Projekt TIN - Dokumentacja

Wiktor Michalski
Przemysław Stawczyk
Maciej Szulik
Kamil Zacharczuk

January 27, 2020

Contents

Chapter 1

Zadanie

1.1 Tresć Zadania

Napisać program obsługujący prosty protokół P2P

1. Zasób to plik identyfikowany pewną nazwą, za takie same zasoby uważa się zasoby o takich samych nazwach i takiej samej wielkości pliku w bajtach.
2. Początkowo dany zasób znajduje się w jednym węźle sieci, następnie może być propagowany do innych węzłów w ramach inicjowanego przez użytkownika ręcznie transferu - raz pobrany zasób zostaje zachowany jako kopia.
3. Po pewnym czasie działania systemu ten sam zasób może się znajdować w kilku węzłach sieci (na kilku maszynach).
4. System ma informować o posiadanych lokalnie (tj. w danym węźle) zasobach i umożliwiać ich pobranie.
5. Program powinien umożliwiać współbieżne:
    - wprowadzanie przez użytkownika (poprzez interfejs tekstowy):
        - nowych zasobów - z lokalnego systemu plików
        - poleceń pobrania nazwanego zasobu ze zdalnego węzła
    - pobieranie zasobów (także kilku jednocześnie)
    - rozgłaszanie informacji o posiadanych lokalne zasobach
6. W przypadku pobierania zdalnego zasobu system sam (nie użytkownik) decyduje skąd zostanie on pobrany.
7. Powinno być możliwe pobranie zasobu z kilku węzłów na raz (tj. "w kawałkach").
8. Zasób pobrany do lokalnego węzła jest kopią oryginału, kopia jest traktowana tak samo jak oryginał (są nierozróżnialne). Istnienie kopii jest rozgłaszane tak samo jak oryginału.
9. Właściciel zasobu może go unieważnić wysyłając odpowiedni komunikat rozgłaszany. Wszystkie kopie zasobu powinny przestać być rozgłaszane. W przypadku trwających transferów zasobów powinny się one poprawnie zakończyć, dopiero wtedy informacja o zasobie może zostać usunięta.

1.2 Wariant zadania dla zespołu

4. Opóźnienia dla wybranego węzła - węzeł reaguje, ale (czasami) z dużym opóźnieniem.

1.3 Interpretacja Zadania

Doprecyzowanie treści i dodatkowe założenia

- W związku z tym, że kopia i oryginał są nierozróżnialne, zasób może być unieważniony przez dowolnego użytkownika, który go posiada.
- Każdy węzeł okresowo rozgłasza informację o posiadanych zasobach. Unieważnienie pliku oznacza, że żaden z węzłów nie będzie już rozgłaszał faktu posiadania tego pliku.
- Unieważnienie wysyłane jest asynchronicznie poprzez broadcast UDP.
- W przypadku unieważnienia pliku w trakcie trwającego przesyłu tego pliku przesyłanie kończy się sukcesem, o ile nie wystąpią inne błędy. Nowy posiadacz pliku nie będzie jednak nigdy rozgłaszał o nim informacji.
- Każdy węzeł przechowuje listy dostępnych zasobów każdego innego węzła. Po odebraniu rozgłoszenia listy zasobów od innego węzła lista ta jest nadpisywana w pamięci węzła odbierającego. Informacje o węźle, w tym lista jego zasobów, są usuwane w przypadku braku, przez ustalony czas, nadchodzącego rozgłoszenia jego listy zasobów.
- W przypadku połączenia z innym węzłem w celu pobrania od niego pliku oczekiwanie na odpowiedź tego węzła ma pewien timeout. Ponadto, jeżeli węzeł przekracza pewien ustalony czas odpowiedzi (nawet jeżeli nie dochodzi do timeout'u), to inkrementujemy licznik "opóźnień" tego węzła (każdy węzeł przechowuje takie liczniki dla każdego innego węzła). Po osiągnięciu ustalonej wartości licznik ten jest zerowany, a węzeł zliczający nie będzie próbował łączyć się z "opóźnionym" węzłem przez pewien określony czas.
- Jeśli węzeł niespodziewanie zakończy połączenie TCP i przestanie rozgłaszać swoją tablicę, to po pewnym czasie pozostałe węzły uznają to za opuszczenie przez niego sieci.
- W przypadku gdy pojawi się błąd w trakcie transferu TCP, usuwamy pobrane dane (segmenty) i kończymy wątek pobierający. Ponowne pobieranie od tego węzła będzie odbywać się po ponownym połączeniu z węzłem.

Chapter 2

Opis Funkcjonalny Projektu

Użytkownik systemu ma wgląd w dwie listy

- lokalny rejestr zasobów - pliki, które użytkownik dodał lub pobrał od innych,
- pliki obecne w systemie - pliki posiadane w lokalnym rejestrze zasobów przez innych użytkowników, które nie zostały unieważnione.

Dla każdego użytkownika generowana jest, oczywiście, odrębna para list.

Użytkownik może wprowadzać tekstowe komendy, aby za ich pomocą

- *wyświetlić listę lokalnych zasobów,*
- *wyświetlić listę zasobów obecnych w systemie,*
- *wyświetlić listę dostępnych komend,*
- *opuścić system,*

a także wykonywać operacje na plikach, wśród których rozróżniamy:

- *dodanie pliku,*
  można dodać do zasobów plik, którego nazwa nie wystąpiła jeszcze wśród plików w lokalnym rejestrze zasobów i reszcie sieci.
  Zakładamy, że nie wystąpi sytuacja, gdy więcej niż jeden użytkownik doda plik o tej samej nazwie “jednocześnie” - to znaczy przed “zauważeniem” przez całą sieć dodania pliku o takiej nazwie przez któregokolwiek z nich.
- *usunięcie pliku,*
  można usunąć plik z własnego rejestru zasobów.
- *unieważnienie pliku,*
  można unieważnić plik, który mamy we własnym rejestrze zasobów. Oznacza to, że zasób nie będzie już widoczny na liście plików dostępnych w systemie, ale dotychczasowi posiadacze nadal będą go posiadali w swoim lokalnym systemie plików.
- *pobranie pliku,*
  można pobrać plik, którego nie mamy jeszcze w rejestrze zasobów, a który jest obecny w systemie.

Chapter 3

Protokół

3.1 Składnia :

```
<Command 1 octet> ::= <REVOKE> | <FILE_LIST> | <REQ_CONN> | <REQ_SEGMENT>

<ResourceHeader> ::=
        <nazwa pliku: 256 octets, NULL terminated> <rozmiar pliku: 64bit>

<Message> ::= <Command> | <Command><ResourceHeader> |
        <Command><ResourceHeader><Options> |
        <Command><No_Of_Files 16bit><ResourceHeader><ResourceHeader>...

<Resource> ::= <1 KB of file>
```

3.2 Komendy :

- *Unieważnienie pliku:*
  Broadcast po UDP:
  `<Command = REVOKE><ResourceHeader = Revoked File>`
- *Rozgłaszanie dostępnych plików:*
  Broadcast po UDP:
  `<Command = FILE_LIST><No_Of_Files = liczba dostępnych plików>`
  `<ResourceHeader = plik1>....`
- *Żądanie utworzenia połączenia TCP:*
  Wysyłane do węzła po TCP: `<Command = REQ_CONN>`
- *Żądanie pobrania segmentu:*
  Wysyłane do węzła po TCP:
  `<Command = REQ_SEGMENT><ResourceHeader = plik><Options = segment number>`

Chapter 4

Organizacja Projektu

4.1 Moduły

1. Moduł CLI odpowiedzialny za komunikację z użytkownikiem.
2. Moduł obsługi sieci.
3. Moduł dispatchera obsługujący protokół.

4.2 Współbieżność

Ogólna koncepcja zakłada istnienie następujących bazowych, działających w pętli wątków:

1. *Obsługa przychodzących żądań przesłania posiadanego zasobu.*
   Wątek ten nasłuchuje na porcie TCP. W przypadku nawiązania połączenia na tym porcie tworzony jest watek potomny. Wątek ten odbiera żądanie przesłania lokalnie posiadanego pliku i nadzoruje to przesłanie.
2. *Odbiór komunikatów broadcast UDP.*
   Komunikaty te obejmują okresowe rozgłaszanie przez każdego użytkownika listy lokalnie posiadanych zasobów, rozgłaszanie unieważnienia zasobu. Po odebraniu komunikatu wątek ten przekazuje otrzymane informacje do wątku synchronizującego dane.
3. *Okresowe rozgłaszanie lokalnej listy zasobów.*
   Wątek, który co pewien czas rozgłasza przez UDP listę zasobów, które udostępnia do pobrania. Wątek blokuje się pomiędzy kolejnymi broadcastami.
4. *Wątek synchronizujący dane.*
   Wątek manipulujący danymi przechowywanymi przez program takimi jak: lista lokalnych zasobów, informacje o pozostałych węzłach - dla każdego z nich lista zasobów, które udostępnia, liczniki opóźnień itp.
5. *Obsługa interfejsu użytkownika.*
   Interakcja z użytkownikiem przez CLI. Odbieranie komend od użytkownika i odpowiednie reagowanie - powoływanie nowych wątków, które mają zająć się realizacją komendy, między innymi:
   - W przypadku chęci pobrania pliku tworzony jest wątek nadzorujący to pobieranie. Na potrzeby połączenia z wieloma węzłami może on stworzyć kilka kolejnych wątków przypisanych do połączeń z węzłami.

W zależności od wyniku pobierania przekaże odpowiednie informacje do wątku synchronizującego dane.

- W przypadku chęci dodania, usunięcia lub unieważnienia pliku tworzony jest wątek, który zajmie się wprowadzeniem tej zmiany i nadzorowaniem wszystkich jej następstw, np.: fizycznie doda plik do systemu i przekaże informację o nowym pliku do wątku synchronizującego dane. W przypadku unieważnienia stworzy nowy wątek, który rozgłosi odpowiednią informację w systemie.
- W przypadku chęci wyświetlenia którejś z list zasobów lub listy dostępnych komend tworzony jest wątek, który odczyta odpowiednie dane i przygotuje je w odpowiedniej formie do wyświetlenia użytkownikowi.

Wyniki działań powyższych wątków przekazywane są z powrotem do wątku obsługującego CLI, który wyświetla je użytkownikowi.

Chapter 5

Implementacja

- Jezyki : *C++*
- Biblioteki : *Boost:Asio, std::thread*