SIK-2018 -- Zadanie 2

Zadanie polega na napisaniu nadajnika i odbiornika internetowego radia.

Zadanie składa się z dwóch części.

"Zmienne" użyte w treści

- MCAST_ADDR - adres rozgłaszania ukierunkowanego, ustawiany obowiązkowym parametrem `-a` nadajnika
- DISCOVER_ADDR - adres używany przez odbiornik do wykrywania aktywnych nadajników, ustawiany parametrem `-d` odbiornika, domyślnie 255.255.255.255
- DATA_PORT - port UDP używany do przesyłania danych, ustawiany parametrem `-P` nadajnika~~ i odbiornika~~, domyślnie 20000 + (numer_albumu % 10000)
- CTRL_PORT - port UDP używany do transmisji pakietów kontrolnych, ustawiany parametrem `-C` nadajnika i odbiornika, domyślnie 30000 + (numer_albumu % 10000)
- UI_PORT - port TCP, na którym udostępniany jest prosty interfejs tekstowy do przełączania się między stacjami, domyślnie 10000 + (numer_albumu % 10000); ustawiany parametrem `-U` odbiornika
- PSIZE - rozmiar w bajtach paczki, ustawiany parametrem `-p` nadajnika, domyślnie 512B
- BSIZE - rozmiar w bajtach bufora, ustawiany parametrem `-b` odbiornika, domyślnie 64kB (65536B)
- FSIZE - rozmiar w bajtach kolejki FIFO nadajnika, ustawiany parametrem `-f` nadajnika, domyślnie 128kB.
- RTIME - czas (w milisekundach) pomiędzy wysłaniem kolejnych raportów o brakujących paczkach (dla odbiorników) oraz czas pomiędzy kolejnymi retransmisjami paczek, ustawiany parametrem `-R`, domyślnie 250.
- NAZWA - nazwa to nazwa nadajnika, ustawiana parametrem `-n`, domyślnie "Nienazwany Nadajnik"

Cześć A (nadajnik)

Nadajnik powinien otrzymywać na standardowe wejście strumień danych z taką prędkością, z jaką odbiorcy są w stanie dane przetwarzać, a następnie wysyłać te dane zapakowane w datagramy UDP na port DATA_PORT na wskazany w linii poleceń adres ukierunkowanego rozgłaszania MCAST_ADDR. Dane powinny być przesyłane w paczkach po PSIZE bajtów, zgodnie z protokołem opisanym poniżej.

Nadajnik powinien przechowywać w kolejce FIFO ostatnich FSIZE bajtów przeczytanych z wejścia tak, żeby mógł ponownie wysłać te paczki, o których retransmisję poprosiły odbiorniki.

Nadajnik cały czas zbiera od odbiorników prośby o retransmisje paczek. Gromadzi je przez czas RTIME, następnie wysyła serię retransmisji (podczas ich wysyłania nadal zbiera prośby, do wysłania w kolejnej serii), następnie znów gromadzi je przez czas RTIME, itd.

Nadajnik nasłuchuje na UDP na porcie CTRL_PORT, przyjmując także pakiety rozgłoszeniowe. Powinien rozpoznawać dwa rodzaje komunikatów:

1. LOOKUP (prośby o identyfikację): na takie natychmiast odpowiada komunikatem REPLY zgodnie ze specyfikacją protokołu poniżej.
2. REXMIT (prośby o retransmisję paczek): na takie nie odpowiada bezpośrednio; raz na jakiś czas ponownie wysyła paczki, według opisu powyżej.

Po wysłaniu całej zawartości standardowego wejścia nadajnik się kończy z kodem wyjścia 0. Jeśli rozmiar odczytanych danych nie jest podzielny przez PSIZE, ostatnia (niekompletna) paczka jest porzucana; nie jest wysyłana.

Uruchomienie nadajnika

Na przykład, żeby wysłać ulubioną MP-trójkę w jakości płyty CD, można użyć takiego polecenia:

```
sox -S "05 Napady.mp3" -r 44100 -b 16 -e signed-integer -c 2 -t raw - | pv -q -L $((44100*4)) | ./nadajnik -a 239.10.11.12 -n "Radio Napady"
```

Pierwsza część polecenia konwertuje plik MP3 na strumień surowych danych (44100 4-bajtowych sampli na każdą sekundę pliku wejściowego), druga część ogranicza prędkość przekazywania danych do nadajnika tak, żeby odbiorniki wyrabiały się z pobieraniem danych.

Żeby wysyłać dane z mikrofonu (w formacie jak powyżej), można użyć takiego polecenia:

```
arecord -t raw -f cd | ./nadajnik -a 239.10.11.12 -n "Radio Accka"
```

Polecenie `arecord` można znaleźć w pakiecie `alsa-utils`.

Część B (odbiornik)

Odbiornik odbiera dane wysyłane przez nadajnik i wyprowadza je na standardowe wyjście.

Odbiornik co ok. 5s wysyła na adres DISCOVER_ADDR na port CTRL_PORT prośbę o identyfikację (komunikat LOOKUP). Na podstawie otrzymanych odpowiedzi (komunikatów REPLY) tworzy listę dostępnych stacji radiowych. Stacja, od której przez 20 sekund odbiornik nie otrzymał komunikatu REPLY, jest usuwana z listy. Jeśli to była stacja aktualnie odtwarzana, rozpoczyna się odtwarzanie innej stacji. Jeśli podano parametr -n, odbiornik

rozpoczyna odtwarzanie stacji o zadanej nazwie, gdy tylko ją wykryje. Jeśli nie podano argumentu -n, odbiornik rozpoczyna odtwarzanie pierwszej wykrytej stacji.

Odbiornik posiada bufor o rozmiarze BSIZE bajtów, przeznaczony do przechowywania danych z maksymalnie ⌊BSIZE/PSIZE⌋ kolejnych paczek.

Rozpoczynając odtwarzanie, odbiornik:

1. Czyści bufor, w szczególności porzucając dane w nim się znajdujące, a jeszcze nie wyprowadzone na standardowe wyjście.
2. Jeśli potrzeba, wypisuje się z poprzedniego adresu grupowego, a zapisuje się na nowy.
3. Po otrzymaniu pierwszej paczki audio, zapisuje z niej wartość pola session_id oraz numer pierwszego odebranego bajtu (nazwijmy go BYTE0; patrz specyfikacja protokołu poniżej), oraz rozpoczyna wysyłanie próśb o retransmisję zgodnie z opisem poniżej.
4. Aż do momentu odebrania bajtu o numerze BYTE0 + ⌊BSIZE*3/4⌋ lub większym, odbiornik nie przekazuje danych na standardowe wyjście. Gdy jednak to nastąpi, przekazuje dane na standardowe wyjście tak szybko, jak tylko standardowe wyjście na to pozwala.

Powyższą procedurę należy zastosować wszędzie tam, gdzie w treści zadania mowa jest o rozpoczynaniu odtwarzania.

Jeśli odbiornik miałby wyprowadzić na standardowe wyjście dane, których jednakże brakuje w buforze, choćby to była tylko jedna paczka, rozpoczyna odtwarzanie od nowa.

Jeśli odbiornik odbierze nową paczkę, o numerze większym niż dotychczas odebrane, umieszcza ją w buforze i w razie potrzeby rezerwuje miejsce na brakujące paczki, których miejsce jest przed nią. Jeśli do wykonania tego potrzeba usunąć stare dane, które nie zostały jeszcze wyprowadzone na standardowe wyjście, należy to zrobić.

Odbiornik wysyła prośby o retransmisję brakujących paczek. Prośbę o retransmisję paczki o numerze n powinien wysłać w momentach $t + k$*RTIME, gdzie t oznacza moment odebrania pierwszej paczki o numerze większym niż n, dla $k = 1, 2, \ldots$ (w nieskończoność, póki dana stacja znajduje się na liście dostępnych stacji). Odbiornik nie wysyła próśb o retransmisję paczek zawierających bajty wcześniejsze niż BYTE0, ani tak starych, że i tak nie będzie na nie miejsca w buforze.

Odbiornik oczekuje połączeń TCP na porcie UI_PORT. Jeśli użytkownik podłączy się tam np. programem Telnet, powinien zobaczyć prosty tekstowy interfejs, w którym za pomocą strzałek góra/dół można zmieniać stacje (bez konieczności wciskania Entera). Oczywiście jeśli jest kilka połączeń, wszystkie powinny wyświetlać to samo, i zmiany w jednym z nich powinny być widoczne w drugim. Podobnie, wyświetlana lista stacji powinna się aktualizować w przypadku wykrycia nowych stacji lub porzucenia już niedostępnych. Powinno to wyglądać dokładnie tak:

```
------------------------------------------------------------------------
  SIK Radio
------------------------------------------------------------------------
    PR1
  > Radio "Disco Pruszkow"
    Radiowa Trojka
------------------------------------------------------------------------
```

Stacje powinny być posortowane alfabetycznie po nazwie. Przy każdorazowej zmianie stanu (aktywnej stacji, listy dostępnych stacji itp.) listę należy ponownie wyświetlić w całości; w ten sposób będzie można w sposób automatyczny przetestować działanie programów.

Uruchomienie odbiornika

```
./odbiornik | play -t raw -c 2 -r 44100 -b 16 -e signed-integer --buffer 32768 -
```

Polecenia `play` należy szukać w pakiecie z programem `sox`.

Protokół przesyłania danych audio

1. Wymiana danych

   Wymiana danych odbywa się po UDP. Komunikacja jest jednostronna -- nadajnik wysyła paczki audio, a odbiornik je odbiera.

2. Format datagramów

   W datagramach przesyłane są dane binarne, zgodne z poniżej zdefiniowanym formatem komunikatów.

3. Porządek bajtów

   W komunikatach wszystkie liczby przesyłane są w sieciowej kolejności bajtów (big-endian).

4. Paczka audio

   ```
   uint64 session_id
   uint64 first_byte_num
   byte[] audio_data
   ```

   Pole `session_id` jest stałe przez cały czas uruchomienia nadajnika. Na początku jego działania inicjowane jest datą wyrażoną w sekundach od epoki.

   Odbiornik zaś zapamiętuje wartość `session_id` z pierwszej paczki, jaką otrzymał po rozpoczęciu odtwarzania. W przypadku odebrania paczki z:

   - mniejszym `session_id`, ignoruje ją,
   - z większym `session_id`, rozpoczyna odtwarzanie od nowa.

Bajty odczytywane przez nadajnik ze standardowego wejścia numerowane są od zera. Nadajnik w polu `first_byte_num` umieszcza numer pierwszego spośród bajtów zawartych w `audio_data`.

Nadajnik wysyła paczki, w których pole `audio_data` ma dokładnie PSIZE bajtów (a `first_byte_num` jest podzielne przez PSIZE).

Protokół kontrolny

1. Wymiana danych odbywa się po UDP.
2. W datagramach przesyłane są dane tekstowe, zgodne z formatem opisanym poniżej.
3. Każdy komunikat to pojedyncza linia tekstu zakończona uniksowym znakiem końca linii. Poza znakiem końca linii dopuszcza się jedynie znaki o numerach od 32 do 127 według kodowania ASCII.
4. Poszczególne pola komunikatów oddzielone są pojedynczymi spacjami. Ostatnie pole (np. nazwa stacji w REPLY) może zawierać spacje.
5. Komunikat LOOKUP wygląda następująco:

   ```
   ZERO_SEVEN_COME_IN
   ```

6. Komunikat REPLY wygląda następująco:

   ```
   BOREWICZ_HERE [MCAST_ADDR] [DATA_PORT] [nazwa stacji]
   ```

   Maksymalna długość nazwy stacji wynosi 64 znaki.

7. Komunikat REXMIT wygląda następująco:

   ```
   LOUDER_PLEASE [lista numerów paczek oddzielonych przecinkami]
   ```

   gdzie numer paczki to wartość jej pola `first_byte_num`, np.

   ```
   LOUDER_PLEASE 512,1024,1536,5632,3584
   ```

Ustalenia dodatkowe

1. Programy powinny umożliwiać komunikację przy użyciu IPv4. Obsługa IPv6 nie jest konieczna.
2. W implementacji programów duże kolejki komunikatów, zdarzeń itp. powinny być alokowane dynamicznie.
3. Programy muszą być odporne na sytuacje błędne, które dają szansę na kontynuowanie działania. Intencja jest taka, że programy powinny móc być uruchomione na stałe bez konieczności ich restartowania, np. w przypadku kłopotów komunikacyjnych, czasowej niedostępności sieci, zwykłych zmian jej konfiguracji itp.

4. Programy powinny być napisane zrozumiale. Tu można znaleźć wartościowe wskazówki w tej kwestii:

   http://www.maultech.com/chrislott/resources/cstyle/LinuxKernelCodingStyle.txt

5. W obydwu aplikacjach opóźnienia w komunikacji z podzbiorem klientów nie mogą wpływać na jakość komunikacji z pozostałymi klientami.

   Patrz też:

   https://stackoverflow.com/questions/4165174/when-does-a-udp-sendto-block

6. Odtwarzany dźwięk musi być płynny, bez częstych lub niewyjaśnionych trzasków.

7. Polecam stosować zasadę niezawodności Postela:

   http://old.rufuspollock.org/2007/02/22/the-robustness-principle

8. Przy przetwarzaniu sieciowych danych binarnych należy używać typów o ustalonej szerokości:

   http://en.cppreference.com/w/c/types/integer

9. W przypadku otrzymania niepoprawnych argumentów linii komend, programy powinny wypisywać stosowny komunikat na standardowe wyjście błędów i zwracać kod 1.

Oddawanie rozwiązania

Można oddać rozwiązanie tylko części A lub tylko części B, albo obu części.

Rozwiązanie ma:

- działać w środowisku Linux;
- być napisane w języku C lub C++ z wykorzystaniem interfejsu gniazd (nie wolno korzystać z libevent ani boost::asio);
- kompilować się za pomocą GCC (polecenie gcc lub g++) – wśród parametrów należy użyć -Wall i -O2, można korzystać ze standardów -std=c++11, -std=c++14, -std=c++17 (w zakresie wspieranym przez kompilator na maszynie students).

Można korzystać z powszechnie znanych bibliotek pomocniczych (np. boost::program_options), o ile są zainstalowane na maszynie students.

Jako rozwiązanie należy dostarczyć pliki źródłowe oraz plik makefile, które należy umieścić na studentsie w katalogu

`/home/students/inf/PUBLIC/SIK/students/ab123456/zadanie2/`

gdzie ab123456 to standardowy login osoby oddającej rozwiązanie, używany na maszynach wydziału, wg schematu: inicjały, nr indeksu. Nie wolno umieszczać tam plików binarnych ani pośrednich powstających podczas kompilacji.

W wyniku wykonania polecenia make dla części A zadania ma powstać plik wykonywalny `sikradio-sender`, a dla części B zadania – plik wykonywalny `sikradio-receiver`.

Ponadto makefile powinien obsługiwać cel 'clean', który po wywołaniu kasuje wszystkie pliki powstałe podczas kompilacji.

Ocena

Za rozwiązanie części A zadania można dostać maksymalnie 2,5 punkta.

Za rozwiązanie części B zadania można dostać maksymalnie 2,5 punkta.

Za rozwiązanie obu części zadania można dostać maksymalnie 6 punktów.

Jeśli student odda obie części zadania, to będą one ocenione osobno. Jeśli obie części współdziałają ze sobą i każda z nich wykazuje działanie zgodne ze specyfikacją, ocena końcowa będzie sumą ocen za poszczególne części pomnożoną przez 1,2.

Ocena każdej z części zadania będzie się składała z trzech składników:

- ocena wzrokowa i manualna działania programu (30%);
- testy automatyczne (50%);
- jakość tekstu źródłowego (20%).

Termin

Termin oddania zadania: piątek 8 czerwca 2018, godzina 19:00 (liczy się czas na serwerze students)

Za spóźnienie do 24 godz. każda rozpoczęta godzina jest warta 0,04 p. Za spóźnienie powyżej 24 godz., ale do 7 dni przed egzaminem – 0,96 p. + 0,02 p. za każdą rozpoczętą godzinę ponad 24 godz.

Rozwiązanie z późniejszą datą można oddać tylko w II terminie.

Punkty za spóźnienia będą odejmowane od końcowego wyniku, jednakże w I terminie za spóźnienia nie odejmuje się więcej niż 2 p.

Rozwiązanie dostarczone w I terminie można poprawić jednokrotnie w II terminie.

W II terminie nie odejmuje się punktów za spóźnienia. Rozwiązania z datą późniejszą niż 7 dni przed egzaminem poprawkowym nie podlegają ocenie.