Techniki Internetowe (TIN)

Dokumentacja

System wymiany plików w sieci P2P

Listopad 2018

1. Nazwa Systemu

System wymiany plików w sieci P2P

2. Tematyka projektu

Zadaniem systemu jest ułatwienie wymiany plików między zaufanymi hostami.

3. Wymagania funkcjonalne

3.1. Wymagania funkcjonalne

3.1.1. Użytkownik może stworzyć nową sieć lub dołączyć do już istniejącej.

3.1.2. Użytkownicy dołączający do sieci powinni być autoryzowani przez osobę będącą już w sieci.

3.1.3. Użytkownik może wyszukać dany plik przy pomocy funkcji Szukaj?

3.1.4. Nieukończone pobieranie pliku zostanie automatycznie wznowione po ponownym uruchomieniu aplikacji klienckiej jeśli plik nadal znajduje się w sieci

3.1.5. Użytkownik może dodać zasoby do sieci poprzez umieszczenie danych plików w folderze udostępnionym w sieci

3.1.6. Użytkownik może wyświetlić listę plików udostępnianych w sieci przez innych użytkowników

4. Przypadki użycia

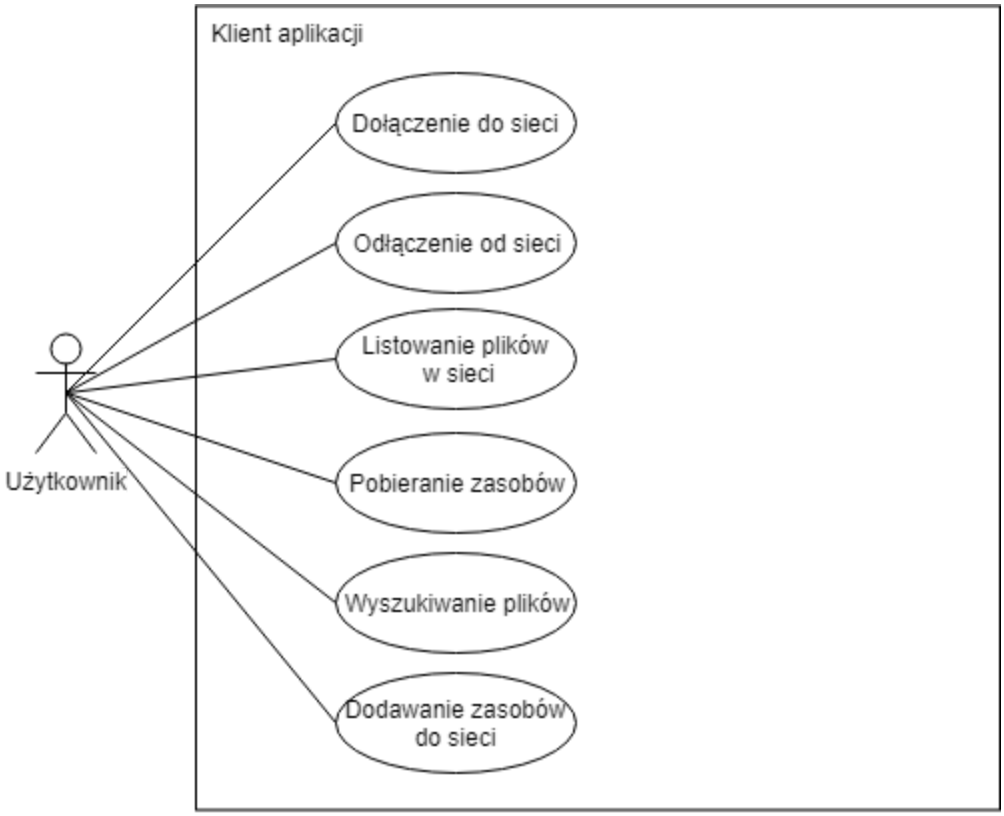

Diagram przypadków użycia

SG - scenariusz główny
SA - scenariusz alternatywny

4.1. Dołączanie węzła do sieci P2P

4.1.1.1. Po uruchomieniu programu wyświetla się lista poprzednio zaufanych IP, które możemy wybrać do połączenia lub wprowadzić własny adres i port.

4.1.2. **SG**: Istnieje węzeł o podanym IP

4.1.2.1. Następuje połączenie sesji TCP z wybranym węzłem.

4.1.2.2. Węzeł wysyła na listę użytkowników w sieci.

4.1.2.3. Nawiązujemy połączenie w wybraną liczbą węzłów.

4.1.3. **SA**: Nie można nawiązać połączenia z podanym adresem

4.1.3.1. Wyskakuje komunikat że nie można połączyć się z wybranym węzłem i należy podać inny węzeł

4.1.4. **SA**: Chcemy założyć sieć, jesteśmy pierwsi w sieci.

4.1.4.1. Użytkownik klika przycisk “załóż własną sieć”

4.1.4.2. Do tablicy węzłów wpisuje samego siebie

4.2. Odłączanie węzła od sieci P2P

4.2.1. **SG**: Odłączenie

4.2.1.1. Użytkownik klika przycisk odłączenia od sieci.
4.2.1.2. Węzły połączone z danym użytkownikiem wyrzucają go z listy obecnych w sieci.
4.2.1.3. Wyświetla się lista zaufanych IP.

4.3. Wprowadzanie przez użytkownika do sieci P2P nowych zasobów.

4.3.1. **SG**: Wprowadzenie plików
4.3.1.1. Użytkownik klika przycisk “Dodaj plik”.
4.3.1.2. Wyświetla się systemowe okno eksploratora plików.
4.3.1.3. Przeciągamy plik do folderu.

4.4. Usuwanie przez właściciela zasobu w sieci

4.4.1. **SG**: Usuwanie plików
4.4.1.1. Usuwamy plik z folderu.

4.5. Wyszukiwanie zasobów w sieci po nazwie

4.5.1. **SG**: Node z którym mamy zestawione połączenie ma plik o danej nazwie
4.5.1.1. Użytkownik wpisuje nazwę szukanego pliku
4.5.1.2. Węzły, z którymi mamy zestawione połączenie deklarują posiadanie pliku
4.5.2. **SA**: Node z którym mamy zestawione połączenie nie ma pliku o danej nazwie
4.5.2.1. Użytkownik wpisuje nazwę szukanego pliku
4.5.2.2. Węzły, z którymi mamy zestawione połączenie odpytują węzły, z którymi one mają zestawione połączenie
4.5.2.3. Zwracane są adresy węzłów posiadających plik

4.6. Pobieranie nazwanych zasobów z sieci P2P

4.6.1. **SG**: Pobieramy plik od jednego Node’a(tylko on ma ten plik)
4.6.1.1. Plik zostaje zapisany do folderu
4.6.2. **SA**: Pobieramy plik od kilku Node’ów (Kilka Node’ów ma interesujący nas plik)
4.6.2.1. Plik zostaje połączony i zapisany w folderze

4.7. Wznowienie pobierania pliku

4.7.1. **SG**: Plik jest obecny w sieci
4.7.1.1. Pobierana jest brakująca część pliku (od zapisanego offsetu)
4.7.1.2. W zależności czy dany plik udostępnia 1 lub więcej osób, pobrane fragmenty są łączone i zapisywane do folderu.
4.7.2. **SA**: Plik nie jest obecny w sieci
4.7.2.1. Jeżeli żaden z węzłów z którymi mamy aktywne połączenie nie ma szukanego pliku, następuje zapytanie przez te węzły do węzłów z którymi mają one aktywne połączenie.
4.7.2.2. W przypadku niepowodzenia wyświetlany jest o tym komunikat.

4.8. Listowanie plików w sieci

4.8.1. **SG**:Użytkownik wyświetla pliki w sieci

Moduł view będzie tylko zawierał szablony widoków w plikach .xml mających odwołania do odpowiednich controllerów.
Moduł controller będzie służył jako “pośrednik” między modułem view, a model, którym będzie biblioteka (dokładniej opisana w osobnym punkcie). Controller będzie obsługiwał żądania użytkownika i odpowiedzi z niższej warstwy aplikacji.
Moduł model będzie występował jako zewnętrzna biblioteka P2PJavaTools.jar. Komunikacja z tą warstwą będzie zachodziła dzięki API tej biblioteki.

6.3. Android

Aplikacja kliencka na system Android zostanie zaimplementowana zgodnie ze wzrocem MVP (Model View Presenter).

Moduł View składać się będzie z “plików layautowych” .xml oraz klas korzystających z API Android dziedziczących po Activity oraz Fragment. View jest odpowiedzialny wyłącznie za poprawne wyświetlanie interfejsu graficznego użytkownika.

Moduł Presenter jest odpowiedzialny za zupełne odseparowanie widoku od modelu. Obsługuje żądania z View zleca wykonanie odpowiednich zadań na Modelu, który za pomocą callback’ów asynchronicznie zwraca rezultaty. Wyniki operacji są prezentowane poprzez metody modułu View.

Moduł Model występuje jako zewnętrzna biblioteka P2PJavaTools.jar. Komunikacja z tą warstwą zachodzi dzięki API tej biblioteki.

Każda z funkcjonalności aplikacji zostanie zaimplementowana w oddzielnym pakiecie oraz będzie posiadała własny moduł View oraz Presenter.

6.4. P2PJavaTools

Biblioteka Javy zawierająca logikę biznesową aplikacji oraz warstwy sieciowe.

Warstwa logiki biznesowej - odpowiedzialna za realizację wymagań funkcjonalnych oraz odpowiednie delegowanie zadań do warstw niższych.

Warstwa komunikacyjna (sieciowa) - wykorzystujące API gniazd umożliwiającym posługiwanie się protokołem TCP w celu nawiązania łączności między hostami oraz przesyłu danych.

Warstwa serializacji i deserializacji - odpowiedzialna za unifikację przesyłanych danych w celu zapewnienia kompatybilności między różnymi technologiami oraz systemami operacyjnymi.

7. Sposób testowania

7.1. Testy jednostkowe

Przetestowane zostaną poszczególne metody krytyczne. Testy jednostkowe umożliwią szybkie wychwytywanie drobnych błędów implementacyjnych bez potrzeby uruchamiania aplikacji.

7.2. Testy automatyczne

Testy integracyjne przy wykorzystaniu narzędzia do symulowania działania użytkownika.

7.3. Testowanie manualne

Testy wykonywane przez użytkownika (testera), który ostatecznie potwierdzi spełnienie wszystkich przypadków użycia oraz funkcjonalności.
Testowanie to będzie wymagało podłączenie innych (niż testowane) fizycznych hostów do sieci lub wykorzystanie narzędzia Docker do symulacji wielu użytkowników sieci na jednym komputerze.

8. Sposób demonstracji rezultatów

8.1. Prezentacja pomyślnie ukończonych testów jednostkowych i automatycznych.
8.2. Prezentacja poprawnego działania aplikacji podczas prezentacji przypadków użycia.

9. Podział prac w zespole

9.1. Michał Mokrogulski - implementacja aplikacji desktopowej (C++)
9.2. Piotr Czyż - implementacja aplikacji desktopowej (C++)
9.3. Irek Wróbel - implementacja aplikacji desktopowej (Java), warstw biznesowych, sieciowych oraz testów jednostkowych i automatycznych.
9.4. Marcin Dziedzic - implementacja aplikacji mobilnej (Android), warstw biznesowych, sieciowych oraz testów jednostkowych i automatycznych. Koordynator ds. Spotkań.

Protokoły

Węzeł A: łączy się z siecią
Węzeł B: jest w sieci

Dołączanie węzła do sieci P2P

1. Węzeł A nawiązuje połączenie TCP z B.
2. Węzeł A wysyła prośbę o wysłanie soli.

OpCode(int 4b) REQUEST_FOR_SALT = 50

3. Węzeł B wysyła sól (losowy int) przypisany do węzła A

OpCode(int 4b) SLAT_FOR_HASH = 52
salt (itn 4b)

4. Węzeł A haszuje wprowadzone przez użytkownika hasło, następnie dodaje do niego sól, po czym ponownie hashuje i wysyła do węzła B

OpCode(int 4b) WANT_TO_JOIN_INIT = 10
passwordHash 64 bit

5. Węzeł B akceptuje hasło

OpCode PASS_RESPONSE = 11
Bool 1 bit

6. Węzeł B wysyła rekordy obecności w sieci.

OpCode LIST_OF_KNOWN_NODES = 12
Int nRecords
32 bitów adres (IP jako int)
...

7. Węzeł A nawiązuje połączenie TCP z węzłami, których adresy i porty otrzymał w poprzednim kroku.

8. Węzeł A wysyła żądanie przesłania soli do w.w. węzłów.

OpCode(int 4b) OPCODE_REQUEST_FOR_SALT_IN_THE_SAME_NET = 51

9. Węzły wysyłają sól (losowy int) przypisaną do węzła A.

OpCode SALT_FOR_HASH_IN_THE_SAME_NET = 53
salt (itn 4b)

10. Węzeł A do hasha hasła dodaje do sól, po czym ponownie hashuje i wysyła do węzłów.

OpCode WANT_TO_JOIN = 13
passwordHash 64 bit

11. Węzeł otrzymujący Opcode 13 weryfikuje hasło.

OpCode PASS_RESPONSE = 11
Bool 1 bit

Listowanie plików w sieci

1. Węzeł A wysyła zapytanie o listę plików:

OpCode FILES_LIST = 20

2. Węzeł B wysyła listę plików

OpCode LIST_OF_FILES = 21		
Int nRecilords		
Nazwa pliku char[64]	Hash pliku char[64] SHA-256 z zawartosci	Rozmiar pliku w Bajtach (8 bajtów)
...	...	...

Pobieranie nazwanych zasobów z sieci P2P

1. Węzeł A wysyła prośbę pobrania części pliku

OpCode FILE_FRAGMENT_REQUEST = 30

Offset pliku 64 bity	Hash pliku char[64] SHA-256 z zawartosci

2. Jeżeli węzeł B nie posiada takiego pliku wysyła o tym informację. W p.p. Wykonuje punkt 3.

OpCode DON_HAVE_FILE = 31

3. Węzeł B wysyła część pliku 1MB

OpCode FILE_FRAGMENT = 32		
dataLength	Offset pliku 64 bity	Hash pliku char[64] SHA-256 z zawartosci
data[dataLength]		

Ochrona przed nieuprawnionym dostępem do sieci

W przypadku kiedy węzeł A wyśle żądanie do węzła B nie będąc uprzednio zautoryzowanym mechanizmem opisanym w sekcji *Dołączenie węzła do sieci P2P* , węzeł B odsyła wiadomość postaci:

OpCode NOT_AUTHORIZED = 40