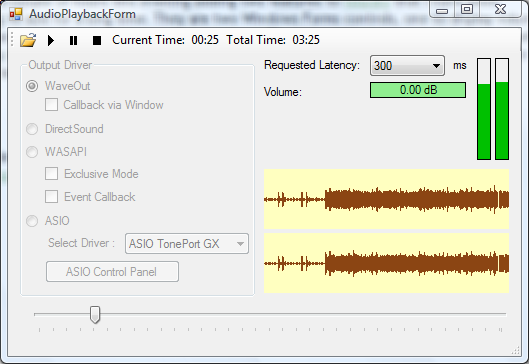
Xây dựng hệ thống streaming audio

1. Tìm hiểu về Naudio

NAudio là một thư viện MIDI âm thanh và MIDI mã nguồn mở, chứa hàng tá các lớp liên quan đến âm thanh hữu ích nhằm tăng tốc độ phát triển các tiện ích liên quan đến âm thanh trong .NET. Nó đã được phát triển từ năm 2002 và đã phát triển bao gồm nhiều tính năng khác nhau. Mặc dù một số phần của thư viện còn khá mới và chưa hoàn chỉnh, các tính năng hoàn thiện hơn đã trải qua thử nghiệm rộng rãi và có thể nhanh chóng được sử dụng để thêm khả năng âm thanh cho ứng dụng .NET hiện có. NAudio có thể nhanh chóng được thêm vào ứng dụng .NET của bạn bằng NuGet.



Hình 1 : Demo chương trình phát nhạc sử dụng naudio

Các tính năng của naudio:

*Chơi audio bằng nhiều api khác nhau như*

* + WaveOut
  + DirectSound
  + ASIO
  + WASAPI

*Đọc âm thanh từ nhiều định dạng tệp tiêu chuẩn*

* + WAV
  + AIFF
  + MP3 (using ACM, DMO or MFT)
  + G.711 mu-law and a-law
  + ADPCM, G.722, Speex (using NSpeex)
  + WMA, AAC, MP4 and more others with Media Foundation

*Ghi lại âm thanh bằng cách sử dụng WaveIn,*

*WASAPI hoặc ASIO Đọc và ghi tệp .*

*WAV Trộn và xử lý luồng âm thanh bằng công cụ trộn nổi 32 bit*

*Hỗ trợ mở rộng để đọc và ghi tệp MIDI*

*Mô hình sự kiện MIDI đầy đủ*

*Hỗ trợ cơ bản cho API Trình trộn*

*Windows Điều khiển biểu mẫu*

*Một số hiệu ứng âm thanh cơ bản, bao gồm nén nhạc*

Streaming trong Naudio

* Thiết kế tổng quan

Thiết kế cơ bản để streaming nhạc của chúng tôi là một luồng sẽ tải xuống âm thanh, trong khi một luồng khác sẽ phát đoạn tải xuống đó. Một trong những vấn đề này đặt ra là chủ đề nào nên thực hiện giải nén. Tôi đã chọn rằng các chủ đề tải về nên làm như vậy. Điều này làm mất đi luồng phát lại và do đó sẽ bảo vệ bổ sung chống phát lại, nhưng có một nhược điểm là nó đòi hỏi nhiều bộ nhớ hơn cho âm thanh được đệm vì nó được lưu trữ trong PCM.

* Bộ đệm audio

Các key class cho phép chúng ta xếp các mẫu thành một hàng đợi trên một luồng để phát lại trên một luồng khác được gọi là BufferedWaveProvider. BufferedWaveProvider thực hiện IWaveProvider để nó có thể được sử dụng làm đầu vào cho bất kỳ thiết bị Wave Out nào. Khi chức năng Read của nó được gọi, nó trả về bất kỳ byte nào được đệm, hoặc thay đổi bộ đệm phát lại nếu không có sẵn. Hàm AddSamples được gọi từ luồng của trình tải xuống để xếp hàng đợi các mẫu khi chúng có sẵn. Dưới mui xe, BufferedWaveProvider trong NAudio 1.4 đã sử dụng một hàng bộ đệm để lưu trữ dữ liệu. Đối với NAudio 1.5 tôi đang chuyển sang bộ đệm vòng an toàn. Điều này có nghĩa là bộ nhớ được phân bổ một lần lên trước và sẽ mang lại hiệu suất tốt hơn. Theo mặc định, nó đệm đủ cho năm giây âm thanh, nhưng điều này có thể dễ dàng thay đổi bằng cách sử dụng BufferDuration (mặc dù khi bạn bắt đầu đọc, kích thước bộ đệm được cố định). Nó cũng có ý nghĩa hơn để theo dõi số lượng byte được đệm (một phương thức của trình trợ giúp, BufferedDuration, biến điều này thành TimeSpan) thay vì số lượng bộ đệm được đưa vào hàng đợi.

* Phân tích frame trong mp3

Một vấn đề nếu mọi người nhanh chóng gặp phải khi cố streaming MP3 là lớp MP3Frame sẽ đưa ra một ngoại lệ nếu nó có thể đọc được đủ byte để tải đầy đủ khung MP3. Rõ ràng trong khi bạn phát trực tuyến, bạn cũng có thể có sẵn một nửa khung MP3. Giải pháp khá đơn giản. Chúng tôi gói luồng mạng của mình trong ReadFullyStream, đây chỉ là một lớp đơn giản kế thừa từ System.IO.Stream sẽ không trả về từ phương thức Read của nó cho đến khi nó đọc được số byte được yêu cầu hoặc luồng nguồn kết thúc. ReadFullyStream hiện đang nằm trong dự án NAudioDemo, nhưng có thể được kết hợp làm tiện ích trợ giúp cho NAudio 1.5

* Giải nén frame

Khi bạn đã phân tích một khung MP3 hoàn chỉnh, đã đến lúc giải nén nó để chúng tôi có thể đệm âm thanh PCM. NAudio bao gồm lựa chọn hai bộ giải nén khung MP3, một sử dụng codec ACM, có trên Windows XP trở đi và một để sử dụng bộ giải mã DMO (DirectX Media Object), có sẵn với Windows Vista trở đi. Hai lớp này là nội bộ với NAudio 1.4, nhưng sẽ được public cho NAudio 1.5. Ngoài ra, bạn có thể sử dụng bộ giải mã MP3 được quản lý hoàn toàn, chẳng hạn như NLayer.

* Giải nén định dạng

Một trong những vấn đề thiết kế khó giải quyết nhất là việc chúng tôi không biết WaveFormat mà BufferedWaveProvider nên xuất cho đến khi chúng tôi đọc được khung MP3 đầu tiên. Hầu hết các MP3 giải nén thành âm thanh nổi 44,1kHz, nhưng một số là 48kHz hoặc 22,1kHz, và một số khác là đơn âm. Do đó, chúng tôi có thể mở ra soundcard vì chúng tôi không biết định dạng PCM mà chúng tôi sẽ cung cấp.

Có hai cách bạn có thể giải quyết vấn đề này. Trước tiên, bạn có thể chọn định dạng PCM (ví dụ: âm thanh nổi 44,1kHz) mà bạn sẽ chuyển đổi các khung MP3 được giải nén thành ngay cả khi chúng có tốc độ mẫu hoặc số kênh khác nhau. Ngoài ra, như tôi đã chọn cho bản demo, bạn giữ lại từ việc tạo BufferedWaveProvider cho đến khi bạn nhận được khung MP3 đầu tiên và do đó biết bạn sẽ chuyển đổi định dạng nào. Trong dự án NAudioDemo, điều này được xử lý bởi một bộ đếm thời gian chạy trên biểu mẫu kiểm tra định kỳ để xem chúng ta đã có BufferedWaveProvider chưa. Khi BufferedWaveProvider xuất hiện, chúng tôi có thể khởi chạy trình phát WaveOut của mình.

* Tăng độ mượt khi streaming

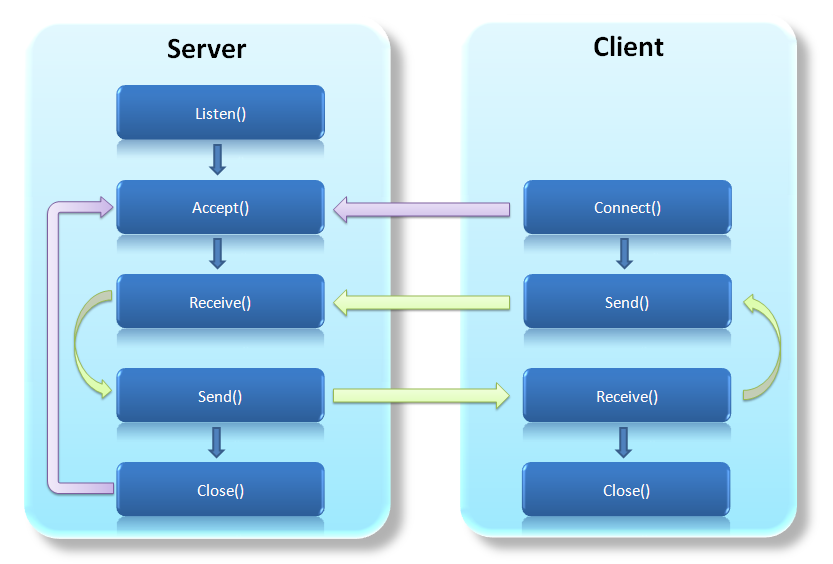
Một trong những vấn đề khi phát lại từ luồng mạng là chúng tôi có thể bị mất âm thanh nếu tệp MP3 không tải xuống đủ nhanh. Mặc dù BufferedWaveProvider sẽ liên tục cung cấp dữ liệu cho soundcard, nhưng nó có thể rất khó xử nếu bạn gặp tốc độ tải xuống chậm vì sẽ có những khoảng trống giữa mỗi khung hình được giải nén. Vì lý do này, một ý tưởng tốt là luồng phát lại tạm dừng khi âm thanh đệm đã hết, nhưng không bắt đầu phát lại cho đến khi có ít nhất vài giây âm thanh được đệm. Điều này được thực hiện trong NAudioDemo với bộ hẹn giờ đơn giản trên luồng GUI bắn bốn lần một giây và quyết định xem chúng ta có nên tạm dừng không.

* Kết thúc stream

Nếu bạn đang phát trực tuyến từ radio internet, thì không có kết thúc phát trực tuyến, bạn chỉ cần nghe cho đến khi bạn đã có đủ. Tuy nhiên, nếu bạn đang phát trực tuyến một tệp MP3, nó sẽ kết thúc vào một lúc nào đó. Hậu quả đầu tiên là nhà xây dựng MP3Frame của chúng tôi sẽ đưa ra ngoại lệ EndOfStream vì nó đã cố đọc một MP3Frame mới và đã đến cuối luồng. Xử lý việc này rất đơn giản - luồng trình tải xuống có thể nắm bắt ngoại lệ này và thoát, biết rằng chúng ta đã có mọi thứ. Tôi đang xem xét cải thiện API cho MP3Frame để bao gồm tùy chọn TryPude để bạn có thể giải quyết trường hợp này mà không cần phải bắt ngoại lệ. Vấn đề khác là luồng phát lại cần phải biết rằng đã kết thúc tập tin, nếu không nó sẽ tạm dừng, chờ thêm âm thanh được đệm sẽ không bao giờ đến. Một cách đơn giản để giải quyết vấn đề này là đặt cờ cho biết toàn bộ luồng đã được đọc và chúng ta không còn cần phải chuyển sang trạng thái đệm.

1. Tìm hiểu về lập trình mạng trong C#

Trong lập trình, Socket là một API (Application Programming Interface) cung cấp các phương thức để giao tiếp thông qua mạng. Trước khi bắt đầu tìm hiểu và viết một ví dụ đơn giản về socket, bạn có thể tham khảo bài viết “Networking – Một số khái niệm cơ bản“ để có cái nhìn sơ lược về những khái niệm cơ bản trong lập trình mạng.



Các lớp .Net cơ bản trong lập trình mạng

Các lớp này được cung cấp trong hai namespace System.Net và System.Net.Sockets. Hai namespace này chứa rất nhiều lớp dùng trong lập trình mạng, nhưng trong phạm vi bài viết ta chỉ quan tâm đến các lớp sau

|  |  |  |
| --- | --- | --- |
| **Class** | **Namespace** | **Desciption** |
| [IPAddress](http://msdn.microsoft.com/en-us/library/system.net.ipaddress.aspx) | [System.Net](http://msdn.microsoft.com/en-us/library/system.net.aspx) | Provides an Internet Protocol (IP) address. |
| [IPEndPoint](http://msdn.microsoft.com/en-us/library/system.net.ipEndpoint.aspx) | [System.Net](http://msdn.microsoft.com/en-us/library/system.net.aspx) | Represents a network endpoint as an IP address and a port number. |
| [TcpListener](http://msdn.microsoft.com/en-us/library/system.net.sockets.tcplistener.aspx) | [System.Net.Sockets](http://msdn.microsoft.com/en-us/library/system.net.sockets.aspx) | Listens for connections from TCP network clients. |
| [Socket](http://msdn.microsoft.com/en-us/library/system.net.sockets.socket.aspx) | [System.Net.Sockets](http://msdn.microsoft.com/en-us/library/system.net.sockets.aspx) | Implements the [Berkeley sockets](http://en.wikipedia.org/wiki/Berkeley_sockets) interface. |
| [TcpClient](http://msdn.microsoft.com/en-us/library/system.net.sockets.tcpclient.aspx) | [System.Net.Sockets](http://msdn.microsoft.com/en-us/library/system.net.sockets.aspx) | Provides client connections for TCP network services. |
| NetworkStream | [System.Net.Sockets](http://msdn.microsoft.com/en-us/library/system.net.sockets.aspx) | Provides the underlying stream of data for network access. |

Kết nối Server-Client với TCP/IP

Khi được chạy, server cần được xác định rõ địa chỉ IP và sẽ “lắng nghe” trên một port cụ thể. Server sẽ nằm trong trạng thái này cho đến khi client gửi đến một yêu cầu kết nối. Sau khi được server chấp nhận, một connection sẽ hình thành cho phép server và client giao tiếp với nhau.

Cụ thể hơn, các bước tiến hành trên server và client mà ta cần thực hiện sử dụng giao thức TCP/IP trong C# (có thể chạy server và client trên cùng một máy):

*Server:*

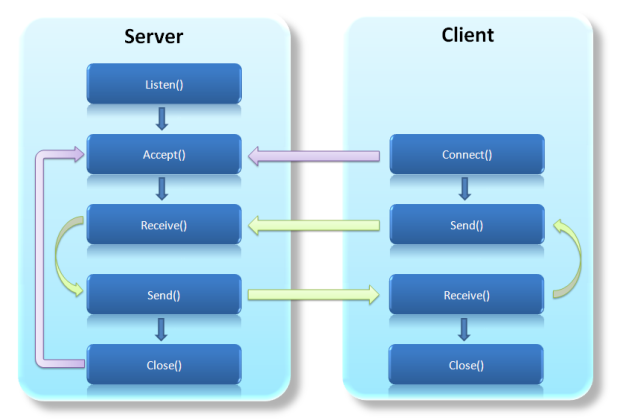
* Tạo một đối tượng System.Net.Sockets.TcpListener để bắt đầu “lắng nghe” trên một cổng cục bộ.
* Đợi và chấp nhận kết nối từ client với phương thức AccepSocket(). Phương thức này trả về một đối tượng System.Net.Sockets.Socket dùng để gửi và nhận dữ liệu.
* Thực hiện giao tiếp với client.
* Đóng Socket.

Thông thường quy trình này sẽ được đặt trong một vòng lặp (lặp lại bước 2) để chấp nhận nhiều kết nối cùng lúc (sử dụng Thread) hoặc các kết nối lần lượt.

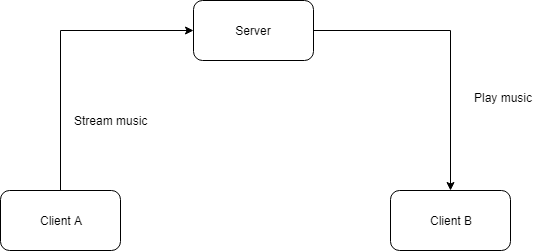
*Client*:

* Tạo một đối tượng System.Net.Sockets.TcpClient
* Kết nối đến server với địa chỉ và port xác định với phương thức TcpClient.Connect()
* Lấy luồng (stream) giao tiếp bằng phương thức TcpClient.GetStream().
* Thực hiện giao tiếp với server.
* Đóng luồng và socket.

Quy trình này có thể được minh họa theo mô hình sau:



1. Xây dựng hệ thống nghe nhạc qua mạng lan



Chương trình của chúng ta gồm 2 phần chính :

* Server : là trung tâm chung chuyển, giao tiếp chung của các client trong hệ thống , có trách nhiệm phát đi thông tin những ai đang phát nhạc trên toàn mạng
* Client : là các ứng dụng trong mạng lan , chịu trách nhiệm phát nhạc và streaming nhạc trên toàng hệ thống.

Server :

* Server chúng tôi sử dụng lớp TCP listener lắng nghe mọi tín hiệu trên cổng 9009
* Khi một client phát nhạc , nó sẽ gửi gói tin thông báo đang stream lên server
* Server thông báo với các client khác rằng có một client đang phát bài hát trên hệ thống

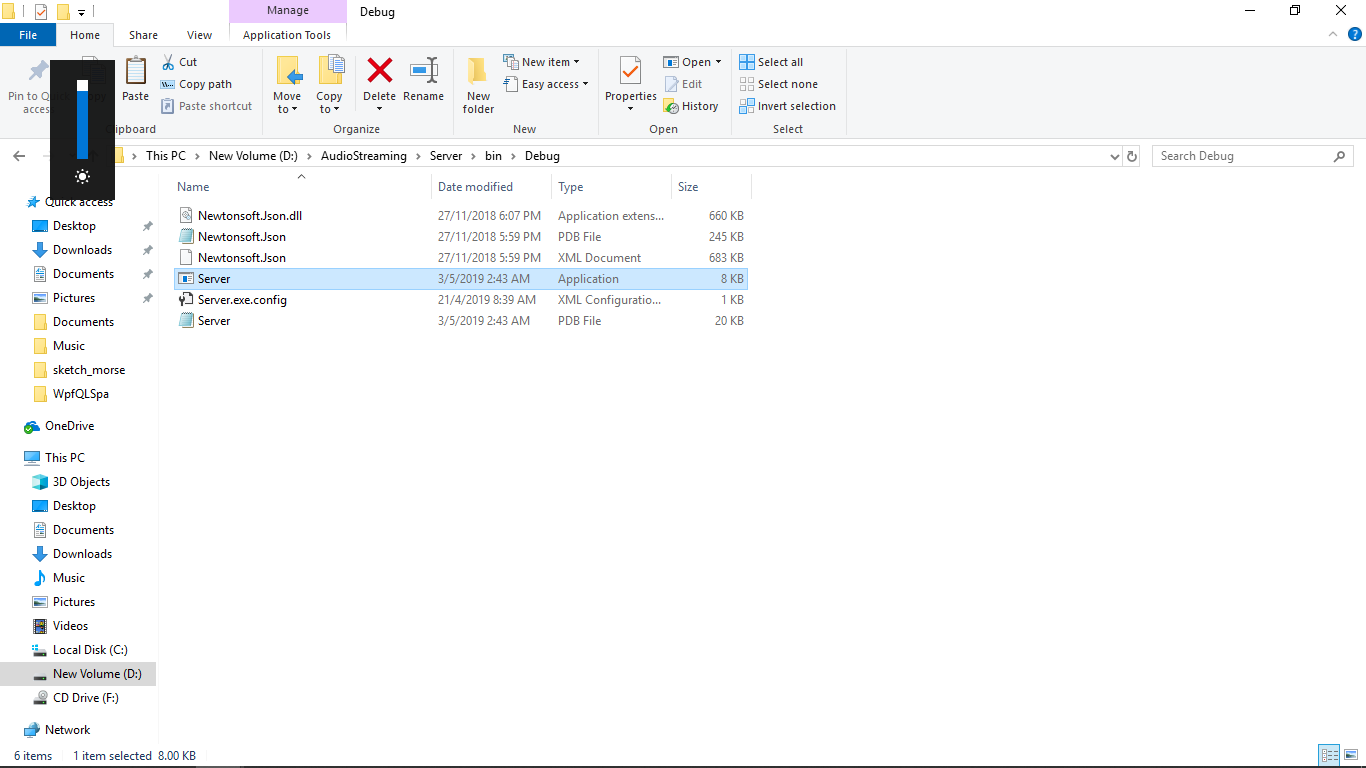
Client :

* Client có chức năng streaming âm nhạc trên hệ thống , khi client mở một bản nhạc n sẽ gửi gói tin phát nhạc tới server để server thông báo nhạc lên toàn hệ thống
* Khi client nhận thông báo từ server để biết client nào đang phát nhạc và vào nghe bài nhạc đó

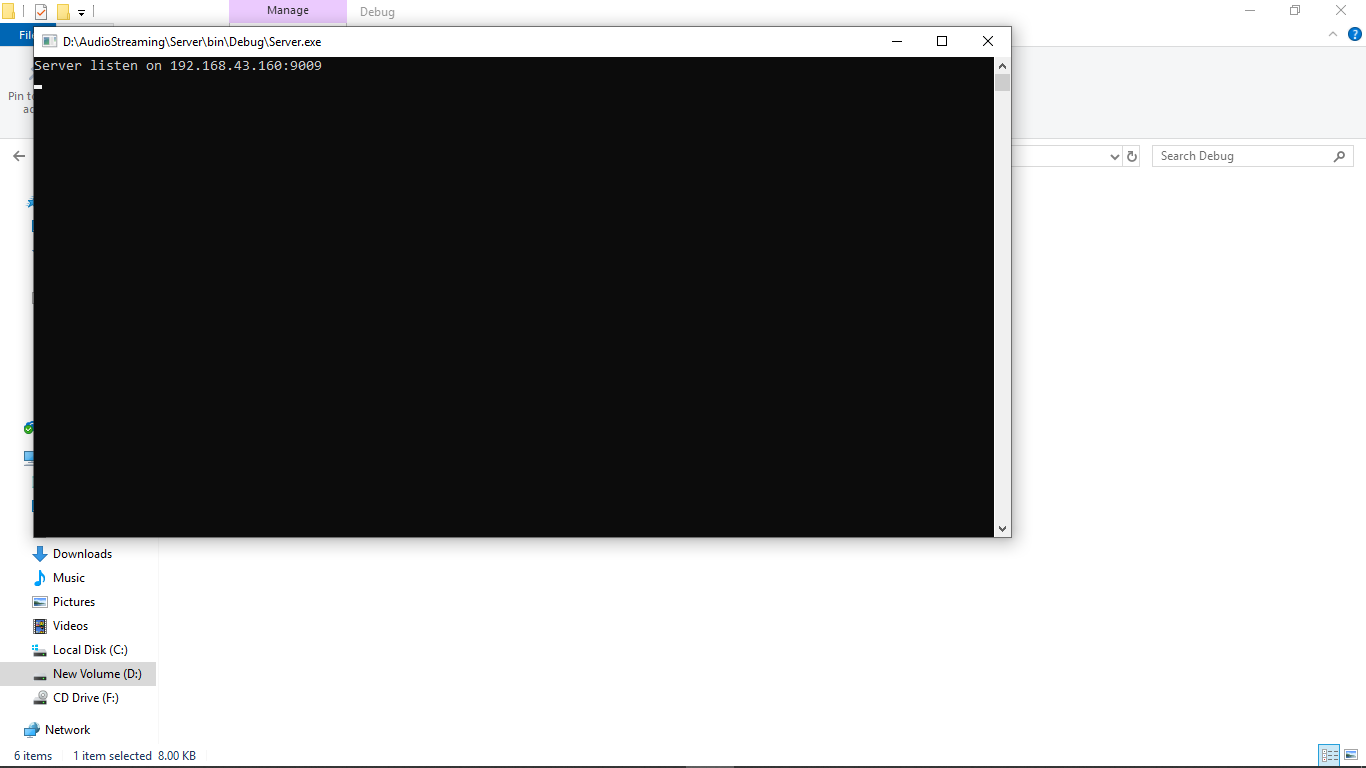
1. Hướng dẫn sử dụng chương trình

Chuẩn bị 2 máy tính

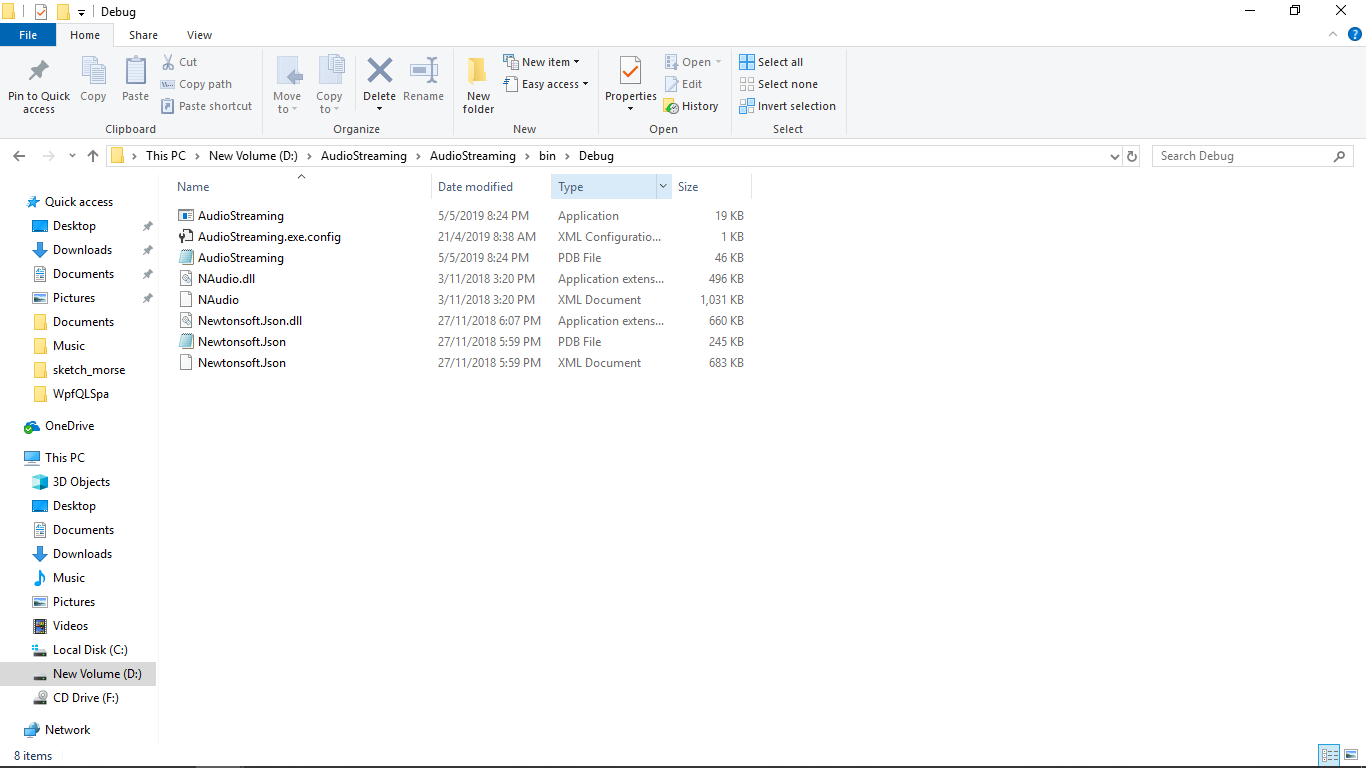
Bước 1 : Trên máy thứ nhất , mở chương trình Server trong folder Server/bin/Debug



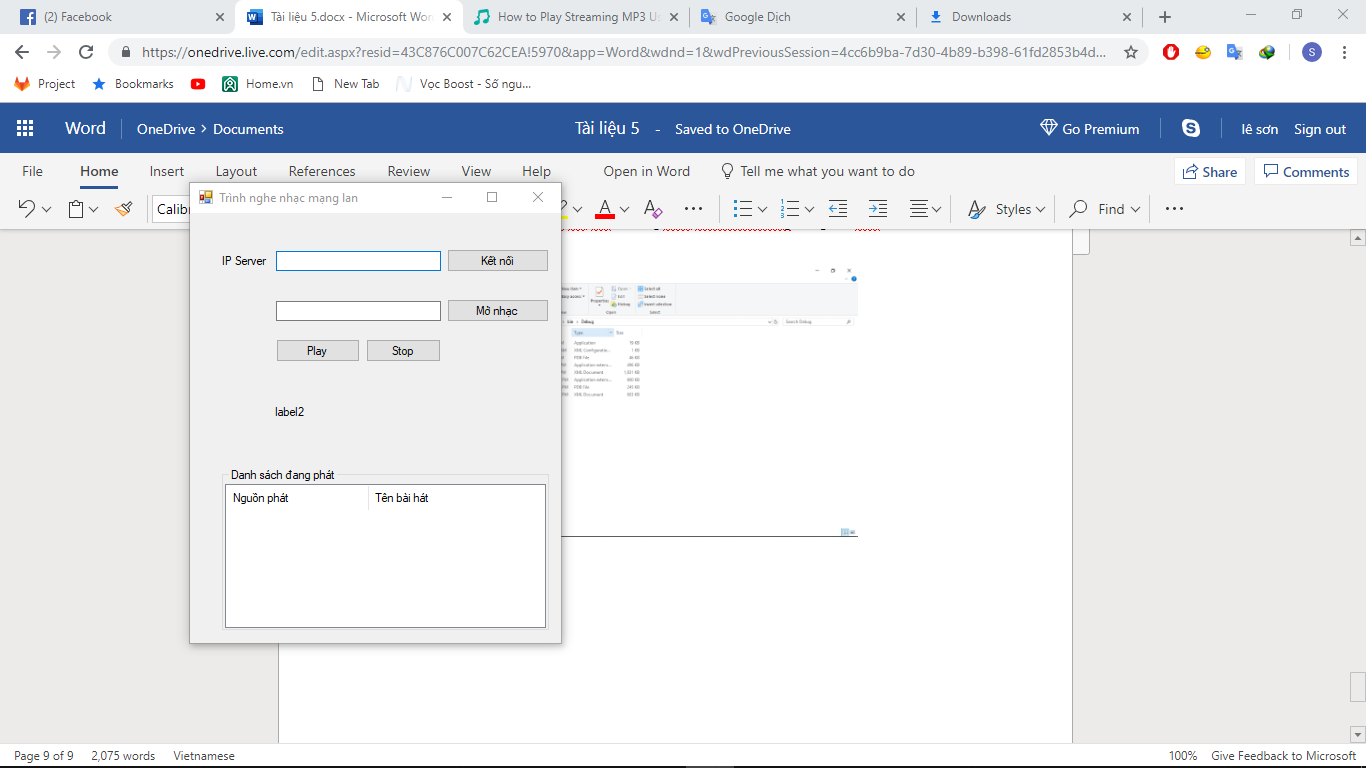
Ta được như sau



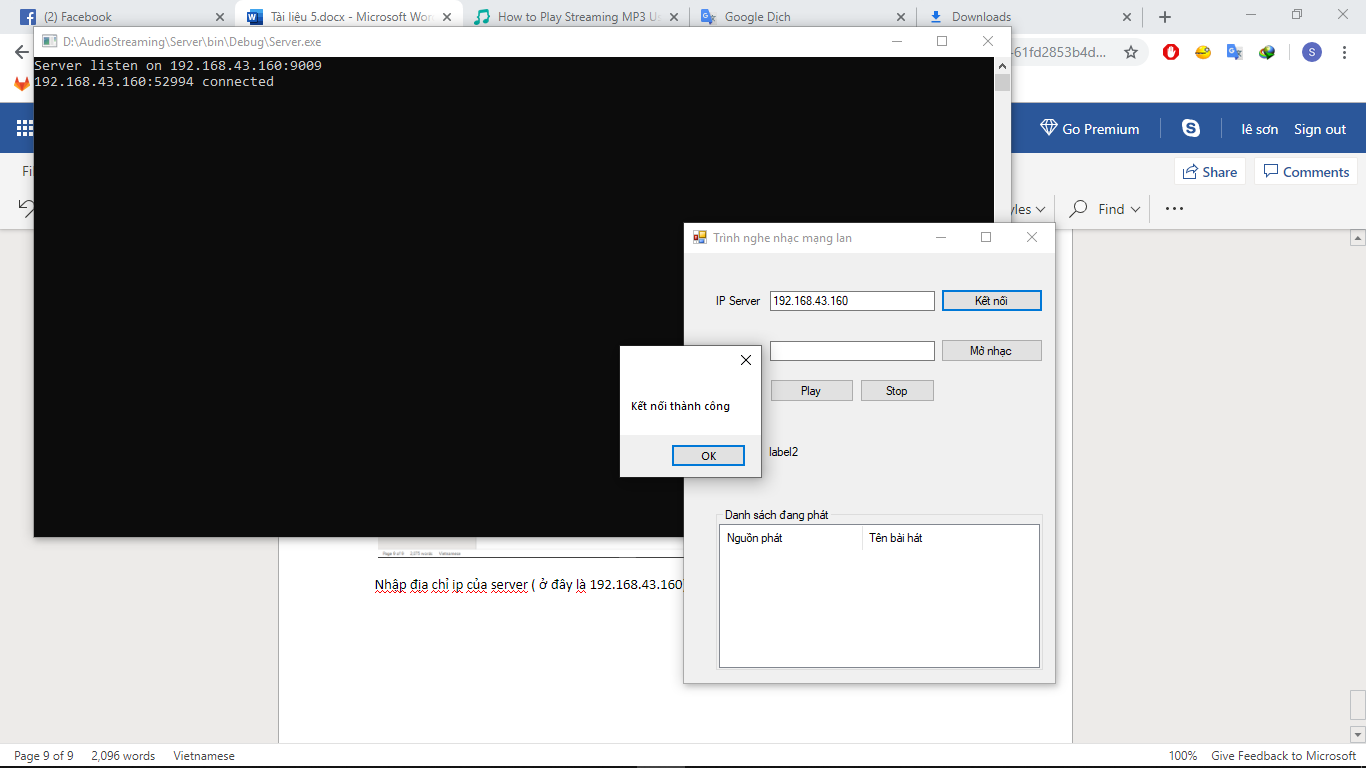
Bước 2 : Trên máy thứ nhất tiếp tục mở chương trình AudioStreaming trong thư mục AudioStreaming/bin/Debug



Ta được hình sau



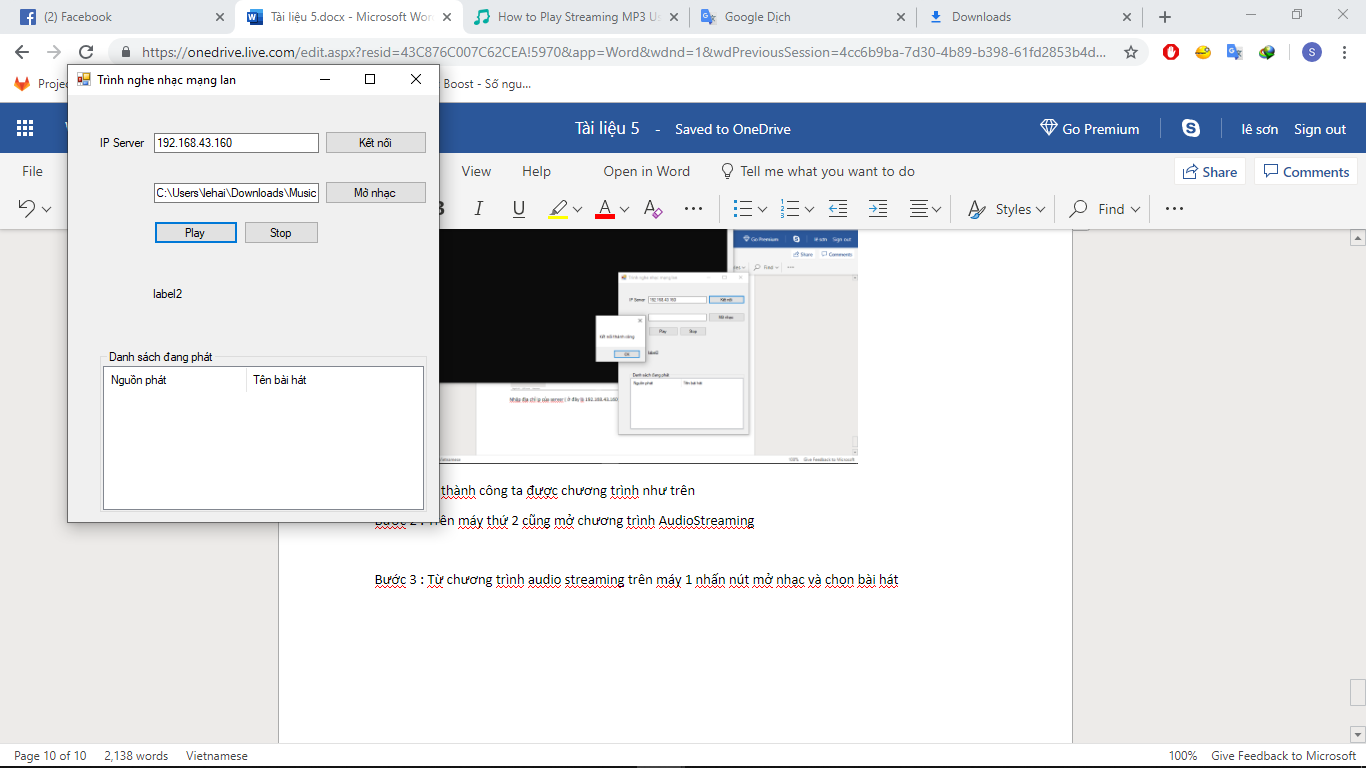
Nhập địa chỉ ip của server ( ở đây là 192.168.43.160) rồi nhấn nút kết nối , ta được kết quả



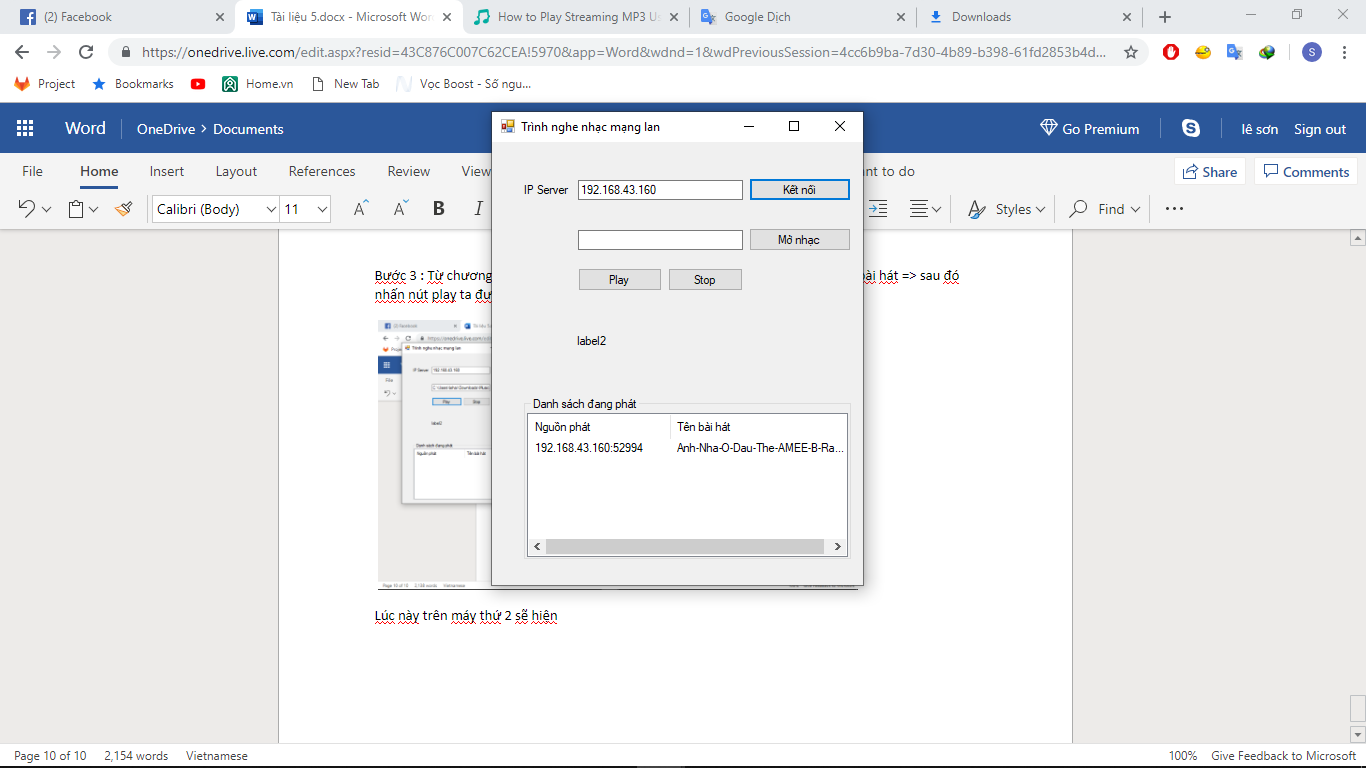
Khi kết nối thành công ta được chương trình như trên

Bước 2 : Trên máy thứ 2 cũng mở chương trình AudioStreaming

Bước 3 : Từ chương trình audio streaming trên máy 1 nhấn nút mở nhạc và chọn bài hát => sau đó nhấn nút play ta được

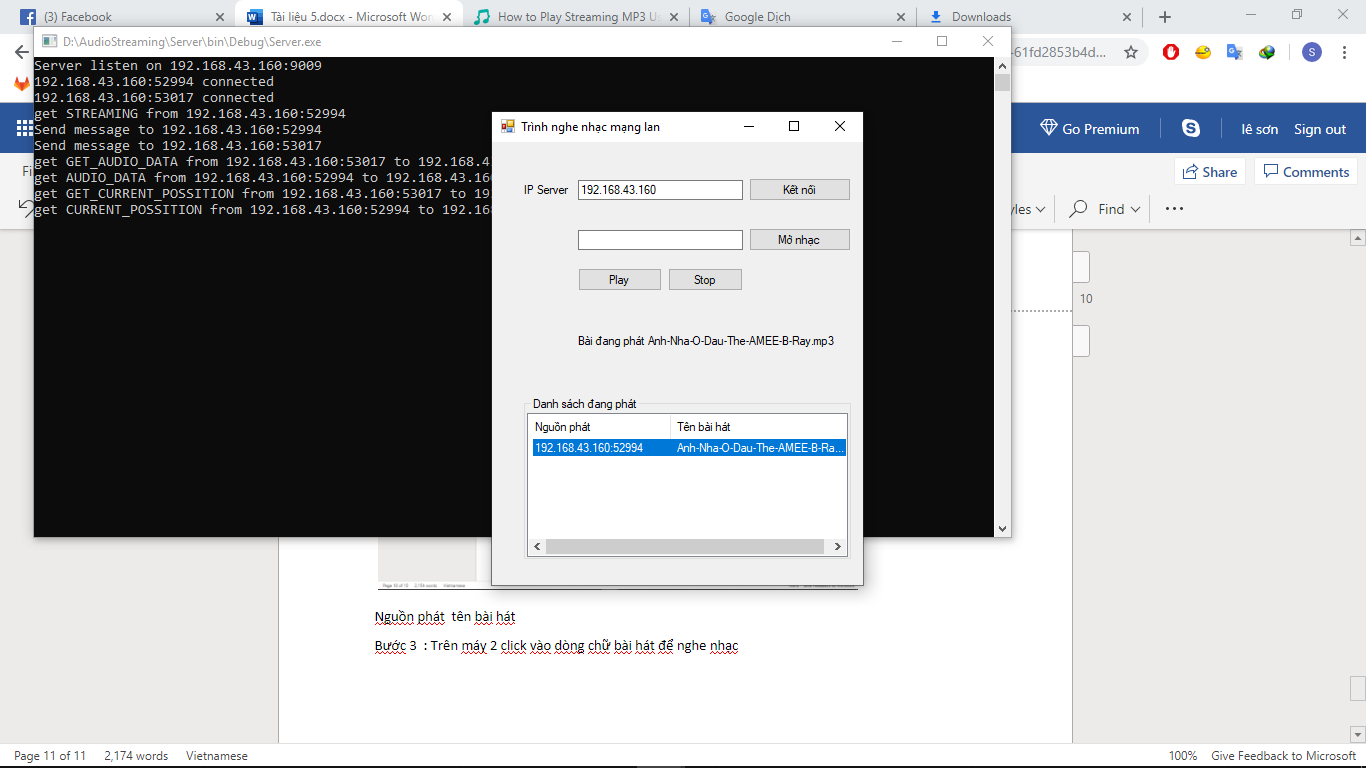


Lúc này trên máy thứ 2 sẽ hiện

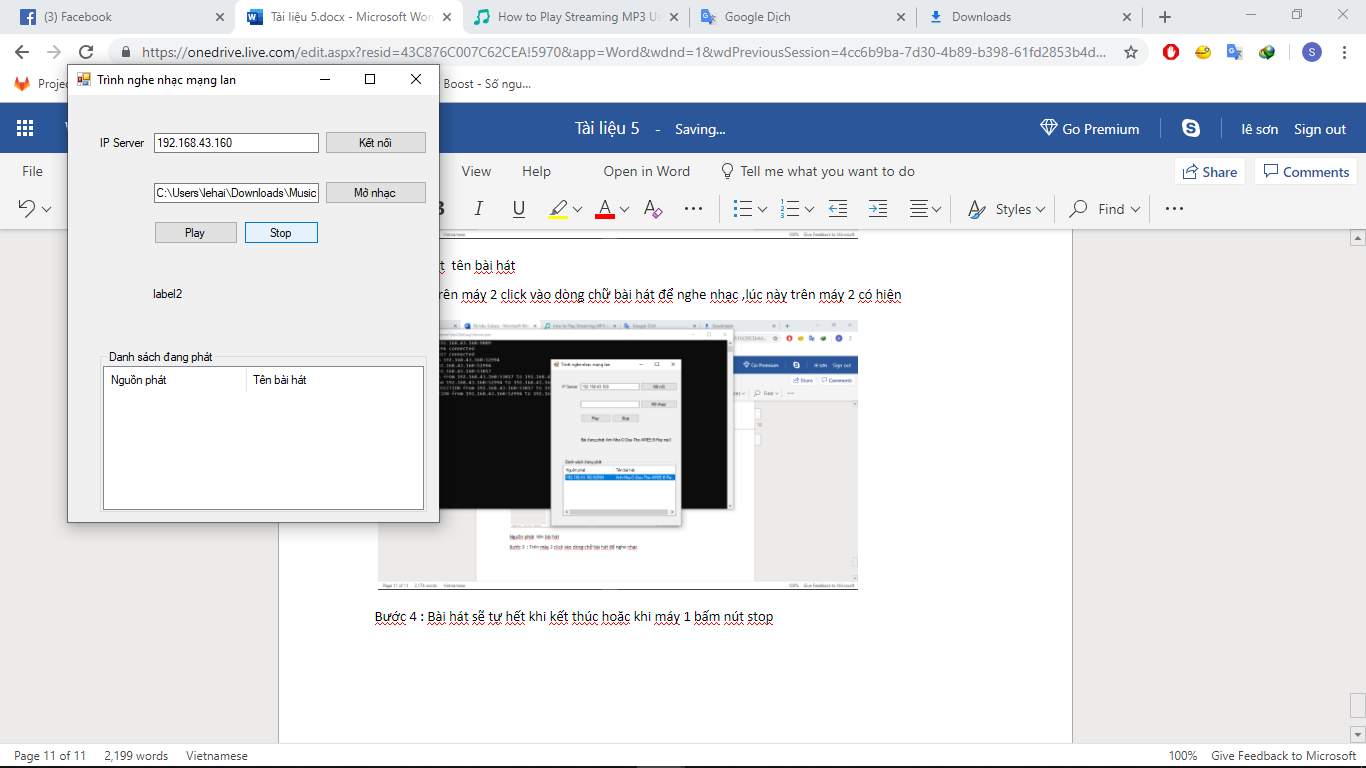


Nguồn phát tên bài hát

Bước 3 : Trên máy 2 click vào dòng chữ bài hát để nghe nhạc ,lúc này trên máy 2 có hiện



Bước 4 : Bài hát sẽ tự hết khi kết thúc hoặc khi máy 1 bấm nút stop



Lúc này trên máy 2 sẽ ko nghe nhạc nữa và sẽ không thấy xuất hiện dòng bài hát nào cả do máy 1 đã dừng nguồn phát

