**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 daemonami a aplikacją kliencką
  2. Klient – aplikacja poprzez którą uzyskuje się dostęp do aliasów i zasobów
  3. Daemon – 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 daemon, 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 daemon. 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 socket-ów. Komunikacja pomiędzy serwerem a klientem i daemonem odbywa się za pomocą autorskiego protokołu będącego niejako „nakładką” na warstwę 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 (Daemon, Client 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.

1. **PODZIAŁ PRAC:**

Po przeanalizowaniu struktury programu i zawartych w nich komponentów dokonano podziału na 4 główne części: Serwer, Klient, Daemon 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 odpowiedzialne za nią na wszystkich trzech modułach.

**WNIOSKI:**

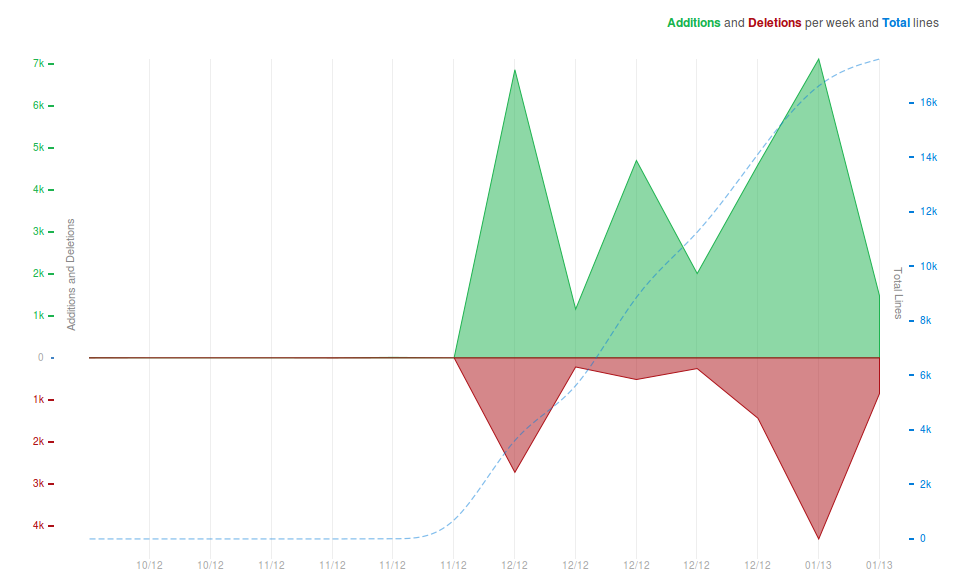
Opracowanie przedstawionego w treści projektu pozwoliło na zanalizowanie i zbadanie szeregu właściwości komunikacji sieciowej. Umożliwiło ocenę i weryfikację sposobów przesyłania plików przez sieć Internet, przetestowanie działania modułów współpracujących ze sobą, lecz umieszczonych na innych stacjach 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:**

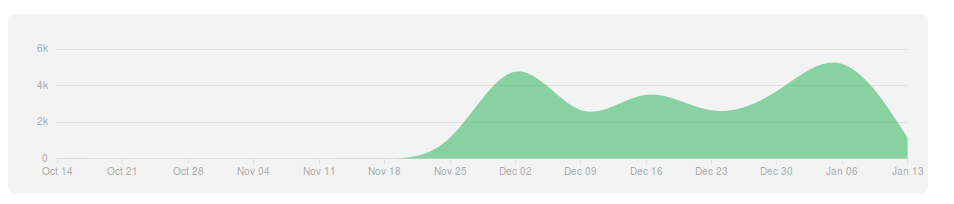
Finalny rozmiar projektu:

120 plików, 18712 linii kodu

Statystyki dotyczące tempa implementacji:



Dodania linii:



Usunięcia:

