

Cezary Zawadka  
Klaudia Zamczewska

## Dokumentacja OINS

Temat projektu: Sieć nakładkowa P2P  
z ochroną informacji.

## Zawartość

1. Cel projektu .....	3
2. Wykorzystane narzędzia .....	3
3. Sposób realizacji.....	3
4. Algorytm dystrybucji zasobów.....	3
5. Bezpieczeństwo .....	4
6. Scenariusz działania programu .....	4
7. Redundancja aktualizacja stanu sieci .....	5
8. Przykład stanu w sieci.....	6
9. Kod źródłowy .....	8
10. Interfejs użytkownika.....	9
11. Podsumowanie .....	11
12. Opis załączonych plików.....	12

## **1. Cel projektu**

Celem projektu jest stworzenie oprogramowania sieciowego, który pozwoli na bezpieczne wysyłanie plików (mp3, tekstowych, JPG, itd.) między równorzędnymi sobie użytkownikami (peerami). Założyliśmy, że komunikacja pomiędzy użytkownikami ma być poufna(szyfrowana), oraz, że użytkownicy będą uwierzytelniani za pomocą certyfikatów podpisanych przez zaufaną trzecią stronę. Ponadto użytkownicy mają tworzyć sieć nakładkową działającą w warstwie aplikacji stosu TCP/IP.

## **2. Wykorzystane narzędzia**

Oprogramowanie zostało napisane w języku Java, w środowisku programistycznym Eclipse. Interfejs użytkownika zaprojektowano w Netbeans IDE. Język ten wybrano ze względu na proste i efektywne tworzenie atrakcyjnie wyglądającego interfejsu użytkownika (GUI) w stosunku do innych języków programowania, a także bogate API umożliwiające tworzenie bezpiecznego oprogramowanie (bezpiecznej komunikacji sieciowej) dostarczone razem ze standardową wersją SDK. Wykorzystane API ze standardowej wersji języka Java to: Java Cryptography Extension oraz Java Secure Socket Extension ponadto wykorzystano biblioteki org.bouncycastle.

## **3. Sposób realizacji.**

W trakcie realizacji projektu zaimplementowano hybrydową sieć P2P - z wyróżnionym węzłem - tak zwanym serwerem. Zadaniem serwera jest przechowywanie informacji o obecnych użytkownikach (zwykłych węzłach sieci – peer'ach ) oraz wystawianie certyfikatów dla zaufanych węzłów, podpisanych przez swój klucz prywatny. Przekazywanie plików pomiędzy użytkownikami sieci odbywa się całkowicie bez udziału serwera dzięki wykorzystaniu algorytmu dystrybucji informacji o zasobach w sieci bazującego na rozproszonej tablicy haszującej wzorowanej na algorytmie Chord. Wybraliśmy architekturę hybrydową, gdyż była prostsza w implementacji, a także pozwalała na łatwą implementację wzajemnego uwierzytelnienia użytkowników.

## **4. Algorytm dystrybucji zasobów**

Zaimplementowany został algorytm dystrybucji zasobów, który ma na celu zminimalizowanie roli serwera. Nasz wzorowany jest na algorytmie Chord, jest jego uproszczoną wersją. Użytkownik po uwierzytelnieniu (za pomocą loginu i hasła) na serwerze (i ewentualnym otrzymaniu ważnego certyfikatu) i otrzymaniu adresów (i portów), na których osiągalni są inne węzły oblicza skrót z nazwy wszystkich plików, które chce udostępnić w

sieci SHA1(nazwaPliku). Następnie przesyła informacje o każdym swoim pliku do węzła sieci, którego skrót adresu (SHA1(address)) ma wartość największą, lecz nie większą od SHA1(nazwaPliku). W ten sposób każdy węzeł w sieci przechowuje informacje o pewnej części zasobów (informacje zawierają skrót z nazwy pliku, oraz adresy i porty węzłów, na których te pliki faktycznie się znajdują).

Aby pobrać plik z sieci węzeł zainteresowany pobraniem oblicza skrót SHA1(nazwaPliku) a następnie sprawdza na najbliższym węźle którego SHA1(address)) < SHA1(nazwaPliku) czy dany plik jest dostępny w sieci a także skąd można go pobrać.

Topologie sieci można wyobrazić sobie, jako punkty na okręgu (węzły) ułożone według rosnącej wartości skrótu ze swojego adresu. Każdy węzeł przechowuje informacje, których skrót „wypada” na łuku pomiędzy danym a kolejnym węzłem.

## **5. Bezpieczeństwo**

W celu zapewnienie poufności komunikacja odbywa się za pomocą protokołu SSL. W komunikacji pomiędzy węzłem a serwerem uwierzytelniany jest jedynie serwer (użytkownik przesyła login oraz hasło, bez tego dalsza komunikacja jest niemożliwa). W komunikacji pomiędzy węzłami uwierzytelniane są obie strony połączenia. Certyfikaty potrzebne do uwierzytelnienia podpisywane są kluczem prywatnym serwera(jest on tak zwaną zaufaną trzecią stroną).

## **6. Scenariusz działania programu**

- Jako pierwszy uruchamiany jest „serwer”. Jego zadaniem będzie m.in. posiadanie informacji, jakie hosty (peer’y) są w sieci. Przechowuje adresy, numery portów a także skróty SHA1 z adresów.

- W momencie, gdy serwer zacznie nasłuchiwanie, możliwe jest połączenie peerów z serwerem. Aby to połączenie się nawiązało należy wpisać zmienne (dane) w interfejsie użytkownika. W przypadku, gdy wszystkie dane będą wypełnione nastąpi próba inicjalizacji Peera i połączenia z serwerem. W przeciwnym razie zostanie wyświetlony komunikat o błędzie. Jeżeli połączenie zostanie nawiązane, serwer zapisuje dane o nowym węźle i przekazuje mu informacje o obecnych w sieci węzłach. Serwer będzie używany jedynie w przypadku odświeżania, aktualizacji listy węzłów należących do sieci oraz do generowania certyfikatów.

- W momencie, gdy węzeł (peer) uzyskuje połączenie z serwerem pojawia się nowe okno, za pomocą którego peer może wyszukać, a później pobrać interesujący go plik. Tworzony jest skrót z nazwy tego pliku, pomijając rozszerzenie, a następnie wysyłane jest zapytanie o posiadacza pliku do węzła znajdującego się w sieci, który ma informacje, na jakim hoście znajduje się poszukiwany plik. Następnie ponownie wysyłane jest zapytanie jednak tym razem

do Peera posiadającego plik. Gdy Peer upewni się, że ten plik rzeczywiście się tam znajduje możemy pobrać plik.

-Po kliknięciu download następuje ściągnięcie pliku, pojawi się komunikat o ściąganiu pliku. Z tego miejsca można ponownie wrócić do panelu wyszukującego i pobierającego plik lub wyjść z programu.

## **7. Redundancja aktualizacja stanu sieci**

W celu ochrony przed utratą informacji o dostępnych zasobach stworzono prosty mechanizm redundancji – wszystkie informacje o zasobach (to, gdzie dostępne są dane pliki) przechowywane są w dwóch węzłach – w węźle, który wynika z algorytmu dystrybucji zasobów oraz w węźle wcześniejszym(pierwszym, którego wartość skrótu adresu jest mniejsza). Jest to tak zwany backup.

Ponadto, w celu bieżącej aktualizacji topologii sieci zaimplementowany został mechanizm sprawdzania kolejnego sąsiada – co pewien czas każdy węzeł sprawdza czy kolejny jest aktywny (następuje również synchronizacja backupu oraz sprawdzanego węzła).

W przypadku zniknięcia węzła poinformowany zostaje serwer. Ponadto zaktualizowane zostają informacje o zasobach (węzeł, który wykrył zniknięcie posiada backup informacji z nieobecnego węzła)

## 8. Przykład stanu w sieci

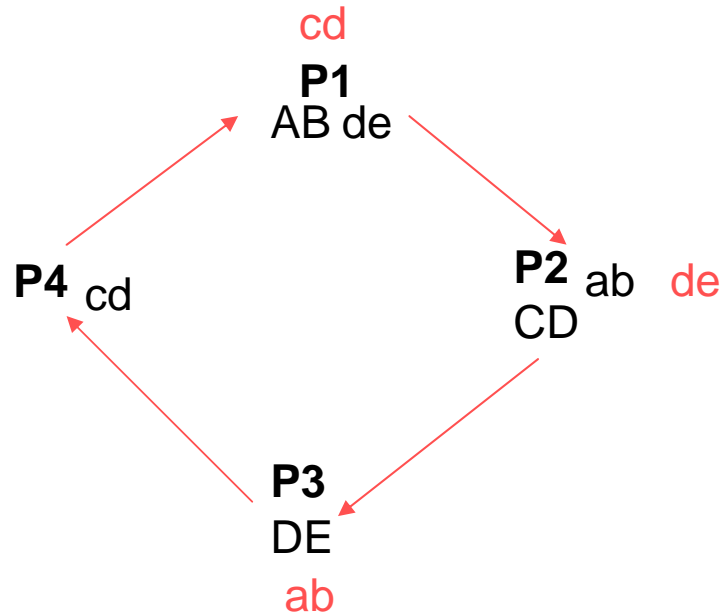
Poniżej pokazany został przykładowy stan sieci.

Oznaczenia:

Pliki: ABCDE [fizyczna obecność w węzłach sieci]

Informacje o plikach: abcde

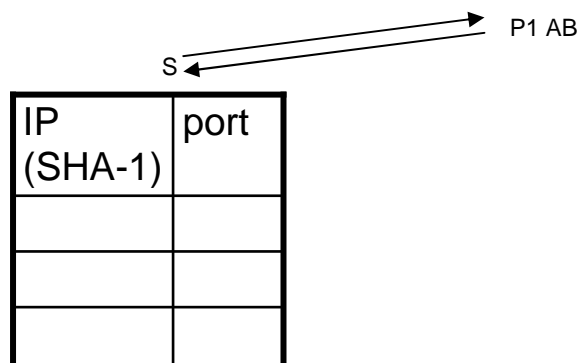
Węzły P1 P2 P3 P4 P5



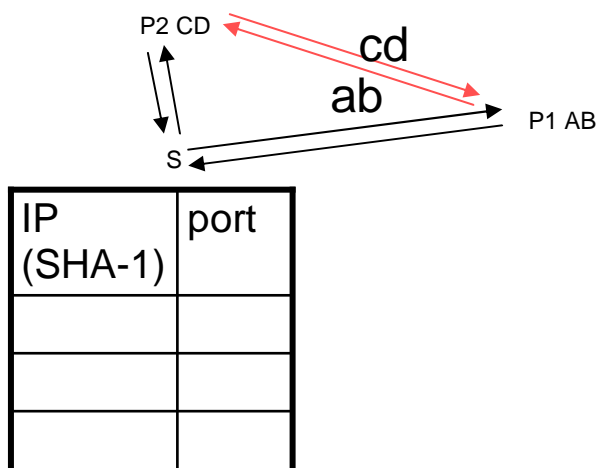
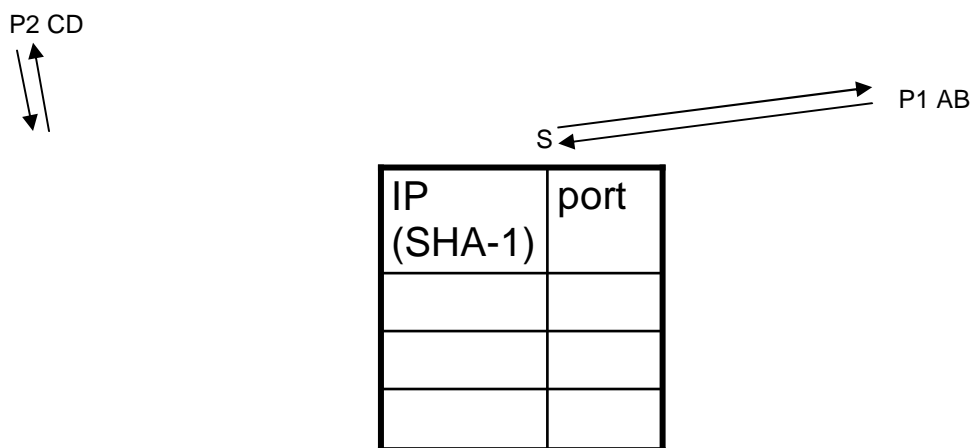
Czerwonym kolorem zostały zaznaczone informacje, które zostały powielone. Gdyby nie zostały powielone w momencie, gdy peer P4 odłączyłby się z sieci utracone zostałyby informacje o plikach C i D i mimo że byłby dostępne w sieci to nie można byłoby ich pobrać. Powielając informacje dane na temat plików cd ma jeszcze peer P1, dzięki czemu pliki cd są dostępne dla użytkowników sieci.

Moment inicjalizacji sieci, etapy:

- 1) Uruchamiamy jest serwer S
- 2) Pierwszy Peer z plikami AB łączy się z serwerem. Serwer zapisuje informacje o istnieniu pierwszego Peera w sieci i udostępnia port i hasz z adresu IP oraz wysyła informacje, że jest jedynym, Peerem w sieci.



- 3) Drugi Peer przyłącza się do sieci analogicznie jak w poprzednim podpunkcie i łączy się z Peerem, aby wysłać informacje o swoich plikach i uzyskać informacje o plikach Pierwszego peera.



4) Analogiczna sytuacja dzieje się dla większej ilości peerów, gdzie nowo przyłączony peer łączy się z jednym z dostępnych w sieci i wysyła do niego informacje o swoich plikach jak również pobiera informacje o plikach znajdujących się w peerze.

## 9. Kod źródłowy

Poniżej znajduje się opis najważniejszych klas.

**CertInfo:**

Klasa z informacjami od peera na temat tego, co ma się znaleźć w certyfikacie

**Connection:**

Klasa bazowa połączeń sieciowych.

**FileInfo:**

Klasa zawierająca informacje o pliku tj. nazwę, typ, rozmiar, skrót nazwy, adres właściciela itp.

**P2PProtocol:**

Protokół komunikacji pomiędzy peerami

**P2SProtokol:**

Protokół komunikacji peera z serwerem

**PeerActionObserver**

Interfejs pozwalający na śledzenie zdarzeń zachodzących w obiekcie klasy Peer (węzeł).

**PeerInfo**

Klasa reprezentująca informacje o węźle – adres sieciowy, numer portu na który można się połączyć, skrót z adresu itp.

**PeerLoginInfo**

Klasa reprezentuje informacje potrzebne do logowania.

\*Wykorzystywana zarówno w klasie Server jak i Peer. Zmienne te są danymi wprowadzanymi przez użytkownika, wykorzystywane do inicjalizacji peera, oraz do poprawnego połączenia peera z serwerem.

**ServerInfo**

Klasa reprezentuje informacje o serwerze.

**Utils**

Zawiera funkcje globalne.



### P2PConnection

Obiekt tej klasy powstaje w czasie połączenia między dwoma węzłami, obsługuje połączenia p2p.

### Peer

Klasa węzła. Implementuje algorytm dystrybucji i wyszukiwania zasobów, algorytm sprawdzania obecności sąsiadów oraz algorytm back-up'u.

### P2SConnection

Obiekt tej klasy obsługuje połączenie między węzłem a serwerem od strony węzła

### S2PConnection

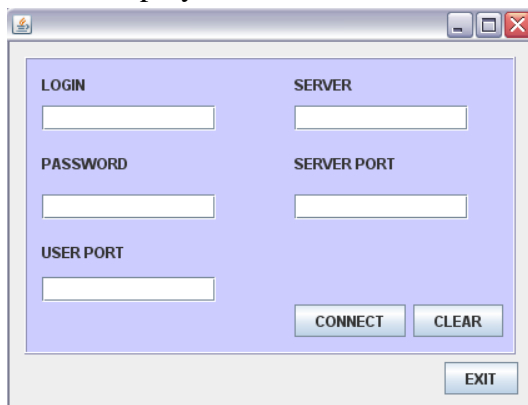
Obiekt tej klasy obsługuje połączenie między serwerem a węzłem od strony serwera

### Serwer

Klasa serwera. Obiekt tej klasy przechowuje informacje o obecnych węzłach, uwierzytelnia węzły, które chcą nawiązać połączenie (za pomocą loginu i hasła) a także generuje certyfikaty dla wzajemnego uwierzytelnienia węzłów

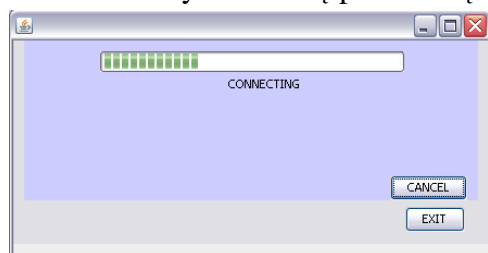
## 10. Interfejs użytkownika

Frame1- pierwsze okno programu, tutaj użytkownik wpisuje dane, aby móc połączyć się do serwera, i przynależać do sieci..

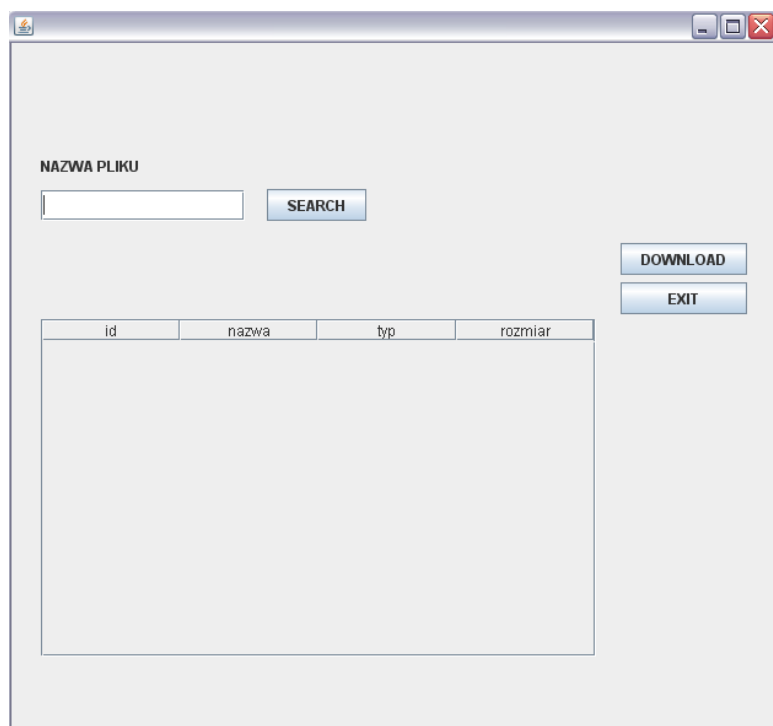


The screenshot shows a graphical user interface window titled 'Frame1'. The window has a light blue background and a white border. It contains several input fields and buttons. On the left side, there are three input fields labeled 'LOGIN', 'PASSWORD', and 'USER PORT'. On the right side, there are two input fields labeled 'SERVER' and 'SERVER PORT'. At the bottom right, there are three buttons: 'CONNECT', 'CLEAR', and 'EXIT'. The 'CONNECT' and 'CLEAR' buttons are grouped together, and the 'EXIT' button is below them.

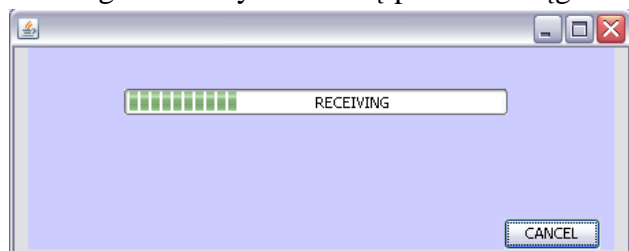
Grk2-okno wyświetla się podczas łączenia zainicjowanego peera z serwerem



Frame3-okno, które ukazuje się po udanym połączeniu s serwerem i pobraniu informacji o peerach przynależących do sieci. W oknie można wpisać nazwę poszukiwanego pliku a w tablicy listę znalezionych pasujących plików od wybranego peera, które można ściągnąć.



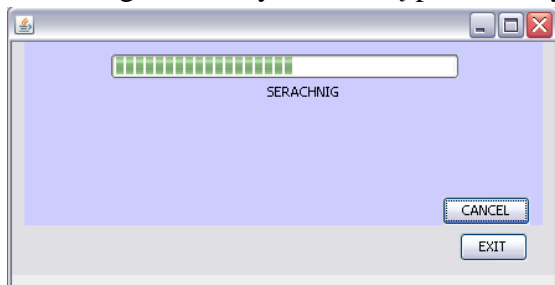
Sending4-okno wyświetla się podczas ściągania wyszukanego pliku



Receiving5-okno wyświetla się, gdy plik zostanie w całości odebrany przez użytkownika



Searching6-okno wyświetla się podczas wyszukiwania konkretnego pliku



## 11. Podsumowanie

Stworzone oprogramowanie implementuje sieć hybrydową P2P umożliwiającą łatwą wymianę plików. Komunikacja odbywa się w sposób szyfrowany, uwierzytelniane są obie strony połączenia.

Możliwe usprawnienia zaproponowanego rozwiązania obejmują:

- zmniejszenie udziału serwera w komunikacji tak, aby potrzebny był jedynie do generowania certyfikatów (bez przechowywania informacji o topologii sieci).
- usprawnienie protokołu sieciowego tak, aby przesyłane było mniej danych o związanych z utrzymaniem sieci (zwłaszcza w początkowej fazie działania węzła).
- usprawnienie protokołu sieciowego tak, aby odporny był na błędne zachowanie węzłów – między innymi wprowadzenie time outów.
- zwiększenie redundancji – w tej chwili szybkie odłączenie dwóch kolejnych węzłów powoduje destabilizację sieci oraz utratę części informacji o zasobach.
- wykorzystanie lepszych algorytmów szyfrujących – w tej chwili w połączeniach wykorzystywany jest domyślny zestaw algorytmów dla gniazdek SSL dostarczony przez Java JDK.
- poprawienie interfejsu użytkownika tak, aby był bardziej estetyczny

- poprawienie obsługi błędów.

## 12. Opis załączonych plików

W folderze z dokumentacją znajdują się również następujące foldery:

- src – kod źródłowy węzłów(folder „peer”) , gui („graf”), oraz serwera(„server”). W folderze „common” znajduje się kod wykorzystywany zarówno przez węzły jak i Server
- bin - skompilowany kod źródłowy do kodu pośredniego dla JVM.

Plik Server.jar służy do uruchomienia serwera. Domyślnie nasłuchuje ona na połączenia przychodzące na porcie 9995. Można to zmienić w pliku /res/serwer/ServerInfo.dat gdzie znajdują się również loginy i skróty sha-512 z haseł użytkowników. Ponadto w folderze /res/serwer/key znajduje się repozytorium kluczy i certyfikatów serwera oraz plik z certyfikatem.

Plik Peer.jar służy do uruchomienia węzła wraz z interfejsem użytkownika. W folderze /res/key/ znajduje się repozytorium zaufanych certyfikatów oraz repozytorium certyfikatów i kluczy węzła.

Aby wystartować sieć należy uruchomić program Server.jar następnie, na dowolnej ilości komputerów w sieci, program Peer.jar.

Przykładowe dane do logowania: login: „czarek”, hasło „12345”.

Na jednym komputerze nie powinien działać więcej niż jeden proces Peer.