Bartłomiej Partyka - Projekt wstępny TIN

Bartłomiej Partyka, Michał Urbański, Krzysztof Blankiewicz, Tomasz Załuska

6 kwietnia 2020

1 Spis treści

2 Treść zadania

Urządzenia przechowują zmienne statyczne w pamięciach ulotnej i nieulotnej. Zmienne dynamiczne przechowywane są w pamięci ulotnej. Projektowany system komunikacji działa niezależnie od systemu wykrywania zaniku zasilania i zachowywania stanu urządzenia. Aplikacja korzystająca z usług tego systemu potwierdza konsumpcję danych. Celem systemu jest retransmisja niepotwierdzonych danych po wznowieniu zasilania. System używa stosu TCP/IPv6. Zaprojektować API systemu komunikacyjnego. Ponadto, należy zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów. (Być może pomocnym będzie przejrzenie RFC 5326 "Licklider Transmission Protocol").

3 Nazwa własna

ICS - Indomitable Communications System

4 Założenia funkcjonalne i niefunkcjonalne

4.1 Założenia funkcjonalne

System ICS używa IPv6 jako protokołu warstwy trzeciej.
Klient:

- Łączy się z serwerem komunikacji
- Wyświetla listę dostępnych klientów
- Akceptuje połączenie z innym klientem lub sam inicjuje połączenie
- Ma możliwość wysyłania i odbierania dużych plików oraz komunikatów tekstowych
- W wypadku zaniku zasilania/połączenia jest w stanie bez przeszkód wznowić komunikację
- Ma możliwość utrzymywania wielu połączeń jednocześnie

Serwer:

- Umożliwia nawiązywanie połączeń między klientami
- Umożliwia autoryzację użytkowników poprzez hasło serwerowe
- Przechowuje w pamięci ulotnej informację o przesłanych danych aby umożliwić wznowienie komunikacji
- Komunikacja jest utrzymywana przez zaniki połączenia aż do zamknięcia połączenia przez użytkownika lub po określonym w konfiguracji czasie nieaktywności
- Kontrola przesyłania danych realizowana jest przez specjalnie zaprojektowane API

Moduł wireshark:

- Służy do testów i debugowania aplikacji
- Wspomaga proces tworzenia systemu komunikacji

4.2 Założenia niefunkcjonalne

- Niezawodna komunikacja
- Prosty w obsłudze interfejs klienta
- Możliwość konfiguracji parametrów serwera
- Możliwość ustawiania parametrów połączenia przez klienta

5 Przypadki użycia

1. Klient łączy się z serwerem, loguje i wybiera z listy innego klienta, z którym inicjuje połączenie. Każdy klient może być połączony z kilkoma klientami jednocześnie.
2. Klient może wysłać wiadomość (tekst lub plik) do dowolnego klienta, z którym jest połączony.
3. Wiadomość po podzieleniu na bloki jest wysyłana na serwer, który podczas odbierania danych przechowuje informację o ilości odebranych dotychczas danych. W przypadku zaniku i wznowieniu zasilania, po ponownym połączeniu się z serwerem nadawca automatycznie wznawia wysyłanie wiadomości.
4. Po udanym wysłaniu pliku przez nadawcę serwer rozpoczyna przesyłanie pliku odbiorcy, prosząc użytkownika o potwierdzenie odbioru każdego wysłanego bloku. Serwer przechowuje informację o ilości poprawnie wysłanych danych, w przypadku przerwania, po ponownym połączeniu, serwer automatycznie wznawia wysyłanie do klienta.
5. Po udanej transmisji wiadomość jest usuwana z serwera (lub po pewnym określonym czasie)
6. Transmisja wiadomości może być anulowana w każdej chwili przez obie strony.
7. Odebrany plik zapisuje się na dysku odbiorcy, odebrany tekst wyświetla się w konsoli.

Użytkownik może się w dowolnej chwili rozłączyć z serwerem.

6 Środowisko

6.1 Systemy operacyjne

Projekt tworzony jest dla systemów Linux. Użyte dystrybucje to:

- Void Linux 64-bit, kernel 5.4

- Debian Linux 10 64-bit, kernel 4.19
- Arch Linux 64-bit, kernel 5.5.10
- Ubuntu Linux

6.2 Języki i biblioteki

Klient i serwer będą pisane w języku C/C++, z wyjątkiem interfejsu użytkownika który będzie pisany w języku Python.

Używane biblioteki:

- boost
- pthread
- ncurses

Moduły do Wireshark napisane w języku Lua.

6.3 Narzędzia testowe

Jedym z ważniejszych narzędzi testowych będzie Wireshark, wraz z specjalnie napisanym modułem do analizy komunikacji w systemie. Do testowania funkcji programu posłuży biblioteka boost.

7 Architektura rozwiązania

System ICS składa się z:

- Serwera
- Aplikacji klienckiej
- Interfejsu użytkownika
- Modułu Wireshark

Serwer ICS ma za zadanie zawiązywać, kontrolować i utrzymywać połączenia między klientami. Umożliwia on również podstawową autoryzację w postaci hasła. Podczas transferu danych, serwer i klient potwierdzają odbiór bloków danych i są w stanie wznowić transmisję po awarii lub utracie połączenia. Serwer ics posiada plik konfiguracyjny, aby umożliwić administratorowi dostosowanie go do swoich potrzeb.

Klient ICS jest w stanie połączyć się z wybranym serwerem ICS, a następnie nawiązać połączenie z dowolnym klientem dostępnym na serwerze. Połączenie te umożliwia niezawodną transmisję danych - plików lub wiadomości tekstowych. Każdy klient ma możliwość utrzymywania wielu połączeń na raz. Użytkownik wchodzi w interakcje z klientem poprzez **prosty interfejs**, umożliwiający

przeglądanie otwartch połączeń, przesyłanie danych i modyfikacje parametrów połączenia.

Moduł Wireshark tworzony jest aby ułatwić zespołowi pracę nad programem i projektowanie API do komunikacji między serwerem a klientami. Posłuży również do zaprezentowania ostatecznej wersji programu.

8 Sposób testowania

Program będzie testowany pod kątem zgodności z głównym przyjętym założeniem - wznowieniem komunikacji w przypadku braków w zasilaniu, połączeniu.

Podczas testowania projektu API komunikacyjnego użyty będzie program Wireshark, aby mieć dokładniejszy wgląd w dane przesyłane między serwerem a klientami.

Prowadzone będą również testy jednostkowe i regresyjne, używając biblioteki boost.

9 Prezentacja

1. Pomyślna kompilacja
2. Uruchomienie serwera i podłączenie klientów
3. Prezentacja komunikacji między klientami
4. Prezentacja komunikacji dla różnych przypadków utraty połączenia/zasilania.
5. Obserwacja ruchu programem Wireshark

10 Podział pracy

- Bartłomiej Partyka - Moduł Wireshark, Interfejs, Klient.
- Michał Urbański - Testy, boost, makefile, pomoc w pracy nad głównymi modułami.
- Krzysztof Blankiewicz - API komunikacji, Serwer(niezawodna komunikacja).
- Tomasz Załuska - Gniazda, Serwer(zarządzanie połączeniami).

11 Harmonogram pracy

Proponowany jest następujący harmonogram pracy:

25.04.2020 - Prototyp - działająca aplikacja klienta i serwera umożliwiające komunikację

10.05.2020 - Działająca wersja wersja z zaimplementowanym mechanizmem przeciwdziałania zanikom zasilania

12 Repozytorium github

Projekt jest publicznie dostępny pod adresem: `https://github.com/six-pd/tin-ics`