Data powstania: 17.01.2013

Ostatnia aktualizacja: 21.01.2013

**Krzysztof Opasiak**

Mikołaj Markiewicz

Marcin Kubik

Jacek Sosnowski

**Dokumentacja końcowa**

**System zarządzania aliasami rozproszonych katalogów.**

1. **TREŚĆ ZADANIA**

Zadanie polega na zaprojektowaniu i implementacji systemu który umożliwia dostęp i synchronizację zawartości katalogów rozmieszczonych na różnych stacjach roboczych.

**Alias** – nazwa symboliczna definiująca katalog rozproszony. Jeden alias może wskazywać kilka katalogów, ale określony katalog może należeć wyłącznie do jednego aliasu.

1. **BUDOWA SYSTEMU**

System składa się z następujących modułów:

* 1. Serwer – pośredniczy w wymianie komunikatów pomiędzy demonami a aplikacją kliencką
  2. Klient – aplikacja poprzez którą uzyskuje się dostęp do aliasów i zasobów
  3. Demon – uruchomiony na stacji klienta odpowiada na żądania serwera

1. **DZIAŁANIE:**

Po uruchomieniu serwera (nasłuchującego na konkretnym porcie) możliwe jest utworzenie przez klienta nowego aliasu (na którego temat informacje przetwarza serwera) lub podłączenie się do aliasu już istniejącego. Możliwe jest dołączenie katalogu do aliasu lub usunięcie go z niego. Po dołączeniu kilku katalogów na różnych komputerach informacje o nich będą dostępne na każdej stacji roboczej. Po zalogowaniu i wylistowaniu zawartości katalogów dane pliki zostaną ukazane w postaci drzewa katalogów. Operacje jakie można wykonać na danym aliasie to pobranie pliku znajdującego się w katalogu zdalnym (innym niż aktualny na komputerze użytkownika), przesłanie pliku znajdującego się w aktualnym katalogu, wyszukanie pliku o nazwie wpisanej przez użytkownika oraz opcje synchronizacyjne. Są to nadpisywanie nowszych oraz przedstawianie listy konfliktów, zezwalające użytkownikowi na wybór odpowiadającego mu pliku. W przypadku pierwszej opcji wybór pliku nadpisującego odbywa się poprzez znalezienie pliku największego i/lub najnowszego.

Bezpośrednio powiązany ze stacją roboczą jest demon, którego zadaniem jest komunikacja zarówno z klientem jak i z serwerem. Odpowiada on za przekazywanie listy plików zgromadzonych w danym katalogu, usuwanie na żądanie danego pliku oraz synchronizację zawartości katalogów.

Komunikacja pomiędzy komponentami odbywa się za pomocą przedstawionego w osobnym dokumencie protokołu komunikacyjnego obejmującego szereg komunikatów, zawierających niekiedy także dane (takie jak lista plików czy informacja o znalezionym/nieznalezionym pliku). Komunikaty wykrywane są za pomocą specjalnie w tym celu przygotowanego modułu do programu Wireshark.

Założeniem zaprojektowanego programu jest, aby w danej chwili na jednej stacji roboczej uruchomiony był co najwyżej jeden klient i jeden demon. Ponieważ program służyć ma przede wszystkim dla użytkownika chcącego synchronizować pliki przechowywane na różnych stacjach roboczych, operacje przez niego wykonywane mają bezpośrednie przełożenie na zawartości katalogów na różnych komputerach.

1. **ŚRODOWISKO URUCHOMIENIOWE, URUCHAMIANIE PROGRAMU**

Program działa w środowisku UNIX, wydatnie korzystając z zaimplementowanych w nim gniazd. Komunikacja pomiędzy serwerem a klientem i demonem odbywa się za pomocą autorskiego protokołu będącego niejako „nakładką” na protokół TCP. Port, na którym obsługiwana jest usług rozproszonych katalogów to 8888. Uruchamianie aplikacji odbywa się z poziomu konsoli.

Program napisany jest w języku C++. Wykorzystane w nim biblioteki to boost (sprytne wskaźniki) oraz Qt (komunikacja sieciowa, operacje na systemie plików). Wykorzystanie ich pozwoliło na sprawniejsze zarządzanie danymi oraz komunikację między komponentami.

1. **SPOSÓB TESTOWANIA:**

Przygotowanie testowej struktury katalogów oraz infrastruktury sprzętowej (serwer, stacje klienckie). Programy (Demon, Klient i Server) uruchamiane były na różnych komputerach. Do sprawdzania komunikatów przekazywanych pomiędzy różnymi stacjami wykorzystano odpowiednio zaimplementowany w tym celu moduł do Wiresharka. Sprawdzanie odbywało się poprzez porównywanie zawartości katalogów na różnych komputerach, przesyłanie plików między nimi oraz wzajemną synchronizację. Powstały również skrypty testujące (komendy przekazywane na standardowe wejście programu aby nie trzeba było wpisywać ciągu komend ręcznie, jednak były one na bieżąco modyfikowane i każdy pisał je według bieżących potrzeb dla swojej struktury katalogów)

1. **PODZIAŁ PRAC:**

Po przeanalizowaniu struktury programu i zawartych w nich komponentów dokonano podziału na 4 główne części: Serwer, Klient, Demon oraz komunikacja między nimi. Prace rozpoczęły się od serii spotkań związanych z opracowaniem architektury całego rozwiązania. Powstały diagramy klas poszczególnych modułów oraz protokół komunikacji pomiędzy nimi. Prace rozpoczęły się od implementacji protokołu komunikacyjnego. Następnie rozpoczęły się równoległe prace nad poszczególnymi modułami. Iteracyjnie każda z opracowanych funkcjonalności była integrowana i testowane były funkcje za nią odpowiedzialne na wszystkich trzech modułach.

Analizując statystyki repozytorium można by dojść do wniosku iż praca w zespole była nie równo podzielona, jednak jest to mylne wrażanie gdyż pierwsze prace odbywały się w grupie, jednak zmiany były zatwierdzane z jednego komputera. Podziału prac nie można oceniać tylko na podstawie linii kodu, gdyż wiele zadań było typu badawczego jak np. zapoznanie się z wireshark’iem czy wyszukanie najlepszej opcji na przesyłanie i synchronizację listy plików. Zadania tego typu są bardzo czasochłonne, jednak czas ten jest konieczny aby rozwiązanie końcowe było optymalne. W końcowej fazie do uzupełniania luk w kodzie wspomagaliśmy się git-owym systemem zadań (Issues), aby nie doszło do zdublowania prac na styku modułów.

Oceniając swój zespół chciałbym zaznaczyć jego duże zaangażowanie i odpowiedzialność, co pomogło mi w znacznym stopniu w zarządzaniu projektem. Każdy z członków wywiązywał się terminowo z powierzonych mu zadań, zwłaszcza tych „badawczych” co umożliwiło systematyczny rozwój projektu bez większych opóźnień. Komunikacja w projekcie przebiegała w sposób wzorowy, nie pojawiły się żadne konflikty interesów czy dublowanie prac. Należy również zaznaczyć dużą samodzielność każdego z członków zespołu, ponieważ mimo zlecenia zadań bardzo ogólnych i wymagających zapoznania się z wieloma technikami, każdy z członków był specjalistą w zakresie swoich kompetencji, a następnie na spotkaniach projektowych ubogacał nas wnioskami z swoich prac. Taki podział umożliwił zespołowi poznanie zarysu całego projektu, jednak nie nakładając na wszystkich obowiązku zapoznawania się z każdym szczegółem implementacyjnym w projekcie, lecz w razie potrzeby korzystano z wiedzy osoby, która sprawowała piecze nad daną funkcjonalnością lub modułem. W związku z powyższym chciałbym ocenić pracę wszystkich członków zespołu na maksymalna ilość 5 punktów.

**WNIOSKI:**

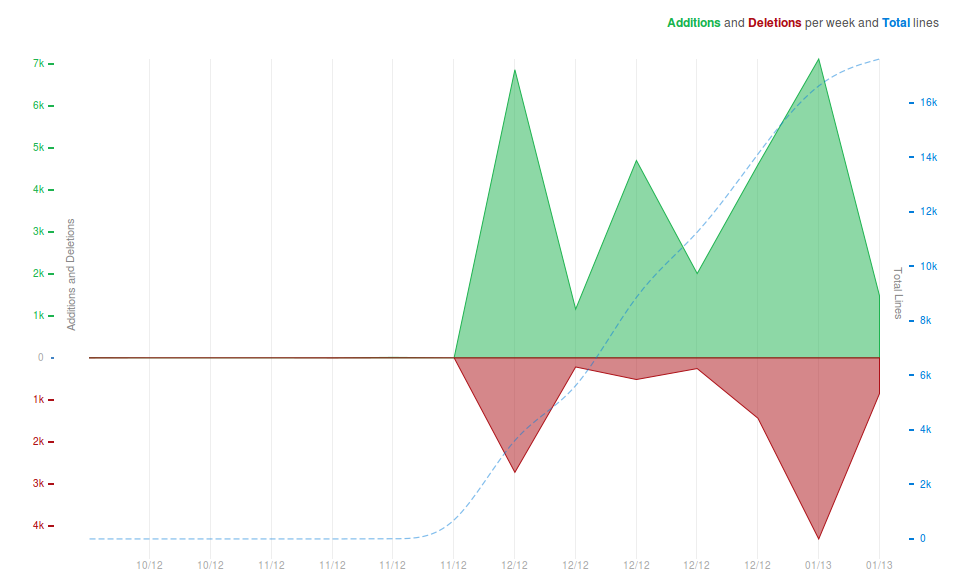
Opracowanie przedstawionego w treści projektu pozwoliło na zanalizowanie i zbadanie szeregu właściwości komunikacji sieciowej. Umożliwiło ocenę i weryfikację sposobów przesyłania plików przez sieć Internet, przetestowanie działania modułów współpracujących ze sobą, lecz umieszczonych na innych stacjach roboczych oraz sprawdzenie działania systemów opierających swoje działania na przesyłaniu danych poprzez sieć. Doświadczenie zdobyte podczas opracowywania protokołu komunikacyjnego i diagramów klas oraz współpraca w grupie pozwoliły nam zdobyć doświadczenie, które z pewnością przyda nam się w przyszłości.

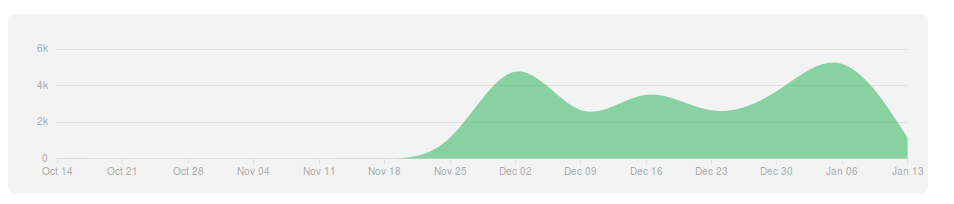
**STATYSTYKI:**

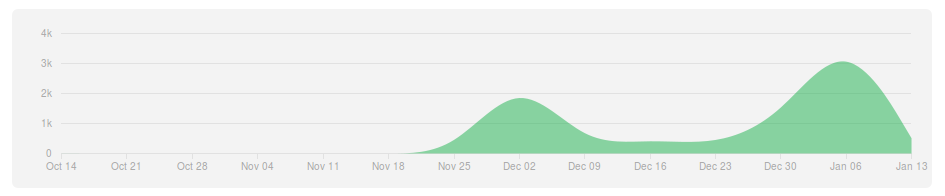
Finalny rozmiar projektu:

122 pliki, 18803 linii kodu.

Statystyki dotyczące tempa implementacji:



Dodania linii:

Usunięcia:

**Czasochłonność projektu:**

Szacujemy iż na spotkaniach projektowych na których poruszane były kwestie dotyczące architektury systemu oraz protokołu komunikacyjnego spędziliśmy około **32** godzin (4 spotkania po 8 godzin).

W kolejnym etapie prac zadania zostały rozdzielone przez lidera i każdy członek zespołu zajmował się przydzielonym mu fragmentem projektu. Członkowie zespołu poświecili następujące ilości czasu na wypełnienie poszczególnych zadań ( przyjęto założenie iż 1 dzień to przeciętnie 8 godzin ):

Opasiak Krzysztof

|  |  |
| --- | --- |
| **Zadanie** | **Czas poświęcony [dni]** |
| Generacja kodu oraz pisanie skryptów i utworzenie projektu w Eclipse | 1 |
| Implementacja klas odpowiedzialnych za przekazywanie komunikatów | 7 |
| Implementacja protokołu transferu plików | 2 |
| Tworzenie dokumentacji, zarządzanie praca grupy i konsultacje z członkami zespołu | 3 |
| Poprawianie błędów zgłoszonych w fazie testów integracyjnych | 1 |
| **Suma:** | **14** |

Kubik Marcin

|  |  |
| --- | --- |
| **Zadanie** | **Czas poświęcony [dni]** |
| Zapoznanie się z konwencjami biblioteki Qt | 2 |
| Zapoznanie się z metodami pisania wtyczek do Wireshark’a | 1 |
| Implementacja wtyczki do Wireshark’a | 1 |
| Zaprojektowanie logiki klienta i interakcji z użytkownikiem | 3 |
| Implementacja logiki modułu klienckiego | 6 |
| Testowanie poprawności współpracy klienta z innymi modułami | 1 |
| **Suma:** | **14** |

Markiewicz Mikołaj

|  |  |
| --- | --- |
| **Zadanie** | **Czas poświęcony [dni]** |
| Poszukiwania najlepszej metody implementacji operacji plikowych | 2 |
| Implementacje operacji plikowych | 3 |
| Poszukiwania najlepszej metody budowania struktury aliasu | 1 |
| Implementacja drzewa plików i katalogów aliasu | 3 |
| Implementacja częściowej funkcjonalności demona (pliki konfiguracyjne XML) | 2 |
| Implementacja komunikacji klient-demon | 3 |
| **Suma:** | **14** |

Sosnowski Jacek

|  |  |
| --- | --- |
| **Zadanie** | **Czas poświęcony [dni]** |
| Zapoznanie się z konwencjami biblioteki Qt | 2 |
| Zaprojektowanie logiki aliasu (część logiki serwera) i interakcji z użytkownikiem | 3 |
| Implementacja logiki aliasu | 4 |
| Implementacja częściowej funkcjonalności demona | 2 |
| Integracja i testowanie integracyjne | 3 |
| **Suma:** | **14** |

W końcowej fazie integracji projektu odbyły się również dwa kolejne spotkania zespołu trwające łącznie **18** godzin.

**Podsumowanie:**

Pracochłonność projektu – **648 osobogodzin**

Średni czas poświęcony przez członka zespołu na wykonanie projektu – **162 godziny.**