Design of IoM

Document Version 1.2

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
2014.11.26	1.2	Bao Trung	Activity Diagram Screenshot
2014.11.24	1.1	Bao Trung	System Flow Class Diagram Top View
2014.11.15	1.0	Bao Trung	Initial Release Introduction

Korea, Seoul, Hanyang University, Department of Computer Engineering

Design of System Program 2, Autumn 08/12/2014

최진식 Professor

2012003426	바오중

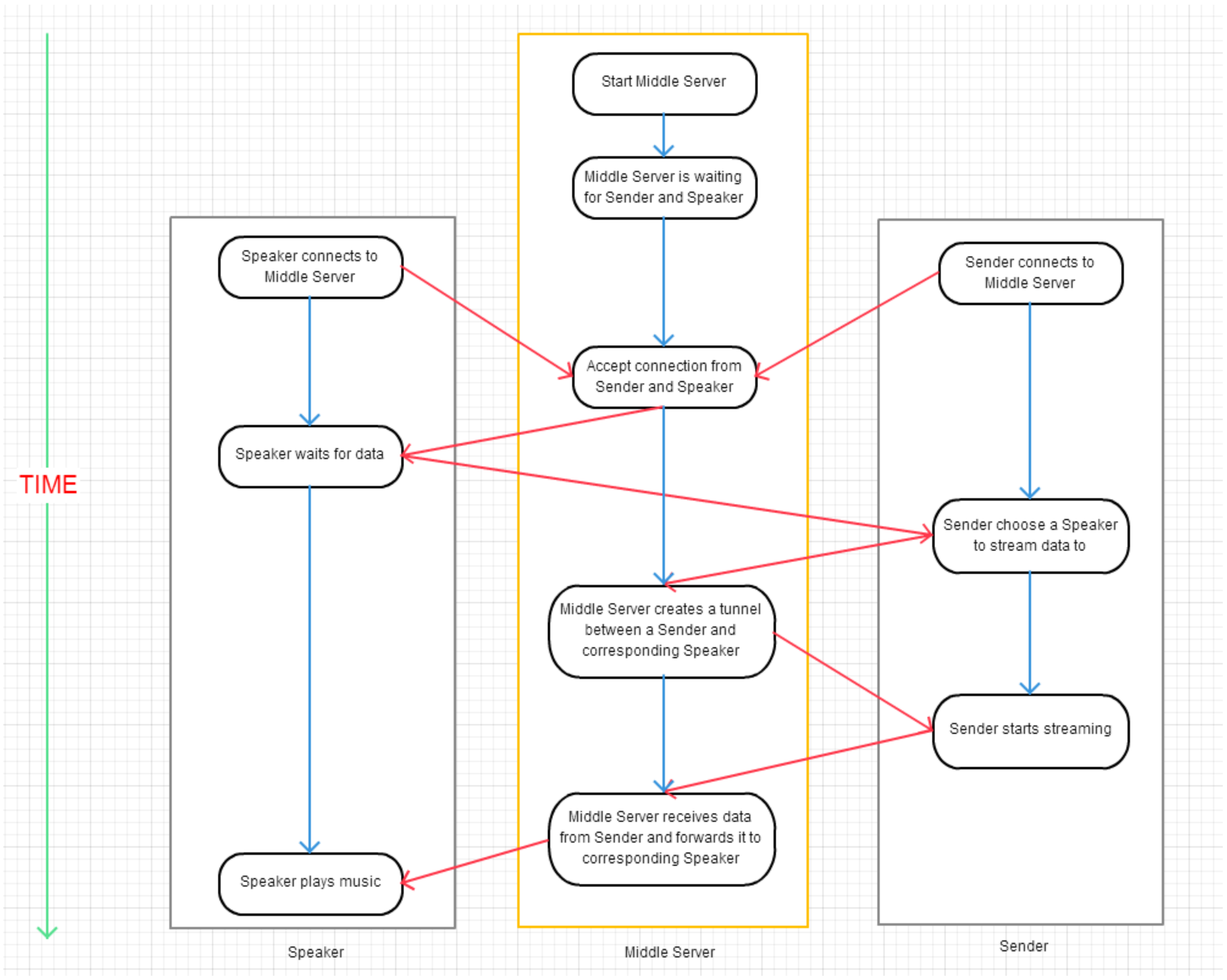
Menu

1. Introduction

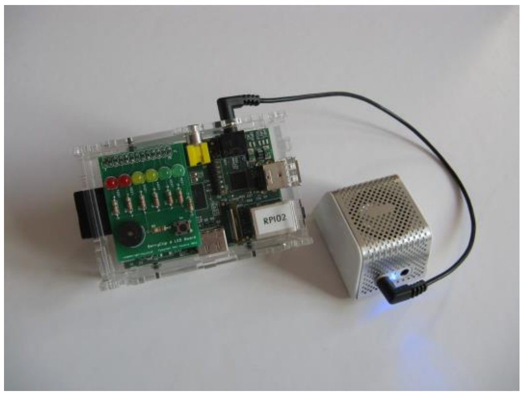
IoM (Internet of Music) is a system for streaming music from a smartphone to a remote speaker via Internet. This is especially useful when 2 friends or a couple who are not living in the same place want to share a song (listen together, real time synchronization, even in different places).

This design document includes System Design, Diagrams, and Screenshots.

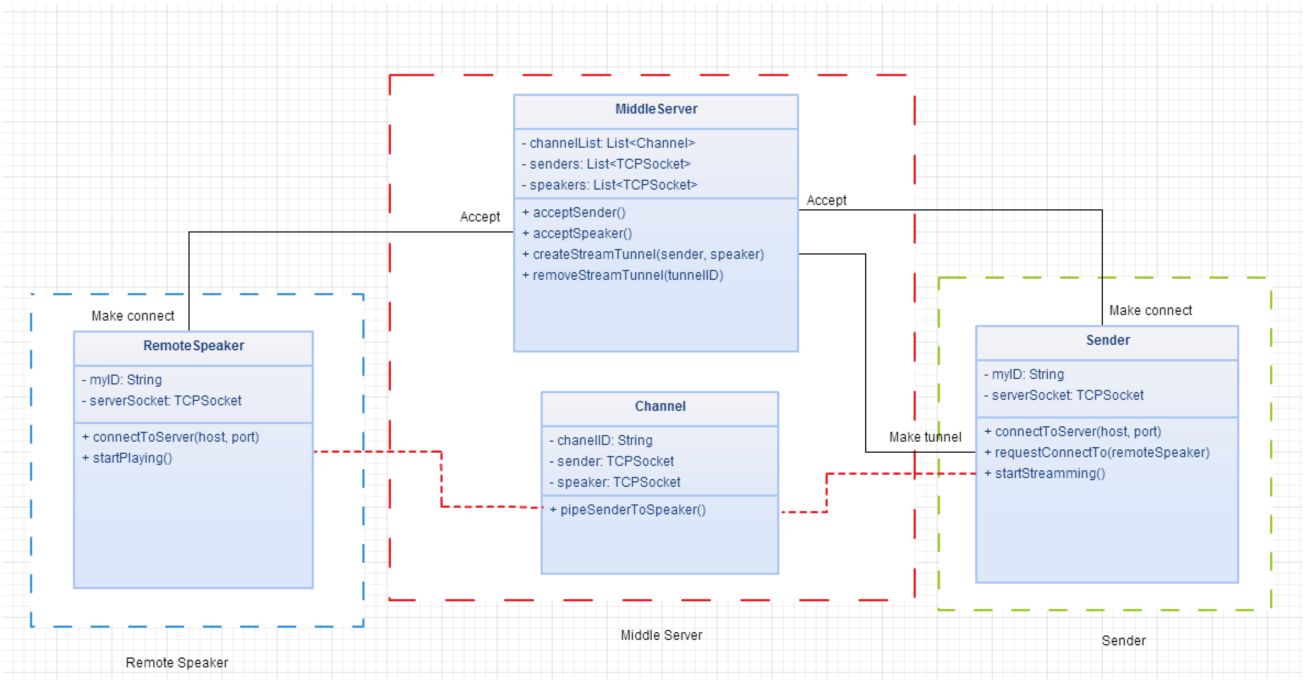
2. System Flow


TIME
Start Middle Server
Middle Server is waiting for Sender and Speaker
Speaker connects to Middle Server
Accept connection from Sender and Speaker
Sender connects to Middle Server
Speaker waits for data
Sender choose a Speaker to stream data to
Middle Server creates a tunnel between a Sender and corresponding Speaker
Sender starts streaming
Middle Server receives data from Sender and forwards it to corresponding Speaker
Speaker plays music
Speaker
Middle Server
Sender

IoM System has 3 components: Middle Server, Sender and Remote Speaker

Middle Server	Middle Server is used to manage Senders and Remote Speakers. This server creates TCP connection with Sender and Remote Speaker and then serves as a virtual tunnel between them (like a pipe). Middle Server receives stream data from a Sender and forwards it to corresponding Remote Speaker.
Sender	Smartphone which streams data to Remote Speaker. Before it can stream any data, it must firstly connect to Middle Server and choose a Remote Speaker to stream data to.
Remote Speaker	A Raspberry Pi with speaker. Before it can receive data and play, it must connect to Middle Server. 

3. Class Diagram



Channel Class

Attributes	
channelID	ID to specify a sender-speaker pair.
sender	Sender's TCP socket.
speaker	Speaker's TCP socket.
Operations	
pipeSenderToSpeaker()	Get data from InputStream of sender's socket and write that data to OutputStream of Speaker's socket. This operation will start a new thread to receive and forward data (stream) without blocking the whole program.

MiddleServer Class

Attributes	
channelList	List of channels (pairs of tunneled sender-speaker).
senders	List of Senders.
speakers	List of Speakers.
Operations	
acceptSender()	Wait and accept new connection from a Sender then save this socket into Sender's List (*senders*). This will start a new thread just for receiving Sender's connection (not to block the whole program).
acceptSpeaker()	Wait and accept new connection from a Speaker then save this socket into Speaker's List (*speakers*). This will start a new thread just for receiving Speaker's connection (not to block the whole program).
createStreamTunnel(sender, speaker)	Create a new Channel object with sender and speaker as corresponding arguments. Store this new object into Channel's List (*channelList*) and then, run the *pipeSenderToSpeaker()* method of this new object to connect these sender and speaker.
removeStreamTunnel(tunnelID)	Remote this tunnel from Channel's List (*channelList*). This also stops the streaming between corresponding sender and speaker.

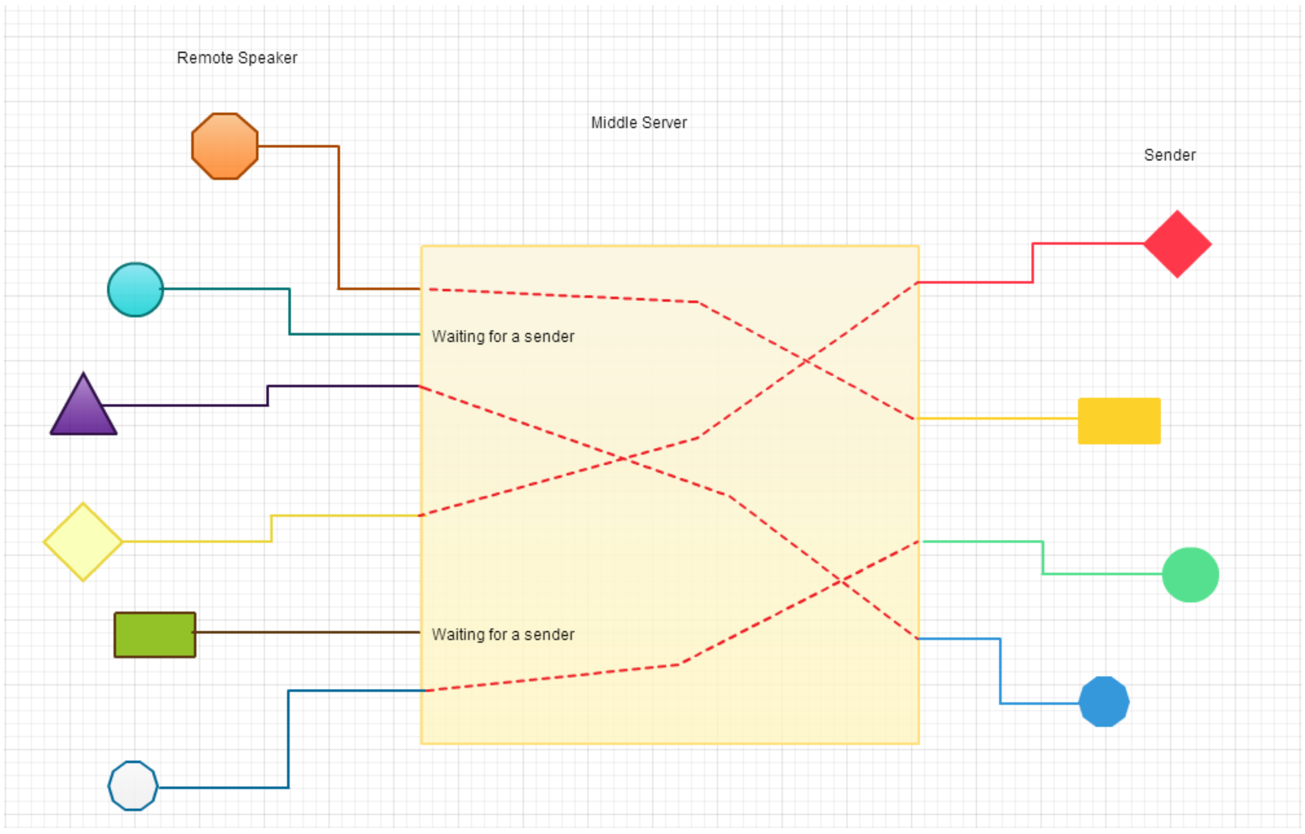
Sender Class

Attributes	
myID	ID of this Sender provided by Middle Server after making connection.
serverSocket	Sender's TCP Socket to Middle Server.
Operations	
connectToServer(host, port)	Make TCP socket connection to Middle Server.
requestConnectTo(remoteSpeaker)	Ask Middle Server to make a tunnel between this Sender and corresponding Speaker.
startStreamming()	Start writing data to serverSocket which, in turn, will be forwarded to corresponding Speaker by Middle Server (after calling *requestConnectTo(remoteSpeaker),* Middle Server already created a tunnel between this Sender and corresponding Speaker).

Speaker Class

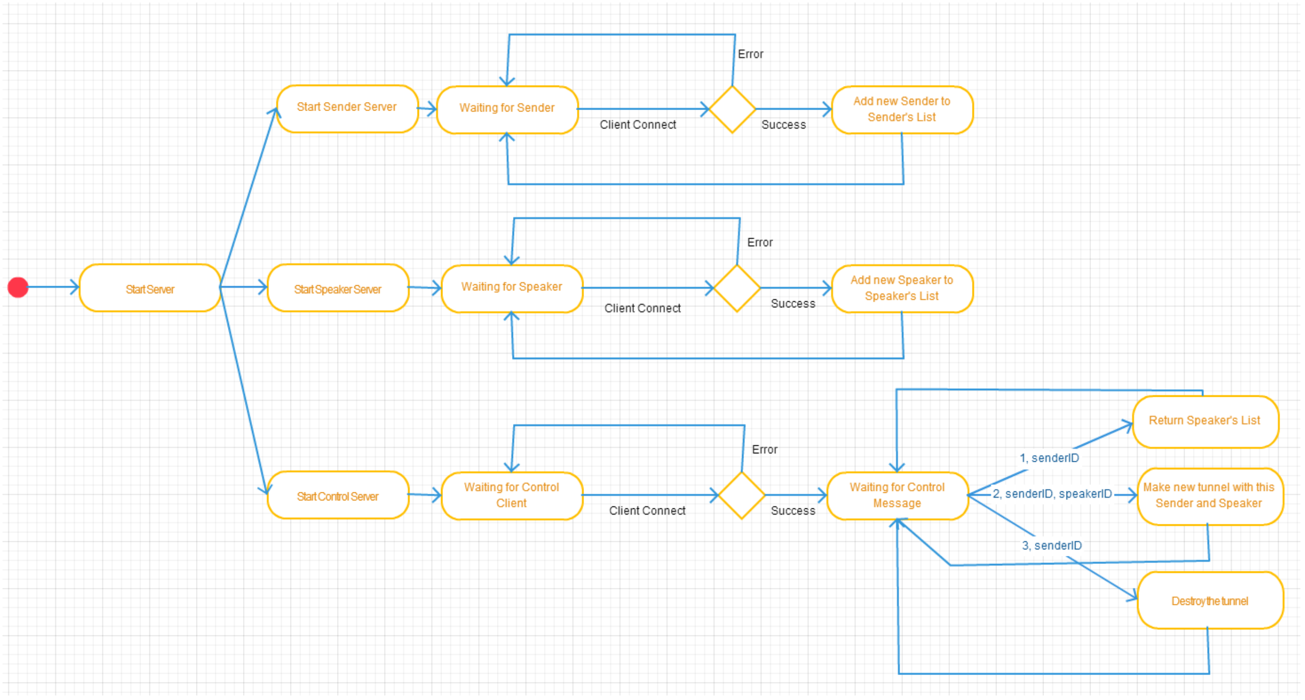
Attributes	
myID	ID of this Speaker provided by Middle Server after making connection.
serverSocket	Speaker's TCP Socket to Middle Server.
Operations	
connectToServer(host, port)	Make TCP socket connection to Middle Server.
startPlaying()	Start playing music with data received from Middle Server.

4. Top View

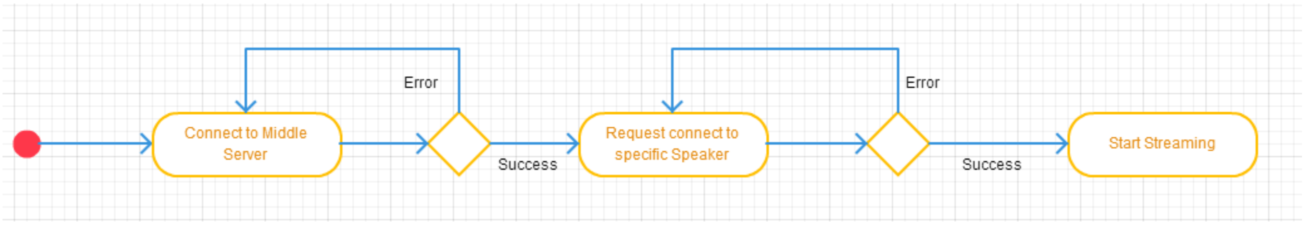

Remote Speaker
Middle Server
Sender
Waiting for a sender
Waiting for a sender

5. Activity Diagram

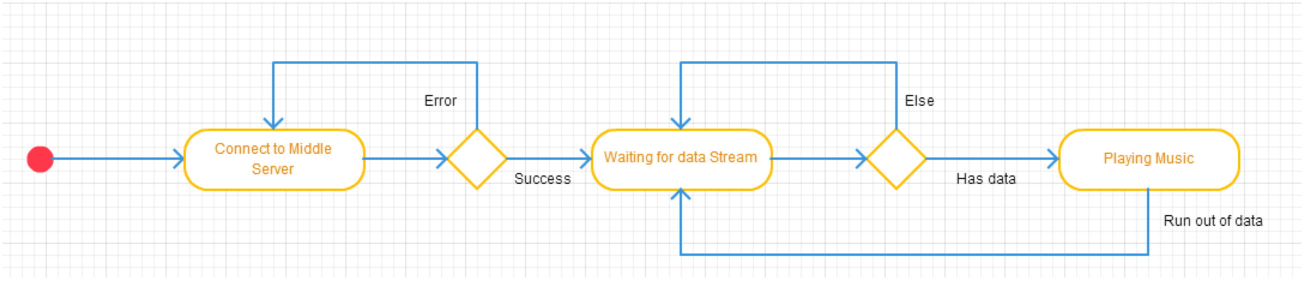
Middle Server



Sender

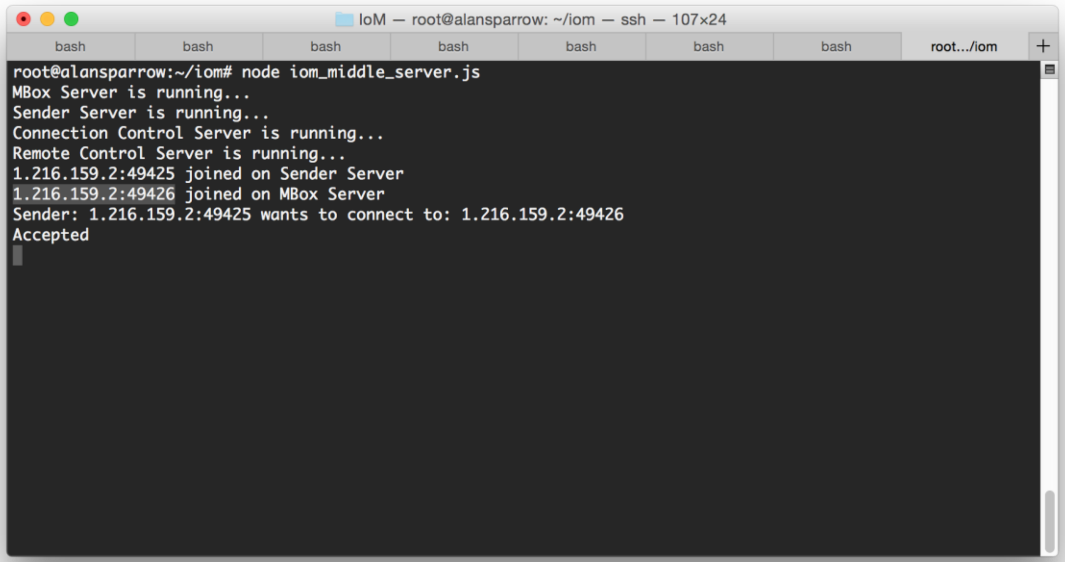


Remote Speaker



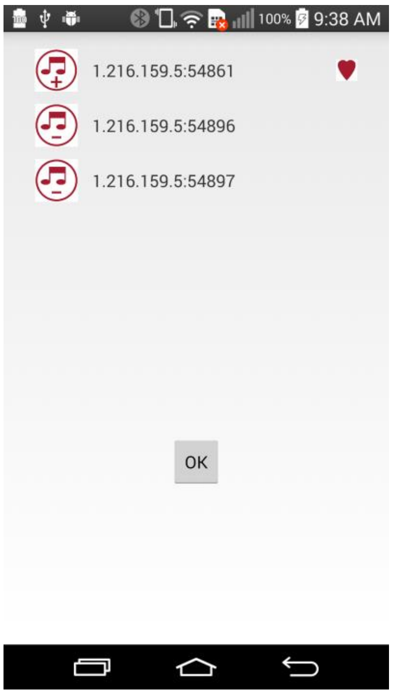
5. Screenshot

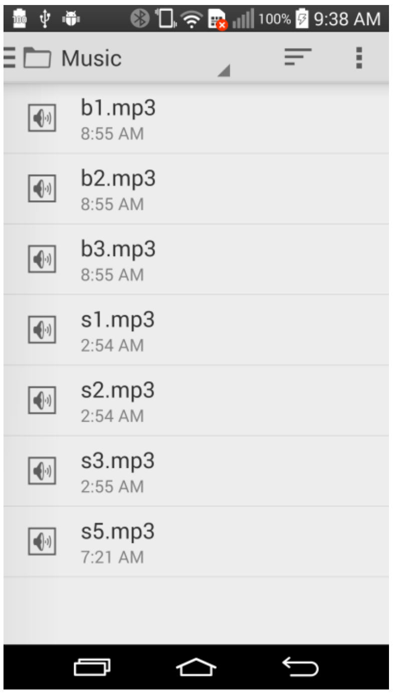
Middle Server

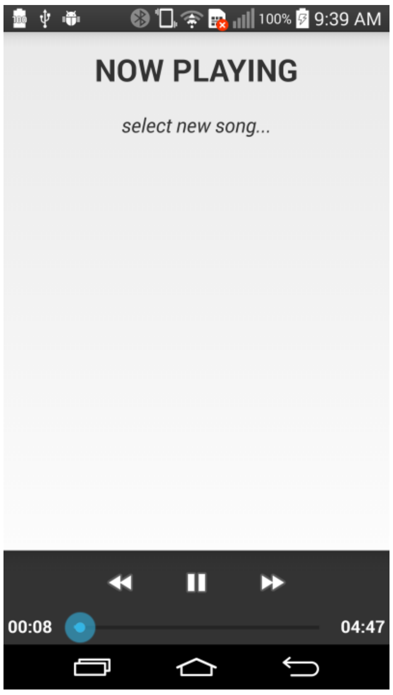


```
root@alansparrow:~/iom# node iom_middle_server.js
MBox Server is running...
Sender Server is running...
Connection Control Server is running...
Remote Control Server is running...
1.216.159.2:49425 joined on Sender Server
1.216.159.2:49426 joined on MBox Server
Sender: 1.216.159.2:49425 wants to connect to: 1.216.159.2:49426
Accepted
```

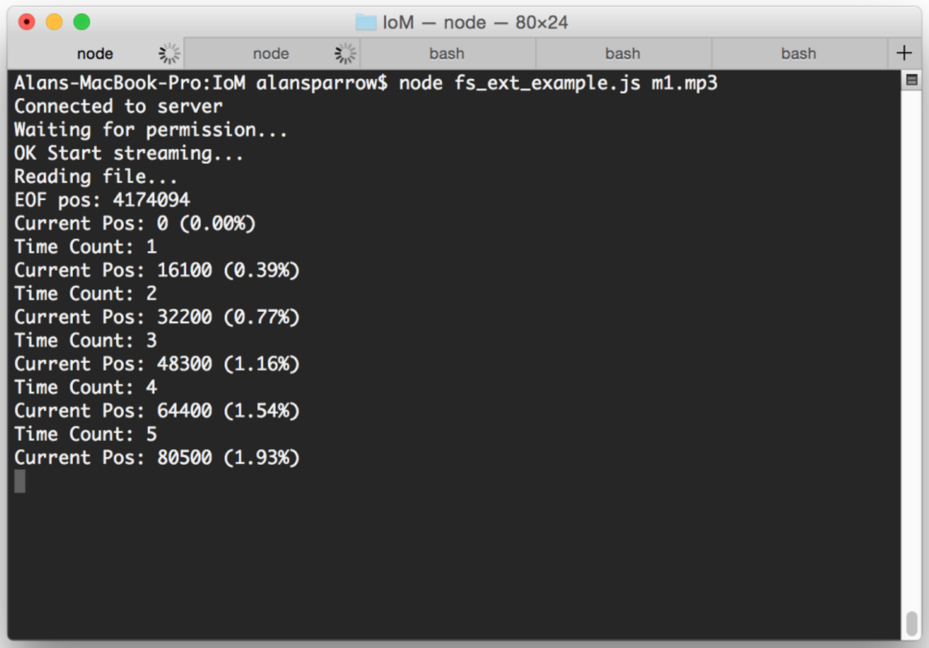
Sender





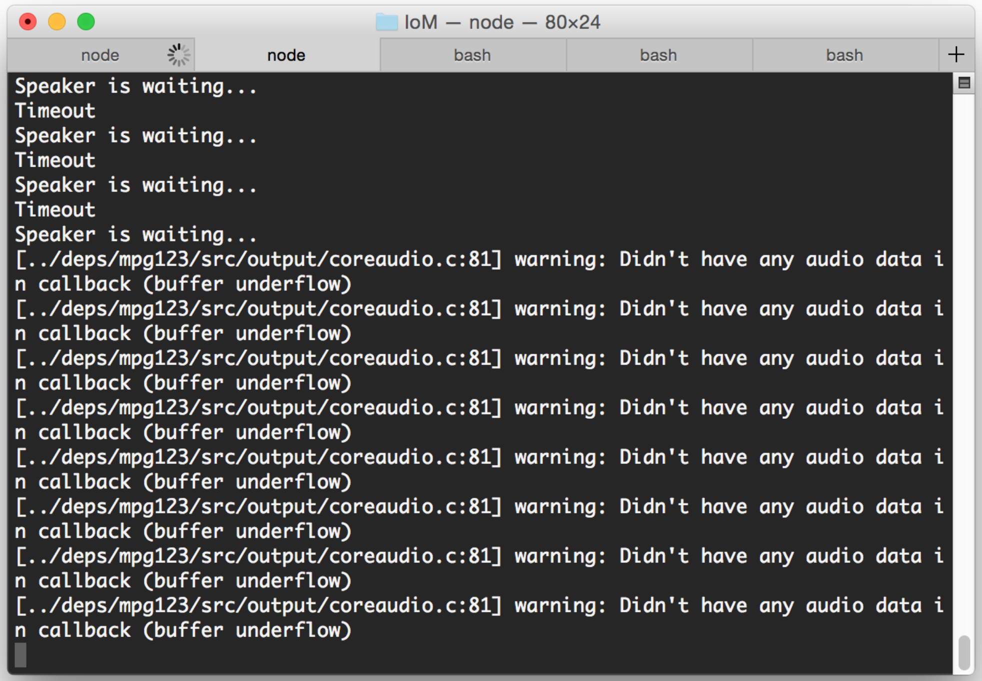


Bonus: Sender runs on PC



```
Alans-MacBook-Pro:IoM alansparrow$ node fs_ext_example.js m1.mp3
Connected to server
Waiting for permission...
OK Start streaming...
Reading file...
EOF pos: 4174094
Current Pos: 0 (0.00%)
Time Count: 1
Current Pos: 16100 (0.39%)
Time Count: 2
Current Pos: 32200 (0.77%)
Time Count: 3
Current Pos: 48300 (1.16%)
Time Count: 4
Current Pos: 64400 (1.54%)
Time Count: 5
Current Pos: 80500 (1.93%)
```

Speaker



```
Speaker is waiting...
Timeout
Speaker is waiting...
Timeout
Speaker is waiting...
Timeout
Speaker is waiting...
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
[../deps/mpg123/src/output/coreaudio.c:81] warning: Didn't have any audio data i
n callback (buffer underflow)
```