Internet radio multicast

The task involves extending the functionality of the transmitter and receiver of an internet radio from task 1.

Variables used in the text:

- MCAST_ADDR: Multicast address, set by the mandatory parameter -a of the transmitter.
- DISCOVER_ADDR: Address used by the receiver to discover active transmitters, set by the -d parameter of the receiver, defaulting to 255.255.255.255.
- DATA_PORT: UDP port used for data transmission, set by the -P parameter of the transmitter, defaulting to 20000 + (student_number % 10000).
- CTRL_PORT: UDP port used for transmission of control packets, set by the -C parameter of the transmitter and receiver, defaulting to 30000 + (student_number % 10000).
- UI_PORT: TCP port on which a simple text interface for switching between stations is provided, defaulting to 10000 + (student_number % 10000); set by the -U parameter of the receiver.
- PSIZE: Size in bytes of the audio_data field in each packet, set by the -p parameter of the transmitter, defaulting to 512B.
- BSIZE: Size in bytes of the buffer, set by the -b parameter of the receiver, defaulting to 64kB (65536B).
- FSIZE: Size in bytes of the transmitter's FIFO queue, set by the -f parameter of the transmitter, defaulting to 128kB.
- RTIME: Time (in milliseconds) between sending successive reports of missing packets (for receivers) and the time between successive retransmissions of packets, set by the -R parameter, defaulting to 250.
- NAME: Name of the transmitter, set by the -n parameter, defaulting to "Unnamed Transmitter".

Part A (Transmitter):

The transmitter is used to send a stream of data received from standard input to the receivers. The transmitter should receive a stream of data from standard input at the rate at which receivers are capable of processing data, then send this data packed into UDP datagrams to port DATA_PORT at the specified multicast address MCAST_ADDR provided in the command line. Data should be transmitted in packets of PSIZE bytes according to the protocol described below. The transmission speed is the same as providing data to the receiver input; the transmitter does not interfere with it in any way.

The transmitter should store the last FSIZE bytes read from the input in a FIFO queue so that it can resend these packets upon request from the receivers.

The transmitter continuously collects requests for packet retransmission from the receivers. It collects them for a time RTIME, then sends a series of retransmissions (while still collecting requests to send in the next series), then collects them again for RTIME, and so on.

The transmitter listens on UDP on the CTRL_PORT, also accepting broadcast packets. It should recognize two types of messages:

- LOOKUP (requests for identification): it should respond immediately with a REPLY message according to the protocol specification below.
- REXMIT (requests for packet retransmission): it does not respond directly to these; it resends packets periodically, according to the description above.

After sending all the contents of standard input, the transmitter terminates with exit code 0. If the size of the read data is not divisible by PSIZE, the last (incomplete) packet is discarded and not sent.

Running the transmitter:

For example, to send a favorite MP3 track with CD quality, you can use the following command:

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

The first part of the command converts the MP3 file to a stream of raw data (44100 4-byte samples per second of the input file), and the second part limits the data transfer rate to the transmitter so that the receivers can keep up with data retrieval.

To send data from a microphone (in the same format as above), you can use the following command:

```
arecord -t raw -f cd | ./sikradio-sender -a 239.10.11.12 -n "Radio Podcast"
```

The arecord command can be found in the alsa-utils package.

Part B (Receiver):

The receiver receives data sent by the transmitter and outputs it to standard output.

Every approximately 5 seconds, the receiver sends a LOOKUP message to the DISCOVER_ADDR address on the CTRL_PORT port. Based on the received responses (REPLY messages), it creates a list of available radio stations. A station from which the receiver has not received a REPLY message for 20 seconds is removed from the list. If this was the currently playing station, playback of another station starts. If the -n parameter is provided, the receiver starts playing the station with the specified name as soon as it is detected. If no -n argument is provided, the receiver starts playing the first detected station.

The receiver has a buffer of size BSIZE bytes, intended to store data from a maximum of ⌊BSIZE/PSIZE⌋ successive packets.

When starting playback, the receiver:

- Clears the buffer, discarding any data in it that has not yet been output to standard output.
- If necessary, unsubscribes from the previous multicast address and subscribes to the new one.
- Upon receiving the first audio packet, it saves the value of the session_id field and the number of the first received byte (let's call it BYTE0; see the protocol specification below), and begins sending requests for retransmission as described below.

Until receiving the byte with number BYTE0 + ⌊BSIZE*3/4⌋ or greater, the receiver does not pass data to standard output. However, once this happens, it passes data to standard output as quickly as standard output allows.

This procedure should be applied wherever the task text mentions starting playback.

If the receiver is supposed to output data to standard output, but some of it is missing from the buffer, even if it's just one packet, it restarts playback. Note: This requirement was not present in task 1.

If the receiver receives a new packet with a number greater than those previously received, it places it in the buffer and, if necessary, reserves space for missing packets that should precede it. If removing old data that has not yet been output to standard output is required to do this, it should be done.

The receiver sends requests for retransmission of missing packets. A request for retransmission of packet number n should be sent at times t + k*RTIME, where t denotes the moment of receiving the first packet with a number greater than n, for k = 1, 2, ... (continuously, as long as the station is on the list of available stations). The receiver does not send requests for retransmission of packets containing bytes earlier than BYTE0, or so old that there will be no space for them in the buffer.

The receiver listens for TCP connections on the UI_PORT. If a user connects there, for example, using telnet, they should see a simple text interface where they can switch between stations using the arrow keys up/down (without having to press Enter). Of course, if there are multiple connections, they should all display the same thing, and changes in one should be visible in the other. Similarly, the displayed list of stations should update if new stations are detected or already unavailable stations are abandoned. It should look exactly like this:

```
------------------------------------------------------------------------

 SIK Radio

------------------------------------------------------------------------

PR1

Radio "357"

 > Radio "Disco Pruszkow"

------------------------------------------------------------------------
```

Stations should be sorted alphabetically by name. Each time the state changes (active station, list of available stations, etc.), the list should be displayed again in full; this way, the operation of the programs can be automatically tested.

Running the receiver:

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

The play commands should be searched for in the package with the sox program.

Audio Data Transmission Protocol:

- Data Exchange: Data exchange occurs via UDP. Communication is one-way—transmitter sends audio packets, and the receiver receives them.
- Datagram Format: Binary data is transmitted in datagrams, consistent with the below-defined message format.
- Byte Order: All numbers in the messages are sent in network byte order (big-endian).

Audio Packet:

- uint64 session_id
- uint64 first_byte_num
- byte[] audio_data

The session_id field remains constant throughout the transmitter's runtime. At the beginning of its operation, it is initialized with the date expressed in seconds since the start of the epoch.

The receiver remembers the session_id value from the first packet it receives after starting playback. Upon receiving a packet with:

- a smaller session_id, it ignores it,
- a greater session_id, it starts playback again.

The bytes read by the transmitter from standard input are numbered starting from zero. The transmitter places the number of the first byte contained in audio_data in the first_byte_num field.

The transmitter sends packets in which the audio_data field has exactly PSIZE bytes (and first_byte_num is divisible by PSIZE).

Control Protocol:

- Data Exchange: Data exchange occurs via UDP.
- Datagram Format: Textual data is transmitted in datagrams, consistent with the below-defined message format.
- Each message is a single line of text terminated by a Unix end-of-line character. Apart from the end-of-line character, only characters with ASCII codes from 32 to 127 are allowed.
- Fields in messages are separated by single spaces. The last field (e.g., station name in REPLY) may contain spaces.
- LOOKUP message looks like this:

ZERO_SEVEN_COME_IN

- REPLY message looks like this:

BOREWICZ_HERE [MCAST_ADDR] [DATA_PORT] [station name]

The maximum length of the station name is 64 characters.

- REXMIT message looks like this:

LOUDER_PLEASE [list of packet numbers separated by commas]

where the packet number is the value of its first_byte_num field, e.g.,

LOUDER_PLEASE 512,1024,1536,5632,3584