Folder współdzielony z notyfikacją zmian

Michał Toporowski Łukasz Pielaszek Bogdan Szkoła

26 kwietnia 2013

Spis treści

1 Cele projektu

Przedmiotem projektu jest zaprojektowanie oraz implementacja sieciowego folderu współdzielonego, przechowywanego na centralnym serwerze oraz aplikacje klienckie uruchamiane na maszynach użytkowników. System umożliwia synchronizację danych między folderami bez udziału użytkownika.

Użytkownik systemu może posiadać zasoby, które będzie chciał przechowywać w utworzonym folderze. Informacje o najbardziej aktualnych danych i o zalogowanych użytkownikach przechowywane są na serwerze. W celu zaktualizowania danych aplikacja kliencja wysyła żądanie do serwera, i jeśli serwer odpowiada zgodą, transfer plików się odbywa (lub odbywa się zmiana nazwy/usunięcie pliku na serwerze). Od razu po zaktualizowaniu folderu na serwerze, zastosowane zmiany są rozsyłane innym zalogowanym użytkownikom, którzy mają dostęp do podanego katalogu.

Przy zalogowaniu użytkownika odbywa się obowiązkowa synchronizacja z serwerem. W razie wynikania konfliktu wersji plików (jednoczesnego modyfikowania plików przez więcej niż jednego użytkownika) serwer tworzy kopie plików i rozsyła pliki oraz informacje o konflikcie do wszystkich zalogowanych klientów.

Aplikacja serwera powinna zapewniać: współbieżne nasłuchiwanie informacji od wielu użytkowników, możliwość wykrycia zmian bez potrzeby wysłania zawartości folderu, możliwość obsługi wielu grup użytkowników. Aplikacja klienta powinna zapewniać: nasłuchiwanie zmian w katalogach lokalnych, synchronizację plików z serwerem oraz interakcję z użytkownikiem.

System musi obsługiwać różne sytuacje wyjątkowe tj. zerwanie połączenia podczas transferu, odłączenie się hosta od sieci, logowanie już zalogowanego użytkownika.

2 Wymagania funkcjonalne

- stworzenie systemu w architekturze klient-serwer
- aktualizacja zmian bez udziału użytkownika
- radzenie sobie z konfliktami wersji
- obsługa drzewiastej struktury katalogów
- obsługa wielu klientów

3 Inne wymagania

- opracowanie protokołu przesyłania plików
- użytkownik należy tylko do jednej grupy współdzielącej katalog
- edycja plików ma miejsce po stronie klientów, serwer pełni rolę pośrednika, nie jest obsługiwana modyfikacja plików na serwerze

4 Analiza wymagań funkcjonalnych

- aplikacja klienta i serwera zostanie stworzona w technologii Java
- aplikację będzie można skonfigurować na dowolnych platformach (Windows, Linux)
- do przesyłania danych użyjemy socketów javowych
- dane będą przesyłane przy użyciu protokołu TCP/IP
- zmiany na lokalnym komputerze wykrywane przy pomocy WatchService

4.1 Wykrywanie zmian

- przechowywanie sumy kontrolnej i daty modyfikacji (po stronie serwera - data ostatniego pobrania pliku od klienta, po stronie klienta - data ostatniego pobrania z serwera
- jeśli sumy kontrolne jednakowe - plik bez zmian
- jeśli sumy różne i daty jednakowe - plik został zaktualizowany, konieczne zastąpienie starego pliku nowym
- jeśli sumy różne i daty różne - konflikt, zapisujemy nowy plik pod inną nazwą

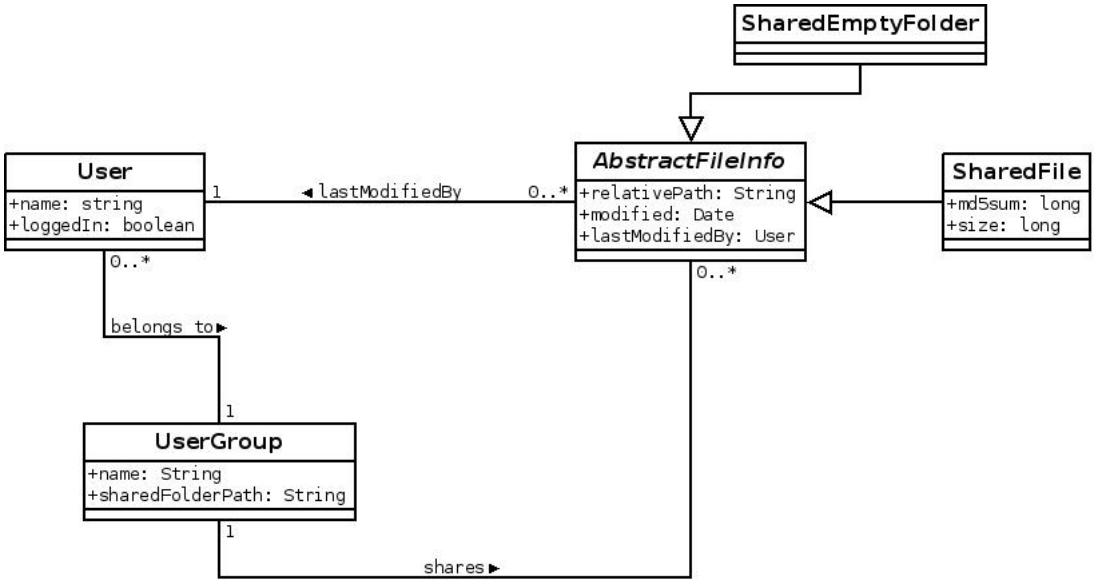
4.2 Obsługa problemów z połączeniem

- utrata łączności - wyświetlenie komunikatu i ponowne połączenie po 1 minucie, plik zapisany jako [plik].temp, a dopiero po odebraniu całego nadpisuje się stary
- przekroczone limity czasowe - komunikat i automatyczne ponowne połączenie się
- błędne pakiety - obsługa zapewniona przez protokół TCP

5 Sposób przechowywania informacji o plikach na serwerze

- każda grupa ma jeden katalog i jest to root dla tej grupy
- metadane plików zapisane w serializowanej klasie SharedFile
- dane o pustych katalogach zapisane w klasie SharedEmptyFolder
- klasy SharedFile i SharedEmptyFolder dziedziczą po AbstractFileInfo
- informacja o niepustych folderach przechowywana w ścieżce pliku
- dane o użytkownikach i grupach przechowywane w serializowanej klasie zapisywanej do pliku users.tin

- metadane wczytywane przy starcie aplikacji serwera, podczas pracy przechowywane w pamięci operacyjnej



6 Protokoły

Przesyłanie przy użyciu klas Socket (na serwerze ServerSocket) oraz ObjectInputStream i ObjectOutputStream odpowiednio do wysyłania i odbierania danych.

6.1 Działanie

1. Przesyłany obiekt typu enum, zawierający informację o typie komunikatu.
2. Przesyłane dodatkowe dane, w zależności od typu komunikatu.
3. Przesyłana informacja o zakończeniu transmisji.

6.2 Rodzaje komunikatów

Rodzaj komunikatu	**Dodatkowe dane**
LOGIN_REQUEST	String username, String password
LOGIN_OK	-
LOGIN_FAILED	-
FILE_CHANGED	AbstractFileInfo metadata
FILEPATH_CHANGED	AbstractFileInfo metadata
FILE_DELETED	AbstractFileInfo metadata
FILE_REQUEST	AbstractFileInfo metadata
FULL_METADATA_TRANSFER	AbstractFileInfo[] metadataList
FILE_TRANSFER	byte[] data

6.2.1 Przesyłanie plików - uszczegółowienie

Pliki zapisywane w postaci tablicy byte[] i wysyłane w paczkach po 100kB.

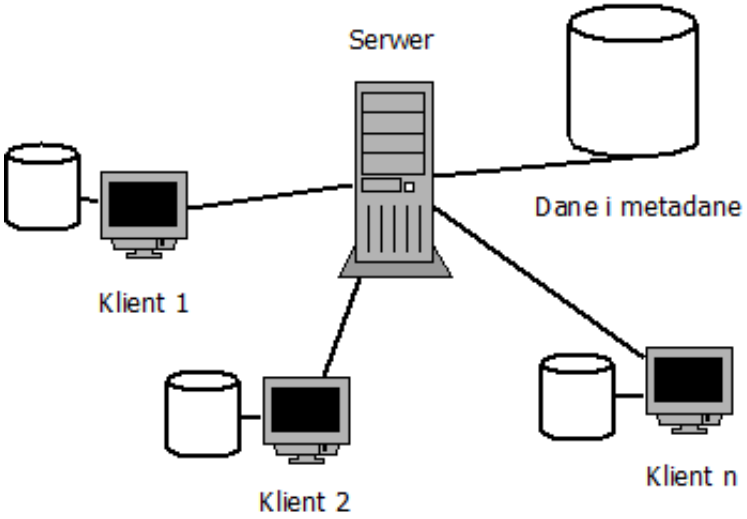
1. Wysłanie komunikatu FILE_TRANSFER.
2. Odczytanie kolejnej paczki.
3. Zapisanie paczki do strumienia sieciowego.
4. Jeśli w pliku są jeszcze dane - powrót do punktu 2.
5. Wysłanie informacji o zakończeniu transmisji.

6.2.2 Dodawanie plików

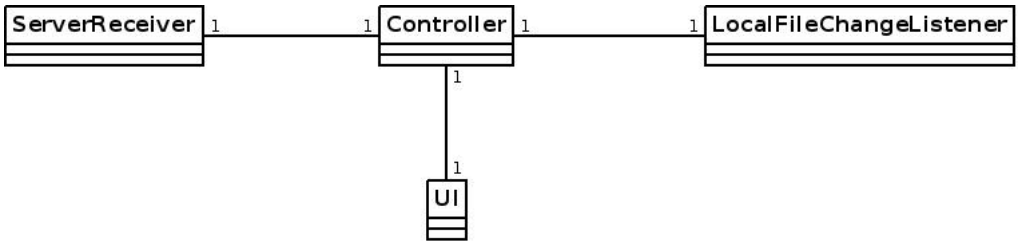
Nie ma osobnego komunikatu FILE_ADDED, ponieważ jest to szczególny przypadek sytuacji FILE_CHANGED, w której na jednej z maszyn plik nie istnieje i należy utworzyć nowy.

7 Architektura

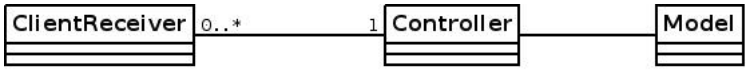
7.1 Schemat ogólny



7.2 Klient



7.3 Serwer



7.4 Synchronizacja wątków

Użycie BlockingQueue do synchronizacji wątków - kolejka będzie pomiędzy obiektami ClientReceiver i Controller na serwerze, ServerReceiver i Controller na kliencie oraz UI i Controller na kliencie.

8 Scenariusze

8.1 Logowanie

1. Użytkownik uruchamia aplikację.
2. Aplikacja kliencka wyświetla komunikat „Login:”.

3. Użytkownik wpisuje „login” i wciska „Enter”.
4. Aplikacja kliencka wyświetla komunikat „Password:”.
5. Użytkownik wpisuje „Password” i wciska przycisk „Enter”.
6. Aplikacja kliencka nawiązuje połączenie z serwerem.
7. Aplikacja kliencka wysyła obudowane dane „login” oraz „password” do serwera.
8. Serwer sprawdza poprawność danych logowania się (oraz czy użytkownik o tej nazwie nie jest już zalogowany), wysyła obudowany pakiet z potwierdzeniem zalogowania się oraz zmienia status użytkownika we własnych danych.
9. Aplikacja kliencka odbiera pakiet, przechodzi do stanu „użytkownik zalogowany”.

8.2 Nieudane logowanie

1-7. Tak jak w scenariuszu [Logowanie]
8. Serwer sprawdza poprawność danych logowania się, wysyła obudowany pakiet z odmową zalogowania się.
9. Serwer kończy połączenie.
10. Aplikacja kliencka wyświetla komunikat o nieudanej próbie logowania.
11. Wracamy do p. 2.

8.3 Klient wprowadza zmiany do pliku

1. Klient zapisuje zmieniony plik.
2. Aplikacja kliencka wykrywa modyfikację pliku.
3. Aplikacja wysyła na serwer informację o modyfikacji pliku.
4. Jeśli data modyfikacji pliku na serwerze jest taka sama jak data poprzedniej modyfikacji przesłanej przez klienta - zastępujemy plik na serwerze. W przeciwnym razie wykrywamy konflikt i tworzymy drugą wersję pliku.
5. Serwer wysyła przeprowadzone zmiany do pozostałych klientów.

8.4 Zmiana ścieżki do pliku

1. Klient zmienia ścieżkę do pliku (zmienia nazwę, bądź przenosi do innego folderu)
2. Aplikacja kliencka wykrywa zmianę ścieżki.
3. Aplikacja wysyła na serwer informację o zmianie ścieżki.
4. Serwer aktualizuje metadane i wysyła je pozostałym klientom.

8.5 Synchronizacja folderu po zalogowaniu

1. [Logowanie się]
2. Klient wysyła swoją listę plików na serwer.
3. Serwer porównuje zawartość, jeśli pliki różne, serwer wysyła aplikacji klienckiej informację o konieczności przesłania różniących się plików
4. Przechodzimy do scenariusza [Klient wprowadza zmianę do pliku].

9 Testowanie

Testy przeprowadzone na podstawie powyżej opisanych scenariuszy oraz wymagań funkcjonalnych.

10 Podział pracy

- Prowadzenie projektu - Łukasz Pielaszek
- Moduł kontrolerów - Bogdan Szkoła
- Moduł sieciowy klienta - Michał Toporowski
- Moduł sieciowy serwera - Bogdan Szkoła
- Moduł logiczny serwera - Łukasz Pielaszek
- Moduł nasłuchujący zmiany w katalogach - Michał Toporowski
- Interfejs użytkownika - Łukasz Pielaszek
- Testowanie i zapewnienie jakości - Bogdan Szkoła
- Dokumentacja końcowa - Michał Toporowski