VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

SNMP2OTEL — EXPORTÉR SNMP GAUGE METRIK DO OPENTELEMETRY (OTEL)

SNMP2OTEL — SNMP GAUGE METRICS EXPORTER TO OPENTELEMETRY (OTEL)

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ANDREJ BOČKAJ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN PLUSKAL, Ph.D.**
SUPERVISOR

BRNO 2025

Obsah

Seznam obrázků

Kapitola 1

Úvod

Cieľom tohto projektu je implementovať program `snmp2otel`, ktorý bude v pravidelných intervaloch dotazovať zadaný SNMP agent na vybrané OID a namerané hodnoty exportovať ako OTEL Metrics pomocou rozhrania OTLP/HTTP (JSON) na špecifikovaný OTEL endpoint (typicky OpenTelemetry Collector). Program musí podporovať iba metriky typu Gauge.

1.0.1 Obmedzenia

- Povolené sú ľubovoľné knižnice.
- Inštalácia knižníc do systému nie je povolená; projekt sa musí preložiť štandardne pomocou `Makefile`.
- Program pobeží bez root oprávnení; nesmie vyžadovať privilegované porty pre lokálne naslúchanie.

1.0.2 Podporované protokoly a dátové typy

- **SNMP v2c**: Podporujte iba operáciu Get pre skalárne OID zakončené `.0`.
- **ASN.1 BER**: Implementujte minimum potrebné pre SNMPv2c Get/Response (INTEGER, OCTET STRING, OID, SEQUENCE).
- **OTEL export**: Dáta sa posielajú pomocou OTLP/HTTP (JSON). Každý nameraný údaj je reprezentovaný ako Gauge.

1.0.3 Výstup aplikácie

- V základnom režime program nič nevypisuje.
- Vo verbose režime (`-v`) vypisujte ladiace informácie (dotazy, odpovede, export).
- Chybové hlášky vypisujte na štandardný chybový výstup.

1.1 Protokol SNMPv2: Popis Funkčnosti

Protokol SNMPv2 (Simple Network Management Protocol verzia 2) je rozšírená a vylepšená verzia štandardného protokolu pre správu a monitorovanie sieťových zariadení, ako sú sme-

rovače, prepínače, servery a tlačiarne. Umožňuje centrálnemu manažérovi (NMS - Network Management Station) získavať informácie a meniť konfiguráciu spravovaných zariadení.

1.1.1 Kľúčové Komponenty

SNMP model pozostáva z troch základných komponentov:

- **Správca Siete (Network Management Station - NMS)**: Centrálna stanica, ktorá spúšťa aplikácie pre správu. Odošle požiadavky a prijíma odpovede a nežiaduce notifikácie (trapy).
- **Spravované Zariadenie (Managed Device)**: Sieťový komponent (napr. router, switch), ktorý obsahuje dáta, ktoré sa majú spravovať.
- **Agent**: Softvérový modul bežiaci na spravovanom zariadení. Zhromažďuje informácie o zariadení, spracováva požiadavky od NMS a posiela odpovede/notifikácie.

1.1.2 Dátová Štruktúra

- **MIB (Management Information Base)**: Je to hierarchická databáza (virtuálna informačná štruktúra) na spravovanom zariadení, ktorá definuje premenné (objekty), ktoré môže NMS spravovať. Každá premenná má jedinečný OID (Object Identifier).
- **OID (Object Identifier)**: Unikátny identifikátor v stromovej štruktúre, ktorý presne odkazuje na konkrétnu premennú v MIB (napr. počet chybových paketov na danom porte).

1.1.3 Komunikácia a Správy

SNMPv2 primárne využíva protokol UDP (User Datagram Protocol) na portoch 161 (požiadavky/odpovede) a 162 (notifikácie/trapy) pre nespoľahlivú a rýchlu komunikáciu.

A. Základné Typy PDU (Protocol Data Unit - Typy Správ)

Tabulka 1.1: Typy SNMPv2 PDU správ

Type	From	To	Desc
GetRequest	NMS	Agent	Žiadosť o jednu alebo viac hodnôt premenných.
GetNextRequest	NMS	Agent	Žiadosť o hodnotu **nasledujúcej** premennej v MIB, umožňuje prechádzanie (walk) celej MIB.
GetBulkRequest	NMS	Agent	**Novinka v v2:** Efektívna žiadosť o prenos veľkého bloku dát.
SetRequest	NMS	Agent	Žiadosť o **zmenu** hodnôt premenných na zariadení (konfigurácia).
Response	Agent	NMS	Odpoveď na akúkoľvek Get alebo Set požiadavku, obsahuje požadované dáta alebo stav operácie.
Pokračovanie na ďalšej strane			

Tabulka 1.1 – pokračovanie od predchádzajúcej strany

Type	From	To	Desc
Trap	Agent	NMS	**Nežiaduce** asynchrónne upozornenie na **významnú udalosť** (napr. reštart zariadenia).
InformRequest	NMS	NMS	**Novinka v v2:** Notifikácia zaslaná medzi dvoma NMS; vyžaduje **potvrdenie** o prijatí.

Pokračovanie na ďalšej strane

B. Mechanizmus Komunít (Communities)

SNMPv2 používa "reťazce komunity"(Community Strings) ako formu základného hesla/autentifikácie.

- Agent prijme požiadavku len vtedy, ak sa reťazec komunity v správe zhoduje s nakonfigurovaným reťazcom na zariadení.

Existujú dva bežné typy:

- **read-only** (len na čítanie): Umožňuje len Get operácie.
- **read-write** (čítanie/zápis): Umožňuje Get aj Set operácie.

1.2 OpenTelemetry (OTEL) a HTTP/JSON Metriky

OpenTelemetry (OTEL) je open source projekt, ktorý poskytuje unifikovaný súbor nástrojov, API a SDK na inštrumentáciu, generovanie, zhromažďovanie a export telemetrických dát (logy, tracy a metriky).

Protokol pre prenos týchto dát sa nazýva OTLP (OpenTelemetry Protocol). Ten podporuje dva hlavné formáty kódovania/prenosu:

- **OTLP/gRPC** (preferovaný): Využíva binárny formát Protocol Buffers pre efektívny a rýchly prenos.
- **OTLP/HTTP/JSON** (vlastná požiadavka): Využíva štandardný protokol HTTP s dátovou záťažou (payload) kódovanou vo formáte JSON.

1.2.1 Čo je to OTEL/HTTP/JSON Metrics?

Ide o štandardizovaný formát a metódu prenosu metriky z aplikácie alebo služby do kolektora alebo monitorovacieho backendu. Používa sa HTTP/HTTPS pre prenos. **Štandardizovaný Formát**: Metriky sú štruktúrované podľa OTLP Data Modelu a odosielajú sa ako JSON objekt v tele HTTP požiadavky (zvyčajne POST).

Štruktúra Metriky

- **Názov** (Name), **Popis** (Description) a **Jednotku** (Unit).
- **Typ metriky** (napr. Counter, Gauge, Histogram).
- **Dátové body** (Data Points): Samotné merané hodnoty, ktoré zahŕňajú časovú značku (timestamp) a sadu Atribútov (Attributes).

- **Atribúty**: Pár kľúč-hodnota, ktoré poskytujú kontext (napr. `http.request.method: POST`, `status_code: 200`, `service.name: nazov-programu`).

1.2.2 Načo sa OTEL/HTTP/JSON Metriky Používajú?

Hlavným cieľom je dosiahnuť observability (pozorovateľnosť) aplikácií a infraštruktúry bez vendor lock-in. Jednotný Štandard: Poskytuje konzistentný spôsob, ako zbierať metriky, bez ohľadu na to, v akom jazyku bola aplikácia napísaná (Java, Python, .NET, Go, C++, atď.). **Agregácia a Analýza**: Metriky sú základom pre monitorovanie stavu, výkonu a dostupnosti dakeho programu (napr. latencia HTTP požiadaviek, počet chýb, využitie pamäte). Flexibilita Exportu: Umožňuje odosielať dáta do rôznych systémov (Prometheus, Grafana, Jaeger, komerčné backends) prostredníctvom jedného protokolu – OTLP.

1.3 Implementácia

Táto sekcia detailne popisuje architektúru programu a techniky implementácie použité pre zber metrík prostredníctvom protokolu SNMPv2c a ich následný export vo formáte OTEL/HTTP/JSON.

1.3.1 1. Konfigurácia a Dátové Štruktúry

Všetky vstupné a konfiguračné parametre programu sú centralizované v dátovej štruktúre `SnmpOtelConfig`. Táto štruktúra slúži ako jediný zdroj pravdy pre nastavenia súvisiace s cieľovými SNMP Agentmi, OID (Object Identifiers) a koncovými bodmi (endpoints) pre export OTEL.

Logika aplikácie je riešená v triede `SnmpOtel_BL` (Business Logic), čím je zabezpečená správna modularita a oddelenie obchodnej logiky od prezentačnej a sieťovej vrstvy.

1.3.2 2. Komunikácia s SNMP Agentmi (SNMPv2c)

Získavanie surových metrík zo sieťových zariadení je realizované prostredníctvom protokolu SNMPv2c.

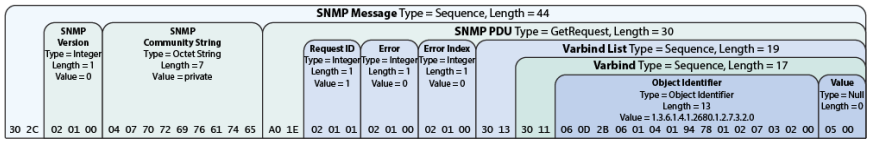
A. Generovanie a Kódovanie SNMPv2c Správ

Knižnica pre Dátové Transformácie: Pre spracovanie dátových štruktúr SNMP využívame knižnicu BasicSNMP (silderan/BasicSNMP). Knižnica je primárne zodpovedná za kódovanie a dekódovanie dát v súlade s oficiálnymi štandardmi protokolu.

Kódovanie ASN.1 BER: Knižnica implementuje algoritmus ASN.1 BER (Basic Encoding Rules). Tento proces je nevyhnutný pre transformáciu internej dátovej štruktúry požiadavky (ktorá zahŕňa verziu protokolu, reťazec komunity – Community String a požadované OID) do korektného binárneho formátu. Tento formát je následne pripravený na bezchybný prenos cez sieť.

B. Sieťový Prenos

Po zakódovaní správy knižnicou BasicSNMP prebieha sieťová komunikácia prostredníctvom vlastného UDP klienta, ktorý je implementovaný v triede `SnmpOtelClient`.



Obrázek 1.1: SNMPV2c protokol

Klient preberá SNMP zakódovanú požiadavku (GetRequest) a odosiela ju pomocou protokolu UDP štandardne na port 161 cieľového SNMP Agenta.

Po odoslaní čaká klient na prichádzajúcu UDP odpoveď, ktorá je následne odovzdaná späť knižnici BasicSNMP na dekódovanie, čím sa získajú finálne hodnoty metrík.

1.3.3 3. Transformácia a Export Metrík (OTEL/HTTP/JSON)

Po úspešnom zbere dát sú SNMP hodnoty transformované do štandardizovaného formátu OpenTelemetry (OTEL).

A. Generovanie Dátovej Štruktúry OTEL

Modelovanie Metrík: Pre zabezpečenie zhody s OTEL OTLP/JSON špecifikáciou je definovaná vlastná hierarchická štruktúra `OpenTelemetryMetrics`. Táto štruktúra vychádza zo vzorového dátového modelu definovaného v OpenTelemetry protokole, konkrétne zo špecifikácie:

https://github.com/open-telemetry/opentelemetry-proto/blob/main/examples/metrics.json

Konverzia na JSON: Transformácia dátovej štruktúry `OpenTelemetryMetrics` do finálneho JSON formátu je zabezpečená pomocou knižnice `nlohmann/json`. Špecificky, funkcia `SnmpOtelJson::CreateOtlpMetricsJson` vykonáva túto konverziu. Funkcia vracia výsledný JSON objekt ako `std::string`, typicky bez formátovania (bez odsadenia) pre optimalizáciu sieťového prenosu a jednoduché porovnanie dát.

B. Export Metrík

Export dát je riadený triedou `SnmpOtelExport`, ktorá využíva knižnicu `httplib.h` (od Yuji Hirose) pre robustnú a spoľahlivú HTTP/S komunikáciu.

Funkcia `SnmpOtelExport::SendHttpMessage` prevezme JSON `std::string` metrík.

Následne iniciuje HTTP POST požiadavku (s `Content-Type: application/json`) na nakonfigurovaný koncový bod OTEL Collectora alebo priameho monitorovacieho backendu. Týmto je zabezpečený finálny prenos spracovaných telemetrických dát.

1.3.4 4. Spracovanie Signálov

Pre riadne ukončenie programu a uvoľnenie zdrojov je implementovaný mechanizmus spracovania signálov. Trieda `SignalHandler` zabezpečuje registráciu prerušenia (napr. kombinácia kláves Ctrl+C). V prípade detekcie signálu je vyvolaná obslužná funkcia, ktorá iniciuje riadené ukončenie logiky programu, čím sa predchádza nekonzistentnému stavu alebo poškodeniu dát.

1.4 Použitie

Program sa spúšťa s nasledujúcimi parametrami:
`snmp2otel` -t target [-C community] -o oids_file -e endpoint [-i interval] [-r retries] [-T timeout] [-p port] [-v] [-O oids_formatingMappingFile]

1.4.1 Parametre

- `-t target` — IP adresa alebo DNS meno SNMP agenta.
- `-C community` — SNMP v2c community string. Výchozí: `public`.
- `-o oids_file` — soubor se seznamem OID, které se mají dotazovat.
- `-e endpoint` — URL OTEL endpointu pro export (OTLP/HTTP JSON), např. `http://localhost:4318/v1/metrics`.
- `-i interval` — perioda dotazování v sekundách (`> 0`). Výchozí: `10`.
- `-r retries` — počet retransmisí při timeoutu. Výchozí: `2`.
- `-T timeout` — timeout SNMP dotazu v ms. Výchozí: `1000`.
- `-p port` — UDP port SNMP agenta. Výchozí: `161`.
- `-v` — verbose režim.
- `-O oids_formatingMappingFile` — volitelný súbor pre mapovanie OID na názvy metrík a jednotky.

Formát souboru se seznamem OID

- Textový ASCII soubor: jedno OID na řádek, numerická forma. Prázdné řádky a řádky začínající `#` ignorujte.

 # System uptime
1.3.6.1.2.1.1.3.0

Formát mapping souboru (volitelný)

{
"1.3.6.1.2.1.1.3.0": { "name": "snmp.sysUpTime", "unit": "ms", "type": "gauge"}
}

- `type` může být pouze `gauge`. Pokud položka chybí, použije se název odvozený z OID.

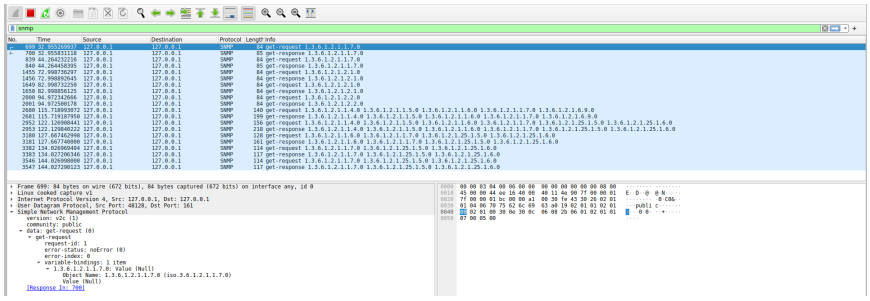
1.5 Testovanie

Táto sekcia popisuje metodológiu a technické nástroje, ktoré boli kľúčové pre komplexnú verifikáciu funkcionality SNMPv2c komunikácie a následnej konverzie dát do formátu OpenTelemetry (OTEL) JSON.

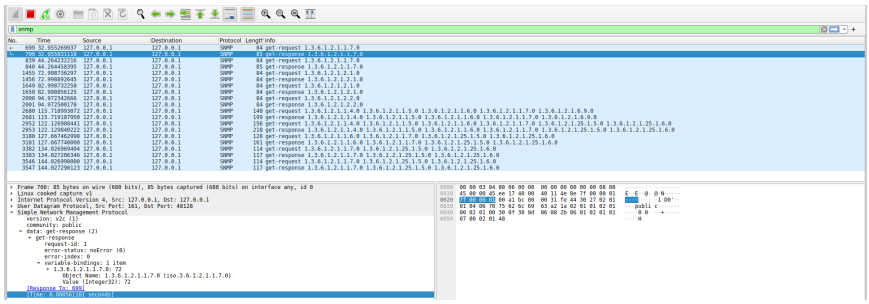
1.5.1 Verifikácia Protokolu SNMPv2c

Primárnym cieľom bolo zabezpečiť správnu implementáciu protokolu SNMPv2c a presné spracovanie dát.

- **Analýza Sieťového Prenosu (Wireshark):**
  - Monitorovaním sieťovej prevádzky sa verifikovalo, či sú požiadavky a odpovede SNMPv2c prenášané v korektnom formáte (UDP) a či je štruktúra PDU v súlade so špecifikáciou protokolu.
  - Testovanie funkčnosti SNMP sme kontrolovali len pomocou **Wireshark** (na začiatku a na konci vývoja), lebo vytváranie SNMP štruktúry zabezpečuje externá knižnica.
- **Referenčný SNMP Agent a Server:**
  - Pre izolačné ‘ bol vyvinutý referenčný SNMP Agent v **C++**, založený na rovnakej knižnici (`silderan/BasicSNMP`) ako testovaný kód.
  - Agent odosielal **konštantné metrické dáta** pre špecifickú sadu požadovaných OID.
  - Tento prístup umožnil dôkladne overiť **UDP komunikáciu** a správne spracovanie prijatých odpovedí testovaným programom.
- **Problémy s Externým Agentom**
  - Pre testovanie bola (vraj) k dispozícii adresa SNMP agenta `isa.fit.vutbr.cz`. Pokusy o komunikáciu s týmto serverom však boli neúspešné. Vzhľadom na to, že ide o školský server, nebolo možné vykonať hĺbkové ladenie (debugging) a identifikovať príčinu zlyhania komunikácie.
  - Pre úspešné testovanie bola preto použitá lokálna inštalácia SNMP agenta na testovacom stroji.
  - Lokálny agent bol nainštalovaný pomocou nasledujúceho príkazu:
    - sudo apt install snmpd snmp
  - Konfigurácia SNMP agenta bola vykonaná úpravou súboru `/etc/snmp/snmpd.conf` podľa inštrukcií zobrazených na sprievodnom obrázku:
    1. **Nastavenie IP adresy (`agentAddress`):** Pre lokálne testovanie bol agent nastavený na počúvanie na lokálnej adrese.
       ```
       agentAddress udp:127.0.0.1:161
       ```
    2. **Nastavenie Komunitného Reťazca (`rocommunity`):** Bol nastavený `read-only` prístup s predvoleným komunitným reťazcom `"public"`, aby bolo možné jednoduché testovanie s aplikáciami používajúcimi túto predvolenú hodnotu.
       ```
       rocommunity public default
       ```
  - Nasledujúce obrázky dokumentujú úspešnú funkčnosť implementácie s lokálnym SNMP agentom a OTEL.
- **OTEL export testovanie**



Obrázek 1.2: Snmp request to local snmpAgent



Obrázek 1.3: Snmp response from local snmpAgent

– Ako automatizované testy sme používali jeden **unit test**, ktorý kontroloval správnu štruktúru JSON formátu pre OTEL export. Dá sa spustiť po zadaní príkazu `make test` alebo ako celok pre `make all`.

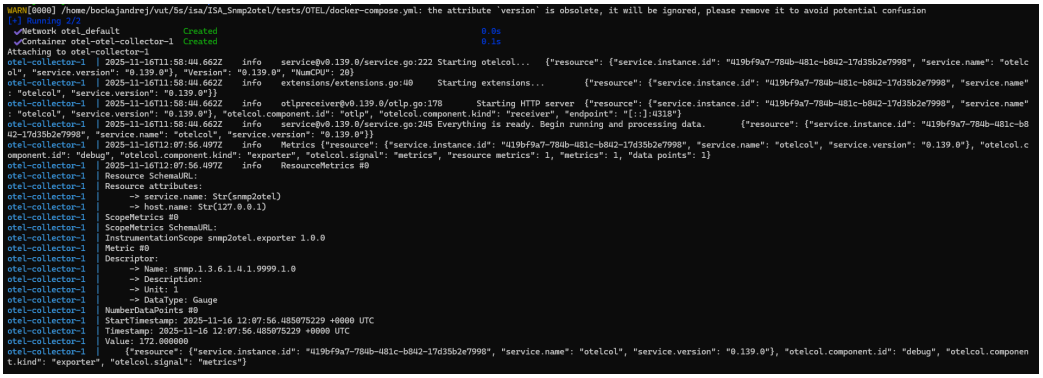
1.5.2 Overenie Formátu OpenTelemetry (OTEL) JSON

Druhou fázou bolo overenie správnej transformácie získaných SNMP dát a ich odoslania v požadovanom formáte OpenTelemetry.

- **Analýza Formátu Metrík (Wireshark):**
  - Wireshark bol opätovne využitý na zachytávanie a kontrolu JSON formátu odosielaného cez **HTTP (POST)** na cieľový OTEL koncový bod.
- **Nasadenie OpenTelemetry Collector (Docker):**

```
otel-collector-1  | 2025-11-16T20:47:43.902Z    info    ResourceMetrics #0
otel-collector-1  | Resource SchemaURL:
otel-collector-1  | Resource attributes:
otel-collector-1  |      -> service.name: Str(snmp2otel)
otel-collector-1  |      -> host.name: Str(127.0.0.1)
otel-collector-1  | ScopeMetrics #0
otel-collector-1  | ScopeMetrics SchemaURL:
otel-collector-1  | InstrumentationScope snmp2otel.exporter 1.0.0
otel-collector-1  | Metric #0
otel-collector-1  | Descriptor:
otel-collector-1  |      -> Name: snmp.1.3.6.1.2.1.1.7.0
otel-collector-1  |      -> Description:
otel-collector-1  |      -> Unit: 1
otel-collector-1  |      -> DataType: Gauge
otel-collector-1  | NumberDataPoints #0
otel-collector-1  | StartTimestamp: 2025-11-16 20:47:43.898074858 +0000 UTC
otel-collector-1  | Timestamp: 2025-11-16 20:47:43.898074858 +0000 UTC
otel-collector-1  | Value: 72.000000
```

Obrázek 1.4: Vysledny OTEL result



Obrázek 1.5: Otel collector

- Bol nasadený plnohodnotný **OpenTelemetry Collector Contrib** v prostredí **Docker**.
- Collector slúžil ako referenčný koncový bod, nakonfigurovaný na **zachytávanie a logovanie prijatých OTEL/HTTP/JSON správ**.
- Úspešné prijatie správ potvrdilo, že výstupné JSON dáta spĺňajú prísne požiadavky formátu OTLP (OpenTelemetry Protocol).

1.6 Odchýlky od Pôvodnej Špecifikácie (Deviations)

V rámci implementácie projektu nastala nasledujúca odchýlka od predpokladanej špecifikácie:

Formát Mapping Súboru (Mapping File Format):

- Pôvodná špecifikácia neurčovala explicitný spôsob definovania mapovacieho súboru pre OID a ich konverziu na metrické názvy a atribúty.
- Z tohto dôvodu bolo prijaté rozhodnutie zaviesť dodatočný voliteľný parameter pre spustenie programu: `[-O oids_formatingMappingFile]`.

Umiestnenie Spustiteľných Súborov:

Kompilácia projektu bola primárne riadená prostredím **CLion (JetBrains)** a systémom **CMake**. Tieto nástroje štandardne uprednostňujú tzv. *out-of-source build* prístup.

Vysvetlenie Adresára `build`

- **Štandardná prax CMake:** Systém `CMake` a IDE ako `CLion` štandardne vytvárajú konfiguračné súbory, dočasné objekty a binárne spustiteľné súbory (exekútory) v samostatnom adresári, typicky nazvanom `build`.
- Z dôvodu integrity projektu a náročnosti spätnej úpravy rozsiahlej konfigurácie kompilácie generovanej nástrojmi CLion a CMake (tzv. *out-of-source build*), bolo zachované existujúce usporiadanie v adresári `build`.

1.7 Použité technológie

Táto sekcia sumarizuje nástroje a prostredia použité pre vývoj, kompiláciu, testovanie a monitorovanie systému.

1.7.1 Vývoj a Build Systém

- **Docker Desktop s WSL 2:** Primárne prostredie pre vývoj a overovanie HTTP komunikácie.
- **Make:** Automatizácia procesov: `setup`, `build`, `run`, `test` a správa Docker kontajnerov.
- **CMake:** Systém na generovanie build skriptov a konfiguráciu projektu.
- **Programovací Jazyk: C/C++**
- **Externé Knižnice:**
  https://github.com/mity/acutest,
  https://github.com/silderan/BasicSNMP,
  https://github.com/yhirose/cpp-httplib,
  https://github.com/nlohmann/json

1.7.2 Monitorovanie a Verifikácia

- **Wireshark:** Nástroj na sledovanie sieťového prenosu (SNMP a HTTP).
- **Docker Kontajnerizácia:** Použitá na spustenie testovacieho **OpenTelemetry Collector Contrib** (na porte 4318) pre príjem a overenie odosielaných metrík OTLP.

Kapitola 2

Literatura

https://support.huawei.com/enterprise/en/doc/EDOC1100174721/684b4c64/snmpv1
https://www.dpstele.com/snmp/tutorial-what-is.php
https://www.dpstele.com/snmp/tutorial/packet-types-structure.php
https://www.ranecommercial.com/legacy/note161.html
https://stackoverflow.com/questions/18119428/c-how-to-exit-out-of-a-while-loop-recvfrom
https://github.com/mity/acutest,
https://github.com/silderan/BasicSNMP,
https://github.com/yhirose/cpp-httplib,
https://github.com/nlohmann/json
https://github.com/open-telemetry/opentelemetry-proto/blob/main/examples/metrics.json