Home Radio

Assignment 1

Network Multimedia Protocols and Services

September 2011

Coding Assignments (1/2)

- 2 Assignments (schedule on the web to be updated)
  - Building on top of one another
  - Interpret standards text and implement packet interpretation/ generation
  - Support parameterization and some visualization (no GUIs!)
  - Interact with real world applications (and their bugs!)
- C/C++ and to some extent Java code supported by us
  - You can also use other languages: on your own and at your own risk
  - Do the work on the Unix machines in the department (must at least work there)
  - Details to follow

Coding Assignments (2/2)

- Small groups: 2 – 3
  - Send one email per group in exactly the following format (one line per group member)
    `"Last name:First name:Student ID:email address"`
- Completion: on **22 December 2011**
  - Send email with tgz or zip archive of source, build environment
  - Present all results interactively in 20-30m per group
  - Special requirements? Just talk to us…
  - Demo can be done by a person of the team [not everyone is needed]

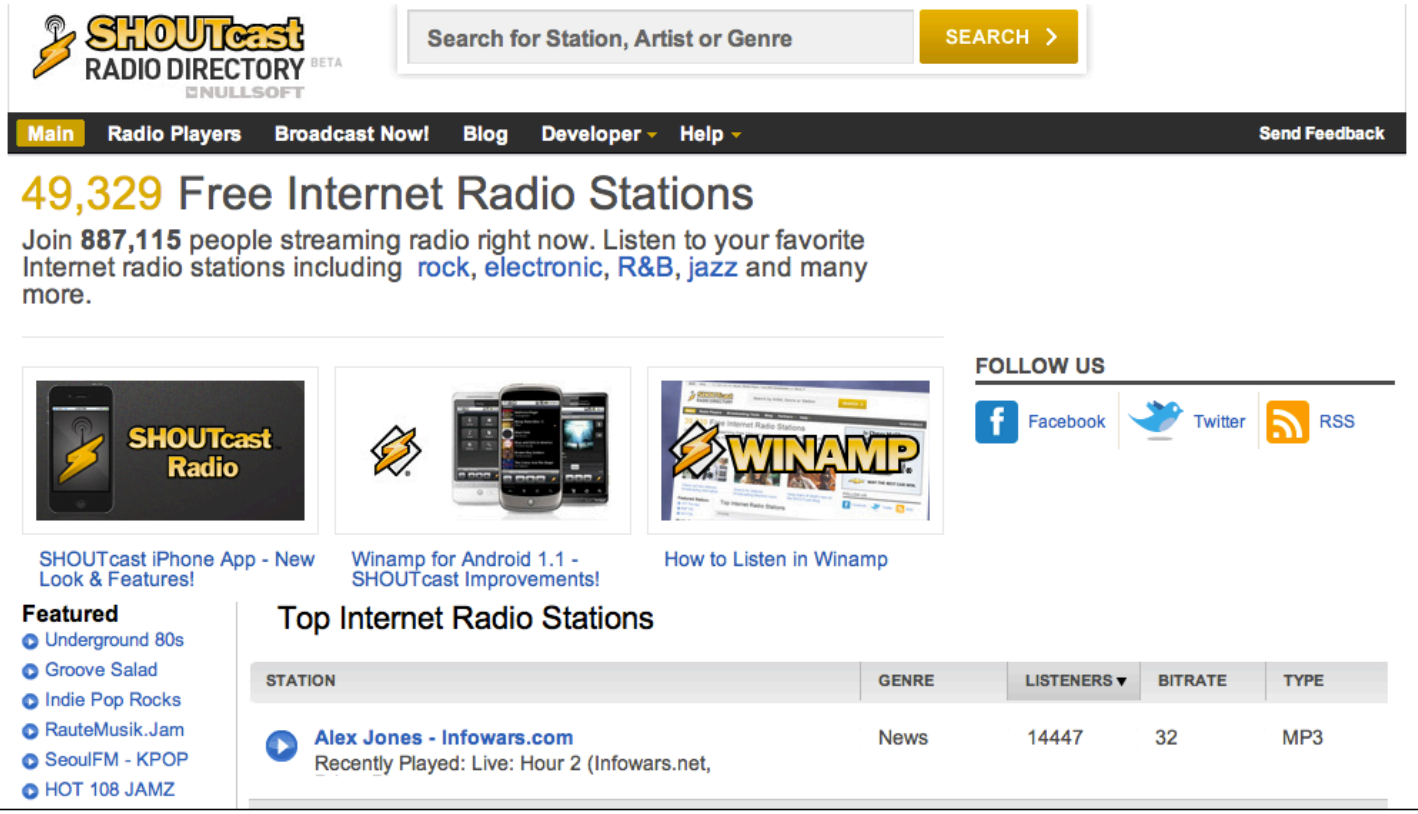
Evaluation

- 2 Assignments (30 points)
  - 15 to pass
- Each assignment is 15 points.
  - Extra points now increase your scores.
- Deadline Midnight of 22th December (after demo day)
- Demo Day: 22th December 1000h to 1700h
  - 20-30 mins for each group
  - Submit code and documentation after the demo

Exercise Classes

- 14.09: This lecture [Assignment 1]
- 21.09: No Lecture*: use this time to arrange meeting or work collaboratively
- 28.09: Q&A Session: Opportunity to ask questions, get clarifications and even hints to solve the problems the teams face. You can help each other as well, but don't exchange code… you can however show others how it is done. If you learn something from each other "acknowledge it in the final report"
- The above cycle of "No Lecture" and "Q&A Session" repeat alternatively
- Week 42 is intentionally left free so that people get time to work together on the assignment.

Internet Radio

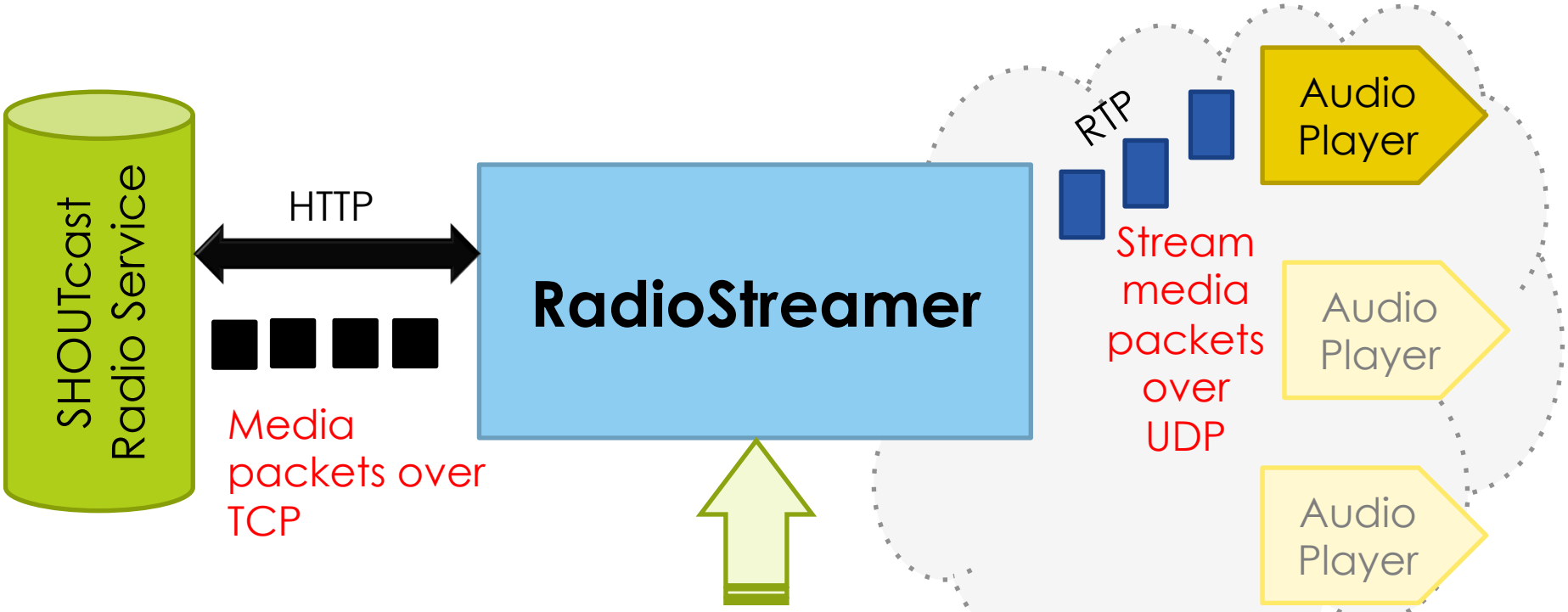


Motivation

- Many new music services
- Many new devices
- But, you don't always want to take your device with you.

- Stream the same music to another device…

Scenario/Use-case



SHOUTcast Radio Service

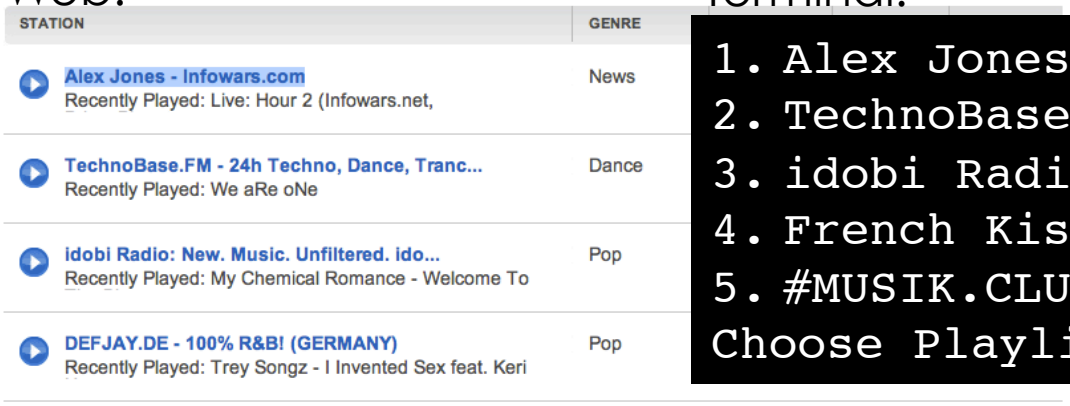
- Goto http://www.shoutcast.com/
- choose a channel, copy the link.
  - Example: http://yp.shoutcast.com/sbin/tunein-station.pls?id=1966819
- Playlist contains multiple links to the station (use `curl` to see a playlist file)
  - All links play the same audio track.
  - Playlist provides a failsafe when a server is inaccessible
- If you want to change the music track, pick another playlist

Note: Shoutcast has AAC playlists as well. Filter them out--**choose MP3**

Parsing and displaying stations

- Fetch and parse the index.html at http://www.shoutcast.com
  - Don't use http://shoutcast.com because it re-directs and unless you implement **`HTTP 3xx`** code
- What are you looking for? --- **Playlist files**
  - Find them and display them to the users. So that they can choose
  - Application UI (more on this later)

Web:



Terminal:

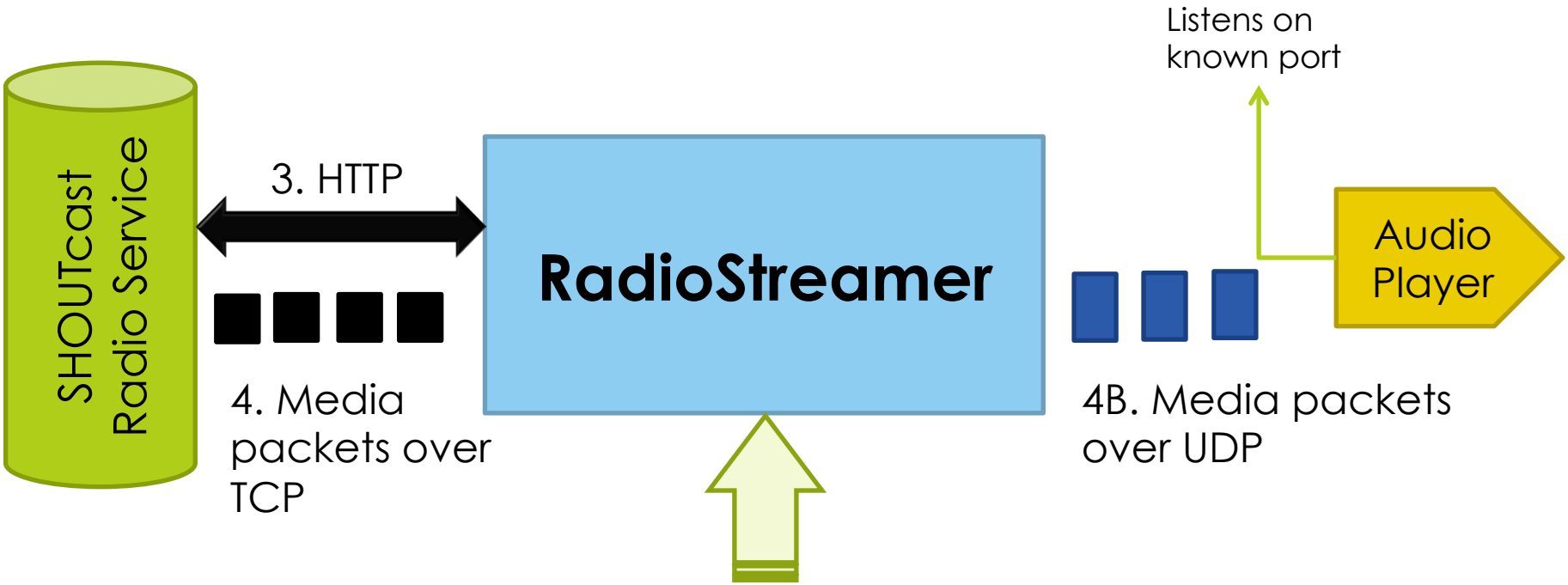
```
1. Alex Jones — Infowa.. [id=1026951]
2. TechnoBase.FM — 24h.. [id=1377200]
3. idobi Radio: New Mu.. [id=847066]
4. French Kiss FM [id=2057543]
5. #MUSIK.CLUB - WWW.R..[id=1275319]
Choose Playlist:_
```

What does playlist.pls contain?

[playlist]
numberofentries=50
File1=**http://87.230.103.85:80**
Title1=(#1 - 221/750) top 100 station - Germanys No.1 Web Hit Station
Length1=-1
File2=**http://78.159.104.130:80**
Title2=(#2 - 230/750) top 100 station - Germanys No.1 Web Hit Station
Length1=-1
(…)
File50=**http://188.72.209.68:80**
Title50=(#50 - 699/700) top 100 station - Germanys No.1 Web Hit Station
Length1=-1
Version=2

NOTE: All <File*N*=URL> stream the same content, so try another if a URL is unavailable/inaccessible

Simple Scenario



Objective: buffer MP3, transcode MP3 to PCM and stream PCM

RadioStreamer

- **Establish** connection over HTTP with SHOUTcast and **Fetch** main page (http://www.shoutcast.com)
- Parse main page to get links to **MP3 playlists** and their titles
- Ask user to select a channel based on title (Remote)
- **Fetch** playlist file, and **establish** connection to one of the URLs [*they may fail to connect or no MP3 packets may appear, so try other URLs on the list*]
- When establishment succeeds, RadioStreamer will start to receive MP3 packets over TCP (**Progressive Download**)
- Convert MP3 packets to PCM (more on this later)
- Stream PCM packets using RTP to clients (more on this later)

HTTP Streaming

- HTTP GET messages to request resource.
- Use tools like CURL, WGET for prototyping.
- Use wireshark to collect traces to understand how HTTP streaming works.
  - However build the REQUEST and RESPONSE by yourselves. (no libraries allowed)
- VLC support HTTP streaming.
  - There are other tools as well, so you can use any to test.
  - i.e goto File->Open Network, in the text box type the HTTP address
    - E.x. http://yp.shoutcast.com/sbin/tunein-station.pls?id=1966819 will play the playlist [i.e., multiple servers but same file]
    - E.x. http://80.237.155.73:80 will play the content only from that server

GET request

GET from a station in http://yp.shoutcast.com/sbin/tunein-station.pls?id=1377200)

Example1:

```
GET / HTTP/1.1\r\n
User-Agent: curl/7.21.7 (x86_64-apple-darwin11.0.0) libcurl/7.21.7
OpenSSL/1.0.0d zlib/1.2.5 libidn/1.22\r\n
Host: 85.17.26.85\r\n
Accept: */*\r\n
\r\n
```

Example2:

```
GET / HTTP/1.1\r\n
Host: 80.237.211.43\r\n
User-Agent: vlc/1.1.11\r\n
Range: bytes=0-\r\n
Icy-MetaData: 1\r\n
\r\n
```

Note: `\r\n` are required to build the http request correctly

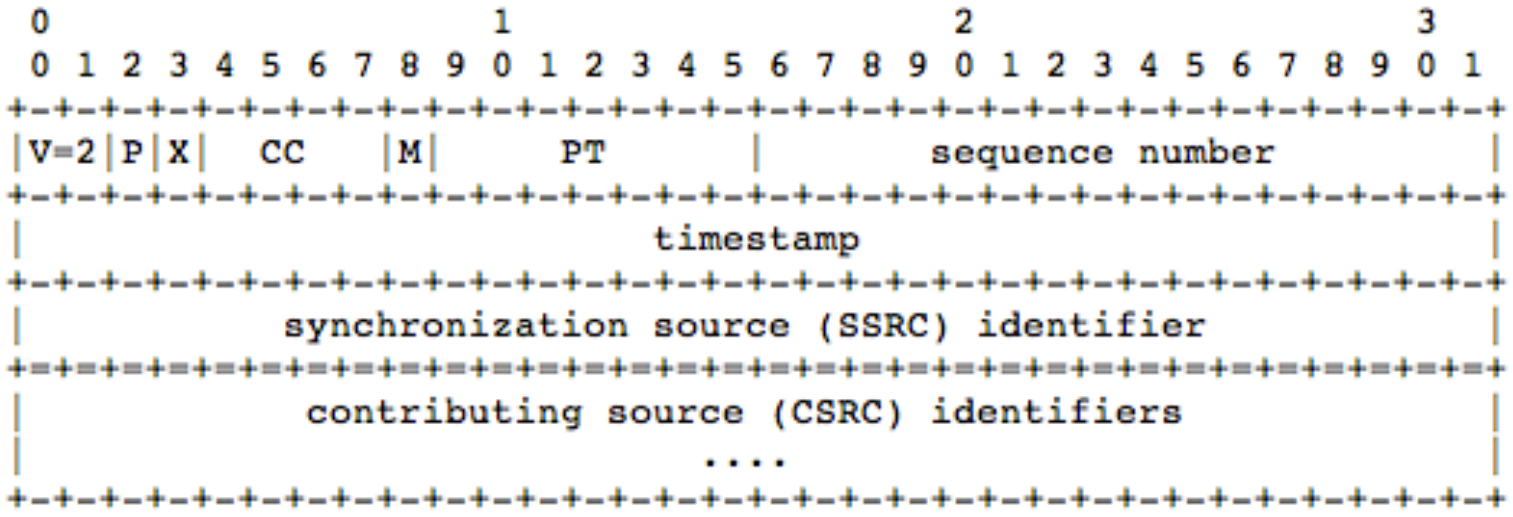
Real-time Transcoding

- SHOUTCast is streaming MP3 data over TCP (it is live content)
- Read MP3 frames from the temporary file/cache and transcode it.
- How to read an MP3 file:
  - http://en.wikipedia.org/wiki/MP3
  - http://www.mpgedit.org/mpgedit/mpeg_format/mpeghdr.htm

Convert MP3 to PCM

- Use FFmpeg or similar library (there are many open source libraries and tons of examples)
  - Here is one example [read tutorials 1 onwards to learn about using the SDL]: http://dranger.com/ffmpeg/tutorial03_print.html
- PCM is Pulse code modulated, you can use PCM-A (used in EU) or PCM-U (used in US). Also known as G.711 (wiki)
  - 8000 Hz means a frame is created every 125us. (1/8000=0.000125s)
  - NOTE: FFmpeg library creates A-law frames with 16-bits, and u-law with 8-bits. Check with the audio player which one is supported.
  - Choose either, tools used in **Assignment 2** will probably affect this choice.

RTP Packet Introduction



…Payload…

Stream RTP packets

- Use the RTP headers as defined in http://tools.ietf.org/html/rfc3550#section-5.1
  - RTP and RTCP will be taught in the following classes.
- PCM requires a specific payload type, read more at:
  - http://www.iana.org/assignments/rtp-parameters
  - http://tools.ietf.org/html/rfc3551#section-4.5.14
- Since you are the sender you may have to implement RTCP sender reports:
  - They are defined here http://tools.ietf.org/html/rfc3550#section-6.1
- Stream RTP packet to client(s)
  - [for simplicity these ports should be configured as command line parameters]
  - You MAY need to fragment large frames if needed (media payload>MTU)

Audio Player

- Connection establishment is a non-goal for assignment 1 and VLC/Gstreamer or any player that supports RTP and PCM (A or U-law) can be used
- Start VLC player to listen on a known port
  - Example: **vlc rtp://localhost:7000**, use any port that is available
  - This configures VLC to listen on port 7000.
  - You can start multiple instances of VLC and listen on different ports
- If the RadioStreamer is not streaming any packets the VLC will timeout because of no packets.
  - If the RadioStreamer is running but not streaming any packets then send dummy RTP packets to keep the connection alive
  - i.e., correct RTP header but 0 payload

RadioStreamer UI

- At startup show a list of channels (show 5 or more)
  - User chooses channel
- After this the user can
  - change change channels
  - Pause/Continue
  - Terminate Session
- Ctrl+C: also terminates session
  - Prints debug stats
  - Uptime
  - MP3 frames received
  - PCM frames created
  - Duration listened, paused
  - IP Address of clients stramed to

Example UI

```
1. Alex Jones – Infowa.. [id=1026951]
2. TechnoBase.FM – 24h.. [id=1377200]
3. idobi Radio: New Mu.. [id=847066]
4. French Kiss FM [id=2057543]
5. #MUSIK.CLUB - WWW.R..[id=1275319]
Choose channel: 2
Press [N]ext, [P]ause, [T]erminate: P
Press [N]ext, [C]ontinue, [T]erminate: _
```

UI actions

- **Next:** chooses the next channel, if it as the end of the list it chooses from the first. [like a carousel]
- **NEXT**: redoes the HTTP establishment and new MP3 stream arrive. If the bit rate of the MP3 changes then make sure that your MP3 to PCM decoder is configured correctly
  - Or only list MP3 playlists with the same bit rates
  - PCM rate should be around 64kbps
- **Pause**: terminates the HTTP connection and stops streaming PCM +RTP data to the client. However, RadioStreamer must still send dummy packets to VLC or else VLC will close the connection
- Pause should turn to **Continue**, when the stream is paused. Continue re-establishes the connection with the SHOUTcast server (the old URL may not work so use a new URL from the playlist file)
  - RTP packets have to resume as before. Audio players may get confused if the sequence numbers jump by a large amount

NOTE: You can add your own command line params, these are just for your guidance

Command Line for RadioStreamer

./RadioStreamer –u <http url:port> -p <TCP Port> -c <client IP addresses: port number>

-u in addition to the HTTP URL, the client MAY need a port address, if not provided it should use 80.

-p is the port number the RadioStreamer will use to make the HTTP requests [can also be random]

-c can have a list of client IP and port addresses. These values are space separated

Use **getopt()** to parse command-line

Remote

- BONUS Points (2)
  - Implement the UI as an alternate program
  - Use TCP or UDP, that is up to you to decide, there are trade-offs.
  - The RadioStreamer should implement a server which allows a client to connect and send and receive commands
  - Remote sends CHANNEL, RadioStreamer sends a list of channels
  - Remote Sends CHOICE X, where X is a [Channel number, Pause, Continue, Next, Terminate], RadioStreamer can send OK, ERR to say if it accepts the command or there is an error in choice.
  - **./remote –r <RadioStreamer IP address:port>**

./radiostreamer –u <http url:port> -p <TCP Port> -c <client IP addresses: port number> -r ServerPortForRemote

Deliverables

- it is a command line utility
- Print meaningful debug messages on stdout.
  - Usually, source and destination IP addr: port number helps in debugging
  - Log the TCP media and RTP media packets in separate logs
  - RTP log: RTP header details for sent RTP packets
- Document the architecture, motivate your design decision.

svn

- For students interested in creating a svn repository in the university unix machines, please refer the below link.
  - http://goblin.tkk.fi/c++/tutorials/svn.html