c/ethernet-ii-frame.

[11] IBM. *Transmission Control Protocol/Intertnet Protocol*. C2020. Dostupné z: https://www.ibm.com/docs/en/aix/7.2?topic=management-transmission-control-protocolinternet-protocol.

[12] IEEE. IEEE Standard for Ethernet. *IEEE Std 802.3-2022 (Revision of IEEE Std 802.3-2018)*. 2022, s. 1–7025. DOI: 10.1109/IEEESTD.2022.9844436.

[13] Kurose, J. F. a Ross, K. W. *Computer networking: a top-down approach*. 5th ed. New York: Addison-Wesley, 2010. ISBN 978-0-13-607967-5.

[14] Mikac, E. *CBT Nuggets*. 2023. Dostupné z: https://www.cbtnuggets.com/blog/technology/networking/single-mode-vs-multimode-fiber-what-are-the-differences.

[15] Mikac, E. *CBT Nuggets*. 2023. Dostupné z: https://www.cbtnuggets.com/blog/communication/half-duplex-vs-full-duplex.

[16] Modicon, I. *Modicon Modbus Protocol Reference Guide: PI-MBUS-300*. Rev.J. c1996. Dostupné z: https://modbus.org/docs/PI_MBUS_300.pdf.

[17] Motoyoshi, S. PROFIBUS and PROFInet technology and standardization activity. In: *Proceedings of the 41st SICE Annual Conference. SICE 2002.* 2002, sv. 2, s. 921–924 vol.2. DOI: 10.1109/SICE.2002.1195287.

[18] ODVA. *EtherNet/IP Adaptation of CIP*. 1.4. Open DeviceNet Vendor Association, Inc, 2007. Dostupné z: https://www.odva.org/subscriptions-services/specifications/.

[19] ODVA. *Network Infrastructure for Ethernet/IP: Introduction and Considerations*. R0. 2007. Dostupné z: https://www.odva.org/wp-content/uploads/2020/05/PUB00035R0_Infrastructure_Guide.pdf.

[20] ODVA. *CIP on Ethernet Technology*. R8. 2024. Dostupné z: https://www.odva.org/wp-content/uploads/2024/04/PUB00138R8_Ethernet.pdf.

[21] ODVA. *Quick Start for Vendors Handbook: A Guide for EtherNet/IP Developers*. Prvé vydanie. c2008. Dostupné z: https://www.odva.org/wp-content/uploads/2020/05/PUB00213R0_EtherNetIP_Developers_Guide.pdf.

[22] ODVA. *CIP on CTDMA Technology*. 1. vyd. c2016. Dostupné z: https://www.odva.org/wp-content/uploads/2020/05/PUB00200R1-Tech-Series-ControlNet.pdf.

[23] Peterka, J. *Ethernet II vs. IEEE 802.3*. 4.5.1999. Dostupné z: https://www.earchiv.cz/anovinky/ai2058.php3.

[24] S, R. J. a S, M. P. *Industrial Ethernet - How to Plan, Install, and Maintain TCP/IP Ethernet Networks: The Basic Reference Guide for Automation and Process Control Engineers (2nd Edition)*. International Society of Automation (ISA), 2004. ISBN 1556178697.

[25] Schiffer, V., Vandesteeg, K., Vasko, D. a Lenner, J. Introduction to DeviceNet safety. In: *2000 IEEE International Workshop on Factory Communication Systems. Proceedings (Cat. No.00TH8531)*. 2000, s. 293–300. DOI: 10.1109/WFCS.2000.882561.

[26] Schiffer, V., VanGompel, D. J., Romito, R. A. a Voss, K. *The Common Industrial Protocol (CIP™) and the Family of CIP Networks*. 1. vyd. ODVA, Inc., 2016. Dostupné z: https://www.odva.org/wp-content/uploads/2020/06/PUB00123R1_Common-Industrial_Protocol_and_Family_of_CIP_Networks.pdf.

[27] s.r.o., S. electronic. *UTP, FTP, S/FTP? Malé zopakovanie isto nezaškodí*. 2019. Dostupné z: https://svetelektro.com/clanky/utp-ftp-sftp-male-zopakovanie-isto-nezaskodi-962/print/.

Zoznam symbolov a skratiek

IEEE	Institute of Electrical and Electronics Engineers
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
MAC	Media Access Control
EMI	Electromagnetic Interference
FTP	Foiled Twisted Pair
S/FTP	Shielded /Foiled Twisted Pair
F/FTP	Foiled /Foiled Twisted Pair
UTP	Unshielded Twisted Pair
SFD	Start Frame Delimiter
CRC	Cycle Redundancy Check
OSI	Open Systems Interconnections
CIP	Common Industrial Protocol
CPF	Common Packet Format
UCMM	Unconnected Message Manager
I/O	Input/Output
ODVA	Open DeviceNet Vendors Association
RPI	Requested Packet Interval
GUI	Graphical User Interface
SNMP	Simple Network Manager Protocol
CSV	Comma-Separated Values
OSGi	Open Service Gateway initiative

Zoznam príloh

A Obsah elektronickej prílohy

Príloha obsahuje archív **BP.zip** v ktorom je kompletný projekt Java aplikácie vytvorený vo vývojovom prostredí IntelliJ IDEA 2024 2.3. Verzia Javy používaná pri vývoji bola Java 23.

Aplikácia sa spúšťa pomocou tlačidla „Run“ alebo stlačením `Ctrl+F5` v súbore `Main.java`. V prípade iného vývojového prostredia je potrebné do vytvoreného projektu zakomponovať všetky zdrojové súbory z podadresára `/src`. Následný stromový graf zobrazuje adresárovú štruktúru projektu a cestu k hlavným zdrojovým súborom:

```
Bachelor_Thesis/ ..... Hlavný adresár (názov projektu)
├── .idea ..... konfiguračné súbory prostredia IDEA
├── out ..... výstupné a konfiguračné súbory prostredia IDEA
├── src ..... zdrojové súbory aplikácie
│   ├── Main.java
│   ├── ENIP_form.java
│   ├── Encapsulation_packet.java
│   ├── Data_segment_parameters.java
│   ├── ENIP_form.form
│   ├── CIP_attribute_format.java
│   ├── CIP_segment_parameters.java
│   ├── CIP_object_format.java
│   └── CIP_object_library.java
├── Bachelor_Thesis.iml
└── .gitignore
```

B Obsah elektronickej prílohy

Ako doplnok bude priložený adresár s názvom **simulacie**, v ktorom sa nachádzajú súbory knižníc OpENer a CIPster. Tieto knižnice je možné stiahnuť aj priamo zo stránok, uvedených v odkaze. V archíve je však aplikácia už pripravená (build je už hotový) na použitie. V súbore readmeme.txt priloženom v archíve je popísaný presný postup a požiadavky na spustenie.

```
simulacie
├── CIPster ..... súbory knižnice CIPster
├── OpENer ..... súbory knižnice OpENer
└── readme.txt ..... návod na spustenie simulácii
```

C Obsah elektronickej prílohy

Ďalším doplnkom sú zdrojové súbory implementácie v OpenMUC.

```
OpenMUC_files
├── MessageHandler
├── EnipConnection
├── CIP_object_library
├── Encapsulation_packet
├── EnipChannelAddress
├── EnipDriver
├── EnipDeviceAddress
├── EnipDeviceSettings
├── CIP_attribute_format
├── CIP_object_format
├── CIP_segment_parameters
└── EnipDeviceScanSettings
```