ZADANIE

Zadanie 1

Otwarto: wtorek, 28 marca 2023, 00:00
Wymagane do: poniedziałek, 24 kwietnia 2023, 23:59

Zadanie polega na napisaniu nadajnika i odbiornika internetowego radia. Stanowi ono istotną część zadania 2, które będzie polegało na rozszerzeniu funkcjonalności nadajnika i odbiornika.

Stałe użyte w treści

- DEST_ADDR - adres odbiornika, ustawiany obowiązkowym parametrem -a nadajnika
- DATA_PORT - port UDP używany do przesyłania danych, ustawiany parametrem -P nadajnika i odbiornika, domyślnie 20000 + (numer_albumu % 10000)
- PSIZE - rozmiar w bajtach pola audio_data paczki, ustawiany parametrem -p nadajnika, domyślnie 512B
- BSIZE - rozmiar w bajtach bufora, ustawiany parametrem -b odbiornika, domyślnie 64kB (65536B)
- NAZWA - nazwa to nazwa nadajnika, ustawiana parametrem -n, domyślnie "Nienazwany Nadajnik"

Cześć A (nadajnik)

Nadajnik służy do wysyłania do odbiornika strumienia danych otrzymanego na standardowe wejście. 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 DEST_ADDR. Dane powinny być przesyłane w paczkach po PSIZE bajtów, zgodnie z protokołem opisanym poniżej. Prędkość wysyłania jest taka jak podawania danych na wejście odbiornika, nadajnik nie ingeruje w nią w żaden sposób.

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 ulubiony utwór w formacie MP3 w jakości płyty CD, można użyć takiego polecenia:

```
sox -S "05 Muzyczka.mp3" -r 44100 -b 16 -e signed-integer -c 2 -t
raw - | pv -q -L \$((44100\*4)) | ./sikradio-sender -a 10.10.11.12 -
n "Radio Muzyczka"
```

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 odbiornik wyrabiał się z pobieraniem danych.

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

```
arecord -t raw -f cd | ./sikradio-sender -a 10.10.11.12 -n "Radio
Podcast"
```

Polecenie arecord można znaleźć w pakiecie alsa-utils.

Część B (odbiornik)

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

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. Podłącza się do nadajnika na port DATA_PORT. Nadajnik jest ustawiany obowiązkowym parametrem -a odbiornika.
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).
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 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 wypisuje na standardowe wyjście informacje dotyczące brakujących paczek. Za każdym razem gdy otrzyma paczkę o numerze *n* sprawdza w buforze i dla każdego *i* < *n* takiego, że nie otrzymał paczki o numerze *i*, mimo, że jest na nią miejsce w buforze, wypisuje:

```
MISSING: BEFORE n EXPECTED i
```

gdzie za *i* i *n* podstawiamy odpowiednie liczby. Komunikaty wypisywane są w kolejności od najmniejszego *i*. Odbiornik nie wypisuje komunikatów dotyczących paczek zawierających bajty wcześniejsze niż BYTE0, ani tak starych, że i tak nie będzie na nie miejsca w buforze.

Uruchomienie odbiornika

```
./sikradio-receiver -a 10.10.11.12 | 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 początku 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).

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: https://www.kernel.org/doc/html/v5.6/process/coding-style.html
5. 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. Polecamy stosować zasadę niezawodności Postela: https://en.wikipedia.org/wiki/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 (5 pkt) lub tylko części B (5 pkt), albo obu części (10 pkt).

Rozwiązanie ma:

- działać w środowisku Linux w LK;
- być napisane w języku C lub C++ z wykorzystaniem interfejsu gniazd (nie wolno korzystać z boost::asio);
- kompilować się za pomocą GCC (polecenie gcc lub g++) – wśród parametrów należy użyć -Wall i -O2, można korzystać ze standardów -std=c2x, -std=c++20 (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 moodle w archiwum ab123456.tgz 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.

Pytania

Pytania do zadania należy kierować na adres: zbyszek@mimuw.edu.pl. Odpowiedzi do najczęściej zadawanych pytań będą pojawiać się na forum: moodle.mimuw.edu.pl/mod/forum/discuss.php?d=8487. Ostatnia odpowiedź pojawi się najpóźniej 3 doby przed terminem oddania zadania.

Oceny

Można otrzymać do 10 p. Ocena każdej z części zadania będzie się składała z trzech składników:

- ocena słuchowa i manualna działania programu (40%);
- testy automatyczne (50%);
- jakość tekstu źródłowego (10%).

Termin

Termin oddania zadania: 24.04, godzina 23:59.

Za każdą rozpoczętą dobę opóźnienia odejmujemy 2 punkty. Można się spóźnić co najwyżej dwie doby.

Dodaj pracę

Status przesłanego zadania

Status przesłanego zadania	Nie przesłano jeszcze zadania
Stan oceniania	Nieocenione
Pozostały czas	Pozostało 26 dni 12 godzin