



Universidade do Minho
Escola de Engenharia

MESTRADO INTEGRADO EM ENGENHARIA DE TELECOMUNICAÇÕES E INFORMÁTICA

TECNOLOGIAS E SERVIÇOS MULTIMÉDIA - BRUNO DIAS

TRABALHO PRÁTICO 3

DISTRIBUIÇÃO DE ÁUDIO *PCM*

Constituição do grupo:

André Lopes - A75363

Augusto Mota - A76563

Hugo Machado - A80362

Guimarães, 25 de Julho de 2020

Índice

1	Introdução	3
2	Fundamentos teóricos	4
2.1	Áudio <i>PCM</i>	4
2.1.1	Formato	5
2.1.2	Codificação	7
2.2	Protocolo de <i>Streaming</i>	8
3	Arquitectura do sistema	9
4	Implementação	10
4.1	Estratégia de codificação	10
4.2	Estratégia de Distribuição de áudio	11
4.3	Protocolo de Controlo	12
4.4	Protocolo de <i>Streaming</i>	12
4.5	Adicional	13
4.5.1	Gestão de clientes	13
4.5.2	Gestão e organização automática das musicas	13
4.5.3	Dependências	13
5	Funcionamento	14
6	Conclusão	20
7	Referências	21

Lista de Figuras

1	Formato canónico.	5
2	Conteúdo do <i>header</i> do ficheiro áudio de exemplo.	7
3	Arquitetura do sistema	9
4	Formato do cabeçalho com duas amostras.	10
5	Representação de uma retransmissão, num diagrama temporal.	11
6	Início de ambos lados(Servidor e Cliente).	14
7	Listagem de géneros.	14
8	Listagem de Artistas.	15
9	Listagem de Álbuns.	15
10	Listagem de Músicas.	16
11	Tocar uma música.	16
12	Listagem da <i>Playlist</i>	17
13	Passar a frente uma música.	17
14	Parar/finalizar a <i>Playlist</i>	18
15	Actualizar lista de música.	18
16	Exemplo de falha de ligação.	19

Lista de Tabelas

1	<i>RIFF header</i>	6
2	<i>FMT header</i>	6
3	<i>DATA header</i>	6
4	Exemplo de codificação	8

1. Introdução

No âmbito da Unidade Curricular de Tecnologias e Serviços Multimédia do curso Mestrado Integrado em Engenharia de Telecomunicações e Informática, foi nos proposto implementar um sistema de distribuição de áudio numa rede com eventuais limitações de ritmo binário disponível e latência estimada.

É pretendido que o servidor de áudio seja capaz de distribuir uma *stream* de áudio com uma qualidade equivalente a um *CD-AUDIO* por um conjunto de clientes finais. Com este projecto, apenas é importante que o cliente tenha uma experiência final seja o melhor possível, sem atrasos, sem interrupções e sem perdas de *frames* de áudio.

Neste trabalho prático, cada cliente vai ser possível executar certos comandos para que permita fazer o controlo de áudio, sendo possível trocar, parar, pausar, repetir e entre outras opções. Os clientes têm a possibilidade de registar num serviço, escolhendo certo grupo alvo e um determinado modo de funcionamento, seja ele *master* ou *slave*.

2. Fundamentos teóricos

2.1. Áudio *PCM*

O PCM (*Pulse Code Modulation*) trata-se da forma mais primitiva de armazenamento de áudio em formato digital, tendo sido introduzido pela Sony e pela Philips no início da década de 70.

Neste método, o áudio é transformado numa série de amostras, cada qual com a sua correspondente amplitude, ou seja, representa digitalmente amostras de sinais analógicos.

No caso dos CD, por exemplo, temos 44100 amostras por segundo (44.1 KHz) com uma amplitude de 16 bits, ou seja, 65536 diferentes valores. Apesar da qualidade do áudio ser extremamente boa, o PCM não prevê nenhum tipo de compressão e é por esta mesma razão que um CD pode armazenar apenas 74 minutos de música em PCM, enquanto que a mesma media pode gravar mais de 10 horas em arquivos *OGG* ou *MP3*.

Apesar do uso mais famoso ser o CD, o PCM possui outras aplicações, como por exemplo o sistema telefónico onde o formato com taxa de amostragem de 8 KHz e 8 bits de resolução é utilizado.

2.1.1. Formato

O formato de um arquivo *WAV* é um subconjunto das especificações *RIFF* da Microsoft para o armazenamento de arquivos multimédia.

Um arquivo *RIFF* inicia com um cabeçalho de arquivo seguido por uma certa sequência de blocos de dados. Geralmente, um ficheiro *WAV* é apenas e unicamente um arquivo *RIFF* com uma parte *WAV*, que consiste em duas sub-camadas, uma camada *fmt* onde é especificado o formato dos dados e uma camada "data" onde está alocado todos os dados reais da amostra. A este formato podemos chamar de "formato canónico".

Em baixo podemos visualizar através da figura(1) como é definido o formato de um arquivo *WAV* sendo de seguida em baixo apresentada uma tabela para melhor compreensão do mesmo.

Endian	File offset (bytes)	Field name	Field size (bytes)
	0	Chunk ID	4
big	4	Chunk Size	4
little	8	Format	4
big	12	Subchunk1 ID	4
big	16	Subchunk1 Size	4
little	20	Audio Format	2
little	22	Number Channels	2
little	24	Sample Rate	4
little	28	Byte Rate	4
little	32	Block Align	2
little	34	Bits Per Sample	2
little	36	Subchunk2 ID	4
big	40	Subchunk2 Size	4
little	44	Data	Subchunk2 Size

Figura 1: Formato canónico.

O formato da *WAV* começa através de um *RIFF header*:

<i>Offset</i>	Tamanho(<i>Bytes</i>)	Nome	Descrição
0	4	<i>ChunkID</i>	Contém as letras <i>RIFF</i> no formato ASCII
4	4	<i>ChunkSize</i>	36 + <i>SubChunk2Size</i> ; Corresponde ao tamanho restante do pedaço após esse número
8	4	<i>Format</i>	Contém as letras da <i>WAV</i> (0x57415645 <i>big-endian form</i>)

Tabela 1: *RIFF header*

O formato da *WAV* consiste em dois *SubChunks*: *fmt* (*fmt Subchunk* descreve o formato da data do som) e *data*:

<i>Offset</i>	Tamanho(<i>Bytes</i>)	Nome	Descrição
12	4	<i>Subchunk1ID</i>	Contém as letras " <i>FMT</i> " (0x666d7420 <i>big-endian form</i>)
16	4	<i>Subchunk1Size</i>	16 para PCM, correspondendo ao tamanho restante do <i>SubChunk</i> que segue esse número.
20	2	<i>AudioFormat</i>	PCM=1, valores superiores a 1 indica alguma forma de compressão
22	2	<i>NumChannels</i>	<i>Mono</i> =1, <i>Stereo</i> =2, etc...
24	4	<i>SampleRate</i>	8000, 4000, etc...
28	4	<i>ByteRate</i>	$(SampleRate * NumChannels * BitsPerSample)/8$
32	2	<i>BlockAlign</i>	$(NumChannels * BitsPerSample)/8$
34	2	<i>BitsPerSample</i>	8 <i>bits</i> = 8, 16 <i>bits</i> = 16, etc...

Tabela 2: *FMT header*

O *SubChunk data* contém o tamanho dos dados e o som real:

<i>Offset</i>	Tamanho(<i>Bytes</i>)	Nome	Descrição
36	4	<i>Subchunk2ID</i>	Contém as letras " <i>DATA</i> " (0x64617461 <i>big-endian form</i>)
40	4	<i>Subchunk2Size</i>	$(NumSamples * NumChannels * BitsPerSample)/8$
44	*	<i>data</i>	Dados do som

Tabela 3: *DATA header*

Em baixo, na figura 4, podemos visualizar o conteúdo do *header* do ficheiro de áudio de exemplo fornecido pelo docente.

00000000	52 49 46 46	B0 CD 88 05	57 41 56 45	66 6D 74 20	RIFF....WAVEfmt
00000010	10 00 00 00	01 00 02 00	44 AC 00 00	10 B1 02 00D.....
00000020	04 00 10 00	4C 49 53 54	C8 00 00 00	49 4E 46 4FLIST....INFO
00000030	49 41 52 54	1A 00 00 00	4E 69 63 6B	20 43 61 76	IART....Nick Cav
00000040	65 20 26 20	54 68 65 20	42 61 64 20	53 65 65 64	e & The Bad Seed
00000050	73 00 49 43	52 44 0B 00	00 00 32 30	31 33 2D 31	s.ICRD....2013-1
00000060	31 2D 32 39	00 00 49 47	4E 52 11 00	00 00 41 6C	1-29..IGNR....Al
00000070	74 65 72 6E	61 74 69 76	65 20 52 6F	63 6B 00 00	ternative Rock..
00000080	49 4C 4E 47	04 00 00 00	65 6E 67 00	49 4E 41 4D	ILNG....eng.INAM
00000090	12 00 00 00	48 69 67 67	73 20 42 6F	73 6F 6E 20Higgs Boson
000000A0	42 6C 75 65	73 00 49 50	52 44 0F 00	00 00 4C 69	Blues.IPRD....Li
000000B0	76 65 20 66	72 6F 6D 20	4B 43 52 57	00 00 49 50	ve from KCRW..IP
000000C0	52 54 02 00	00 00 31 00	49 53 46 54	0E 00 00 00	RT....1.ISFT....
000000D0	4C 61 76 66	35 38 2E 32	39 2E 31 30	30 00 49 53	Lavf58.29.100.IS
000000E0	52 43 0D 00	00 00 47 42	53 47 34 31	32 30 30 30	RC....GBSG412000
000000F0	30 38 00 00	64 61 74 61	BC CC 88 05	00 00 00 00	08..data.....
00000100	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00

Figura 2: Conteúdo do *header* do ficheiro áudio de exemplo.

2.1.2. Codificação

<—Esclarecimento do algoritmo sugerido pelo stor, terminar a tabela—>

A codificação numa *Stream* de áudio PCM *stereo* armazenada num ficheiro WAV deve tomar um determinado conjunto de estratégias básicas muito semelhantes às estabelecidas na codificação *FLAC*.

Sabendo que ambos os canais numa *Stream* áudio *stereo* possuem conteúdo muito semelhante, uma das soluções para a codificação passa por transformar a sequência dos dois canais individuais em duas sequências alternativas que precisarão de menos *bits* para a sua correcta apresentação. Essas sequências podem ser:

- Uma sequência com a média de dois canais mais uma outra com a diferença da média para o valor de um dos canais;
- Uma sequência com um canal mais uma sequência mais uma sequência com a diferença para o outro canal;

Após a obtenção dos dois valores acima indicados (média e *offset*), são calculados os deltas através da diferença do valor atual com o valor anterior. A seguinte tabela mostra de forma gráfica o que foi acima mencionado.

	A0	A1	A2	A3	A4	A5	A6
Canal L	0	25	33	45	67	89	67
Canal R	0	23	34	56	67	78	56
$M = (L+R)/2$	0	24	33.5	50.5	67	83.5	61.5
$O = L-M$	0	1	-0.5	-5.5	0	5.5	5.5
$DM = M_f - M_i$		24	9.5	17	16.5	16.5	-22
$DO = O_f - O_i$		1	-1.5	-5	5.5	5.5	0

Tabela 4: Exemplo de codificação

2.2. Protocolo de *Streaming*

O protocolo de *Streaming* pode ser encapsulado em TCP ou em UDP, sendo que este último proporciona um protocolo aplicacional com um menor atraso e, caso as redes sejam fiáveis, permite ainda um ritmo de dados mais elevado. Contudo, o uso do UDP apenas pode não conseguir assegurar fiabilidade em redes pouco fiáveis, devido ao número elevado de perdas possíveis e, neste caso, o uso do TCP é preferível. De forma a ser possível a utilização do UDP de forma fiável, o protocolo aplicacional deve estabelecer mecanismos adicionais para:

- Assegurar a correcta ordenação dos *frames*/blocos de áudio;
- Baixar a perda de *frames*/blocos de áudio por falta de fiabilidade nos níveis de transporte e rede;

Posto isto, os dois últimos tópicos acima descritos podem ser alcançados através do uso de mecanismos para:

- Correcta numeração de *frames*/blocos/datagramas de nível aplicacional;
- *Bufferização* de *frames*/blocos/datagramas aplicacionais tanto nos *gateways* como nos clientes;
- Envio de uma ou mais cópias de cada *frame*/bloco/datagrama aplicacional de forma a prevenir a perda;
- Envio dos datagramas numa sequência aleatória;

3. Arquitectura do sistema

A representação ilustrativa da nossa arquitectura pode ser visualizada na seguinte imagem:

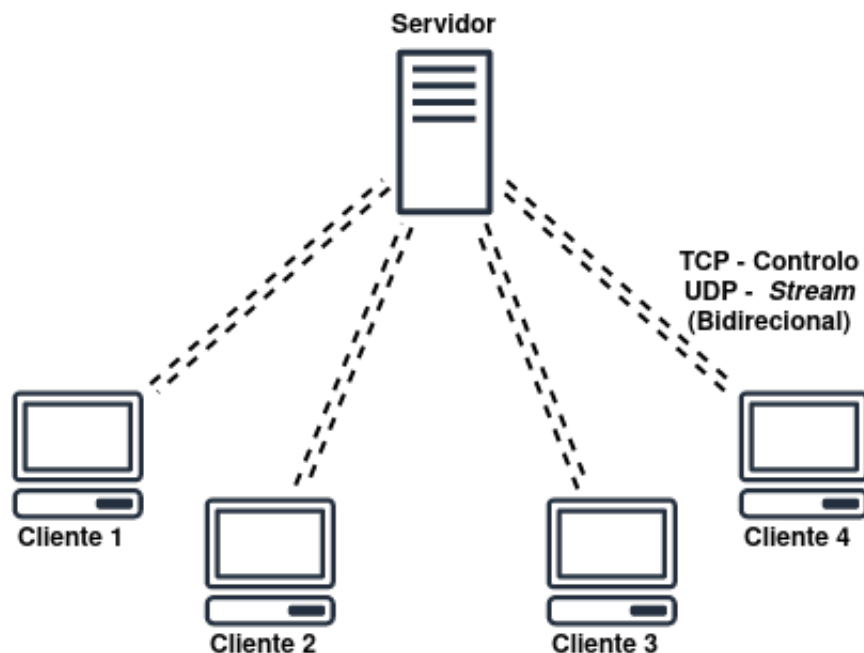


Figura 3: Arquitetura do sistema

Através da figura 3, entende-se que este sistema faz uso de dois protocolos da camada de transporte, TCP e UDP. O grupo escolheu esta alternativa, de forma a poder ter a segurança e a qualidade de serviço que o protocolo TCP oferece e ao mesmo tempo poder usufruir da baixa latência oferecida pelo protocolo UDP.

Desta forma, sob o protocolo TCP apenas passam pacotes de controlo, sendo estes responsáveis por iniciar uma *stream*, atualizar a lista de musicas e parar uma *stream*.

Como resultado, sob o protocolo UDP pode-se limitar mais os pacotes, usufruindo melhor do espaço disponível em cada pacote, minimizando a largura de banda gasta.

4. Implementação

4.1. Estratégia de codificação

Após o entendimento do formato de áudio *Stereo PCM* e do estudo do algoritmo de compressão sugerido pelo docente, o grupo começou a fazer as principais decisões, de forma a minimizar os tempos e os valores de compressão.

Com este objectivo, a primeira decisão foi ler um bloco grande(2 *Mega Bytes*) de dados, comprimi-lo e só depois encapsula-lo em pacotes para envio. Isto, porque a leitura de blocos mais pequenos de dados cíclica, torna-se um processo lento em ficheiros grandes.

De seguida, ficou decido calcular dois tamanhos diferentes entre para cada amostra(*Delta Average e Delta Offset*). Este calculo dos tamanhos é simplesmente o tamanho em *bits* da média e do valor máximo, sendo estes representados por *Min* e *Max*, respectivamente.

A seguinte decisão, foi separar este bloco maior em blocos mais pequenos, acompanhados de um cabeçalho com o tamanho médio e máximo em *bits* das amostras calculadas(cada tamanho ocupa 5 *bits*). Depois após de alguns testes foi decidido que cada bloco tivesse 1024 amostras, ou seja 4096 *Bytes* cada bloco antes da compressão.

Por fim, não esquecendo, cada amostra calculada tem um *bit* a anunciar o tamanho, e por cada valor da amostra existe um primeiro *bit* responsável pelo complemento(se é negativo ou não), e o segundo *bit* é responsável pelo resto da divisão(se tem 0.5 ou não).

Exemplo:

```
Delta Average Min: 4
Delta Offset  Min: 3
Delta Average Max: 6
Delta Offset  Max: 5
```

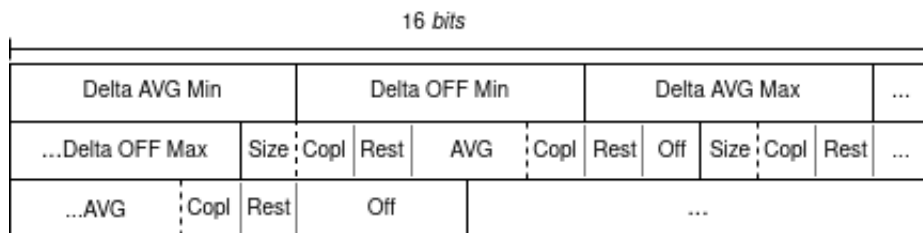


Figura 4: Formato do cabeçalho com duas amostras.

4.2. Estratégia de Distribuição de áudio

Depois da leitura das sugestões do docente, da análise do processo anterior e de uma breve pesquisa, ficou decidido a utilização de *burst's* de pacotes numerados, para a transmissão de cada bloco (de 2 *Mega Bytes*) comprimido.

Em primeiro lugar, e como sugestão do docente, encontrou-se o MTU (*Maximum Transfer Unit*), para definir o tamanho máximo de cada pacote (1472 *Bytes*). Isto para evitar a fragmentação dos pacotes que pode levar à perda dos mesmos.

Em segundo lugar, definir o tamanho dos *burst's*, este ficou definido a 100 pacotes. Apesar de ser um número pequeno, o grupo teve em consideração questões mais realistas tais como a diversidade de máquinas onde poderá vir a correr o programa. Sabendo que por atribuir 100 pacotes de *burst's* e tendo cada pacote 1472 *Bytes*, isto significa atribuir mais de 147200 *Bytes* ao *Recv Buffer* da *Socket*. E como medida de segurança é importante definir mais um pouco para garantir que todos os pacotes cheguem, nosso caso foi definido o dobro do tamanho, ou seja 294400 *Bytes*.

E por fim, para garantir a fiabilidade da transmissão, foi utilizado uma trama de *ACK/NACK* a cada 100 pacotes (a cada *burst*), avisando a falha ou o sucesso. Nesta retransmissão foi introduzida uma técnica para que o atraso causado pela perda fosse minorado. Esta técnica, basicamente, entra quando há perdas e faz o reenvio dos pacotes perdidos mais os seguintes pacotes de forma a que a soma de ambos tenha o tamanho de um *burst*.

Exemplo:

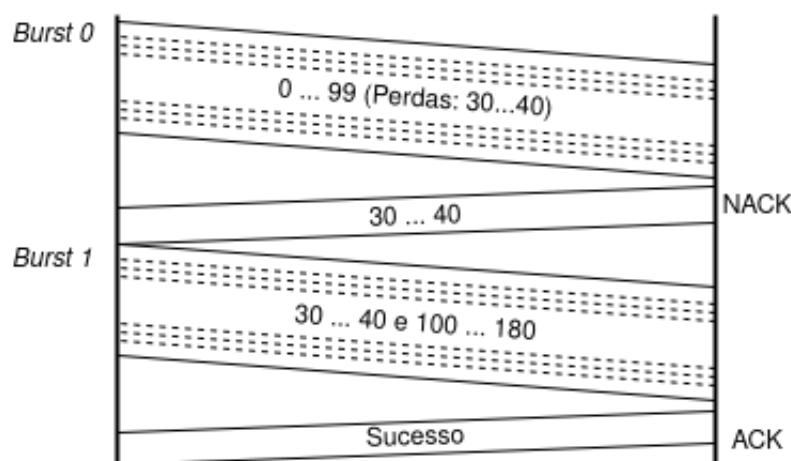


Figura 5: Representação de uma retransmissão, num diagrama temporal.

4.3. Protocolo de Controlo

Neste programa foi utilizado um protocolo de controlo de reprodução sob TCP. Este é responsável pelas tramas de actualização da lista de musicas, tocar e parar a o *Streaming* de áudio, e também o fim de ligação, sabendo que caso a ligação TCP seja interrompida, o *Streaming* de áudio é da mesma forma terminado.

Existe também uma trama que é transmitida no inicio da ligação, que informa ao cliente de que grupo *Multicast* ele se deve juntar.

As tramas utilizadas possuem a seguinte definição:

IP Group:	< Multicast IP Group >	4 Bytes
Update:	< 0 - Opcode > < Lista de Paths das musicas disponveis >	1 Byte Resto
Play:	< 1 - Opcode > < Path da musica a tocar >	1 Byte Resto
Stop:	< 2 - Opcode >	1 Byte
Exit:	< 3 - Opcode >	1 Byte

4.4. Protocolo de *Streaming*

Este é o protocolo principal deste programa, é através dele que passam os dados de cada *Stream*. Foi desenvolvido de forma a ser o mais simples possível e com uma estrutura mais mínima, de forma a maximizar a quantidade de dados por trama.

As tramas utilizadas possuem a seguinte definição:

Info:	< 0 - Opcode >	< N. Canais >	< Frequencia de Amostragem >	< N. Bits por amostra >
	2 Bits	2 Bytes	4 Bytes	2 Bytes
Data:	< 1 - Opcode >	< N. Pacote >	< DATA >	
	2 Bits	14 Bits	1470 Bytes	
Last:	< 2 - Opcode >	< N. Pacote >	< DATA >	
	2 Bits	14 Bits	1470 Bytes	
ACK/NACK:	< 3 - Opcode >	(n* x < N. Pacote perdido >)	< Identificador de fim >	
	2 Bits	2 Bytes	2 Bytes	

(n* representa a existência de nenhum ou múltiplos identificadores)

4.5. Adicional

4.5.1. Gestão de clientes

De forma a conseguir gerir mais que um cliente e de forma dinâmica, o programa faz uso múltiplos processos para tratar individualmente a informação, e utiliza diferentes grupos de *Multicast*.

O intervalo de grupos *Multicast* utilizado é do 224.0.0.189 até ao 224.0.0.255, sendo incrementativo por cada cliente.

Por cada cliente existe um ligação TCP e um processo que faz o controlo da *Stream* conforme o pedido do cliente, e por cada *Stream* é iniciado um processo para se responsabilizar pela transmissão da *Stream*.

4.5.2. Gestão e organização automática das musicas

Sendo este um serviço de de *Stream* de áudio, é necessário a organização de todo o áudio existente, e para automatizar este processo foi desenvolvido um *Script* em Python.

Este é inicializado automaticamente com o servidor e cria uma directoria para onde todos ficheiros de áudio podem ser arrastados, num estilo *Drag and Drop*.

Este *script* fica á escuta de novos ficheiros, e quando recebe um novo, verifica e cria directorias conforme a *metadata* nele existente, no final se o ficheiro já estiver em formato *.wav* apenas o move para a respectiva directoria, caso contrario, converte-o para *.wav* através do comando *ffmpeg*, directamente para a respectiva directoria e elimina o ficheiro original.

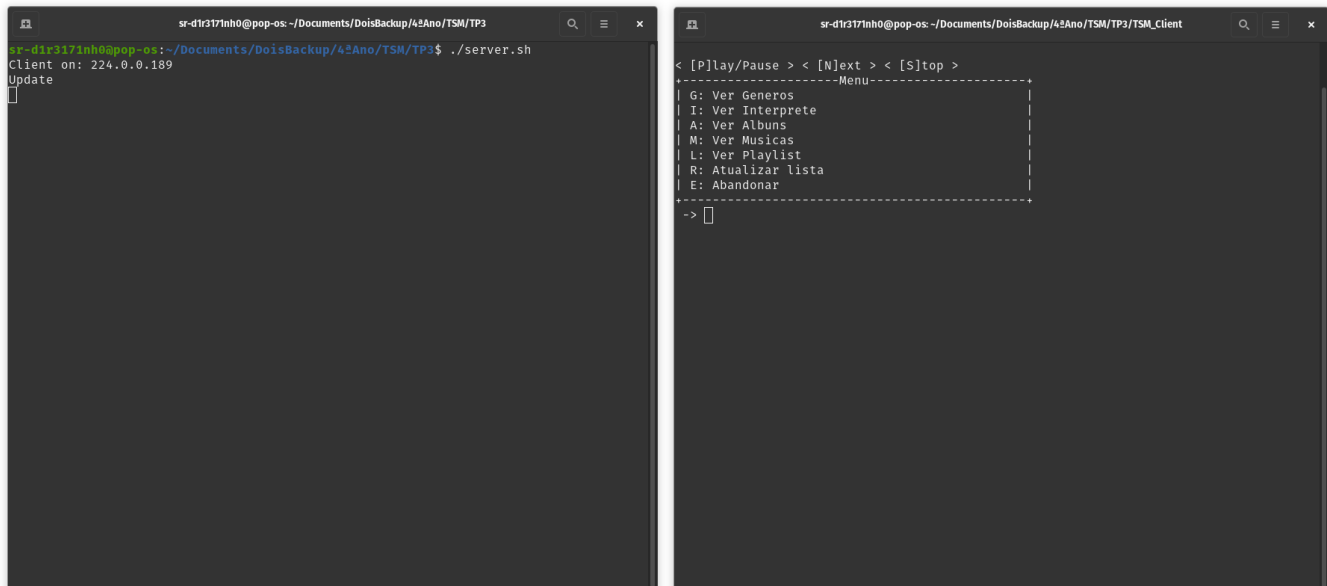
No fim de organizar todos os ficheiros que foram arrastados, o *script* actualiza o ficheiro com a lista de áudios existentes.

4.5.3. Dependências

Para o funcionamento correto deste programa, é necessário um conjunto de *Software* adicional, listado no seguinte esquema:

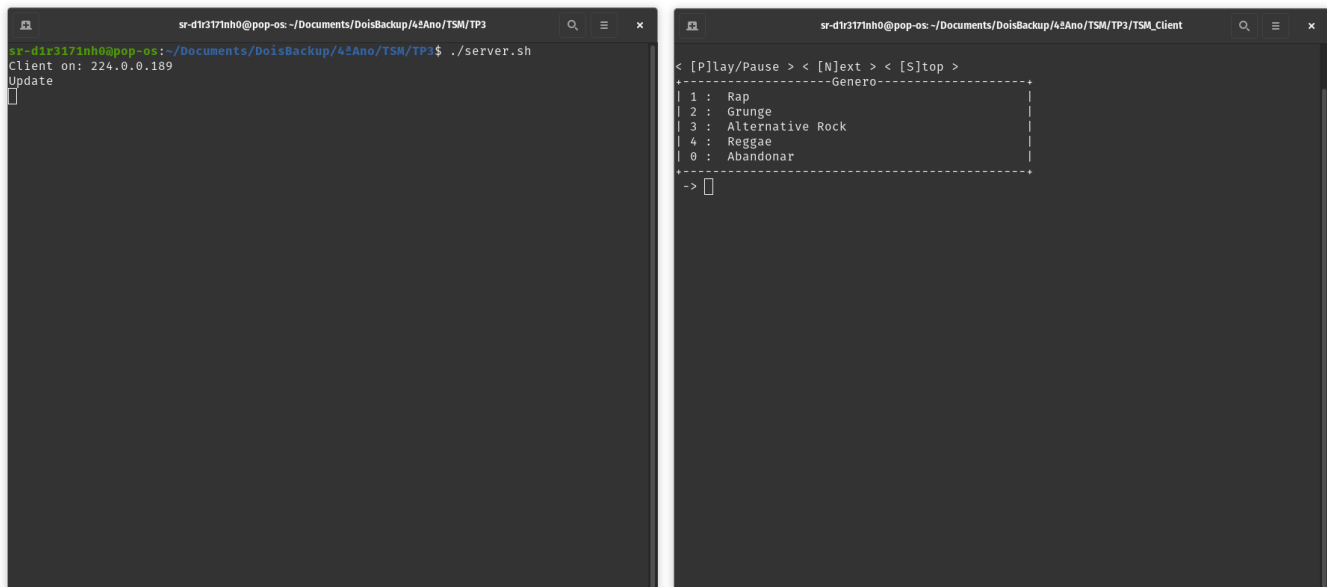
```
>Linux
  >build-essential(GCC)
  >ffmpeg
  >openjdk-8-jdk(JAVA 8)
  >python3
    >mutagen
    >watchdog
```

5. Funcionamento



The image shows two terminal windows side-by-side. The left window is titled 'sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3' and shows the execution of './server.sh'. The output includes 'Client on: 224.0.0.189' and 'Update'. The right window is titled 'sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3/TSM_Client' and shows a menu with options: 'G: Ver Generos', 'I: Ver Interprete', 'A: Ver Albuns', 'M: Ver Musicas', 'L: Ver Playlist', 'R: Atualizar lista', and 'E: Abandonar'. The prompt is '->'.

Figura 6: Inicio de ambos lados(Servidor e Cliente).



The image shows two terminal windows side-by-side. The left window is the same as in Figure 6, showing the server output. The right window is the same as in Figure 6, showing the menu. The prompt is '->'.

Figura 7: Listagem de géneros.

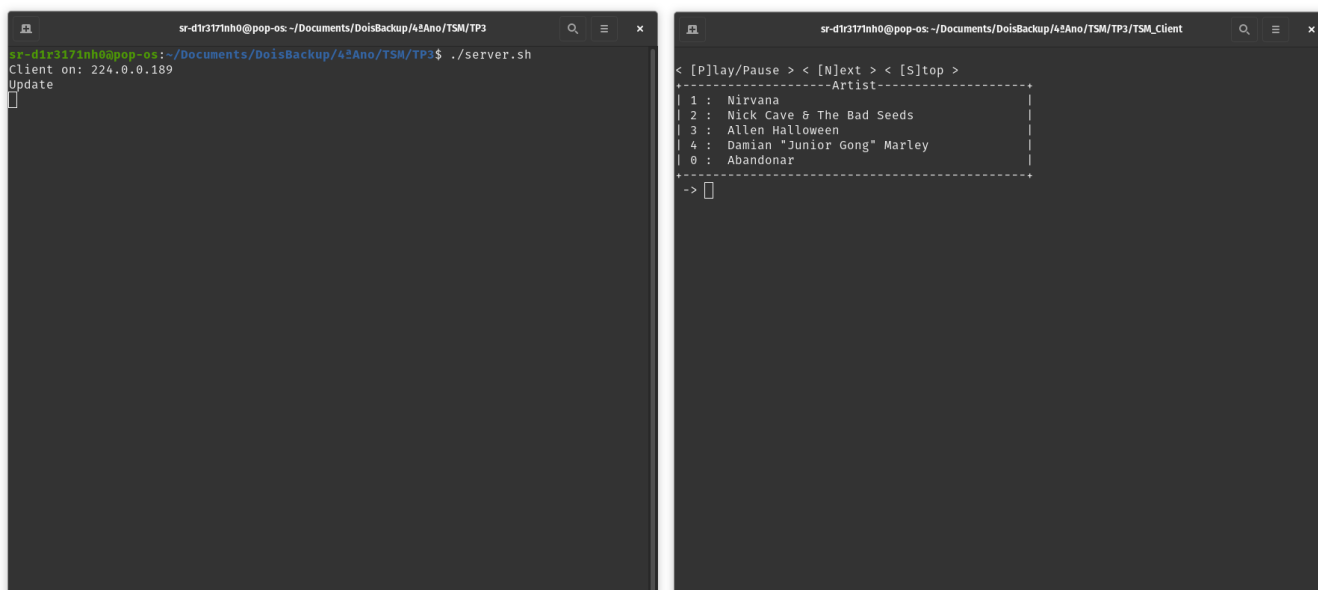


Figura 8: Listagem de Artistas.

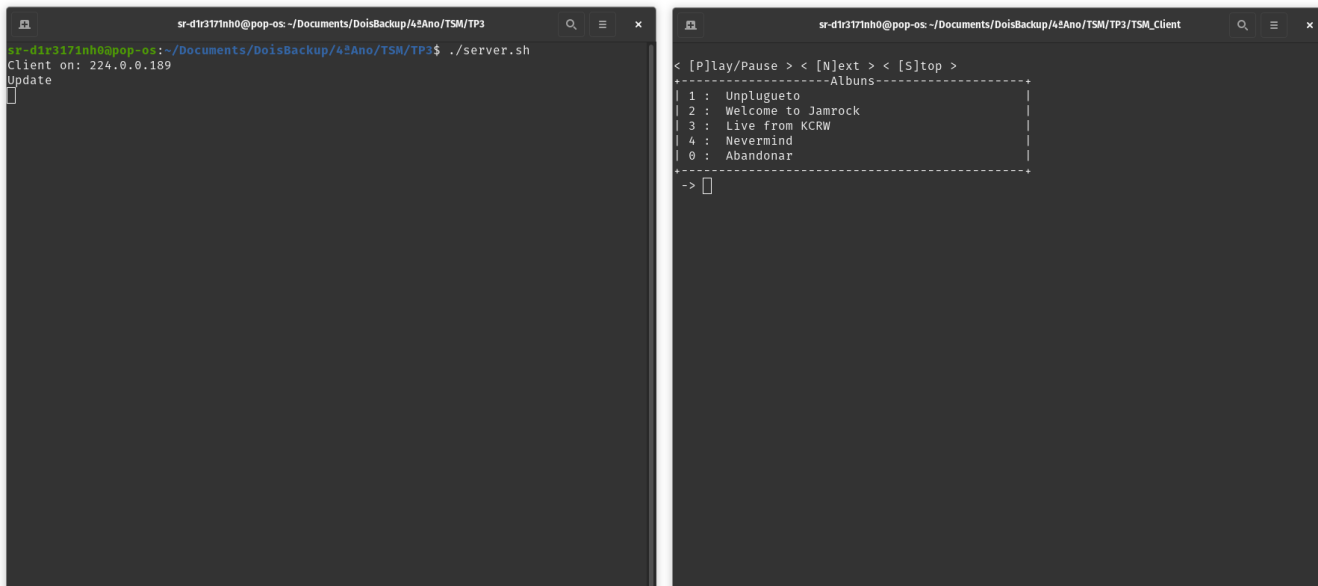
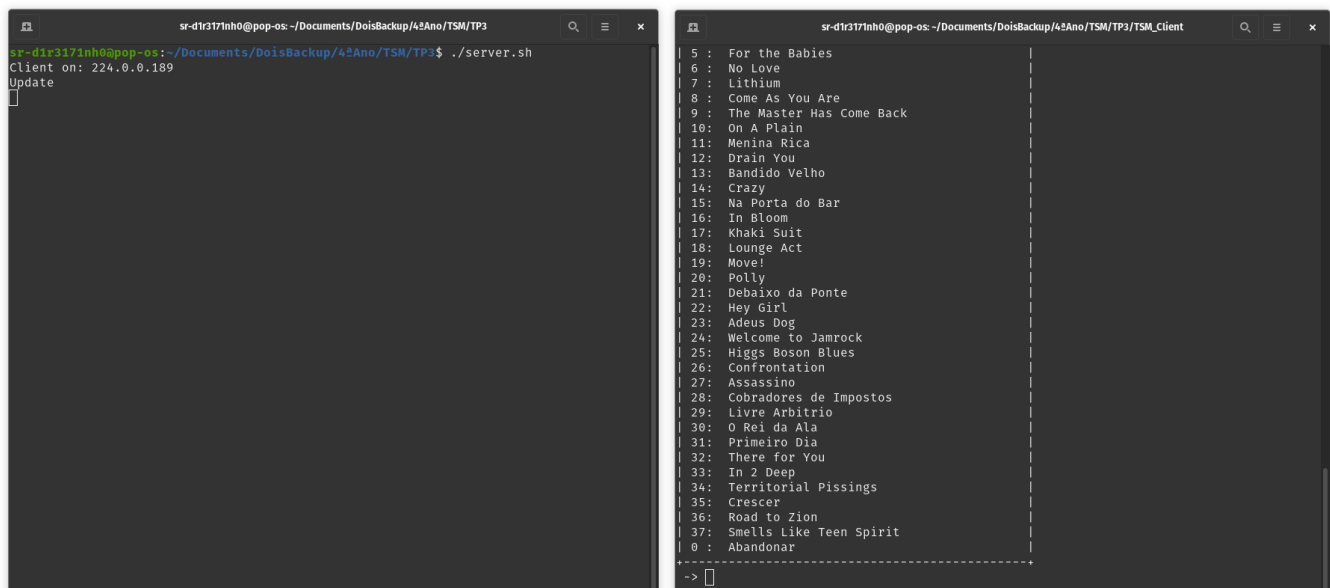


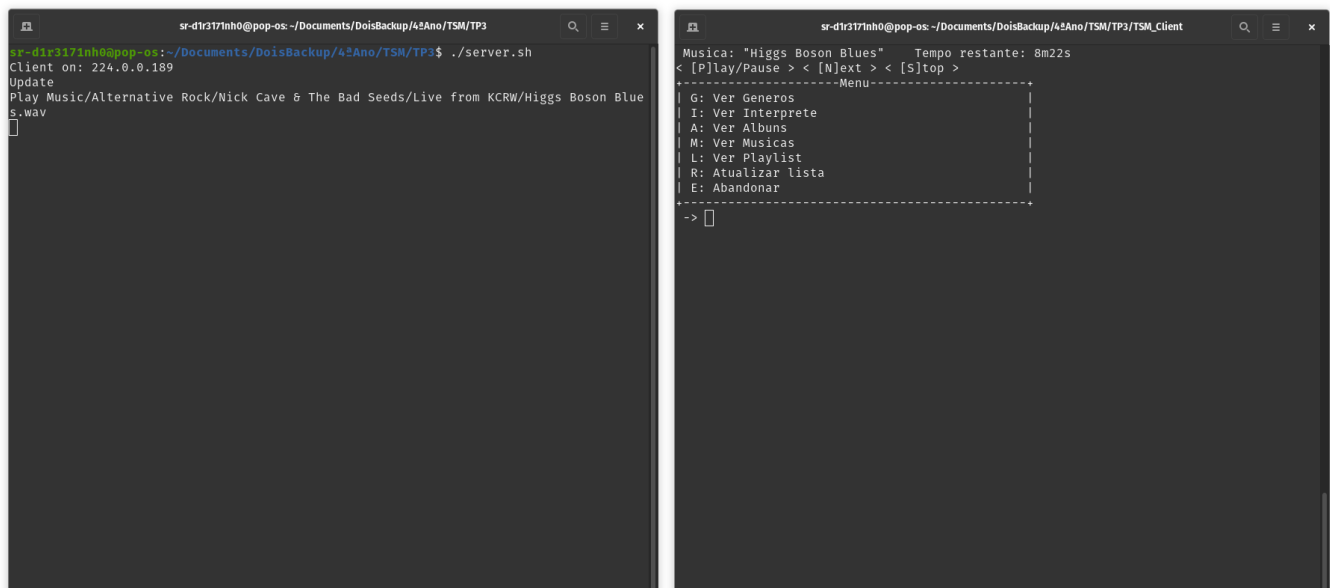
Figura 9: Listagem de Álbuns.



```
sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/4ºAno/TSM/TP3$ ./server.sh
Client on: 224.0.0.189
Update
█

sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3/TSM_Client
5 : For the Babies
6 : No Love
7 : Lithium
8 : Come As You Are
9 : The Master Has Come Back
10: On A Plain
11: Menina Rica
12: Drain You
13: Bandido Velho
14: Crazy
15: Na Porta do Bar
16: In Bloom
17: Khaki Suit
18: Lounge Act
19: Move!
20: Polly
21: Debaixo da Ponte
22: Hey Girl
23: Adeus Dog
24: Welcome to Jamrock
25: Higgs Boson Blues
26: Confrontation
27: Assassino
28: Cobradores de Impostos
29: Livre Arbitrio
30: O Rei da Ala
31: Primeiro Dia
32: There for You
33: In 2 Deep
34: Territorial Pissings
35: Crescer
36: Road to Zion
37: Smells Like Teen Spirit
0 : Abandonar
--> █
```

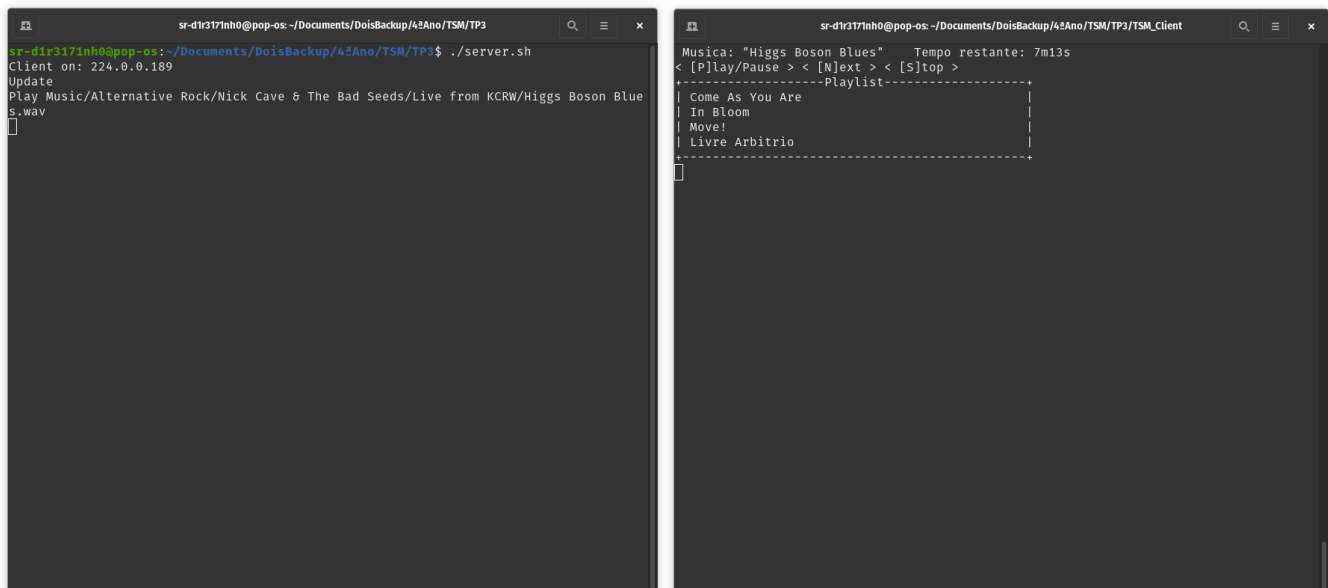
Figura 10: Listagem de Músicas.



```
sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/4ºAno/TSM/TP3$ ./server.sh
Client on: 224.0.0.189
Update
Play Music/Alternative Rock/Nick Cave & The Bad Seeds/Live from KCRW/Higgs Boson Blues.wav
█

sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/4ºAno/TSM/TP3/TSM_Client
Musica: "Higgs Boson Blues"      Tempo restante: 8m22s
< [P]lay/Pause > < [N]ext > < [S]top >
-----Menu-----
G: Ver Generos
I: Ver Interprete
A: Ver Albuns
M: Ver Musicas
L: Ver Playlist
R: Atualizar lista
E: Abandonar
--> █
```

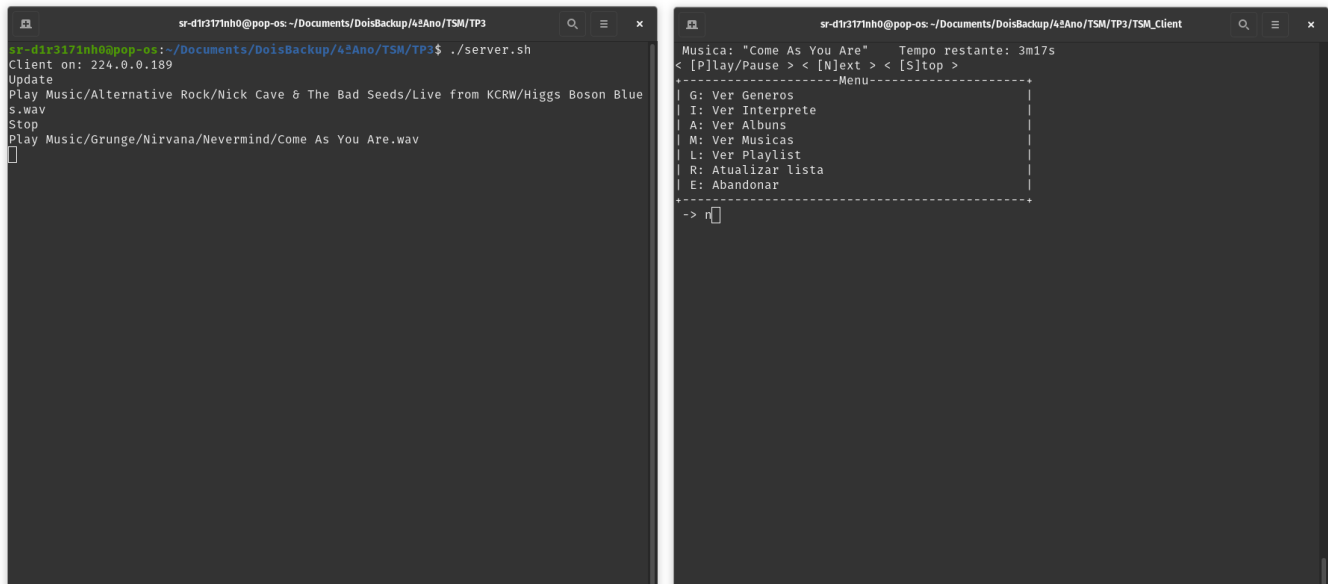
Figura 11: Tocar uma música.



The figure consists of two terminal windows side-by-side. The left window shows the execution of a server script. The right window shows the client interface displaying a playlist.

```
sr-d1r3171nh0@pop-os: ~/Documents/DoisBackup/42Ano/TSM/TP3
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/42Ano/TSM/TP3$ ./server.sh
Client on: 224.0.0.189
Update
Play Music/Alternative Rock/Nick Cave & The Bad Seeds/Live from KCRW/Higgs Boson Blue
s.wav
[]
```

```
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/42Ano/TSM/TP3/TSM_Client
Musica: "Higgs Boson Blues"    Tempo restante: 7m13s
< [P]lay/Pause > < [N]ext > < [S]top >
-----Playlist-----
| Come As You Are
| In Bloom
| Move!
| Livre Arbitrio
|
[]
```

Figura 12: Listagem da *Playlist*.

The figure consists of two terminal windows side-by-side. The left window shows the server script being updated with a new music file. The right window shows the client interface displaying a menu with various options.

```
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/42Ano/TSM/TP3
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/42Ano/TSM/TP3$ ./server.sh
Client on: 224.0.0.189
Update
Play Music/Alternative Rock/Nick Cave & The Bad Seeds/Live from KCRW/Higgs Boson Blue
s.wav
Stop
Play Music/Grunge/Nirvana/Nevermind/Come As You Are.wav
[]
```

```
sr-d1r3171nh0@pop-os:~/Documents/DoisBackup/42Ano/TSM/TP3/TSM_Client
Musica: "Come As You Are"    Tempo restante: 3m17s
< [P]lay/Pause > < [N]ext > < [S]top >
-----Menu-----
| G: Ver Generos
| I: Ver Interprete
| A: Ver Albuns
| M: Ver Musicas
| L: Ver Playlist
| R: Atualizar lista
| E: Abandonar
|
-> r[]
```

Figura 13: Passar a frente uma música.

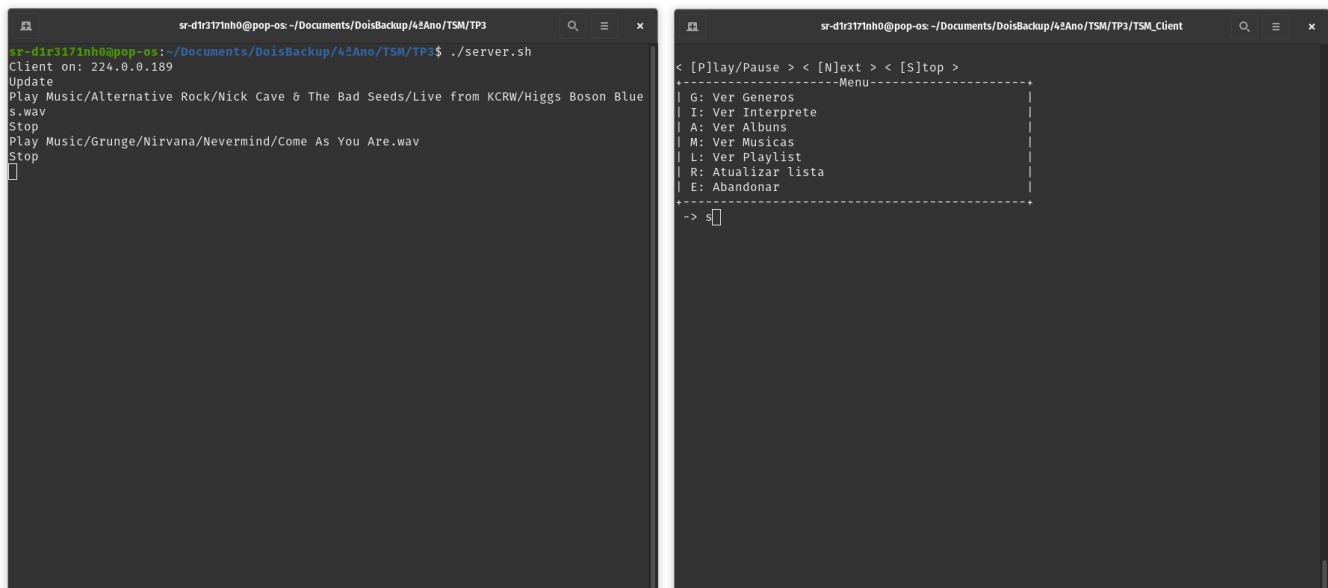
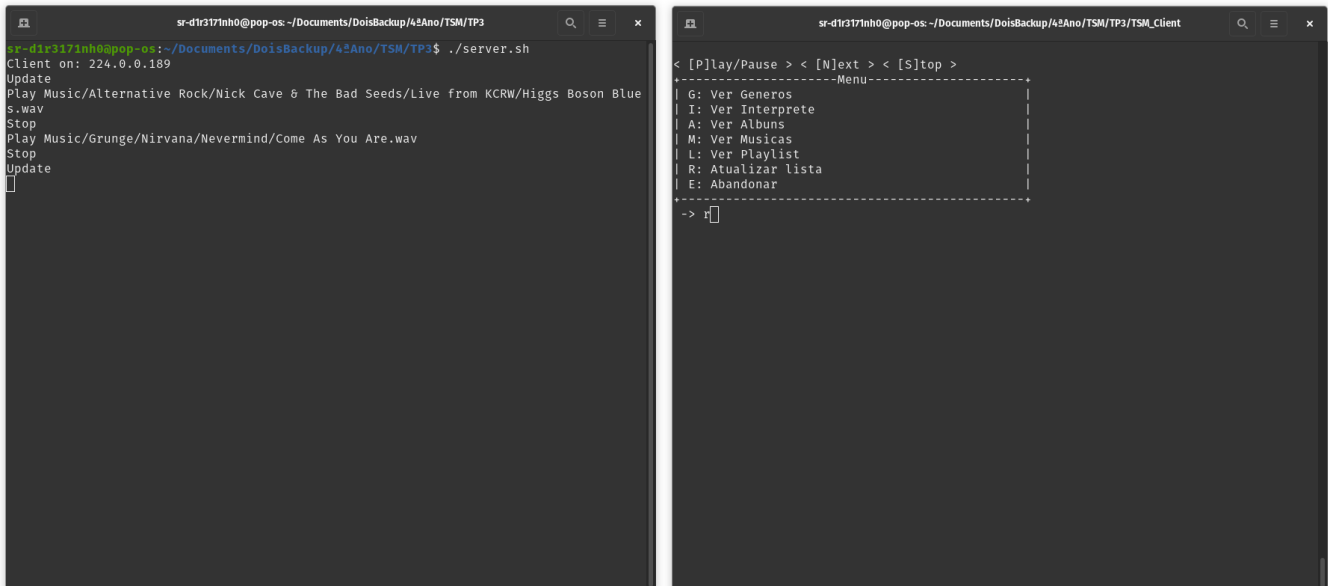
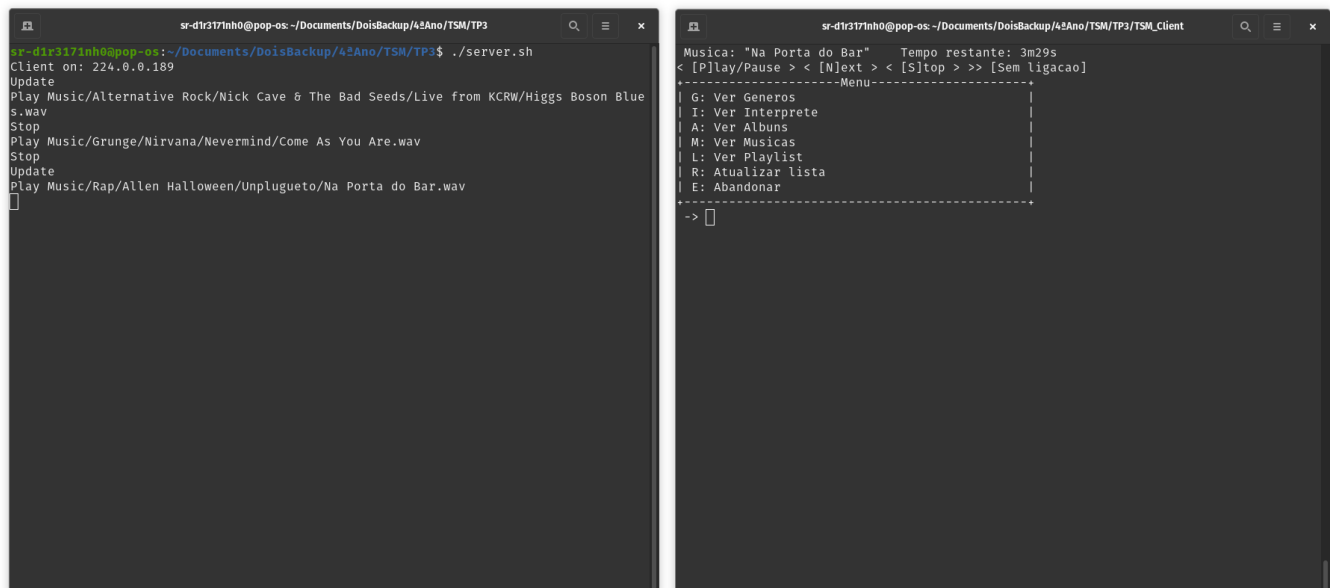
Figura 14: Parar/finalizar a *Playlist*.

Figura 15: Actualizar lista de música.



```
sr-dir3171nh0@pop-os: ~/Documents/DoisBackup/44Ano/TSM/TP3
sr-dir3171nh0@pop-os:~/Documents/DoisBackup/44Ano/TSM/TP3$ ./server.sh
Client on: 224.0.0.189
Update
Play Music/Alternative Rock/Nick Cave & The Bad Seeds/Live from KCRW/Higgs Boson Blue
s.wav
Stop
Play Music/Grunge/Nirvana/Nevermind/Come As You Are.wav
Stop
Update
Play Music/Rap/Allen Halloween/Unplugueto/Na Porta do Bar.wav
[]

sr-dir3171nh0@pop-os: ~/Documents/DoisBackup/44Ano/TSM/TP3/TSM_Client
Musica: "Na Porta do Bar"      Tempo restante: 3m29s
< [P]lay/Pause > < [N]ext > < [S]top > >> [Sem ligacao]
-----Menu-----
| G: Ver Generos              |
| I: Ver Interprete          |
| A: Ver Albuns               |
| M: Ver Musicas              |
| L: Ver Playlist             |
| R: Atualizar lista          |
| E: Abandonar                |
-----
-> []
```

Figura 16: Exemplo de falha de ligação.

6. Conclusão

Concluindo este projecto, apesar dos resultados positivos obtidos e do funcionamento correto, o grupo não ficou totalmente satisfeito, pois não foi capaz de concluir todas as etapas previstas.

A construção do algoritmo, de volta do protocolo definido, tornou-se um processo demorado, acabando por nos ocupar tempo crucial. E devido a esta nossa implementação algorítmica, o uso de *gateways* era possível, mas não prático.

Contudo, após todo este desenvolvimento, todo o nosso trabalho provou-se funcional, cumprindo todos os objectivos funcionais propostos.

7. Referências

[1] Enunciado Trabalho Prático Nº3

"DISTRIBUIÇÃO DE ÁUDIO PC", Bruno Dias

[2] <http://soundfile.sapp.org/doc/WaveFormat/>

"WAVE PCM soundfile format"

[3] <https://docs.oracle.com/javase/8/docs/api/>

"Java™ Platform, Standard Edition 8 API Specification"