

# **Dokumentacja projektu z przedmiotu Programowanie Sieciowe**



## ***Autorzy***

Soroka Hubert  
Łukasz Jaremek  
Daniel Kobiałka  
Radosław Kostrzewski

## ***Data***

5 stycznia 2023

# Polecenie

Napisać program obsługujący prosty protokół P2P (Peer-to-Peer).  
Założenia:

- Zasób to plik 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 węźle sieci, następnie może być propagowany do innych węzłów w ramach inicjowanego przez użytkownika ręcznie transferu (patrz dalej) – raz pobrany zasób zostaje zachowany jako kopia.
- Tak więc, po pewnym czasie działania systemu ten sam zasób może znajdować się w kilku węzłach sieci (na kilku maszynach).
- Program ma informować o posiadanych lokalnie (tj. w danym węźle) zasobach i umożliwiać ich pobranie.
- Program powinien umożliwiać:
  - wprowadzanie przez użytkownika nowych zasobów – z lokalnego systemu plików,
  - pobieranie konkretnych nazwanych zasobów ze zdalnego węzła (jednego naraz)
  - rozgłaszanie informacji o posiadanych lokalnie zasobach.
- W przypadku pobierania zdalnego zasobu użytkownik decyduje skąd zostanie on pobrany.
- Zasób pobrany do lokalnego węzła jest kopią oryginału, kopia jest traktowana tak samo jak oryginał (są nierozróżnialne) – tj.: istnienie kopii jest rozgłaszane, tak samo jak oryginału.
- Należy zwrócić uwagę na różne obsługę różnych sytuacji wyjątkowych – np. przerwanie transmisji spowodowane błędem sieciowym.
- Lokalizacja zasobów ma następować poprzez rozgłaszanie – wskazówka: użyć prot. UDP, ustawić opcje gniazda `SO_BROADCAST`, wykorzystać adresy IP rozgłaszające (same bity "1" w części hosta).
- Interfejs użytkownika – wystarczy prosty interfejs tekstowy, powinien on jednak obsługiwać współbieżny transfer zasobów (tj. Nie powinien się blokować w oczekiwaniu na przesłanie danego zasobu)

Warianty indywidualne:

1. W1 – zasymulować błędy protokołu (zgubienie rozgłaszanego UDP)
2. W2 – zasymulować błędy protokołu (zerwanie sesji TCP)
3. W3 – całość komunikacji (przesyłania zasobu) zrealizować na UDP, dodatkowo – zasób mieści się w całości w jednym datagramie (datagram danych może być zgubiony – należy to uwzględnić)

## Informacje Ogólne

Skład zespołu:

- Soroka Hubert – [01158909@pw.edu.pl](mailto:01158909@pw.edu.pl)
- Łukasz Jaremek
- Daniel Kobiałka
- Radosław Kostrzewski

Data przekazania: 05.01.2023

Realizowany wariant: W1

## Interpretacja treści zadania

Celem zadania jest napisanie programu z interfejsem tekstowym, który umożliwi obsługę sieci Peer-to-Peer, przez którą przekazywane są pliki pomiędzy węzłami. Użytkownik może łatwo uzyskać dostęp do pliku, który fizycznie znajduje się na węźle sieci. Jako węzeł rozumiemy instancję programu uruchomioną pod unikalnym adresem sieciowym. Warto zaznaczyć, że sieć nie umożliwia aktualizacji istniejących plików inaczej, niż przez zmianę nazwy, ponieważ zasoby o tych samych nazwach, są traktowane jednakowo, a zmiana zawartości pliku bez zmiany nazwy, nie będzie widoczna dla węzła sieci. Każdy węzeł będzie posiadał własną tablicę zasobów, w której będą znajdowały się informacje o dostępnych zasobach w każdym węźle sieci.

## Opis funkcjonalny

Przewidujemy udostępnienie użytkownikowi następujących funkcji:

- `downloaded_files` – wylistowanie zasobów fizycznie znajdujących się w lokalnym węźle
- `available_files` – wylistowanie unikalnych zasobów znajdujących się we wszystkich węzłach sieci
- `download_file [nazwa pliku]` – rozpocznij pobieranie pliku o danej nazwie dostępnego w sieci (w wypadku wielu wystąpień w sieci, wymaga podania węzła źródłowego spośród dostępnych)

- `upload_file [nazwa pliku]`– umożliwienie pobrania pliku z tego węzła pozostałym węzłom z użyciem `download_file`
- `download_progress [nazwa pliku]` – wyświetla komunikat o stanie przetwarzania danego pliku (dostępny lokalnie / pobierany / niepobierany)

## Komunikacja

Użyjemy protokołu UDP do rozgłaszania tablicy dostępnych zasobów wszystkim węzłom z użyciem opcji BROADCAST oraz protokołu TCP do przesyłania zasobów pomiędzy węzłami.

Rozgłaszanie w UDP zostanie zrealizowane z użyciem gniazda nasłuchującego, które będzie czekać na sygnały rozgłaszające w wątku programu do tego przeznaczonym, a następnie synchronicznie je obsługiwać poprzez aktualizacje tablicy zasobów. Gniazdo nadające sygnał rozgłaszający będzie tworzone po dodaniu nowego zasobu w węzle, a po nadaniu sygnału, będzie zamknięte.

Przesyłanie plików w TCP również będzie wymagało stale nasłuchującego gniazda w osobnym wątku. Różnica jest taka, że połączenia będą obsługiwane wspólnie z użyciem wątków. Węzeł, po otrzymaniu komunikatu zawierającego żądanie pobrania zasobu, zacznie przysyłać żądany plik lub wyśle komunikat błędu. Gniazda inicjujące połączenia z innymi gniazdami będą tworzone w osobnych wątkach po wywołaniu funkcji `download_file` przez użytkownika. Program nie będzie czekał na pobranie pliku, lecz umożliwi użytkownikowi sprawdzenie stanu pobierania w funkcji `download_progress`, poprzez sprawdzenie stanu wątku pobierającego dany zasób.

Aktualizacja tablicy zasobów, a więc wyznacznika stanu dostępności zasobów w węzłach, będzie aktualizowana dopiero po pobraniu pliku, aby nie doszło do sytuacji, gdzie użytkownik otrzymuje informacje o dostępności pliku, który jeszcze nie jest kompletny. Po udanym pobraniu pliku, aktualizacja tablicy zasobów będzie rozgłaszana po UDP, wg opisu powyżej.

Po wywołaniu `upload_file`, plik o danej nazwie jest kopiowany z dysku do węzła, po czym jest rozgłaszana informacja o jego dostępności, analogicznie jak dla pobrania.

Gniazda słuchające obu protokołów będą nasłuchiwać na tym samym porcie.

Przy pobieraniu pliku z innego węzła, węzeł inicjujący połączenie wyśle komunikat, który będzie zawierał tylko nazwę żadanego zasobu. Węzeł docelowy najpierw odpowie mu prostym komunikatem zawierającym potwierdzenie / zaprzeczenie dostępności pliku

Rozgłaszanie UDP będzie realizowane z użyciem tylko jednego komunikatu, który będzie zawierał adres węzła oraz listę zasobów na nim dostępnych. Każdy inny węzeł po otrzymaniu takiego komunikatu zaktualizuje tablicę zasobów, poprzez ustawienie wartości dla nadającego węzła na otrzymaną listę zasobów.

# Struktura

The diagram illustrates the system architecture, divided into three main sections: **Użytkownik** (User), **Węzeł** (Node), and **Inny węzeł z sieci** (Other network node).

**Użytkownik (User):** Contains input/output elements: *downloaded\_files*, *available\_files*, *download\_file*, *wybór węzła* (node selection), *download\_progress*, and *upload\_file*.

**Węzeł (Node):** Contains internal components:
 

- Lokalne pliki** (Local files): A database of files stored on the node.
- Tablica zasobów wszystkich węzłów** (Resource table of all nodes): A table storing information about resources available across the network.
- Zapytaj o węzeł, jeśli plik niedostępny lokalnie. Pobierz w wątku z wybranego węzła** (Ask for node if file is not locally available. Download in a thread from the selected node): A process that initiates file retrieval from other nodes.
- Wątek pobierający** (Downloading thread): A thread responsible for downloading files from other nodes.
- Sprawdź, czy plik dostępny lokalnie, jeśli nie, sprawdź wątki.** (Check if file is locally available, if not, check threads): A process that checks local availability before attempting to download.
- Jeśli plik niedostępny lokalnie, przekopiuj z dysku** (If file is not locally available, copy from disk): A process that handles local file copying.
- Rozgłoś zmianę tablicy zasobów** (Broadcast resource table change): A process that updates the resource table across the network.

**Inny węzeł z sieci (Other network node):** Contains **Lokalne pliki** (Local files) and **Tablica zasobów wszystkich węzłów** (Resource table of all nodes).

**Data Flow and Interactions:**

- Yellow Line (User to Node):**
  - downloaded\_files* sends *Lista plików* (List of files) to **Lokalne pliki**.
  - available\_files* sends *Lista plików bez duplikatów* (List of files without duplicates) to **Tablica zasobów wszystkich węzłów**.
  - download\_file* sends *Nazwa pliku do pobrania* (File name to download) to the **Zapytaj o węzeł...** process.
  - wybór węzła* sends *Nazwa węzła* (Node name) to the **Zapytaj o węzeł...** process.
  - download\_progress* sends *Komunikat o dostępności / stanie pobierania* (Availability/status message) and *Nazwa pobieranego pliku* (Name of the file being downloaded) to the **Sprawdź, czy plik dostępny lokalnie...** process.
  - upload\_file* sends *Nazwa/ścieżka do pliku* (Name/path to file) to the **Jeśli plik niedostępny lokalnie, przekopiuj z dysku** process.
- Node Internal Flow:**
  - Lokalne pliki** sends *Nazwa szukanego pliku* (Name of the searched file) to **Tablica zasobów wszystkich węzłów**.
  - Tablica zasobów wszystkich węzłów** sends *Lokalizacja pliku* (File location) to the **Zapytaj o węzeł...** process.
  - The **Zapytaj o węzeł...** process sends *Nowy plik* (New file) to the **Wątek pobierający**.
  - The **Wątek pobierający** sends *Po udanym pobraniu* (After successful download) to **Rozgłoś zmianę tablicy zasobów**.
  - The **Sprawdź, czy plik dostępny lokalnie...** process sends *Nowy plik* to the **Wątek pobierający** and *Nowy plik* to **Rozgłoś zmianę tablicy zasobów**.
  - The **Jeśli plik niedostępny lokalnie, przekopiuj z dysku** process sends *Po udanym kopiowaniu* (After successful copying) to **Rozgłoś zmianę tablicy zasobów**.
  - Rozgłoś zmianę tablicy zasobów** sends *Zmiana tablicy zasobów* (Resource table change) to **Tablica zasobów wszystkich węzłów** in the **Inny węzeł z sieci**.
- Node to Other Network Node:**
  - The **Wątek pobierający** sends *Połączenie z węzłem posiadającym plik* (Connection to node with file) to **Lokalne pliki** in the **Inny węzeł z sieci**.
  - Lokalne pliki** in the **Inny węzeł z sieci** sends *Plik* (File) to the **Wątek pobierający**.

Owale oznaczają operacje użytkownika, prostokąty operacje programu, błękitny oznacza zasoby lokalnego węzła, zielony to zasoby innego węzła z sieci, napis nad strzałką oznacza zwracaną wartość, strzałka to przepływ sterowania.

## Implementacja

Projekt będzie realizowany w języku Python na środowisku linuxowym. Do serializacji komunikatów zostanie wykorzystana biblioteka `pickle`.

Stworzymy klasę `Node`, która będzie reprezentować jeden węzeł w sieci. Każdy węzeł będzie posiadać plik `config.json`, który będzie informował o adresach dostępnych w sieci węzłów. Przykładowy wygląd pliku:

```
{
  "nodes": [
    {
      "node_name": "Luki",
      "node_addr": "192.168.1.212",
      "node_port": "4200"
    },
    {
      "node_name": "Bomba",
      "node_addr": "192.168.1.211",
      "node_port": "4200"
    }
  ]
}
```