Internet radio

The task consists of writing a transmitter and receiver for an internet radio. It is an essential part of task 2, which will involve expanding the functionality of the transmitter and receiver.

Constants used in the text:

- DEST_ADDR: Receiver address, set by the mandatory parameter -a of the transmitter.
- DATA_PORT: UDP port used for data transmission, set by the parameter -P of the transmitter and receiver, defaulting to 20000 + (student_number % 10000).
- PSIZE: Size in bytes of the audio_data field in each packet, set by the parameter -p of the transmitter, defaulting to 512B.
- BSIZE: Size in bytes of the buffer, set by the parameter -b of the receiver, defaulting to 64kB (65536B).
- NAZWA: Name of the transmitter, set by the parameter -n, defaulting to "Unnamed Transmitter".

Part A (Transmitter):

The transmitter is used to send a stream of data to the receiver received from standard input. 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 DEST_ADDR address 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.

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 song in MP3 format 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 10.10.11.12 -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 receiver 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 10.10.11.12 -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.

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.
- Connects to the transmitter on port DATA_PORT. The transmitter is set by the mandatory parameter -a of the receiver.
- 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).
- 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 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 outputs diagnostic information (stderr) regarding missing packets to standard output. Each time it receives a packet with number n, it checks the buffer and, for each i < n such that it has not received a packet with number i, even though there is space for it in the buffer, it outputs:

MISSING: BEFORE n EXPECTED i

where appropriate numbers are substituted for i and n. The messages are output in ascending order of i. The receiver does not output messages concerning packets containing bytes earlier than BYTE0 or so old that there will be no space for them in the buffer.

Running the receiver:

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