Dokumentacja wstępna

System zarządzania aliasami rozproszonych katalogów.

1. Treść zadania:

Zadanie polega na zaprojektowaniu i implementacji systemu który umożliwi a 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.

2. Budowa systemu.

System składa się z następujących modułów:

2.1. Serwer – pośredniczy w wymianie komunikatów pomiędzy daemonami a aplikacją kliencką
2.2. Klient – aplikacja poprzez którą uzyskuje się dostęp do aliasów i zasobów
2.3. Daemon – uruchomiony na stacji klienta odpowiada na żądania serwera

3. WYMAGANIA FUNKCJONALNE

3.1. **SERWER**
- 3.1.1. Przechowywanie informacji na temat aliasów aliasów utrzymywanych na tym serwerze
- 3.1.2. Utrzymywanie połączeń z daemonami stacji klienckich
- 3.1.3. Pośredniczenie w komunikacji pomiędzy klientami a daemonami obsługującymi dany alias
- 3.1.4. Dodanie aliasu do zbioru na żądanie klienta
- 3.1.5. Usunięcie aliasu ze zbioru na żądanie klienta

3.2. **DAEMON**
- 3.2.1. Utrzymanie połączenia z serwerami, które obsługują aliasy dostępne na danym komputerze
- 3.2.2. Odpowiadanie na żądania otrzymywane z serwera
  - 3.2.2.1. Usuwa pliki
  - 3.2.2.2. Listuje katalogi – ujawnia zawartość katalogów
  - 3.2.2.3. Synchronizację danego katalogu z innymi katalogami zdalnymi
- 3.2.3. Odpowiadanie na żądania lokalnego klienta

3.3. **KLIENT**

3.3.1. Dodanie aliasu razem z hasłem dostępu.

3.3.2. Usunięcie aliasu.

3.3.3. Dowiązanie lokalnego katalogu do aliasu.

3.3.4. Usunięcie dowiązania lokalnego katalogu.

3.3.5. Operacje na aliasie:

3.3.5.1. Wylistowanie zawartości

3.3.5.2. Wyszukanie pliku w obrębie aliasu

3.3.5.3. Odczyt istniejącego pliku

3.3.5.4. Usunięcie pliku z aliasu

3.3.6. Synchronizacja katalogów w ramach jednego aliasu w zależności od ustalonych opcji działania na wypadek konfliktu plików:

3.3.6.1. Nadpisywanie starszego pliku przez nowszy

3.3.6.2. Dla każdego pliku rozstrzyganie konfliktu przez użytkownika

4. WYMAGANIA NIEFUNKCJONALNE

4.1. Obsługiwane będą adresy IPv4 oraz IPv6

4.2. Implementacja odpowiedniego modułu do programu Wireshark w celu prezentacji i analizy przesyłanych komunikatów

4.3. W danej chwili na stacji roboczej może być uruchomiony co najwyżej jeden klient i jeden daemon

5. ŚRODOWISKO PRACY SYSTEMU

5.1. Platforma: UNIX

5.2. Język programowania: C++

5.3. Używane biblioteki: BOOST, Qt

6. SPOSÓB TESTOWANIA:

Przygotowanie testowej struktury katalogów oraz infrastruktury sprzętowej (serwer, stacje klienckie).

7. Podział prac

Zadania projektowe związane z zaprojektowaniem struktury programu (diagramu klas) oraz zaprojektowanie protokołu komunikacji pomiędzy modułami zostaną wykonane wspólnie. Podział prac w fazie pisania kodu, będzie polegał na przydzielaniu członkom zespołu konkretnych klas w celu ich implementacji.