[TIN] Projekt

Data przekazania:

**26.01.2018**

# Program obsługujący prosty protokół P2P

Program konsolowy umożliwiający współbieżne wykonywanie operacji:

* Dołączanie węzła do sieci P2P
* Odłączanie węzła od sieci P2P
* Wprowadzanie przez użytkownika do sieci P2P nowych zasobów
* Pobieranie nazwanych zasobów z sieci P2P (w tym kilku jednocześnie)
* Rozgłaszanie informacji o posiadanych lokalnie zasobach
* Usuwanie (unieważnianie) przez właściciela zasobu w sieci

Na maszynie uruchamiany jest program do obsługi sieci P2P, po czym odpowiednią komendą użytkownik zgłasza chęć dołączenia do sieci.

Po dołączeniu do sieci możliwe jest poprzez odpowiednie komendy wykonanie wyżej wymienionych operacji (wyłączając ponowne dołączenie do sieci).

W sieci P2P każdy z węzłów zna adres każdego innego węzła, posiada informacje o liczbie węzłów w sieci oraz może poprosić dany węzeł o przekazanie mu informacji o jego lokalnie posiadanych zasobach (po takiej operacji wykonanej na wszystkich węzłach – wiadomo jakie pliki znajdują się w sieci).

Funkcja odwzorująca hashe plików na identyfikator węzła jest zmienna, w zależności od tego ile węzłów znajduje się w danej sieci.

Przy wprowadzeniu pliku do sieci, właściciel musi obliczyć jego unikalny hash, którym bedzie identyfikowany w sieci, a następnie zmapować hash na numer odpowiedniego węzła i przesłać mu dany plik.

Unieważnienie pliku w sieci może nastąpić przez jego właściciela. Wysyła on odpowiedni komunikat do posiadającego plik węzła. Jeśli podczas odebrania komunikatu, plik był transferowany do innego węzła, transfer jest dokańczany, a dopiero potem plik jest usuwany (jego wpis w tablicy plików przechowywanych lokalnie oraz fizycznie na dysku).

Dostępne dla użytkownika komendy (wykonywane w osobnych wątkach procesu):

* *join –* dołączenie do sieci P2P lub w razie braku – utworzenie nowej
* *leave –* odłączenie się od sieci
* *add <filename>–* dodanie do sieci pliku o podanej nazwie, jako jego właściciel
* *get <hash> –* pobranie z sieci pliku o podanym hashu
* *remove <hash> –* unieważnienie zasobu o podanym hashu
* *list –* wypisanie na konsoli identyfikatorów (hashy) wszystkich plików w sieci
* *stat –* wypisanie na konsoli identyfikatorów wszystkich węzłów w sieci wraz z ich adresami IP

Każdy węzeł sieci posiada:

* *size\_t nodeId* - identyfikatorktórym posługuje się w sieci P2P
* *size\_t nodeCnt* - liczbę węzłów w sieci
* *map<size\_t, struct in\_addr> nodeMap* - tablicę mapującą *nodeId* wszystkich węzłów w sieci na ich adresy IP (*nodeId* jest indeksem tablicy, zaś jej wartością jest adres IP węzła o *nodeId* odpowiadającym indeksowi przechowywany w strukturze *in\_addr*)
* *map<string, FileInfo> nodeFiles* - tablicę identyfikatorów plików lokalnych posiadanych przez dany węzeł (hash MD5), wraz ze strukturą przechowującą *nodeId* ich właścicieli oraz zmiennymi odpowiadającymi za dokańczanie transferu w razie usunięcia plik

# **Formaty komunikatów:**

### **Informacyjne**: **Danowe:**

Format **K0**: Format **KD0**:

|  |
| --- |
| opcode |
| fileId0 | fileId1 |
| fileId2 | ... |

|  |
| --- |
| opcode |

Format **K1**: Format **KD1**:

|  |
| --- |
| opcode |
| fileId | Owner nodeId |
| data | |

|  |  |
| --- | --- |
| opcode | nodeCnt |
| nodeId sender | nodeId receiver |

Format **K2**:

|  |  |
| --- | --- |
| opcode | fileId |

Opcode:

* **100** – prośba o przyłączenie do sieci (K0)
* **101** – prośba o odłączenie od sieci (K0)
* **102** – informacje dla nowego węzła: *nodeId* nadawcy komunikatu, nadane *nodeId* dla nowego węzła oraz liczba węzłów w sieci *nodeCnt* (K1)
* **103** – prośba o informacje o plikach lokalnych w danym węźle (K2)
* **200** – poprawna redystrybucja zasobów (K0)
* **201** – poprawne przyłączenie węzła (K0)
* **203** – poprawne zbudowanie szkieletu sieci - *nodeMap* (K0)
* **300** – unieważnienie pliku o identyfikatorze *fileId* (K2)
* **301** – prośba o przesłanie pliku o identyfikatorze *fileId* (K2)
* **302** – przesłanie tablicy identyfikatorów plików (KD0)
* **304** – przesłanie pełnego pliku o identyfikatorze *fileId,* wraz z *nodeId* jego właściciela oraz zawartością *data* (KD1)
* **400** – błędne dołączenie do sieci, odbiorca komunikatu ma się niezwłocznie odłączyć

# Struktura programu:

Tworzony jest proces wielowątkowy, początkowo z 1 wątkiem bazowym, który:

* obsługuje komendy wprowadzane przez użytkownika
* po rozpoznaniu komendy tworzy nowy wątek do obsługi danej komendy o zadanych parametrach bądź uruchamia obsługę danej komendy we własnym wątku

Do edycji struktury *nodeFiles,* która może zmieniać się przez komendy *add* i *remove*, potrzeba zapewnienia synchronizacji między wątkami przy użyciu mutexa w przypadku zmiany tej struktury, tak samo używany jest mutex przy zmianie struktury *nodeMap*.

Do pozostałych nie potrzeba synchronizacji, ponieważ podczas innych komend struktury są jedynie odczytywane, nie zmieniane, zaś restrukturyzację sieci i zmianę wszystkich struktur obsługuje tylko jeden wątek.

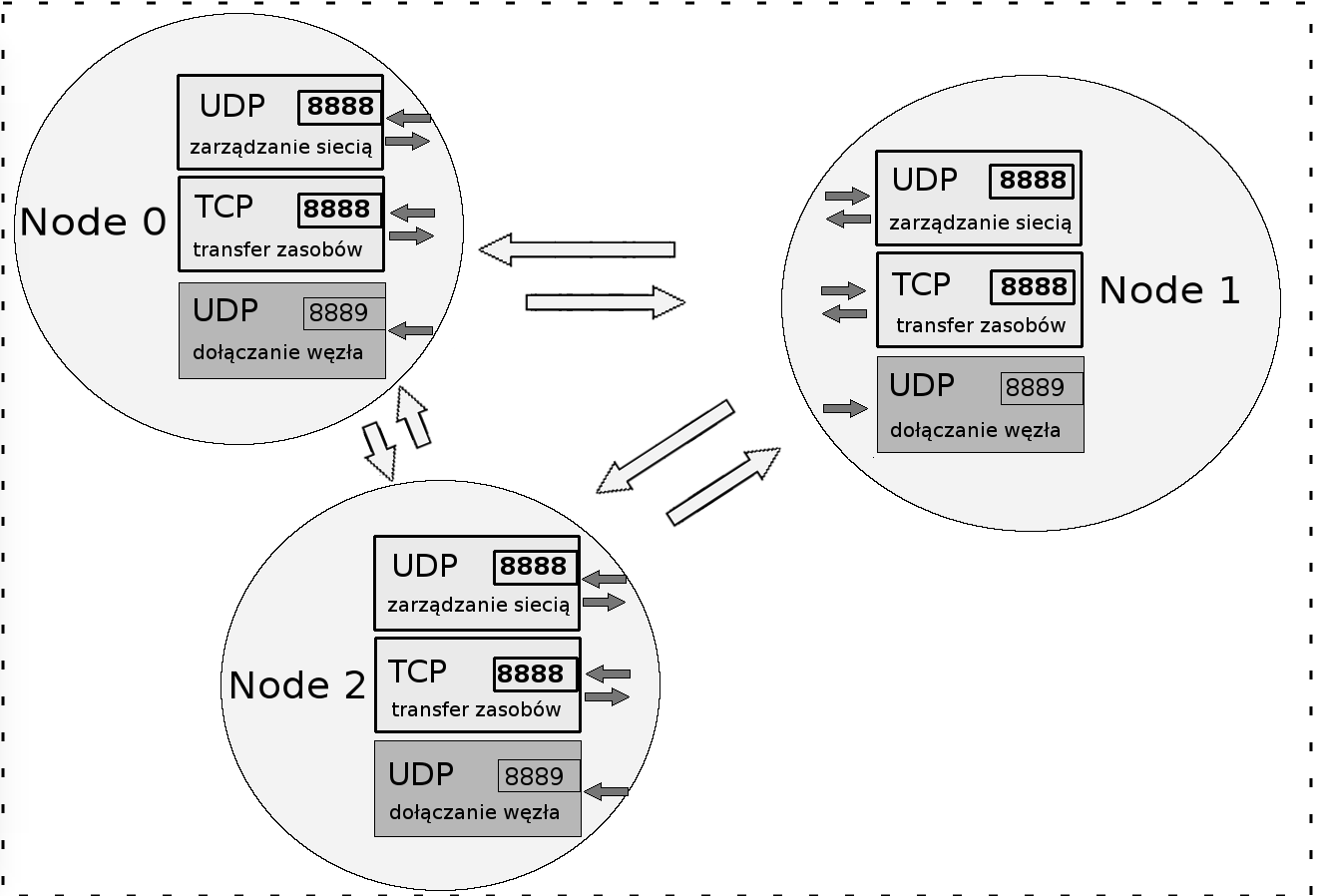
Szczególnie po uruchomieniu komendy *join*, nowo powstały wątek (*NetMainThread*) obsługuje nie tylko dołączenie do sieci, ale także już po dołączeniu/utworzeniu sieci:

* tworzy wątek serwera współbieżnego, odbierającego komunikaty po TCP
* reaguje na zmiany sieci i odpowiednio modyfikuje struktury w węźle:
  + dodaje nowy węzeł (+ rekonfiguruje sieć)
  + usuwa odłączający się węzeł (+ rekonfiguruje sieć)
  + przesyła komunikat o sieci dla nowego węzła
* jeśli potrzebny, tworzy nowy wątek do przesyłania danych do innych węzłów w sieci
* jeśli potrzebny, tworzy nowy wątek do usunięcia pliku z sieci

Komunikaty o chęci dołączenia/odłączenia od sieci, oraz potwierdzenie prawidłowego dołączenia do sieci są przesyłane protokołem UDP przez port 8888. Komunikaty węzłów o sieci dla nowego węzła, przesyłane są także protokołem UDP ale przez port 8889.

Po wydaniu komendy *leave*, nie powstaje nowy wątek. Wątek obsługujący komendy wysyła odpowiedni komunikat broadcastem i wszystkie węzły rozpoczynają procedurę jego odłączenia.

# Struktura komunikacji:



# Do podstawowej komunikacji w sieci używane są 3 gniazda utworzone na bazie API gniazd BSD.

# Do zarządzania siecią wykorzystywana jest komunikacja obustronna bazująca na protokole UDP i porcie 8888. W szczególności przez port 8888 UDP przesyłane są komendy do innych węzłów w celu wykonania danej akcji (np. przesłania pliku, rekonfiguracji sieci) oraz informacje o sieci dla nowych węzłów.

# Do przesyłania zasobów służy gniazdo obsługujące protokół TCP z portem 8888. Za jego pomocą są wysyłane i odbierane pliki oraz listy plików w sieci. Port 8888 protokołu TCP jest portem podstawowym z którego i do którego kierowane są wszystkie żądania tego typu, natomiast w przypadku transferu kilku zasobów jednocześnie wątek obsługujący port 8888 protokołu TCP deleguje osobne wątki do równoległych transferów przydzielając im osobne gniazda.

# W początkowej fazie “życia” węzła, podczas dołączania do sieci wykorzystywany jest osobny port protokołu UDP nr 8889. Przez ten port węzeł chcący dołączyć do sieci otrzymuje informacje o sieci od każdego z węzła będącego już w sieci. Rodzielenie komunikatów na dwa różne porty podyktowane było tym, iż podczas przyłączania nowego węzła nie musi on obsługiwać wszystkich komunikatów tj. odłączanie się wezła od sieci czy komunikat o następnym nowym węźle. Potwierdzenie poprawnego przyłączenia do sieci jest jednak wysyłane już standardowym dla aplikacji, portem 8888 protokołu UDP.

# Klasa NodeInfo

Klasa ta jest najważniejszym obiektem w programie. Przechowuje ona wszystkie informacje o aktualnej sieci P2P oraz udostępnia metody do zarządzania owymi strukturami.

Zmienna *size\_t nodeId* jest zmienną przechowywującą identyfikator danego węzła. Jest nadawana przy dołączaniu węzła do sieci, ale może być zmieniana podczas rekonfiguracji sieci (tj. odłączania/przyłączania węzła). W strukturze *nodeFiles* znajduje się także przy każdym pliku, wskazując na to, kto jest jego właścicielem. Przy zmianie *nodeId* na inny zgodnie zmieniają się identyfikatory właścicieli przy danych plikach.

Zmienna *size\_t nodeCnt* wskazuje na liczbę węzłów w sieci. Jest zwiększana za każdym razem gdy do sieci dołącza węzeł i zmniejszana gdy któryś z węzłów się odłącza. Jest to zmienna niezwykle krytyczna dla programu ponieważ na jej podstawie wylicza się wartość funkcji mapującej hash pliku na *nodeId* węzła, w którym powinien się znajdować.

Składowa *nodeFiles* jest kontenerem przechowującym informacje o plikach w danym węźle. Jest to słownik (*std::map*), który każdemu kluczowi będącemu hashem pliku przyporządkowuje obiekt *FileInfo* reprezentujący informacje o pliku. Słownik *nodeFiles* jest chroniony mutexem, który zapewnia atomowość operacji na zbiorze plików danego węzła. Obiekt *FileInfo* przechowywany w słowniku dla każdego identyfikatora (hasha) pliku jest krotką o 3 elementach. Pierwszy element jest identyfikatorem węzła, który jest właścicielem tego pliku. Drugi element krotki to licznik aktualnych transferów danego pliku. Ostatnim elementem jest zmienna warunkowa, która służy do sygnalizacji zakończenia wszystkich transferów danego pliku dla wątku oczekującego na usunięcie pliku z kontenera. Dzięki zmiennej warunkowej każdy transfer pliku ma gwarancję poprawnego zakończenia ponieważ wątek chcący usunąć plik ze zbioru najpierw zajmuje mutex dedykowany dla kontenera *nodeFiles* a następnie, w przypadku aktywnych transferów danego pliku zawiesza się na zmiennej warunkowej do momentu sygnalizacji zakończenia transferu. Na końcu, gdy licznik transferów jest równy 0, wątek może usunąć wpis danego pliku.

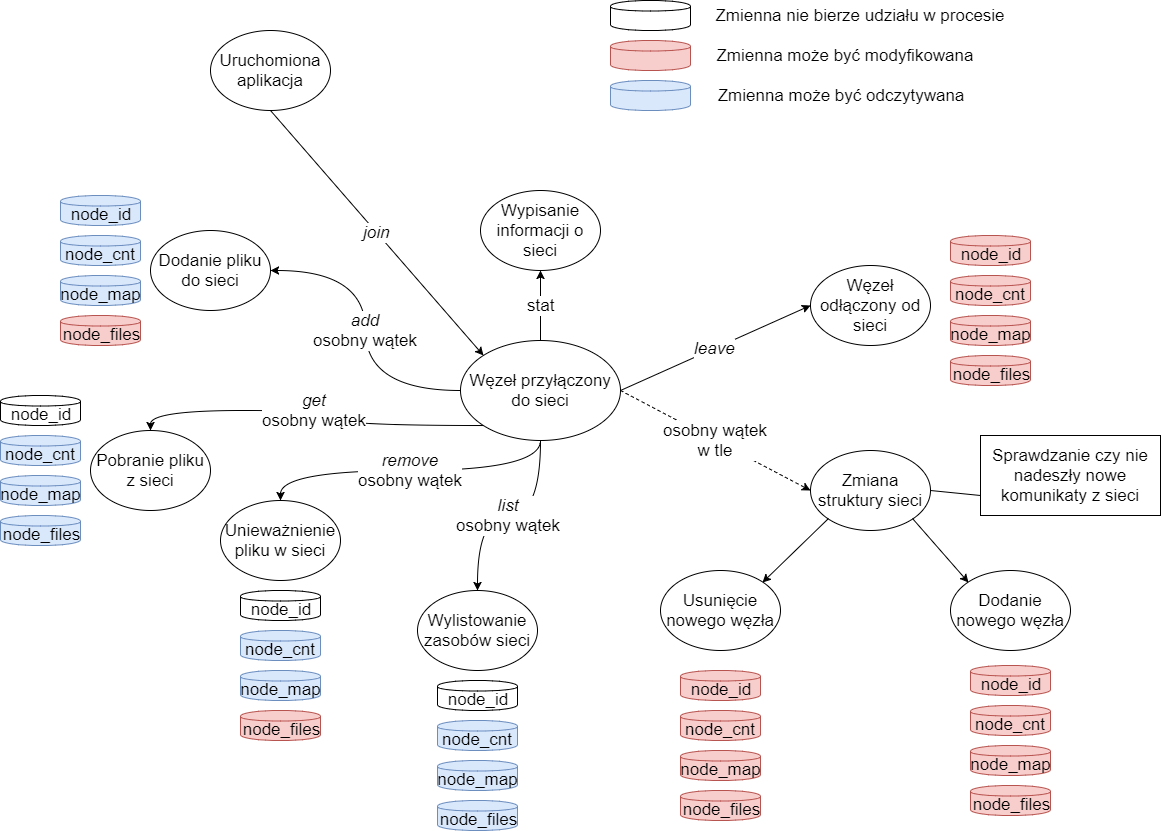
Składowa *nodeMap* przechowuje identyfikatory węzłów wraz z ich adresami IP, które umieszczone są bezpośrednio w strukturze *in\_addr*, tak aby łatwo można było do nich sięgnąć przy wysyłaniu komunikatów do innych węzłów przez sieć. Obsługa tej struktury w klasie *NodeInfo* nie jest banalna, z racji węzłów, które mogą odłączyć się nie będąc ostatnio dodanymi. W takich sytuacjach protokół wymaga aby ostatni węzeł, zamienił się identyfikatorem z węzłem odłączającym się. Oczywiście w strukturze *nodeFiles* dla plików, których właścicielem jest węzeł, którego identyfikator się zmienia, także muszą być zamienione odpowiednie zmienne w obiekcie *FileInfo*.

# Interfejs aplikacji

Aplikacja posiada jedynie interfejs tekstowy. Jest on zaprojektowany w sposób wygodny dla użytkownika, czyli:

* przy starcie aplikacji wyświetlana jest lista dostępnych komend wraz z ich krótkim opisem
* przy niewłaściwie wpisanej komendzie wyświetlany jest komunikat co należy wpisać aby zobaczyć ponownie listę dostępnych komend

Po uruchomieniu aplikacji wyświetlany jest komunikat o dostępnych komendach, którymi są:

* *join* – wymagany jako pierwsza komenda, powoduje próbę dołączenia do sieci, a jeśli ta się nie powiedzie – buduje nową sieć. Komenda może nie powieść się w ogóle, ze względu na to, iż inny węzeł będzie w tym samym czasie budował swoją sieć, bądź ze względu na nieprawidłową próbę dołączenia poprzedniego węzła, węzły w sieci spodziewają się innego komunikatu
* *add <filename>* - komenda mogąca być użyta tak jak każda następna (oprócz *leave* i *exit*) dopiero po komendzie *join*. Powoduje dodanie pliku o zadanej nazwie, znajdującego się w folderze aplikacji, do sieci P2P, wraz z informacją, iż węzeł, który go dodał, jest jego właścicielem
* *get <hash>* - powoduje pobranie z sieci pliku o zadanym identyfikatorze (hash’u). Po poprawnie odebranym pliku, zostaje wyświetlony komunikat o jego dodaniu (ale bez wpisu do *nodeFiles* danego węzła)
* *remove <hash>* - powoduje wysłanie komunikatu z żądaniem usunięcia pliku z sieci. Komunikat zadziała jedynie jeśli zawiera identyfikator węzła będącego właścicielem pliku. Po żądaniu usunięcia, transfery pliku są dokańczane, a dopiero potem jest on usuwany (zarówno jego wpis w tablicy *nodeFiles*, węzła go przechowywującego, jak i fizycznie z dysku danego węzła)
* *list* – powoduje wypisanie wszystkich identyfikatorów plików, będących aktualnie w sieci
* *stat* – powoduje wypisanie informacji o aktualnym stanie sieci P2P, takich jak: liczba węzłów w sieci, oraz identyfikatory poszczególnych węzłów wraz z ich adresami IP
* *leave* – powoduje wysłanie komunikatu o chęci odłączenia się od sieci, a po poprawnym odłączeniu się (dokończenie wszystkich transferów, rekonfiguracja plików w sieci) powoduje zakończenie programu
* *exit* – wymusza zamknięcie programu w trybie natychmiastowym, bez możliwości rekonfiguracji sieci i poprawnego odłączenia węzła. Komenda powinna być używana jedynie w przypadku błędów, gdy nie można odłączyć się od sieci komendą *leave*
* ****

Język: C/C++, środowisko Linux

Biblioteki:

* openssl (do obliczenia hashu pliku – MD5)
* pthread (do współbieżnego wykonywania funkcji)
* socket (do obsługi komunikacji sieciowej)
* inet (do obsługi adresów IP)

## **F1. Dołączanie węzła**

1. Użytkownik wydaje polecenie dołączenia do sieci P2P.
2. Program znając adres broadcastowy sieci, rozgłasza żądanie dołączenia do sieci P2P.
3. Każdy z węzłów w sieci wysyła odpowiedź w postaci swojego identyfikatora wraz z nadanym identyfikatorem nowo przyłączonego węzła oraz liczbą węzłów w sieci na porcie UDP 8889, a następnie ‘’oczekuje’’ (w zasadzie nie oczekuje, ale pamięta, iż takie żądanie się pojawiło) na potwierdzenie na porcie UDP 8888
4. Nowo przyłączony węzeł uzupełnia informacje o sieci P2P, do której dołączył oraz jeśli wszystko się zgadza tworzy wątek obsługujący odbiór plików po TCP i wysyła komunikat o poprawnym przyłączeniu do sieci.
5. Pozostałe węzły po otrzymaniu komunikatu potwierdzającego:
   1. Dodają wpis o nowym węźle do swojej *nodeMap*, oraz modyfikują liczbę węzłów w sieci - *nodeCnt*
   2. Hashują identyfikatory plików posiadanych lokalnie i wysyłają do odpowiednich węzłów, zostawiając te, które nadal powinni trzymać (funkcja hashująca jako argument przyjmuje liczbę węzłów w sieci)

Sytuacje wyjątkowe:

## E1.1. Brak odpowiedzi

Jeśli węzeł nie dostanie żadnej odpowiedzi po rozgłoszeniu broadcastem chęci dołączenia do sieci (tzn. nie ma sieci P2P), po określonym czasie, zaczyna tworzenie własnej:

1. Ustala swój identyfikatorna 0
2. Ustala liczbę węzłów w sieci na 1
3. Po czym nasłuchuje na porcie 8888 UDP na komunikaty o chęci dołączenia do jego sieci innych maszyn (możliwe dodawanie/usuwanie zasobów z sieci w tym samym czasie)

## E1.2. Przyłączenie węzła podczas przyłączania innego

W przypadku kiedy węzeł wysłał żądanie dołączenia ale nie wysłał jeszcze komunikatu potwierdzającego poprawne dołączenie, może zdarzyć się, iż inny węzeł także wyśle żądanie dołączenia, wtedy *nodeId* zostanie mu przydzielone bazując na lokalnym liczniku, który z każdym żądaniem jest zwiększany o 1, a z każdym odłączonym węzłem zmniejszany o 1.

W tej sytuacji wymuszona jest kolejność dodawania węzłów, w przypadku kiedy kolejność zostanie zaburzona (węzeł 1 wysłał chęć dołączenia -> węzeł 2 wysłał chęć dołączenia -> węzeł 2 wysłał potwierdzenie poprawnego dołączenia) licznik jest resetowany do ustawień sprzed wysłania żądań przez oba węzły, a do węzła, który wysłał potwierdzenie poza kolejnością wysyłany jest komunikat powodujący automatyczne odłączenie węzła od sieci.

## E1.3. Odłączanie węzła podczas przyłączania innego

Operacje wykonywane są sekwencyjnie, a mutexy zapewniają, że nie będą sobie przeszkadzać. W wypadku kiedy nowy węzeł wysłał żądanie dołączenia ale nie wysłał jeszcze potwierdzenia poprawnego przyłączenia, a inny węzeł wysłał w tym czasie chęć odłączenia, potwierdzenie nowego węzła oczekuje w kolejce, a obsługiwane jest żądanie odłączenia od sieci. Po poprawnym odłączeniu i rekonfiguracji sieci, następuje procedura przyłączania nowego węzła i ponowna rekonfiguracja.

## **F2. Odłączenie węzła**

1. Użytkownik wydaje polecenie odłączenia od sieci P2P.
2. Program broadcastuje informację o chęci odłączenia się od sieci.
3. Każdy z węzłów w sieci (włącznie z węzłem odłączającym się):
   1. Jeśli posiada lokalnie pliki oznaczone jako własność węzła o identyfikatorze węzła chcącego się odłączyć - usuwa je
   2. Jeśli węzeł chcący się odłączyć, nie dołączył do sieci jako ostatni to:
      1. Jego wpis w strukturze przechowywującej mapowania identyfikatorów na adresy IP zamieniany jest z ostatnio dołączonym
      2. Wszystkie pliki, których właściecielem jest ostatni węzeł mają zmieniane identyfikatory właściciela na jego nowy identyfikator
   3. Następuje redystrybucja plików w sieci, węzeł chcący się odłączyć musi przesłać wszystkie pliki, które nie są jego do odpowiednich węzłów

Sytuacje wyjątkowe:

## E2.1. Ostatni węzeł w sieci

Jeśli węzeł jest ostatnim węzłem w sieci (jego identyfikator równa się 0), usuwa struktury służące sieci i nie reaguje na komunikaty o chęci przyłączenia się do sieci innej maszyny.

## E2.2. Fizyczne odłączenie węzła od sieci

Jeśli przed odłączeniem węzła od sieci użytkownik nie wywoła komendy *leave*, a np. zamknie program, to komendy skierowane do tego węzła nie zostaną wykonane, zaś przy próbie przesłania pliku do odłączonego węzła zostanie wyświetlony komunikat o uszkodzonej sieci.

## E2.3. Przyłączanie nowego węzła w trakcie odłączania

Jeśli podczas odłączania przyjdzie komunikat od nowego węzła z prośbą o przyłączenie, rekonfiguracje sieci wykonywane są sekwencyjnie – najpierw odłączenie, następnie przyłączenie nowego węzła.

## **F3. Dodawanie zasobów do sieci**

1. Użytkownik wydaje polecenie dodania zasobu od sieci P2P.
2. Program oblicza hash wskazanego pliku, a następnie na podstawie informacji o liczbie węzłów w sieci, mapuje hash pliku na odpowiedni węzeł w sieci, po czym przesyła plik wraz z identyfikatorem i informacją kto jest jego właścicielem do odpowiedniego węzła.

Sytuacje wyjątkowe:

## E3.1. Zmiana sieci podczas transferu pliku

Jeśli podczas transferu pliku pojawi się komunikat o zmianie sieci (dołączenie/odłączenie węzła) to wszystkie transfery są dokańczane, a następnie sieć jest rekonfigurowana.

## **F4. Usuwanie zasobów z sieci**

1. Użytkownik wydaje polecenie usunięcia zasobu z sieci P2P.
2. Program na podstawie liczby węzłów w sieci oblicza odpowiednią funkcją mapującą identyfikator węzła, w którym znajduje się zasób.
3. Program przesyła do węzła komunikat z prośbą o unieważnienie danego pliku.
4. Jeśli węzeł stwierdzi, że węzeł proszący o unieważnienie pliku jest jego właścicielem, dokańcza transfery danego pliku jeśli istnieją, a następnie usuwa jego wpis z lokalnej tablicy oraz usuwa plik fizycznie z dysku.

Sytuacje wyjątkowe:

## E4.1. Zmiana sieci podczas transferu pliku

Jeśli podczas transferu zasobu pojawi się komunikat o zmianie sieci (dołączenie/odłączenie węzła) to wszystkie transfery są dokańczane, a następnie sieć jest rekonfigurowana.

## **F5. Pobieranie zasobów z sieci**

1. Użytkownik wydaje polecenie pobrania zasobu o danym identyfikatorze z sieci P2P.
2. Program na podstawie liczby węzłów w sieci oblicza odpowiednią funkcją mapującą identyfikator węzła, w którym znajduje się zasób.
3. Program przesyła do węzła komunikat z prośbą o plik o danym identyfiaktorze.
4. Węzeł przesyła plik wraz z identyfikatorem i identyfikatorem jego właściciela.

Sytuacje wyjątkowe:

## E3.1. Zmiana sieci podczas transferu pliku

Jeśli podczas transferu pliku pojawi się komunikat o zmianie sieci (dołączenie/odłączenie węzła) to transfer jest dokańczany niezależnie od struktury sieci.

## **F6. Informacja o zasobach w sieci**

1. Użytkownik wydaje polecenie pobrania listy zasobów będących w sieci P2P.
2. Program odpytuje wszystkie węzły o ich lokalnie posiadane pliki.
3. Węzły przesyłają swoje struktury zawierające identyfikatory plików.
4. Wszystkie identyfikatory są łączony w jedną listę i wyświetlane na konsoli programu.

# Testy aplikacji

Przy testowaniu poprawności normalnego działania, w sieci znajdowały się 4 węzły i 18 plików. Podczas testów udało uzyskać się wymagane sytuacje wyjątkowe i odpowiednie na nie reakcje:

1. Przy 2 węzłach próbujących wspólnie budować sieć, jeden z nich zawsze był w stanie wysłać komunikat drugiemu o budowanej sieci, tym samym przerywając jego pracę i wymuszając na nim próbę ponownego dołączenia do sieci
2. Przy komendzie *leave* w węźle, którego pliki były akurat przesyłane, pliki pozostały w sieci, aż do zakończenia transferu, a w zależności czy węzeł był akurat w ich posiadaniu i to on udostępniał transfer czy też były one w innym węźle, węzeł właściciela czekał na koniec transferu (pliki w węźle właściciela) bądź też od razu odłączał się od sieci (pliki właściciela w innym węźle)
3. Przy próbie przeplatanego dołączania węzłów do sieci, sieć nie przyjmowała żadnego z nich i resetowała swoje liczniki równocześnie wysyłając do węzłów komunikat (opcode = 400) o natychmiastowym odłączeniu się od sieci