Dokumentacja TIN

Jan Piotrowski, Daniel Chmielewiec, Kamil Jabłonowski, Wojciech Moczydłowski

Treść zadania

W dynamicznie zmieniającej się grupie węzłów powinien być aktywny lider i lider zapasowy. Obaj liderzy okresowo rozsyłają komunikaty potwierdzające swą aktywność. Jeżeli przez określony czas lider jest niesłyszalny, to jego funkcję przejmuje lider zapasowy i aktywowana jest procedura wyboru nowego lidera zapasowego. Jeżeli przez określony czas lider zapasowy jest niesłyszalny, to rusza procedura jego ponownego wyboru. Konieczny jest mechanizm wykluczenia zdublowanego lidera. Należy użyć protokołów UDP i IPv6. Należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów.

Nazwa własna projektowanego systemu

Umarł król niech żyje król! - Long live the king

Założenia funkcjonalne.

1. Rozsyłanie komunikatu potwierdzającego obecność lidera głównego
2. Rozsyłanie komunikatu potwierdzającego obecność lidera zastępczego
3. Procedura wyboru lidera głównego
4. Procedura wyboru lidera zastępczego
5. Procedura mianowania lidera zastępczego jako lidera głównego
6. Odbieranie komunikatu o aktywności lidera głównego
7. Odbieranie komunikatu o aktywności lidera zastępczego
8. Tworzenie i usuwanie węzłów
9. Monitorowanie wysyłanych komunikatów (moduł w wiresharku)
10. Testy integracyjne

Założenia niefunkcjonalne.

1. Komunikacja oparta na gniazdach BSD
2. Użyte protokoły: UDP, IPv6
3. Wybór nowego lidera zajmuje mniej niż 1s. Wybór lidera zapasowego (po wyborze lidera głównego) zajmuje również mniej niż 1s
4. Komunikaty przesyłane są przynajmniej co 0.1s
5. Interfejs użytkownika przy pomocy CLI
6. Jednoczesna praca do 10 węzłów
7. Każdy węzeł posiada unikalne id będące liczbą oraz port na którym nasłuchuje

Przypadki użycia

Start systemu

Aktor: Użytkownik

1. Użytkownik uruchamia program
2. Program inicjuje węzły

Tworzenie nowego węzła

Aktor: Użytkownik

1. Program pyta użytkownika o wybór opcji
2. Użytkownik tworzy nowy węzeł
3. System inicjalizuje nowy węzeł
   Ścieżka alternatywna
   a. System posiada już maksymalną ilość węzłów
   b. System zgłasza błąd. Powrót do kroku 1.

Wylistowanie węzłów

Aktor: Użytkownik

1. Program pyta użytkownika o wybór opcji
2. Użytkownik prosi o wylistowanie węzłów
3. System zwraca listę węzłów z Id, adresem pod którym nasłuchuję oraz informacji czy węzeł jest liderem. bądź liderem zastępczym

Usuwanie węzła

Aktor: Użytkownik

1. Program pyta użytkownika o wybór opcji
2. Użytkownik wpisuje Id węzła do usunięcia
3. System usuwa węzeł
   Ścieżka alternatywna
   a. System nie posiada węzła o danym Id
   b. System zgłasza błąd. Powrót do kroku 1.

Kończenie działania programu

Aktor: Użytkownik

1. Program pyta użytkownika o wybór opcji
2. Użytkownik wpisuję komendę zakończenia programu
3. Program kończy działanie wszystkich wątków
4. Program kończy działanie

Podgląd komunikatów

Aktor: Użytkownik

1. Użytkownik uruchamia odpowiedni moduł w programie wireshark do poglądu komunikatów wysyłanych przez węzły.

2. Program wyświetla ostatnie komunikaty wysłane przez węzły

Aktywność lidera głównego

Aktorzy: Węzeł, Lider główny, Lider zapasowy

1. Lider główny wysyła komunikat o swojej obecności
    a. Nie ma lidera, który wysyła komunikat.
    b. Następuje wybór lidera głównego.
2. Węzeł otrzymuje informacje o aktywności lidera

Aktywność lidera zapasowego

Aktorzy: Węzeł, Lider zapasowy

1. Lider zapasowy wysyła komunikat o swojej obecności
    a. Nie ma lidera zapasowego, który wysyła komunikat.
    b. Następuje wybór lidera zapasowego
2. Węzeł otrzymuje informacje o aktywności lidera

Wybór nowego lidera zapasowego

Aktorzy: Węzły

1. Węzły rozsyłają informację o swoim id
2. Węzły wybierają węzeł o najniższym id
3. Wybrany węzeł zostaje liderem zapasowym.

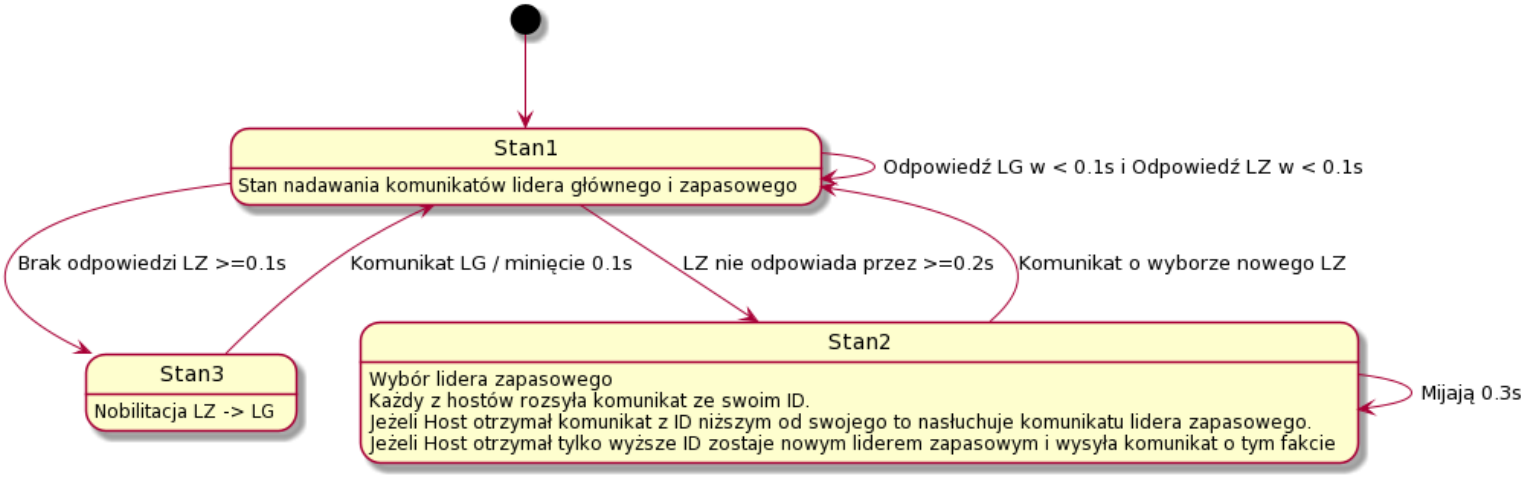
Wybór nowego lidera głównego

Aktorzy: Węzły

1. Następuje zidentyfikowanie lidera zapasowego
    a. Nie ma lidera zapasowego
    b. Następuje wybór lidera zapasowego
    c. Wybrany lider zapasowy zostaje liderem głównym
2. Lider zapasowy zostaje liderem głównym

W miejscu gdzie pojawia się “użytkownik” w testach bedziemy zastępować skrypytami automatycznymi zbudowanymi w środowisku mininet.org

Diagram stanów



Rys.1 Diagram stanów, na którym LZ to Lider zastępczy, LG to Lider główny
https://bit.ly/33qIAUg

Analiza możliwych sytuacji błędnych i proponowana ich obsługa.

1. **Istnieje więcej niż jeden lider główny**
   Uruchomiona zostaje procedura usunięcia wszystkich liderów głównych. Następnie pojedynczy lider główny zostaje wybrany zgodnie z algorytmem wyboru lidera głównego.

2. **Istnieje więcej niż jeden lider zastępczy**
   Uruchomiona zostaje procedura usunięcia wszystkich liderów zastępczych. Następnie pojedynczy lider zastępczy zostaje wybrany zgodnie z algorytmem wyboru lidera zastępczego.

3. **Istnieje węzeł, który przestał wysyłać komunikaty**
   Zostaje on usunięty z grupy węzłów.

Wybrane środowisko sprzętowo-programowe:

Docelowy system operacyjny: Linux (Ubuntu 20.04 LTS)

Język programowania: C++, Python

Wykorzystane biblioteki: std

Debugowanie: debugger wbudowany w środowisko (CLion/Visual Studio Code)

Testowanie: skrypty w Pythonie + analiza wysyłanych komunikatów w programie Wireshark

Dokumentacja: Doxygen

Budowanie aplikacji: CMake

Środowisko do analizy ruchu sieciowego: Wireshark

Konteneryzacja: Docker

Architektura rozwiązania:

Architektura rozwiązania zostanie podzielona na następujące moduły:

- **Zarządzanie węzłami**

  Moduł pozwalający zarządzać wielkością sieci. Umożliwia dodawanie i usuwanie węzła przyjmując jako wejście ID węzła oraz listowanie aktywnych węzłów wraz z ich identyfikatorami i rolami, jeśli je posiadają (lider, lider zastępczy).

- **Odbiorca komunikatów**

  Moduł służący do odbioru rozsyłanych komunikatów. Sprawdza poprawność otrzymywanych danych, wykrywa potencjalne próby ataku/złośliwe komunikaty.

- **Nadawca komunikatów**

  Moduł służący do rozsyłania komunikatów. Wykorzystywany do powiadamiania węzłów przez lidera i wicelidera o obecności oraz do procedury wyboru nowych lidera i wicelidera.

- **Obsługa algorytmu wyboru lidera i lidera zastępczego**

  W przypadku wykrycia braku lidera zapasowego uruchamiana jest procedura dążąca do wyboru nowego węzła pełniącego tę rolę.

  Wykrycie braku lidera zapasowego następuje poprzez nieotrzymanie od niego komunikatu potwierdzającego jego obecność w sieci w czasie *0.1s*. Po upływie tego czasu, każdy z hostów odczekuje jeszcze *0.1s*.

  Jeżeli po upływie czasu *0.2s* od rozesłania ostatniego komunikatu lider nie stał się aktywny, każdy z hostów rozsyła komunikat, w którym przekazuje swoje ID. Jeśli host otrzymał komunikat z ID niższym od swojego, nasłuchuje aktywności nowego lidera zapasowego w czasie nie przekraczającym *0.2s*, po upływie tego czasu ponownie rozsyła komunikat zawierający jego ID. W przypadku jeżeli host nie otrzymał komunikatu z ID niższym od swojego (jego ID jest najniższe ze wszystkich które zostały rozesłane) w czasie *0.1s*, staje się on nowym liderem i rozsyła komunikat o swojej aktywności.

  W przypadku wykrycia braku lidera głównego, lider zapasowy staje się nowym liderem głównym, a uruchamiana jest procedura wyboru lidera zapasowego.

W przypadku wykrycia braku zarówno lidera głównego oraz lidera zapasowego, najpierw wybierany jest lider zapasowy, który automatycznie staje się liderem głównym, a następnie wybierany jest lider zapasowy.

Typy wiadomości:

LEADERS_MESSAGE: 0 {id} {rola}
Wiadomość wysyłana przez lidera informująca o jego obecności.

ID_MESSAGE 1 {id}
Wiadomość wysyłana przez węzły w momencie gdy następuje wybór lidera zapasowego.

SESSION_CONTROLLER_KILL_MSG 2 {id}
Komunikat wysyłany przez zarządcę do usuwania węzłów.

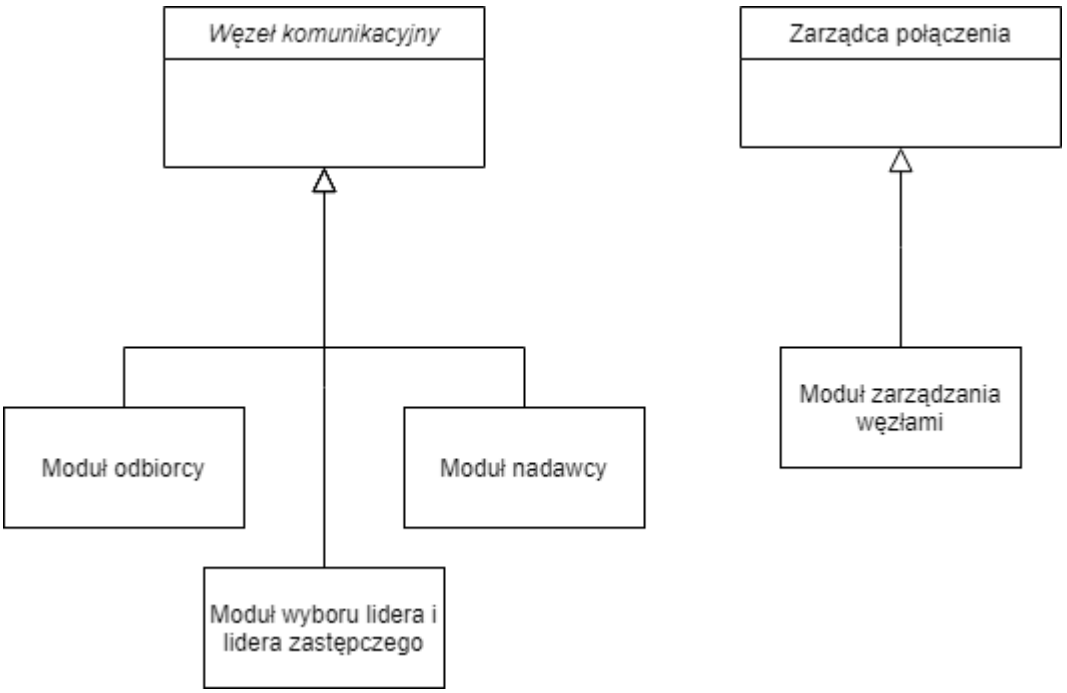
MORE_THAN_ONE 3 {id} {rola}
Wiadomość wysyłana w momencie gdy węzeł stwierdzi, że istnieje więcej niż jeden lider

- **Moduł do Wireshark**

  Umożliwia analizę, wyświetlanie oraz śledzenie komunikatów przesyłanych w sieci przy wykorzystaniu programu Wireshark. Moduł taki wymaga zdefiniowania protokołu komunikacyjnego, którym będą się posługiwać węzły.

- **Skrypty testujące**

  Skrypty umożliwią przeprowadzanie testów rozwiązania. Będą sprawdzały poprawność wyboru lidera/wicelidera, usuwania zdublowanych liderów/wiceliderów, dodawania i usuwania węzła, wysyłania/odbierania komunikatów. Będą uruchamiały odpowiedni scenariusz i sprawdzały, czy efekty są zgodne ze spodziewanymi. W razie braku zgodności poinformują o miejscu błędu.
  Nie zostały one zaimplementowane



Rys.2 Diagram zależności

Lista komunikatów z określeniem nadawców i odbiorców:

Wszystkie węzły są odbiorcami każdego komunikatu.

- Komunikat o znaczeniu “Jestem liderem” - wysyłany przez aktualnego lidera.
- Komunikat o znaczeniu “Jestem liderem zastępczym” - wysyłany przez lidera zastępczego.
- Id węzła - Wysyłane przez każdy węzeł w sieci, kiedy lider zastępczy pozostaje niesłyszalny przez określony czas lub został mianowany liderem.

Sposób testowania.

1. Testy integracyjne sprawdzające komunikację między węzłami
2. Testy integracyjne sprawdzające poprawność algorytmu wyboru lidera
3. Testy integracyjne sprawdzające poprawność algorytmu wyboru lidera zastępczego

Nie zostały one zaimplementowane

Sposób demonstracji rezultatów, tj. scenariusze testów akceptacyjnych do zaprezentowania przy odbiorze projektu.

Scenariusz pierwszy: wybór lidera głównego oraz zastępczego podczas inicjalizacji

1. Uruchomienie węzłów
2. Uruchomienie modułu do wireshark oraz obserwowanie komunikacji między węzłami
3. Wylistowanie węzłów oraz sprawdzenie, czy mianowany został lider główny oraz lider zastępczy
4. Zamknięcie programu

Scenariusz drugi: ponowny wybór lidera głównego

1. Uruchomienie 5 węzłów
2. Systematyczne listowanie węzłów, aż do czasu pojawienia się lidera głównego
3. Usunięcie węzła oznaczonego jako lider główny
4. Obserwowanie algorytmu wyboru lidera głównego przy użyciu modułu wireshark
5. Wylistowanie węzłów
6. Zamknięcie programu

Scenariusz trzeci: ponowny wybór lidera zastępczego

1. Uruchomienie 5 węzłów
2. Systematyczne listowanie węzłów, aż do czasu pojawienia się lidera zastępczego
3. Usunięcie węzła oznaczonego jako lider zastępczy
4. Obserwowanie algorytmu wyboru lidera zastępczego przy użyciu modułu wireshark
5. Wylistowanie węzłów
6. Zamknięcie programu

Scenariusz czwarty: jednoczesny wybór lidera głównego i zastępczego

1. Uruchomienie 10 węzłów
2. Systematyczne listowanie węzłów, aż do czasu pojawienia się lidera głównego oraz lidera zastępczego
3. Usunięcie węzła oznaczonego jako lider główny oraz węzła oznaczonego jako lider zastępczy
4. Obserwowanie komunikacji przy użyciu modułu do programu wireshark
5. Wylistowanie węzłów
6. Zamknięcie programu

Harmonogram prac oraz podział pracy

- Faza I - Prototyp do 10.05.2021
- Faza II - Działający system do 26.05.2021 (ostatecznie do 2.06.2021)
- Faza III - Poprawki do 7.06.2021

Faza I

Stworzenie podstawowej komunikacji przez loopback

Cel: Węzły jako procesy komunikują się na interfejsie loopback
Odpowiedzialny: Daniel Chmielewiec, Wojciech Moczydłowski

Stworzenie modułu do Wiresharka

Cel: Moduł do wiresharka, pomagający nam filtrować pakiety oraz analizować role węzłów oraz ich wiadomości
Odpowiedzialny: Jan Piotrowski

Stworzenie modułu zarządzania węzłami (CLI)

Cel: Możliwość dodawania, usuwania oraz wylistowywania węzłów - moduł zarządcy oraz CLI
Odpowiedzialny: Kamil Jabłonowski

CI/CD

Cel: Sprawdzenie czy jest możliwe CI/CD w ramach wydziałowego gitlaba
Odpowiedzialny: Jan Piotrowski

Niestety nie udało się. Okazuje się, że nasz wydziałowy gitlab nie ma wykupionego żadnego “workera” na którym mogłyby zostać uruchomione skrypty do CI/CD. Niestety też nie łapiemy się do programu darmowych tokenów na start jako Politechnika Warszawska.
Stworzenie podstawowych testów

Testy

Cel: Stworzenie podstawowych testów - konfiguracja CMake’a oraz instalacja i zapoznanie się z biblioteką
Odpowiedzialny: Kamil Jabłonowski

Faza II

Implementacja algorytmu Lidera i Wicelidera

Cel: Stworzenie podstawowej funkcjonalności opisanej w dokumentacji, algorytm, który zakłada, że jeden z węzłów zostaje liderem, a jeden wiceliderem i odbywa się odpytywanie. Komunikacja odbywa się na adresie multicast ff02::1.
Odpowiedzialny: Daniel Chmielewiec

Obsługa sytuacji wyjątkowych

Cel: Implementacja obsługi sytuacji wyjątkowych (np. w jednym czasie aktywni są dwaj liderzy lub dwaj liderzy zastępczy)
Odpowiedzialny: Kamil Jabłonowski

Dopracowanie zarządcy - uniezależnienie go od procesów

Cel: Zarządca wysyła wiadomości o zabiciu węzła oraz otrzymuje informacje o węzłach za pomocą pakietów, a nie wskaźników.
Odpowiedzialni: Daniel Chmielewiec, Kamil Jabłonowski

Konteneryzacja

Cel: Każdy węzeł jako kontener dockera. Dostowanie SessionControllera do tworzenia i usuwania kontenerów.
Odpowiedzialny: Jan Piotrowski

Uaktualnienie modułu do wiresharka

Cel: Dostosowanie modułu do wiresharka do aktualnych wiadomości (wersja po implementacji algorytmu lidera i wicelidera).
Odpowiedzialny: Jan Piotrowski

Faza III

Dokumentacja

Cel: Udokumentowanie działania naszego systemu. Stworzenie README
Odpowiedzialny: Wojciech Moczydłowski

Nie zrealizowano:

Testy integracyjne

Cel: Implementacja testów integracyjnych (scenariuszy) oraz dokładne otestowanie algorytmu wyboru lidera i wicelidera.

Analiza podatności bezpieczeństwa

Utworzony system posiada kilka podatności:

1. Komunikaty nie są szyfrowane, więc atakujący jest w stanie odczytać przesłaną wiadomość
2. Uprzednio analizując komunikaty, atakujący jest w stanie stworzyć węzeł, który zostanie podłączony do sieci i będzie rozsyłał błędne komunikaty

Wnioski wykonanego testowania

W projekcie nie zdążyliśmy zrobić testów automatycznych, testowaliśmy za pomocą testów manualnych. W celu polepszenia jakości testowania utworzyliśmy moduł nasłuchujący. Pozwala on na zbieranie informacji o obecności węzłów oraz przypisanych im

ról. Dzięki niemu byliśmy w stanie przetestować poprawność wysyłanych komunikatów oraz algorytm wyboru lidera/vice lidera.

Testowanie aplikacji składającej się z wielu komunikujących się wzajemnie węzłów jest zdecydowanie trudniejsze niż jednego uruchomionego programu. Trudność wynikała z braku możliwości zaimplementowania testów związanych z konkretnym węzłem. Potrzebny był oddzielny moduł nasłuchujący.

Uruchomienie

Sposób uruchomienia aplikacji został przedstawiony w pliku readme.md na serwerze kontroli wersji.

Podsumowanie

Realizując projekt zdobyliśmy wiedzę na temat programowania aplikacji z wykorzystaniem gniazd BSD oraz dokładniej zrozumieliśmy sposób działania komunikacji sieciowej. Ponadto podnieśliśmy kompetencje w tworzeniu kilku elementowej architektury. Programując w języku C++ zgłębiliśmy jego mechanizmy, powtórzyliśmy api wątków oraz mutexów. Użyliśmy konteneryzacji przy pomocy Dockera oraz zbudowaliśmy moduł do wiresharka co było ciekawym poznaniem nowych obszarów.

Współpraca odbywała przy użyciu serwera kontroli wersji GitLab, komunikatorów: Slack oraz Messenger. W dobie luźnych stosunków studenckich trudno jest się przestawić na Slacka, więc używaliśmy trochę tego trochę tamtego, co raczej było błędem. Messenger jest dobry by kogoś “obudzić z letargu”, lecz nie po to by dobrze się komunikować i wyszukiwać stare wiadomości, w których były bardzo informacje.

Bardzo przydatnym narzędziem okazał się gitlab, chociaż już pod koniec II fazy nie korzystaliśmy z mechanizmu Issues, jest to pewnie kwestia lenistwa i niechęci programistów do rozliczania się z godzin. Miał on na początku jednak swoje wielkie plusy. Można było odpowiednio stworzyć małe zadanie, o określonym zakresie oraz celu i przyporządkować zadania do poszczególnych osób oraz logować czas. Było to szczególnie przydatne, gdy każdy z nas siadał do projektu w innym momencie, ale widział jakie issues są już zrealizowane, kto ile spędził czasu w projekcie, oraz mógł szybko przeczytać cel swojego Issue.

Udało nam się pracować “agile” i dobrze zorganizować się wokół projektu. Wydaję mi się, że jest to zasługa bardzo dobrego rozeznania w trakcie projektu wstępnego i zbudowania wyobrażenia projektu, rozdzielenia zadań i podzielenia go na etapy.

W ramach projektu stworzyliśmy zbiór plików ważących sumarycznie: 778KB

Projekt zajął ok. 25-35h na każdego członka zespołu.

Adres projektu na serwerze kontroli wersji.

https://gitlab-stud.elka.pw.edu.pl/jpiotro1/tin-leader-and-viceleaders

Każda osoba zarejestrowana wydziałowym mailem powinna mieć dostęp. Jeżeli jednak są jakieś problemy proszę dać znać, spróbujemy przekazać w inny sposób.