PROJEKTOWANIE SYSTEMÓWI I SIECI KOMPUTEROWYCH

System rozproszonego przetwarzania danych edge-cloud z użyciem kamery stereo OAK-D i frameworka ROS 2

Autor: Jakub Świątek
Opiekun projektu: dr inż. Marek Bolanowski

Spis treści

1 Opis projektu

Celem projektu jest stworzenie systemu rozproszonego przetwarzania danych dla aplikacji robotycznych, wykorzystującego architekturę edge-cloud. System ma umożliwiać przesyłanie, kompresję i przetwarzanie danych wizualnych z **kamery stereo OAK-D** w środowisku **ROS 2**.

Dane z kamery (obrazy RGB) będą przesyłane z urządzenia typu edge (np. laptop lub Raspberry Pi) do drugiego urządzenia – lokalnego serwera lub maszyny wirtualnej – gdzie będą przetwarzane przez moduły oprogramowania mapowania terenu W projekcie wykorzystane zostaną istniejące pakiety ROS 2, w szczególności:

- depthai_ros_driver – do integracji kamery OAK-D z ROS 2,
- rtabmap_ros – do mapowania 3D (SLAM) i generowania chmur punktów,
- rgbd_sync – do synchronizacji i kompresji danych z czujników RGB i Depth.
- rgbd_relay – do dekokmpresji danych z czujników RGB I Depth

Zadaniem projektu będzie modyfikacja konfiguracji i kodu źródłowego wymienionych paczek w taki sposób, by umożliwić kompresję danych wizualnych przesyłanych pomiędzy urządzeniami z ROS 2 (np. z jednej wirtualnej maszyny do drugiej). System powinien zapewniać płynną transmisję danych w sieci lokalnej oraz poprawną rekonstrukcję po stronie serwera.

Projekt ten stanowi krok w kierunku budowy skalowalnej infrastruktury do rozproszonego przetwarzania danych dla robotów autonomicznych, w której zadania mogą być dynamicznie rozdzielane pomiędzy urządzenia edge (zbierające dane) i cloud (analizujące dane i podejmujące decyzje).

2 MVP (Minimum Viable Product)

MVP (Minimum Viable Product) Minimalna działająca wersja projektu będzie polegała na:

- konfiguracji i uruchomieniu kamery OAK-D z pakietem depthai_ros_driver,
- uruchomieniu mapowania 3D z użyciem rtabmap_ros,
- przesyłaniu danych z jednej maszyny (urządzenia edge) do drugiej (urządzenia cloud) za pomocą sieci ROS 2,
- implementacji podstawowej kompresji I dekompresji danych z paczki rgbd_sync I rgbd_relay w celu zmniejszenia przepustowości transmisji,
- wizualizacja uzyskanych zdekompresowanych danych, wraz z zmniejszonym zużyciem sieci na urządzeniu odbierającym transmisje

MVP potwierdzi poprawność integracji poszczególnych modułów ROS 2 oraz skuteczność kompresji danych w środowisku rozproszonym.

Rozszerzona wersja projektu

W rozszerzonej wersji system zostałby rozwinięty o:

- uruchomienie algorytmów mapowania terenu zdalnie na serwerze koła naukowego Machine Learning, przy uruchomionym zbieraniu danych z kamery OAK-D I wizualizacji wyników lokalnie na laptopie

Dzięki temu system stanie się kompletnym środowiskiem testowym dla rozproszonego przetwarzania danych w zastosowaniach robotyki mobilnej i autonomicznej.

3 Progresja

3.1 Instalacja ROS2

Wykorzystano tutorial znajdujący się na oficialnej stronie

3.2 Pierwszy workspace oraz test komunikacji

W celu nauki używania komend oraz workspace'ów stworzono przykładowy odpowiednik, którego zadaniem było publikowanie "Hello world".

3.3 Uruchomienie mapowania lokalnego z kamery

Wykorzystano kod znajdujący się w examples.

3.4 Użycie własnego workspace'u do mapowania lokalnego za pomoca kamery

Napisany kod (napisany wraz z poradnikiem) został wykorzystany w celu odebrania danych z kamery.

3.5 Modyfikacja kodu w celu zbierania infomacji z nagranych wcześniej danych przez rosbag (bez kamery)

Domyślny program (znajdujący się w examples) nie jest dostosowany do odbierania danych publikowanych przez grania rosbag. Dokonano remapowania w celu umożliwienia korzystania z tego typu rozwiązania (wykonane przez AI).

3.6 Uruchomienie rosbag oraz workspace'u na oddzielnych urządzeniach

Na dwóch urządzeniach (laptopach) uruchomiono wcześniej wspomniany workspace oraz nagranie z rosbag. W sieci (WIFI) pojawiły się nadawane tematy dzięki czemu udało się odtworzyć nagranie na innym urządzeniu.

3.7 Testy jakości połączenia

Obecnie w celu weryfikacji sukcesu "użycie dwóch oddzielnych urządzeń"do przyjmowania danych, prowadzone są próby ulokowania kodu oraz nadającego rosbag na dwóch maszynach wirtualnych.

3.8 Użycie RasberryPi 4 model B

Po wielu nie opisanych tutaj problemach z połączeniem w trakcie transmicji wideo dla testów zdecydowano się na użycie połączenia kablem RJ45. Transmisja kamery ma tendencje do wiesznaia się lub rozłączania z siecią hotspot telefonu, dlatego aby skutecznie korzystać z projektu wymagane byłoby urządzenie sieciowe o podanych później specyfikacjach.

3.9 Użycie sieci

Aby odbierać zwykły obraz który nadaje RasberryPi wymagane jest łącze zdołające utrzymać **60MB/s** (liczba bazowana na eksperymentach przeprowadzonych w późniejszym etapie). Po dodaniu interfejsu dla Ethernet'u RasberryPi za pomocą ip addr add 192.168.1.11/24 dev eth0, jesteśmy w stanie utrzymać z urządzeniem stabilny kontakt. Do wizualizacji jesteśmy w stanie użyć jedynie tematów, które nie są skompresowane. Obciążają one mocno połączenie, a celem pozostaje zmniejszenie jego użycia.

3.10 Paczka image$_{t}ransport$

Dzięki użyciu paczki image_transport z łatwością osiągnelibyśmy założony cel. Okazuje się, że zależności i ścieżki paczki napisane były dla wcześniejszej wersji ROS'a i w postaci widniejącej w internecie nie są one "użyteczne". Dzięki remapowaniom i zmianach w kodzie mimo przestarzałej wersji, paczka powinna funkcjonować poprawnie.

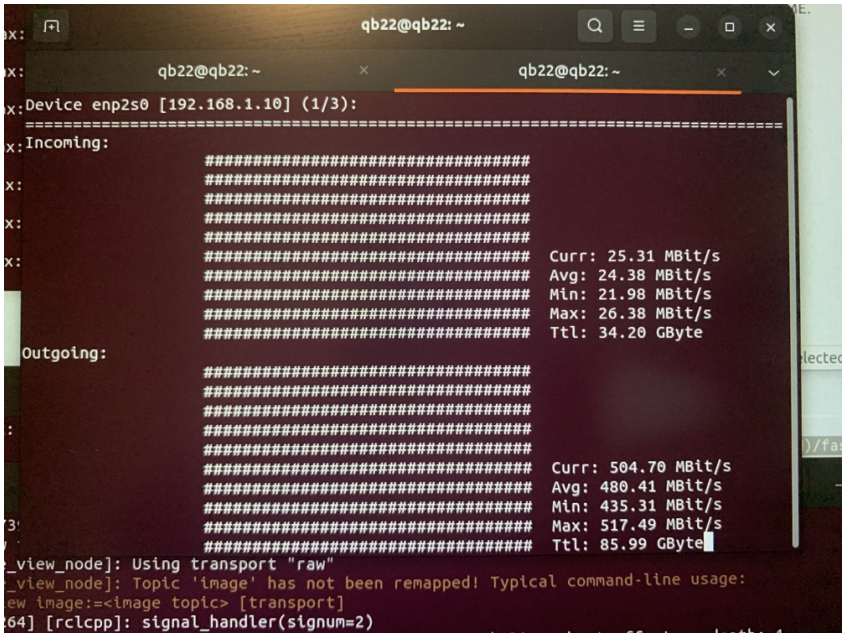
3.11 Rezygnacja z image_transport

Mimo początkowych założeń okazuje się, że image_transport nie nadaje się do tego zadania. W jego miejsce wykorzystano inne rozwiązania:

- ręczna dekompresja - za pomocą cv::imdecode , skompresowany "samodzielnie"rozpakowujemy aby uniknąć problemów z paczką image_transport
- QoS : Best Effort - aby nie dochodziło do sytuacji, że system czeka na paczki które zgubił
- mechanizm Shared Memory - dzięki użyciu tego mechanizmu publikujemy dane do innych programów w obrembie laptopa bezpośrednio przez RAM, a nie sieć
- za pomocą cv_bridge wykorzystujemy cv_bridge::CvImage, aby w najprostszy sposób zamieni macierz cv::Mat na wiadomość ROS sensor_msgs/msg/Image

3.12 Test nowego rozwiązania

Dzięki powyższym rozwiązaniom udało się opublikować temat image_topic, którym jest rozpakowany obraz (jako macierz). Ilość danych przychodzących jest "normalna", zapakane dane można przesyłać łączem, które utrzymuje 25 Mb/s.



Rysunek 1: Przeładowanie sieci

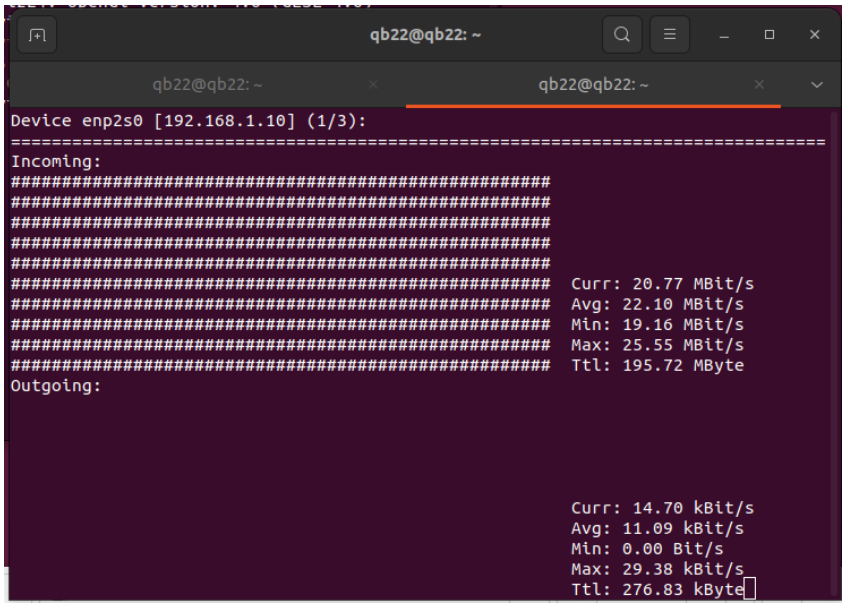
Okazuje się jednak, że sieć jest zapchana. Dane wychodzące osiągają ilość nawet 500 Mb/s, mimo że liczba ta powinna wynosić 0. Okazuje się że na ten moment konfiguracja połączenia powoduje, że nasze "proxy"publikuje wszystkie dane w całej sieci, zapychając ją (docelowo nie miał wysyłać nic).

3.13 Zmiana DDS

Głównym problemem było to, że laptop po dekompresji obrazu próbował wysyłać surowe dane (60 MB/s) z powrotem w sieć, co ją zapychało. Rozwiązaniem okazuje się konfiguracja DDS (Data Distribution Service)

- Wymuszenie Shared Memory (SHM): Skonfigurowano DDS tak, aby duże wiadomości (surowy obraz 2.7 MB) przesyłał wyłącznie przez pamięć RAM laptopa bezpośrednio do Rviz/Rqt. Dzięki temu dane te w ogóle nie trafiają na kartę sieciową.
- Knebbel na sieć (UDP): W pliku fastdds_config.xml ograniczono maksymalny rozmiar paczki danych wysyłanych przez kabel (UDP) do 16 KB. Ponieważ klatka obrazu jest znacznie większa, DDS jest zmuszony zostawić ją w pamięci RAM, zamiast pchać w sieć.
- Stabilizacja QoS: Użyto profilu Best Effort, dzięki czemu system nie marnuje czasu i pasma na ponawianie prób wysłania zgubionych klatek.

Po tych zmianach ruch z urządzenia wygląda tak:



Rysunek 2: Przepływ danych z laptopa

4 Problemy

4.1 Problem 1: Niewidoczność Pakietu (Package not found)

- Kod Błędu / Komunikat: Package not found lub błędy ImportError.
- Opis: Jest to najczęściej spotykany problem, pojawiający się po kompilacji lub otwarciu nowego terminala. Mimo że pliki pakietu fizycznie istnieją na dysku, system uruchomieniowy ROS nie jest w stanie ich zlokalizować ani zaimportować. Problem ten często wiąże się również z załadowaniem niepoprawnego lub innego workspace'u.
- Rozwiązanie: Wszystkie workspace'y muszą mieć inne nazwy. Source'ować plik /opt/ros/humble/setup.bash oraz docelowy workspac'e. (~/NAZWA_WORKSPACE/install/setup.bash)

4.2 Problem 2: Błędna Definicja w Pliku package.xml

- Kod Błędu / Komunikat: Package not found (nawet po wykonaniu source) lub błędy kompilacji colcon.
- Opis: Błąd pliku (`package.xml`) . Błędne, niekompletne lub nieaktualne informacje w tym pliku, w szczególności dotyczące zależności (`<depend>`) lub brak poprawnego typu kompilacji (`build_type` dla pakietów Pythonowych), powodują, że narzędzie `colcon` niepoprawnie przetwarza pakiet. Skutkiem jest brak poprawnej instalacji pakietu, co prowadzi do jego niewidoczności dla całego systemu ROS 2.
- Rozwiązanie: Najczęściej okazuje się, że do pakietu przypisany zły typ kompilacji.

4.3 Problem 3: Brak Zależności (package 'nodelet' not found)

- Kod Błędu / Komunikat: [ERROR] [launch]: Caught exception... "package 'nodelet' not found".
- Opis: Ten błąd jest wynikiem próby użycia archaicznego mechanizmu `nodelet`, który był standardem w ROS 1 i pozwalał na wydajne grupowanie węzłów. W ROS 2 został on zastąpiony przez natywny system **Komponentów**.
- Rozwiązanie: Wynik halucynacji AI lub pomyłka użytkownika przy próbie odtworzenia schematu z ROS 1 w środowisku ROS 2

4.4 Problem 4: Błąd Synchronizacji Danych

- Kod Błędu / Komunikat: [WARN] [rgbd_sync]: Did not receive data since 5 seconds!
- Opis: Ten błąd pojawia się, gdy brakuje jednego z krytycznych tematów. Najczęściej (do tej pory) spotykaną przyczyną była **niezgodność formatów wiadomości** – węzeł oczekiwał surowych danych (`sensor_msgs/msg/Image`), ale był błędnie podłączony do tematu skompresowanego (`/compressed`), co uniemożliwiało synchronizację i dalsze przetwarzanie danych. Możliwą przyczyną jest także brak nadawania jakichkolwiek tematów.
- Rozwiązanie: W zależności od sytuacji, wymaga to np. zmiany remapowania

---

4.5 Problem 5: Krytyczny Błąd Uruchomienia Rosbaga (ModuleNotFoundError)

- Kod Błędu / Komunikat: ModuleNotFoundError: No module named 'rclpy._rclpy_pybind11'.
- Opis: Jest to poważny błąd systemowy, który całkowicie blokuje działanie narzędzi ROS 2 opartych na Pythonie. Błąd wynika z faktu, że dystrybucja ROS 2 (np. Humble) została skompilowana, opierając się na konkretnej wersji Pythona (np. 3.10), natomiast w systemie uruchamiana jest inna, nowsza wersja (np. 3.12). Ta niezgodność uniemożliwia załadowanie kluczowych modułów implementacji języka C, które są niezbędne do komunikacji ROS.
- Notatka: Błąd wystąpił w wyniku naprawy krytycznego błędu bootowania systemu. W trakcie ratowania systemu, przez halucynacje AI lub niedopatrzenia użytkownika, system zauktualizował się do wersji 24.04 (zamiast zostać na Ubuntu 22.04 na której działa ROS2 Hubmle).
- Rozwiązanie: Wykorzystano distrobox, aby sam terminal funkcjonował w Ubuntu 22.04

4.6 Problem 6: Uszkodzenie Pliku Bazy Danych Rosbaga

- Kod Błędu / Komunikat: std::runtime_error: Failed to read from storage lub komunikaty o Database corruption / Sqlite error.
- Opis: Rosbag2 przechowuje nagrane dane w pliku bazy danych **SQLite** (`.sqlite3`). Ten błąd wskazuje, że plik bazy danych został uszkodzony.
- Notatka: Użytkownik przerzucił plik rosbag'a bez spakowania go
- Rozwiązanie: Spakowane pliki nie wykazują tego błędu

4.7 Problem 7: Uprawnienia/sterowniki kamery

- Kod Błędu / Komunikat: Brak
- Opis: Kamera po uruchomieniu jakiegokolwiek skryptu (kamera podłączona kablem USB 3.0) zaczyna "podłączać się i odłączać"bez przerwy.
- Notatka: Problem ten nie pojawiał się na Ubuntu 22.04. Wspomniany problem zaobserwowano na maszynie wirtualnej oraz distrobox.
- Rozwiązanie: Wymagana była emigracja na nowo utworzony system Ubuntu 22.04, na którym problem ustał.

4.8 Problem 8: Zapchanie sieci

- Kod Błędu / Komunikat: Zerwanie połączenia SSH lub brak wyników z poleceń ros2
- Opis: Komunikacja pomiędzy urządzeniami jest niemożliwa lub "pingi"przychodzą po ponad 1000ms (mimo połaczenia poprzez RJ45)
- Notatka: Problem zdawał się być kompletnie losowy i pojawiać się bez wyraźnego powodu.
- Rozwiązanie: Okazuje się, że laptop zapychał sieć chcąc publikować swoje dane w całej sieci, a CherryPi było zmuszone nasłuchiwać przez np. komendy mające wyświetlić dostępne topic'i. Zmiana DDS usunęła problem.

5 Dotychczasowe zasoby

- https://github.com/luxonis/depthai-ros
- https://github.com/introlab/rtabmap_ros
- https://docs.ros.org/en/humble/index.html
- https://docs.luxonis.com/software/depthai/getting-started
- https://wiki.ros.org/rtabmap_ros/TutorialsOldInterface/SetupOnYourRobot
- https://docs.ros.org/en/humble/p/image_transport/