TinyAd
Projekt wstępny

Rafał Kulus (Lider),
Damian Kolaska,
Kamil Przybyła

https://bitbucket.org/BlueAlien99/tinyad

6 maja 2021

Spis treści

1 Treść zadania

W systemie pracuje stacja zarządzająca i zbiór (do kilkudziesięciu tysięcy) sterowników paneli reklamowych. Stacja multicastowo dystrybuuje treści i harmonogramy ich wyświetlania. Harmonogram określa okresy wyświetlania i czas przechowywania. Sterowniki unicastowo raportują swój stan i zgłaszają błędy. Błąd w transmisji multicastowej obsługiwany jest retransmisją multi- lub unicastową w zależności od liczby otrzymanych komunikatów NAK. System ma pracować w prywatnej sieci IPv6. Stacja zarządzająca może wysyłać unicastowo ważne treści informacyjne do natychmiastowego wyświetlenia przez wybrane sterowniki. Należy też zaprojektować moduł do Wireshark umożliwiający wyświetlanie i analizę zdefiniowanych komunikatów.

2 Format plików

2.1 Konfiguracja dla paneli

`panel_name` i `tags` są opcjonalne i służą do identyfikowania paneli w celu wyświetlenia ważnych treści. Są wysyłane do stacji zarządzającej podczas rejestrowania panelu. Domyślna nazwa i lokalizacja pliku to `./config.lfd`.

`num_retries` — ilość ponowień komunikatu (`-1` = nie przerywaj), jeśli stacja zarządzająca nie odpowiada
`timeout` — odstęp pomiędzy komunikatami w sekundach

```
server_ip=fe80::1234
panel_name=Lobby Front
tags=[oled][huge][orange apricot]
num_retries=5
timeout=30
```

2.2 Harmonogram

`expire`, `weekday`, `hour` są opcjonalne. W przypadku braku `expire`, zawartość nigdy nie wygaśnie. W przypadku braku pozostałych dwóch pól, zawartość będzie wyświetlana niezależnie od dnia tygodnia i / lub godziny. Pierwszeństwo ma pierwsza zawartość, która spełnia wszystkie warunki. Dni tygodnia są indeksowane od 1.

```
file="./img.jpg"
expire="2021-05-15 1930"
weekday=[1][2][3]
hour=[0530-0830][1500-1900]
---
file="./vids/video.mp4"
expire="2021-06-30"
```

3 Interfejs użytkownika

3.1 Stacja zarządzająca

Stacja zarządzająca będzie posiadała prosty CLI. Dane mogą być wprowadzone przez użytkownika:

```
./station
$ <command1>
$ <command2>
...
$ <commandN>
```

Dostępne polecenia:

1. `schedule ./my_sched.sch`
   Spowoduje rozesłanie nowego harmonogramu i wymaganych plików do paneli.
   `./my_sched.sch` to ścieżka do pliku zawierającego harmonogram.
2. `panic ./emergency.mkv --name "Lobby Front"`
3. `panic ./emergency.mkv --tags [oled][huge]`
   Spowoduje wysłanie ważnych treści do paneli, których nazwa to `Lobby Front` lub które posiadają co najmniej jeden z podanych tagów. Ważne treści będą wyświetlane tak długo, aż panel nie otrzyma nowego harmonogramu poleceniem `schedule`
4. `bye`
   Spowoduje wyłączenie stacji zarządzającej
5. `top 10`
   Wyświetlenie 10 ostatnich linii logów o poziomie info lub wyższym.

3.2 Panele

Panele nie będą posiadały interfejsu użytkownika. Wszystkie potrzebne informacje będą wczytywane z prostego pliku konfiguracyjnego.

4 Pozostałe założenia funkcjonalne

Proces uruchomienia paneli powinien się ograniczyć tylko i wyłącznie do startu aplikacji klienta, która sama wykona wszystkie wymagane czynności, aby panel mógł zacząć wyświetlać zawartość. Po udanej rejestracji panelu w stacji zarządzającej zostanie do niego natychmiastowo wysłany ostatnio użyty harmonogram (jeśli taki istnieje), a także adres multicast do nasłuchiwania.

5 Założenia niefunkcjonalne

Stacje zarządzające i panele powinny sobie radzić ze wszystkimi przewidzianymi błędami, aby zapewnić nieprzerwaną pracę, stabilność i niezawodność, w szczególności błędy transmisji danych nie powinny uniemożliwić poprawnego działania systemu.

6 Przypadki użycia

6.1 Rejestracja panelu reklamowego

1. Użytkownik uruchamia panel reklamowy (opcjonalnie podając plik konfiguracyjny)
2. Panel rejestruje się, wysyłając komunikat stacji zarządzającej
3. Panel otrzymuje harmonogram i treści do wyświetlenia
4. Panel wyświetla zlecane treści

6.2 Zarządzanie harmonogramem oraz wyświetlanymi treściami

1. Użytkownik, za pośrednictwem stacji zarządzające, modyfikuje informacje o harmonogramie lub treściach
2. Stacja zarządzająca wysyła aktualne dane do paneli
3. Panele wyświetlają treści zgodnie z otrzymanym harmonogramem

6.3 Zlecenie natychmiastowego wyświetlenia informacji

1. Użytkownik, za pośrednictwem stacji zarządzające, zleca natychmiastowe wyświetlenie informacji podanej podgrupie paneli
2. Stacja zarządzająca wysyła treść do natychmiastowego wyświetlenia
3. Panele wyświetlają żądaną treść, ignorując treści wynikające z harmonogramu

7 Obsługa błędów

7.1 Brak harmonogramu / błędny harmonogram

Panel będzie wyświetlał `default content`, w logach zostanie zarejestrowana odpowiednia informacja i będą następowały regularne próby uzyskania harmonogramu.

7.2 Stacja zarządzająca nie odpowiada na komunikat

1. Cykliczne ponawianie komunikatu
2. Restart aplikacji po wyczerpaniu prób

Konfigurowalne parametry zostały opisane w sekcji 2.1.

7.3 Brak pliku konfiguracyjnego

Wyświetlenie odpowiedniej informacji zwrotnej użytkownikowi i zamknięcie aplikacji.

7.4 Błąd transmisji danych

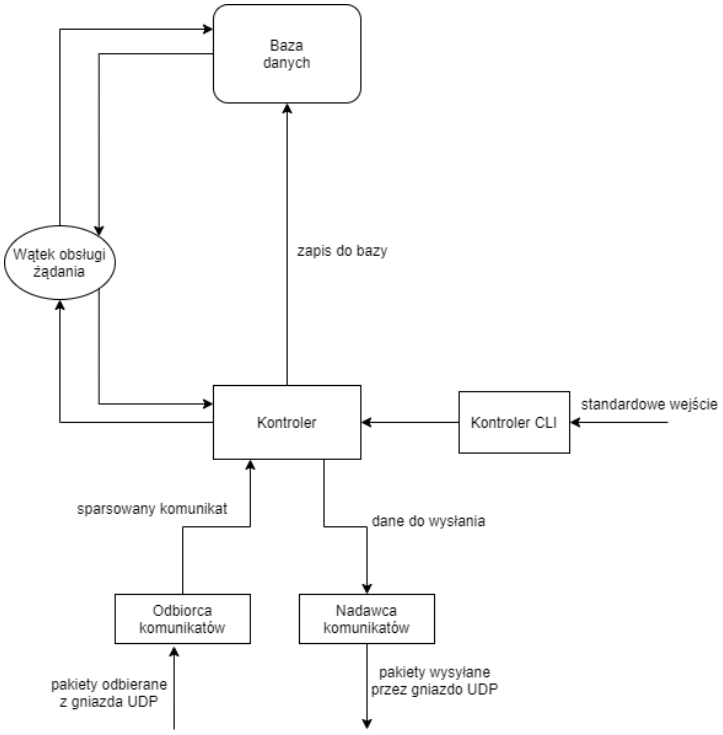
W przypadku gdy w wyniku błędu sieciowego jeden lub więcej pakietów zostanie zagubiony, odbiorca wyśle komunikat z prośbą o ponowne przesłanie brakujących segmentów.

8 Środowisko programistyczne

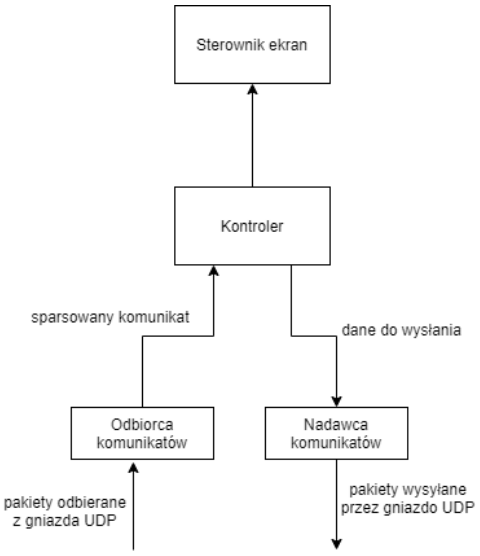
Projekt zostanie zrealizowany na systemie operacyjnym Ubuntu 20.04 LTS, w języku C++17. Do testów zostanie wykorzystana biblioteka GoogleTest, a do debugowania programy gdb, valgrind oraz Wireshark. Do budowania projektu zostaną wykorzystane CMake oraz g++. Do logowania użyjemy biblioteki Loguru (https://github.com/emilk/loguru).

9 Architektura

9.1 Stacja zarządzająca



9.2 Panel



Odbiorca komunikatów: Odbiera pakiety z gniazda UDP, formuje z nich komunikaty, weryfikuje je, a następnie wysyła do kontrolera.
Nadawca komunikatów: Odbiera dane od kontrolera, formuje z nich pakiety i wysyła je przez gniazdo UDP.
Kontroler: Główny moduł zarządzający. Koordynuje on pracę reszty modułów, przekierowuje komunikaty, tworzy wątki do obsługi żądań, przekazuje polecenia do sterownika ekranu.
Kontroler CLI: Odbiera polecenia podawane na standardowe wejście.
Baza danych: Baza danych przechowująca informacje na temat: harmonogramu, wyświetlanych treści. Aby nie komplikować niepotrzebnie rozwiązania użyjemy SQLite.

10 Przesyłanie danych

10.1 Architektura komunikatów

Problemy

- jak obsłużyć komunikaty różnych typów
- aby łatwo obsługiwać współbieżność, należałoby zapewnić bezstanowość komunikacji po stronie serwera
- jak czytać komunikaty o różnych długościach

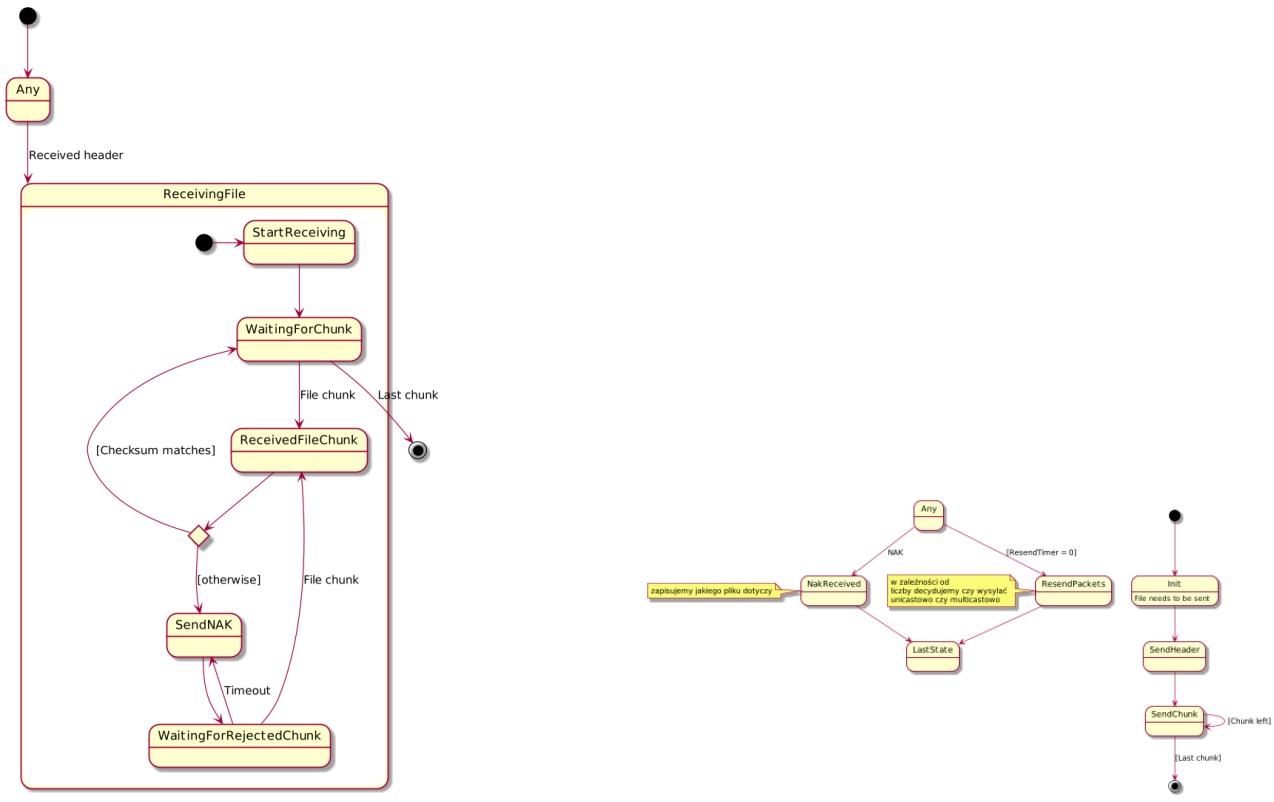
Rozwiązania

- 4 pierwsze bajty -> identyfikator komunikatu (np. MDAT - dane multimedialne)
- w każdym komunikacie do serwera klient podaje czego oczekuje, np. trzeba dosłać fragmenty pliku o nr 3, 5, 13
- czytamy po 4 bajty do momentu aż przeczytamy cały identyfikator, a następnie odczytamy tyle bajtów, ile zajmuje odpowiednia struktura

Przykładowy komunikat

```
struct __attribute__ ((__packed__)) Multimedia {
    uint8_t tag[4] = "MDAT";
    uint8_t data[508];
};
```

10.2 Pliki multimedialne



(a) Klient (b) Serwer

Rysunek 1: Automaty stanów dla niezawodnej transmisji plików

Pliki multimedialne są zbyt duże, żeby przesyłać je w jednym datagramie. Należy je zatem podzielić. Przesyłając plik w kawałkach, może się okazać, że część pakietów nie dotrze albo pakiety dotrą w innej kolejności. Aby rozwiązać ten problem zamierzamy:

- numerować pakiety
- w każdym pakiecie podawać numer pierwszego i ostatniego pakietu w sekwencji
- w ramach optymalizacji uprzednio podzielić pliki na kawałki i tak przechowywać je w bazie danych

Transmisja pliku rozpoczyna się od wysłania przez serwer wiadomości z nagłówkiem. Nagłówek zawiera informacje takie jak: nazwa pliku, jego rozmiar, liczba części, na które zostanie podzielony, oraz identyfikator (np. UUID), po którym będziemy rozpoznawać, czy pakiety dotyczą danego pliku. Po wysłaniu nagłówka rozpoczyna się transmisja pliku. W ramach transmisji wysyłane są wiadomości zawierające identyfikator (pliku), numer części, dane i sumę kontrolną. Klient odbierając pakiet danych sprawdza, czy dotyczy on obecnie przesyłanego pliku (porównując identyfikator). Następnie weryfikuje sumę kontrolną. Jeśli suma się nie zgadza, wysyła komunikat NAK do serwera. Serwer nie odsyła pakietów od razu, a zbiera komunikaty NAK przez pewien czas, po którym decyduje, czy odesłać pakiety unicastowo czy multicastowo.

11 Sposób testowania

Do testowania wykorzystany zostanie moduł do Wiresharka, za pomocą którego badać będziemy, czy stacja zarządzająca i panele wysyłają odpowiednie komunikaty oraz czy ich zawartość jest zgodna z oczekiwaniami. Na przykład podczas uruchamiania panelu, oczekiwać będziemy, że zostanie wysłany komunikat do stacji zarządzającej, którego zawartość będzie zgodna z konfiguracją panelu, oraz że stacja kontrolna odeśle adekwatną odpowiedź.
Przydatne również mogą okazać się logi stacji zarządzającej, z których będziemy mogli próbować odczytać, jakie akcje stacja wykonuje.
Przygotowane zostaną również testy regresyjne zautomatyzowane skryptami, które przetestują działanie systemu z wykorzystaniem maszyn wirtualnych symulujących stację zarządzającą i panele.

12 Sposób demonstracji rezultatów

12.1 Demonstracja działania dystrybucji harmonogramów oraz treści

1. Uruchomienie stacji zarządzającej
2. Podłączenie panelu do działającej stacji zarządzającej
3. Zlecenie przesłania harmonogramu oraz treści, za pośrednictwem interfejsu tekstowego stacji zarządzającej
4. Zaprezentowanie panelu wyświetlającego wskazane treści, zgodnie z harmonogramem

12.2 Demonstracja działania dystrybucji ważnych treści

1. Uruchomienie stacji zarządzającej
2. Podłączenie kilku paneli o różnych nazwach i tagach do działającej stacji zarządzającej
3. Zlecenie wyświetlenia pilnej treści panelowi o wskazanej nazwie
4. Zaprezentowanie panelu wyświetlającego żądaną treść
5. Zlecenie wyświetlenia pilnej treści podzbiorowi paneli o wskazanym tagu
6. Zaprezentowanie paneli wyświetlających żądaną treść

7. Wysłanie żądania wyświetlania treści zgodnie z harmonogramem
8. Pokazanie paneli, wyświetlających treści według harmonogramu
9. Próba zlecenia wyświetlenia ważnej treści nieistniejącemu panelowi
10. Prezentacja informacji o nieistniejącym panelu w interfejsie tekstowym

12.3 Demonstracja działającego mechanizmu wyłączania paneli

1. Uruchomienie stacji zarządzającej
2. Podłączenie kilku paneli do działającej stacji zarządzającej
3. Zlecenie wyświetlenia pilnej treści panelowi o wskazanej nazwie
4. Zaprezentowanie paneli wyświetlających treści
5. Wyłączenie jednego z paneli
6. Zlecenie wyświetlenia pilnej treści wyłączonemu panelowi
7. Prezentacja informacji o nieistniejącym panelu w interfejsie tekstowym oraz pozostałych, działających paneli

12.4 Podział prac w zespole

- **Rafał Kulus** – warstwa abstrakcji na systemowe gniazda UDP (multicast, unicast), wielowątkowa obsługa żądań
- **Damian Kolaska** – odbieranie i nadawanie komunikatów, interfejs tekstowy, testy integracyjne
- **Kamil Przybyła** – moduł do Wiresharka, baza danych, obsługa harmonogramów

13 Harmonogram prac

- Do 27 kwietnia – dokumentacja wstępna
- 19-26 maja – działająca dystrybucja harmonogramów oraz wyświetlanie treści przez panele
- Do 1 czerwca – wysyłanie ważnych treści oraz komunikacja z warstwą składowania danych
- Do 7 czerwca – dokumentacja końcowa