PSI 24Z - Wstępna dokumentacja projektu

Maksym Bieńkowski (01178511@pw.edu.pl) | Jędrzej Grabski | Aleksander Drwal | Tomasz Kowalski

14.12.2024

1. Temat i treść zadania

Program obsługujący prosty protokół P2P (Peer-to-Peer)

Założenia

- Zasób to obiekt z danymi binarnymi identyfikowany pewną nazwą. Za takie same zasoby uważa się zasoby o takich samych nazwach.
- Rozmiar zasobu jest znaczny (tj. większy od jednorazowego transferu sieciowego).
- Początkowo dany zasób znajduje się w jednym hoście, a następnie może być propagowany do innych hostów w ramach inicjowanego przez użytkownika "ręcznie" transferu. Raz pobrany zasób zostaje zachowany jako kopia.
- Po pewnym czasie działania systemu ten sam zasób może znajdować się w kilku hostach sieci.
- Program ma informować o posiadanych lokalnie (tj. w danym węźle) zasobach i umożliwiać ich pobranie.

Funkcjonalność programu

- Dodawanie nowych zasobów przez użytkownika – wprowadzanie z lokalnego systemu plików.
- Pobieranie zasobów:
  - Użytkownik może pobrać konkretny zasób po nazwie ze zdalnego hosta (jeden zasób na raz).
  - Użytkownik decyduje, z którego hosta dany zasób zostanie pobrany.
- Rozgłaszanie informacji o posiadanych lokalnie zasobach.

Dodatkowe założenia

- Zasób pobrany do lokalnego hosta jest kopią oryginału. Kopia jest traktowana tak samo jak oryginał (są nierozróżnialne), tj.:
  - Istnienie kopii jest rozgłaszane w taki sam sposób jak istnienie oryginału.
- Program powinien obsługiwać różne sytuacje wyjątkowe, np. przerwanie transmisji spowodowane błędem sieciowym.
- Lokalizacja zasobów odbywa się poprzez rozgłaszanie:
  - Wskazówka: użyć protokołu UDP, ustawić opcje gniazda `SO_BROADCAST`, wykorzystać adresy IP rozgłaszające (same bity "1" w części hosta).

Interfejs użytkownika

- Wystarczy prosty interfejs tekstowy.
- Interfejs powinien obsługiwać współbieżny transfer zasobów – tj. nie powinien się blokować w oczekiwaniu na przesłanie danego zasobu.

Wariant zadania W13/W22

- całość komunikacji (przesyłania zasobu) zrealizować na UDP, dodatkowo dla uproszczenia można przyjąć, że zasób mieści się w całości w jednym datagramie (datagram danych może być zgubiony – należy to uwzględnić)
- implementacja w C++

2. Interpretacja treści zadania

Główne funkcje programu:

1. **Udostępnianie zasobów lokalnych:**

- Na urządzeniu hosta istnieje folder roboczy programu, którego zawartość będzie udostępniana innym.
- Udostępnienie zasobu jest równoznaczne z umieszczeniem go w tym katalogu.

```
> share /path/to/resource
# równoznaczne ze skopiowaniem pliku /path/to/resource do /shared_folder

user@machine launch-program

Sharing resources from folder /shared_folder.
```

```
Shared resources:
file1.txt
file2.txt
```

2. **Rozgłaszanie udostępnianych zasobów:**

- Periodyczne wysyłanie listy dostępnych zasobów przy użyciu protokołu UDP w trybie broadcast.

3. **Aktualizacja dostępnych zasobów:**

- Po otrzymaniu wiadomości broadcastowej z listą udostępnianych przez hosta A zasobów, host B aktualizuje listę dostępnych u A zasobów.

```
> list-resources

file1.txt
hosts: [host1, host2]

# otrzymanie wiadomości o file2.txt od hosta 1
> list-resources

file1.txt
hosts: [host1, host2]

file2.txt
hosts: [host1]
```

4. **Pobieranie zasobu:**

- Transfer zasobu z wybranego hosta inicjowany przez użytkownika programu.
- Plik pobierany jest do katalogu `shared_folder` i wiadomość o dostępności wysyłana jest przy następnym rozgłoszeniu.

```
> download file1.txt host1

Downloading file1.txt from host1...

>

Completed download of file1.txt from host1.
Available at shared_folder/file1.txt
```

5. **Obsługa sytuacji wyjątkowych:**

- Ponawianie zapytań w przypadku utraty datagramów, stosowanie prostego rozwiązania ACK.
- Informowanie użytkownika o niepowodzeniach transferu.

```
> download file1.txt host1

Downloading file1.txt from host1...

Error while downloading file1.txt: could not connect to host1. Retrying...
Error while downloading file1.txt: could not connect to host1. Retrying...
Error while downloading file1.txt: could not connect to host1. Retrying...
Fatal: Error while downloading file1.txt: could not connect to host1. Retried 3 times.
# Usunięcie użytkownika host1 z pamięci do otrzymania od niego kolejnego broadcastu
```

6. **Sprawdzenie listy dostępnych zasobów**

- Po wpisaniu określonej komendy, wypisywane są adresy hostów wraz z udostępnianymi przez nich zasobami.

```
> list-resources

file1.txt
hosts: [host1, host2]

file2.txt
hosts: [host1, host3]
```

- Po wpisaniu innej komendy, wypisywane są adresy hostów udostępniających konkretny zasób.

```
> find file1.txt

file1.txt
hosts: [host1, host2]
```

3. Krótki opis funkcjonalny (“black-box”)

Użytkownik

- Może udostępniać zasoby ze swojego systemu plików.
- Może wywołać pobieranie zasobu o podanej nazwie z określonego hosta.
- Może przeglądać udostępniane przez określonego hosta zasoby.
- Może przeglądać wszystkie dostępne w sieci zasoby.
- Jest informowany o wystąpieniu błędów i niepowodzeń.

Program

- Rozgłasza informacje o udostępnianych zasobach.
- Obsługuje prośby o przesłanie zasobów od innych hostów.
- Przesyła i odbiera zasoby.
- Zarządza transferami bez blokowania interfejsu użytkownika.
- Informuje użytkownika o wystąpieniu błędów i niepowodzeń.

5. Opis i analiza protokołów komunikacyjnych

Bazowy protokół - UDP

Struktura nagłówka pakietu

PROT_VERSION (8 bitów)

- wersja protokołu

MSG_TYPE (8 bitów)

- `RESOURCE_ANNOUNCE` - ogłoszenie lokalnie posiadanych zasobów
- `RESOURCE_REQUEST` - prośba o udostępnienie określonego zasobu od konkretnego hosta
- `RESOURCE_DATA` - przesłanie zasobu do węzła, który go zażądał

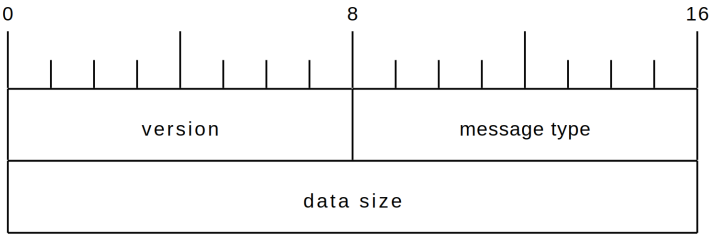
DATA_SIZE (32 bity)

- długość danych spoza nagłówka w bajtach.

Za maksymalny rozmiar wysyłalnych danych uznajmy $65507 - 1 - 1 - 4 = 65501$ bajtów (zgodnie z założeniem, że dane mieszczą się w jednym datagramie UDP na podstawie przydzielonego wariantu zadania). Uznajemy, że zgubienie pakietu rozgłoszeniowego nie jest warte obsłużenia ze względu na ich periodyczne wysyłanie. Zgubienie pakietu `RESOURCE_REQUEST` obsługiwane jest mechanizmem ponownego odpytywania n razy do otrzymania zasobu, co rozwiązuje także przypadek zgubienia datagramu `RESOURCE_DATA`.

```cpp
class PacketHeader {
public:
    enum class MsgType : uint8_t {
        RESOURCE_ANNOUNCE = 0,  // Ogłoszenie lokalnie posiadanych zasobów
        RESOURCE_REQUEST = 1,   // Prośba o udostępnienie określonego zasobu od konkretnego hosta
        RESOURCE_DATA = 2       // Przesłanie zasobu do węzła, który go zażądał
    };

    uint8_t PROT_VERSION; // Wersja protokołu (8 bitów)
    MsgType MSG_TYPE;     // Message type (8 bitów)
    uint32_t DATA_SIZE;   // Rozmiar danych do odebrania w bajtach (32 bitów)
};
```



Rysunek 1: Struktura nagłówka protokołu

Dane poza nagłówkiem

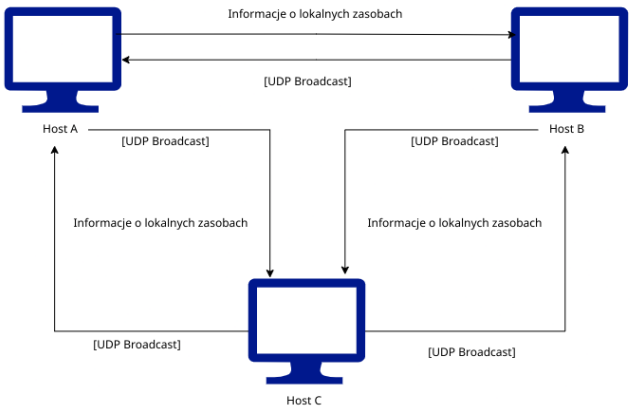
- W przypadku pakietu `RESOURCE_ANNOUNCE`, lista zakodowanych w ASCII nazw plików, separowanych przez bajt `\0`
- W przypadku pakietu `RESOURCE_REQUEST`, nazwa żądanego pakietu zakończona bajtem `\0`
- W przypadku pakietu `RESOURCE_DATA`, binarne dane zasobu o długości `DATA_SIZE`

```
class Message {
public:
    PacketHeader header;            // Nagłówek pakietu
    std::vector<uint8_t> data;      // Dane poza nagłówkiem (zmienny rozmiar)
};
```

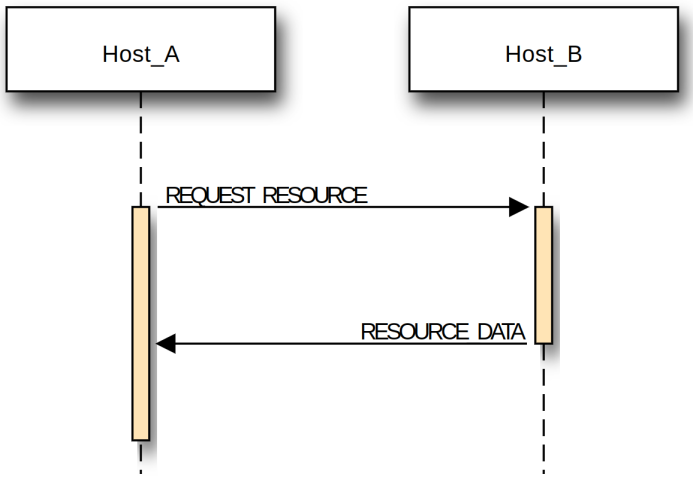
Przypisanie portu

- Zakładamy, że na potrzeby protokołu wszystkim hostom przypisany jest stały port.

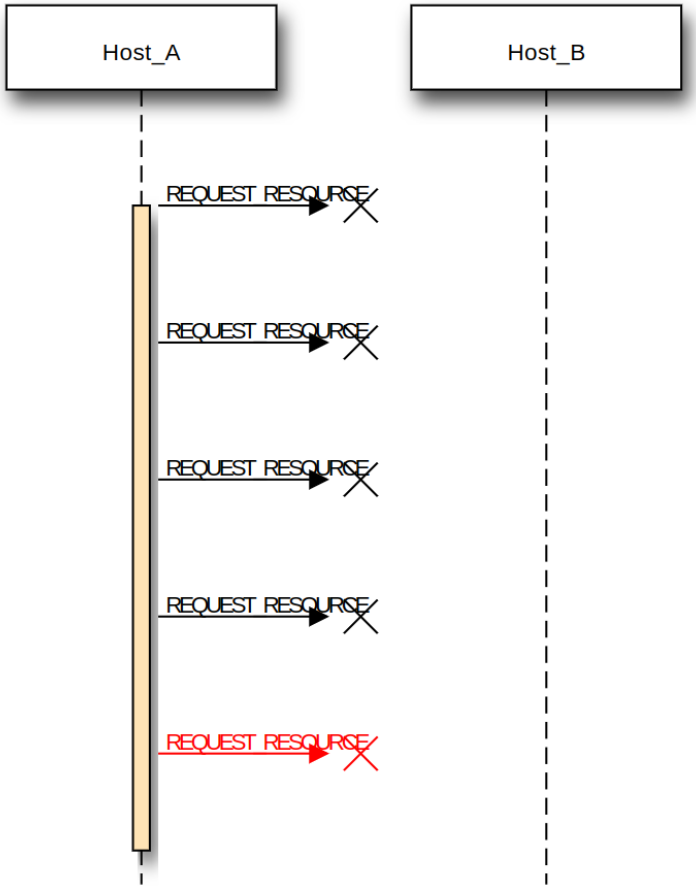
Diagramy komunikacji



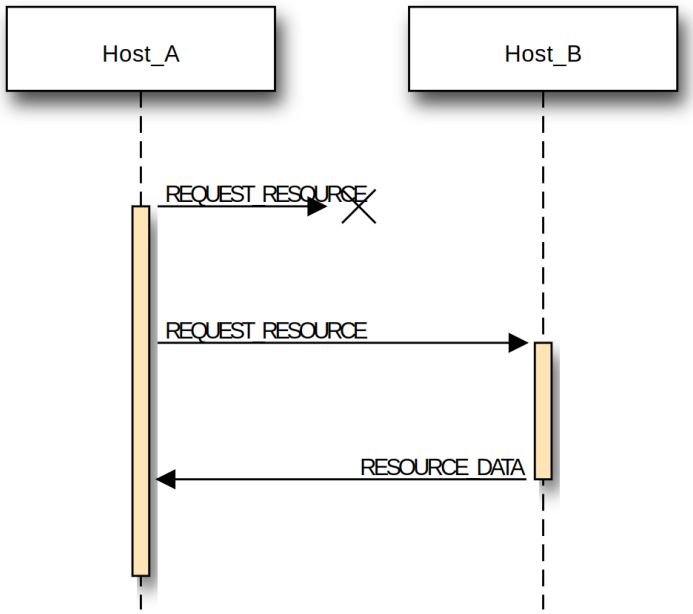
Rysunek 2: Rozgłaszanie zasobów



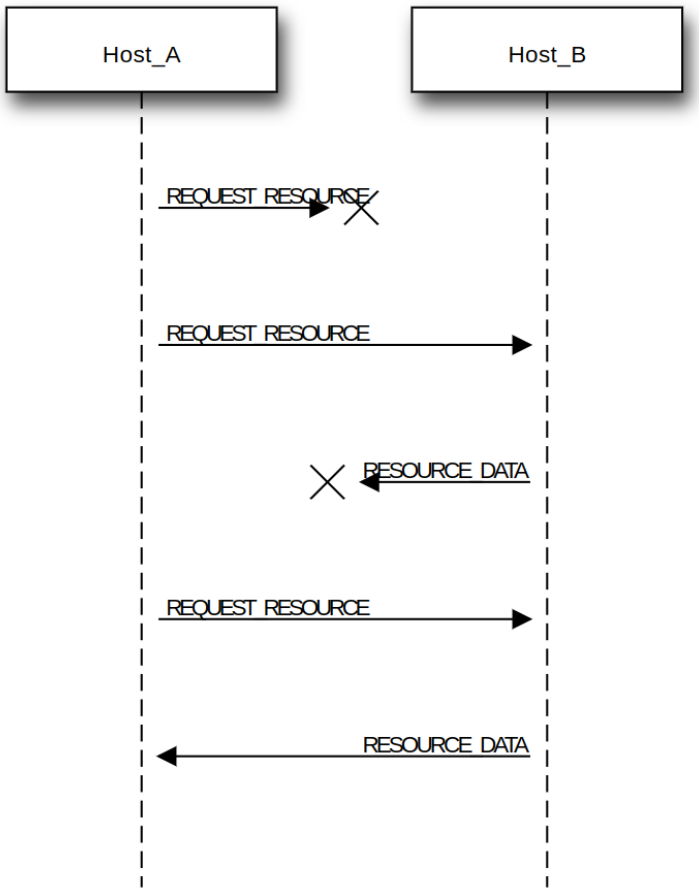
Rysunek 3: Bezproblemowy przesył danych



Rysunek 4: Niemożnośc osiągnięcia hosta



Rysunek 5: Utrata prośby o zasób



Rysunek 6: Utrata prośby i zasobu

6. Planowany podział na moduły

1. Moduł zarządzania zasobami:

- Obsługa dodawania, usuwania i przeglądu zasobów lokalnych

2. Moduł sieciowy:

- Rozgłaszanie zasobów
- Obsługa zapytań i transferu zasobów
- Zarządzanie retransmisją utraconych datagramów i obsługą błędów

3. Moduł interfejsu użytkownika:

- Prosty tekstowy interfejs umożliwiający wykonywanie operacji przez użytkownika, na wzór wcześniej pokazanych zrzutów z terminala

7. Zarys koncepcji implementacji

- Implementacja w języku C++

Biblioteki

- **Boost.Asio**: Obsługa gniazd sieciowych i operacji asynchronicznych.
- **STL**: Przechowywanie i zarządzanie danymi lokalnymi.

Narzędzia

- **CMake**: System budowy projektu.
- **GCC/Clang**: Kompilator.
- **Clang-tidy, clang-format**: Linter i formater.
- **Google test**: Testy jednostkowe

Ogólne podejście do implementacji

W celu zapewnienia nieblokujących operacji IO zamierzamy zastosować wielowątkową architekturę programu:

- wątek odpowiedzialny za interakcję przez CLI z użytkownikiem
- wątek odpowiedzialny za cykliczne wysyłanie rozgłoszeń
- dynamicznie powoływane wątki *workers* odpowiedzialne za pobieranie i przesyłanie zasobów

Zamierzamy zastosować obiektowe podejście przy implementacji programu w celu testowalnego i wyraźnego podziału odpowiedzialności między komponentami. Testowanie poprawności implementacji przeprowadzone będzie przy pomocy testów ręczych i jednostkowych.