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
- 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 (zostanie wyświetlony komunikat na konsoli, zawierający listę węzłów, na których

szukany zasób jest dostępny). Nie powoduje pobrania, jeśli zasób już dostępny.

- 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 synchroniczne je obsługiwać poprzez aktualizacje tablicy zasobów. Gniazdo nadające sygnał rozgłaszający będzie tworzone po dodaniu nowego zasobu w węźle, 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 współbieżnie z użyciem wątków. Węzeł, po otrzymaniu komunikatu zawierającego żądanie pobrania zasobu, zacznie przesyłać żądany plik lub wyśle komunikat. 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 żądanego zasobu

(download_request). Węzeł docelowy najpierw odpowie mu prostym komunikatem (download_response) zawierającym potwierdzenie / zaprzeczenie lokalnej dostępności pliku (available) oraz, jeśli zasób nie jest dostępny na węźle docelowym, ale jest dostępny na innym w jego tablicy, adresem węzła, który posiada szukany zasób (node_addr). W tym przypadku, nastąpi też wysyłanie pliku z użyciem istniejącego połączenia. Węzeł żądający pobrania najpierw sprawdzi, czy otrzymał potwierdzenie dostępności, a potem zacznie odbierać plik lub przerwie połączenie i skieruje się do wskazanego węzła.

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

Komunikaty będą miały w kodzie formę prostych obiektów, które będą serializowane przed przesłaniem, z użyciem biblioteki pickle.

Struktura komunikatów

Nazwy struktur i pól są robocze. Adresy węzłów zawierają też ich porty.

Komunikaty używane w TCP:

```
struct download_request {
    char[64] name;
}
```

```
struct download_response {
    bool available;
    char[24] node_addr;
}
```

Komunikat używany w UDP:

```
struct broadcast_msg {
    char[24] node_addr;
    char[64][512] resources;
}
```

Symulacja błędów

W ramach zadania, zasymulujemy błędy protokołu UDP. UDP będzie używane tylko i wyłącznie do rozgłaszania informacji o aktualizacji zasobów na danym węźle. W przypadku, gdy zostaną zgubione tylko komunikaty do pojedynczych węzłów, ale przynajmniej jeden z nich dotrze do celu, nadal możliwe jest przesyłanie plików. Podczas próby połączenia celem pobrania pliku z innego węzła (opisane w **Komunikacja**), w przypadku braku pliku na węźle

docelowym, nastąpi sprawdzenie dostępności zasobu na innych węzłach wg tablicy zasobów węzła docelowego. Jeśli użytkownik zechce pobrać zasób, którego obecność nie została dobrze rozgłoszona, i jako węzeł docelowy ustali węzeł, który otrzymał poprawne rozgłoszenie, nastąpi przekierowanie do węzła zawierającego zasób.

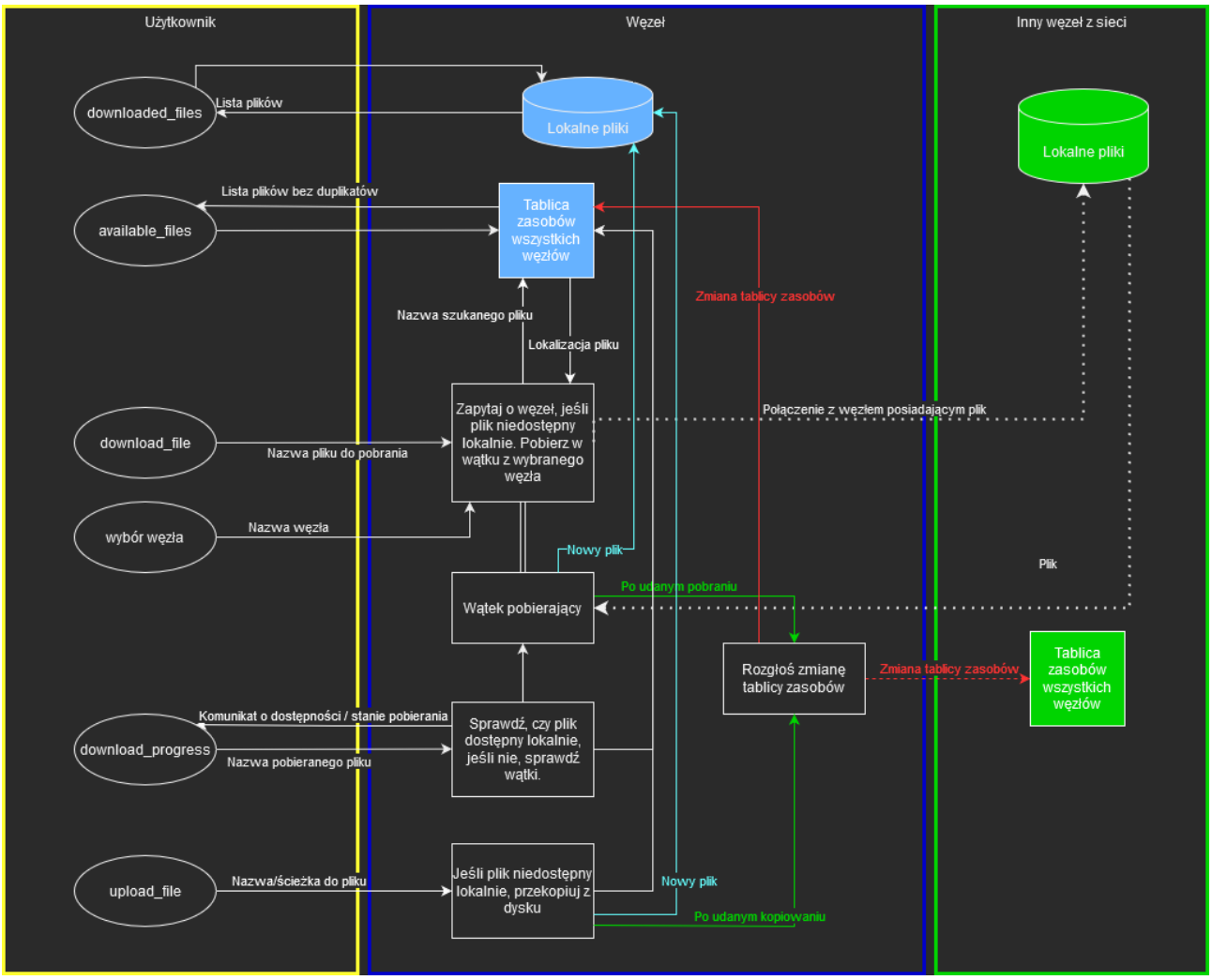
W przypadku, gdy żaden węzeł nie otrzyma informacji o dostępności nowego zasobu, są dwa scenariusze umożliwiające naprawienie sytuacji:

1. Ponowne wywołanie upload_file lub download_file dla dowolnych nowych zasobów na węźle, z którego rozgłaszanie się nie udało. Obie funkcje, po udanej operacji, rozgłaszają swój stan innym węzłom. Przy rozgłaszaniu informują o wszystkich zasobach, więc te niedodane ostatnio, również będą rozgłoszone.
2. Próba pobrania zasobu „na ślepo”, czyli żądanie pobrania pliku, który, z punktu widzenia węzła żądającego, jest niedostępny (program ostrzeże o takiej sytuacji, ale nie będzie blokował, właśnie ze względu na taką sytuację).

W programie, aby móc wymuszać takie sytuacje na potrzeby testów, dodamy flagi przy wołaniu operacji download_file lub upload_file, których użycie będzie skutkowało zupełnym lub częściowym (losowe węzły nie otrzymają komunikatu) niewysłaniem komunikatu rozgłoszenia. Oczywiście, użytkownik nie powinien z takiej flagi skorzystać.

Struktura

Schemat planowanej architektury aplikacji jest zobrazowany na grafie:



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. Warto zaznaczyć, że węzły muszą mieć unikalne kombinacje adresów i portów. Przykładowy wygląd pliku:

```
{
   "nodes": [
      {
         "node_name": "Luki",
         "node_addr": "192.168.1.212:4200"
      },

      {
         "node_name": "Bomba",
         "node_addr": "192.168.1.211:4200 "
      }
   ]
}
```