Stanisław Zieliński 145321
Jakub Zmuda 145205

Sprawozdanie SK2

Ciągła synchronizacja katalogów

1. Opis projektu

Nasz projekt miał polegać na stworzeniu aplikacji w architekturze klient-serwer, która synchronizowałaby pliki z folderów otwartych przez klientów. Chcieliśmy zrealizować ten pomysł tworząc serwer w języku cpp i klienta w języku python. W celu obsługi połączeń TCP/IP korzystamy z interfejsu BSDSockets. Do zarządzania utworzonymi socketami korzystamy z funkcji poll, aby uniknąć problemów ze współbieżnością.

2. Opis komunikacji między klientem i serwerem

Najpierw serwer czeka na połączenie z dowolnym klientem. Po nawiązaniu połączenia, klient wysyła nazwy i sumy hashowe wszystkich swoich plików w swoim folderze przeznaczonym do synchronizacji, a serwer zapisuje je w swoim folderze. Kiedy z serwerem połączenie nawiąże inny klient, on też wysyła nazwy i sumy hashowe wszystkich plików, które następnie są porównywane z istniejącymi w folderze na serwerze. Jeżeli któryś z klientów nie ma w swoim folderze danego pliku lub ma plik o tej samej nazwie ale z inną sumą hashową, to plik ten jest do niego wysyłany.

3. Podsumowanie

Największe trudności sprawiło zadbanie o synchronizację między klientem i serwerem oraz poprawne zabezpieczenie kodu w celu ominięcia problemów ze współbieżnym odczytem. Synchronizację między klientem i serwerem zapewniliśmy dzięki wysyłaniu trzybajtowych kodów statusu. Są one zapisane w pliku constant.py oraz w kodzie serwera jako define. Są one np wysyłane przed wysłaniem pakietu danych przez klienta, aby mógł się on przygotować na jego odbiór. Służą one także do przekazania informacji co będzie w danej chwili wykonywane od serwera do klienta. Większą trudność sprawiło poprawne zabezpieczenie kodu do współbieżnego działania. Z tego powodu zdecydowaliśmy się użyć funkcji poll. Niestety przez trudności wynikłe z obsługi funkcji poll nie udało nam się zaimplementować ciągłej synchronizacji folderów. Informacje o zawartości folderów są przekazywane tylko przy łączeniu się do serwera. NIe powinno być żadnych problemów przy połączeniu się dwóch klientów w tym samym czasie ponieważ accept() nie zaakceptuje dwóch socketów na raz i dopiero przy kolejnej iteracji po liście deskryptorów dołączy drugi.