Zadanie radio

Zadanie składa się z trzech części.

Część A

Zaimplementować program radio-proxy umożliwiający odtwarzanie radia internetowego lub zapisywanie odbieranego dźwięku do pliku. Po uruchomieniu program podłącza się do wskazanego serwera i rozpoczyna ściąganie z niego strumienia audio i ewentualnie metadanych o aktualnie odtwarzanej treści. Otrzymywane dane audio program wypisuje na standardowe wyjście. Otrzymywane metadane program wypisuje na standardowe wyjście diagnostyczne. Zakończenie programu następuje po wysłaniu mu sygnału za pomocą kombinacji klawiszy `Ctrl-C`. Program powinien wtedy zamknąć połączenie z serwerem.

Wywołanie programu:

`./radio-proxy parameters`

Parametry:

`-h host` – nazwa serwera udostępniającego strumień audio, obowiązkowy;

`-r resource` – nazwa zasobu na serwerze, zwykle sam ukośnik, obowiązkowy;

`-p port` – numer portu, na którym serwera udostępniającego strumień audio, liczba dziesiętna, obowiązkowy;

`-m yes|no` – wartość `yes`, jeśli program powinien żądać od serwera przesyłania metadanych, domyślnie wartość `no` oznaczająca, że program nie powinien żądać od serwera metadanych, opcjonalny;

`-t timeout` – czas w sekundach, po którego upłynięciu, jeśli serwer nic nie przysłał, pośrednik uznaje, że serwer przestał działać, domyślna wartość 5 sekund, opcjonalny.

Przykładowe użycia:

```
./radio-proxy -h waw02-03.ic.smcdn.pl -r /t050-1.mp3 -p 8000 >waw.mp3
./radio-proxy -h ant-waw-01.cdn.eurozet.pl -p 8602 -r / -m yes >ant.mp3 2>meta.txt
./radio-proxy -p 8000 -h waw02-03.ic.smcdn.pl -r /t043-1.mp3 -m no | mplayer -cache 2048 -
```

Część B

Dodać do programu `radio-proxy` funkcjonalność pośrednika, który, zamiast wypisywać dane na standardowe wyjście danych i standardowe wyjście diagnostyczne, rozsyła je za pomocą UDP. Pośrednik serwuje tylko jedno radio, ale powinien umożliwiać obsługę wielu klientów równocześnie.

Dodatkowe parametry:

`-P port` – numer portu, na którym program ma nasłuchiwać komunikatów UDP od klientów i z którego ma wysyłać im, też za pomocą UDP, dane, liczba dziesiętna, obowiązkowy;

`-B multi` – adres rozsyłania grupowego, na którym program ma nasłuchiwać, opcjonalny;

`-T timeout`– czas w sekundach, po którego upłynięciu, jeśli klient nie odzywa się, pośrednik przestaje mu wysyłać cokolwiek, domyślna wartość 5 sekund, opcjonalny.

Program przed zakończeniem powinien oprócz zamknięcia połączenia z serwerem zwolnić też port, na którym nasłuchuje.

Przykładowe użycia:

```
./radio-proxy -h waw02-03.ic.smcdn.pl -r /t050-1.mp3 -p 8000 -P 10000 -t 10
./radio-proxy -h ant-waw-01.cdn.eurozet.pl -p 8602 -r / -B 239.10.11.12 -P 54321
```

Część C

Zaimplementować program `radio-client`, który używa rozgłaszania UDP w celu rozpoznania, jakie programy `radio-proxy` są uruchomione, a po wykryciu pośrednika odbiera od niego dane. Odbierany strumień audio klient wysyła na standardowe wyjście. Sposób wyświetlania metadanych jest opisany poniżej.

Wywołanie programu:

```
./radio-client parameters
```

Parametry:

`-H host` – adres, na którym nasłuchuje `radio-proxy`, lub adres rozgłoszeniowy, lub adres rozsyłania grupowego, obowiązkowy;

`-P port` – numer portu UDP, na którym nasłuchuje `radio-proxy`, liczba dziesiętna, obowiązkowy;

`-p port` – numer portu TCP, na który można podłączyć się programem `telnet` w celu sterowania klientem, obowiązkowy;

`-T timeout`– czas w sekundach, po którego upłynięciu, jeśli pośrednik przestał przysyłać dane, klient uznaje, że pośrednik przestał działać, domyślna wartość 5 sekund, opcjonalny.

Przykładowe użycia:

```
./radio-client -H 127.0.0.1 -P 10000 -p 20000 >abc.mp3
./radio-client -H 255.255.255.255 -P 10000 -p 20000 -t 10 >/dev/null
./radio-client -H 239.10.11.12 -P 10000 -p 20000 | mplayer -cache 2048
```

Po uruchomieniu klient czeka na podłączenie się do niego za pomocą TCP programem `telnet`, a po podłączeniu udostępnia menu, za którego pomocą zarządza się jego pracą. Klient komunikuje się w danym momencie tylko z jednym pośrednikiem, ale powinien umożliwiać zmianę pośrednika. Klient obsługuje tylko jedno połączenie sterujące TCP w danym momencie, ale to połączenie może być wielokrotnie zamykane i ponownie otwierane.

Protokół między pośrednikiem a serwerem radia internetowego

Program `radio-proxy` ściąga strumień audio w formacie MP3 za pomocą protokołu Shoutcast znanego też pod nazwą ICY. Komunikacja wygląda jak w HTTP. Klient (w tym przypadku `radio-proxy`) łączy się z serwerem radia za pomocą TCP i wysyła zapytanie GET. Odpowiedź serwera składa się (jak to w HTTP) z: linii statusu, ciągu tekstowych pól nagłówka oraz pustej linii. Następnie serwer rozpoczyna wysyłanie binarnego strumienia audio ewentualnie zmultipleksowanego z tekstowymi metadanymi. Tu jest kilka dokumentów wyjaśniających, jak wygląda protokół:

- http://www.garymcgath.com/streamingprotocols.html
- https://web.archive.org/web/20160729115551/http://www.indexcom.com/streaming/player/SHOUTcast.html
- https://web.archive.org/web/20190317190956/https://www.smackfu.com/stuff/programming/shoutcast.html
- https://people.kth.se/~johanmon/dse/casty.pdf

Warto też zwrócić uwagę, że serwer zamiast `ICY 200 OK` może odpowiedzieć `HTTP/1.0 200 OK` lub `HTTP/1.1 200 OK`, co należy uznać za poprawne zachowanie.

Protokół między klientem a pośrednikiem

Komunikacja między pośrednikiem a klientem odbywa się za pomocą UDP. Komunikat składa się z nagłówka, który ma cztery oktety i pola z danymi o zmiennej długości. Nagłówek zawiera kolejno dwa pola: `type` i `length`, każde zajmuje dwa oktety. Pole `type` koduje typ komunikatu. Pole `length` koduje długość pola z danymi. Liczby są zapisywane w sieciowym porządku bajtów.

Typy komunikatów, wartości pola `type`:

`1` – `DISCOVER` – klient rozsyła ten komunikat w celu wykrycia działających pośredników, nie ma pola z danymi, pole `length` ma wartość 0;

`2` – `IAM` – odpowiedź pośrednika, dane zawierają informacje o serwowanym radiu

`3` – `KEEPALIVE` – klient wysyła do pośrednika informację, że nadal działa, nie ma pola z danymi, pole `length` ma wartość 0;

`4` – `AUDIO` – pośrednik wysyła porcję danych audio;

`6` – `METADATA` - pośrednik wysyła metadane.

Protokół sterujący klientem

W połączeniu sterującym klient pełni rolę serwera, a klientem jest program `telnet`. W programie `telnet` strzałkami w górę i w dół poruszamy się po menu, a enterem wybieramy opcję.

Po zaakceptowaniu połączenia TCP klient wypisuje menu początkowe:

```
Szukaj pośrednika
Koniec
```

Wybranie opcji `Koniec` kończy działanie klienta, co oczywiście powoduje zamknięcie połączenia z `telnetem`. Program powinien też wtedy zwolnić wszystkie używane porty.

Wybranie opcji `Szukaj pośrednika` powoduje, że klient rozsyła na adres podany w parametrze `-H` i port podany w parametrze `-P` komunikat `DISCOVER`.

Jeśli klient odbierze jakieś komunikaty `IAM` od pośredników, dodaje do menu opcje umożliwiające wybranie jednego z nich, przykładowo:

```
Szukaj pośrednika
Pośrednik Radioaktywne
Pośrednik Dobreradio
Koniec
```

Po wybraniu pośrednika klient wysyła już bezpośrednio do niego komunikat `DISCOVER` i rozpoczyna odbieranie od niego komunikatów `AUDIO` i `META` oraz wysyła komunikaty `KEEPALIVE` w regularnych odstępach czasu co trzy i pół sekundy.

W menu klient zaznacza wybranego pośrednika gwiazdką, a pod menu wyświetla linię statusu, w której pojawiają się otrzymywane metadane, przykładowo:

```
Szukaj pośrednika
Pośrednik Radioaktywne
Pośrednik Dobreradio *
Koniec
Maanam - Nic dwa razy
```

W każdym momencie można zmienić pośrednika, rozpocząć poszukiwanie nowych pośredników lub zakończyć działanie klienta. Jeśli klient uzna, że pośrednik przestał przysyłać dane, usuwa go z menu.

Informacje dodatkowe

Programy powinny być możliwie jak najbardziej odporne na wszelakie błędy, jakie się mogą pojawić. Powinny tolerować błędy, które nie uniemożliwiają kontynuowania pracy. W kwestiach nieopisanych w treści zadania należy kierować się przyjętymi w Linuksie konwencjami i zdrowym rozsądkiem.

Programy przy poprawnym zakończeniu powinny się zwracać kod 0. Błędne uruchomienie programu lub wystąpienie błędu, którego nie da się tolerować, powinno powodować zakończenie programu kodem 1. Jeśli program `radio-proxy` nie obsługuje funkcji pośrednika, powinien przy próbie jego wywołania jako pośrednika zakończyć się kodem 2.

Wymagamy jedynie obsługi IP4.