Concert for Computer Network / One Bit Motion Mix

Guide for software installation and use

Andreja Andric

2019

The project

One Bit Dance, One Bit Motion Mix and Concert for Computer Network are different titles for a family of musical works which are at the same time a composition cycle, a piece of computer software and an improvisation practice. Concert for Computer Network is a version of the work for a networked laptop ensemble. One Bit Motion Mix is a solo work for laptop or smartphone, based on the same software without the network part. One Bit Dance refers to a series of sound files created with an offline version of the software.

For details about the music and how to perform, see the Performance Guide. This Guide covers the installation and use of software for laptop solo and for networked laptop ensemble. For smartphone solo version, refer to the documentation on the following web page: http://andrejaandric.altervista.org/OneBitMotionMix.html

The laptop software

The software for network performance consists in two parts: Client and Server. In one performance there can be exactly one Server and up to 10 Clients. The network is structured as shown in Figure 1 below. In the case of a solo laptop performance, the Server program is not necessary, while the Client program works independently. The role of the Client program is to play the music and provide an interface for the performer's input. The role of the Server is to aggregate all the inputs, merge them together and send the combined result to each Client. The merging of melodic fragments is done like mixing stacks of cards. The compound melody played by all the Clients consists of: first notes from each melodic fragment in order of enrollment; afterwards, second notes for each melodic fragment, and so on. If there are differences in lengths of melodies, then shorter melodies are repeated until the length of the longest melody is reached. The compound melody has therefore the length of the longest individual melodic fragment multiplied by the number of performers.

The software is optimized for working on long distance networks as well, so it is possible to have performers who perform live and who are not at the same location or even in the same country. It is therefore also possible to give a simultaneous concert on multiple location around the world.

In a performance, all the Clients will emit the exact same sound, so only one should be connected to a sound amplification system. The music flow will be synchronized at all times

and eventual network delays will not cause interruptions in sound. However, the sound output from different clients will not be possible to synchronize exactly. This will result in blurry sound if more laptops were to be connected to the sound system. If, by chance, the sound outputs are exactly synchronized, then they will be exactly the same, and then again there is no point in having more laptops connected to the sound system. If a simultaneous concert at multiple locations takes place, then one laptop per space should be amplified and the others should be muted.

Each performer runs the Client software on their laptop. If all the performers are on the same local network (like in the same space), then the Server program can be run on one of the performers' laptops. If the performers are in different locations, then the Server program has to be run on a public server. I run the Server program on a UNIX server at sdf.org. (http://www.sdf.org).

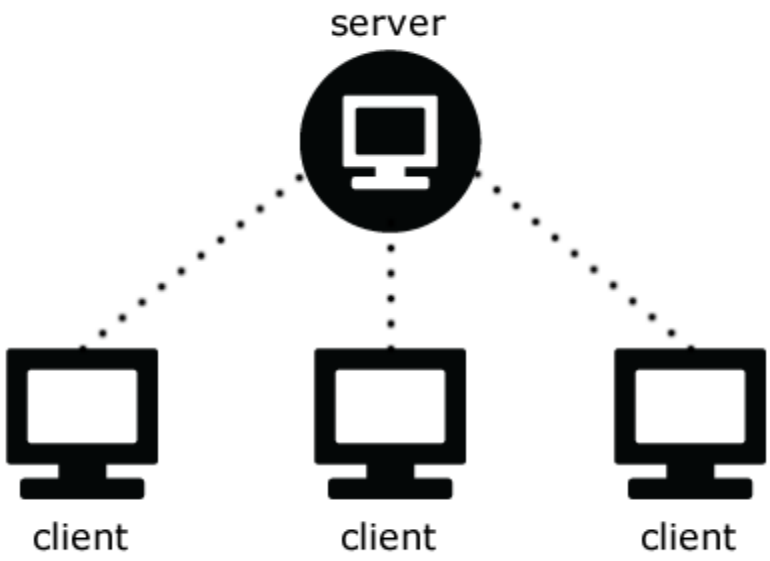

Figure 1. Client-Server architecture of One Bit Net Dance

The Server program

The Server program is implemented in two different versions to accommodate for different capabilities that the server computer may have. Namely, if you are using a public server to run your Server code, then you might have limited control on what you can install there or what options may be offered to you for running your program. Both versions are fully compatible with each other and the only difference is that one is implemented in Java and the other in C.

The Java version of the Server program requires Java Runtime Environment. Download it and

install it from here: https://java.com/en/download/. Verify that it has been installed correctly by opening Terminal Window (MacOS) or Command Prompt (Windows) and typing java. The command should display a list of instructions. If this is not possible to do (because perhaps you don't have sufficient rights on the Server to install Java yourself), consider using C version as described further below.

The Java version of the Server program consists of the following three files:

```
NetDanceServer.class
```

```
NetDanceServerThread.class
```

```
ServerConfiguration.txt
```

The class files contain executable code of the Server for JRE. File ServerConfiguration.txt contains the port number that the server will use to wait for connections from Client programs.

The second parameter is the name of the Master of Ceremonies (see the Performance Guide). If you do not wish to specify the Master of Ceremonies, then put dash ('-') in this place. If Master of Ceremonies is specified then only that person can change the global parameters of music. Otherwise everyone can do it. The name specified here has to be the same as the name specified in the ClientConfiguration.txt file of the player who is to be the Master of Ceremonies.

Example content of this file is:

```
4444
```

```
Andreja
```

Run the Server program from the Terminal Window (MacOS) or Command Prompt (Windows) with the following command:

```
java NetDanceServer
```

If your Server computer does not support Java and you cannot install it, then use the C implementation. Upload the source code NetDanceServer.c to your Server machine and build it with the following command:

```
gcc NetDanceServer.c -lpthread -o NetDanceServer.
```

Run it with:

```
./NetDanceServer
```

When configured correctly and run locally, the Server appears as in the screenshot on Figure 2 below.

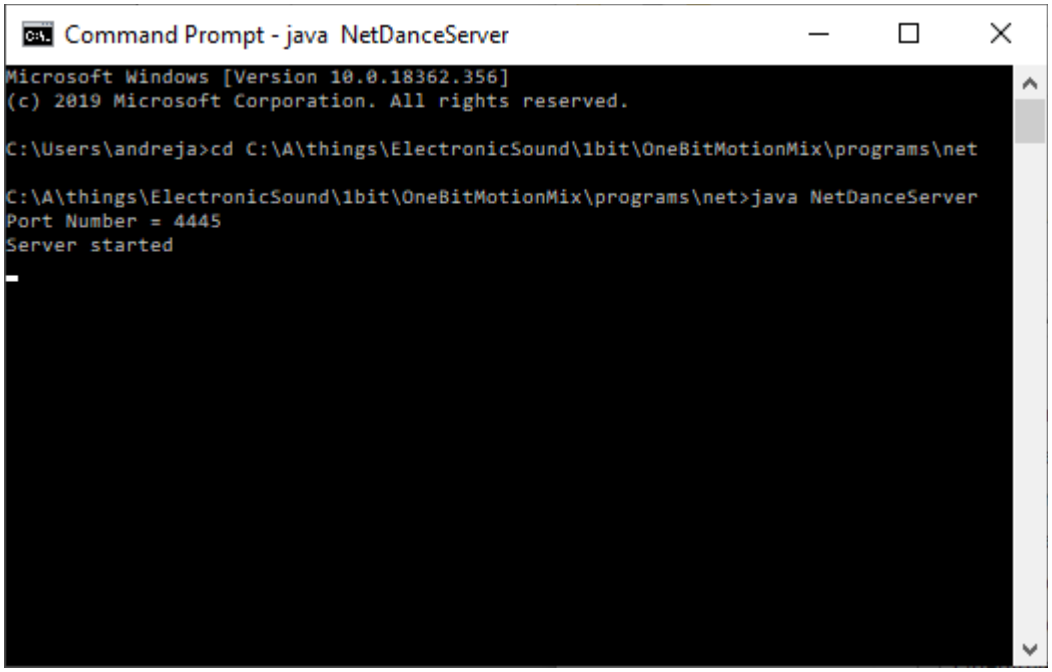

Figure 2. Running the server

The C version of the Server program run on a UNIX server looks as shown in Figure 3.

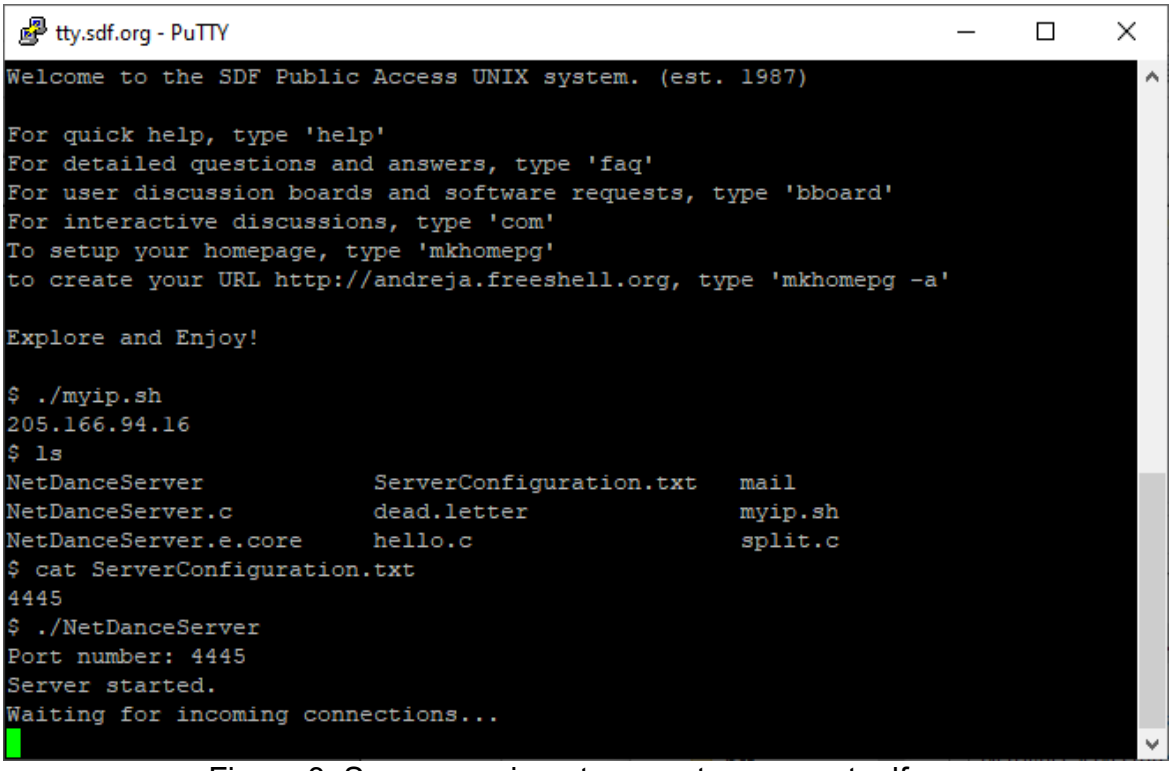

Figure 3. Server running at a remote server at sdf.org

In both cases, when no clients are connected, the Server will wait for connections. When

some clients have connected and afterwards disconnected, it will exit automatically after the exit of the last client. If that doesn't happen you'll have to stop the program manually with Ctrl-C.

The Client program

The Client program exists only in Java version. It also requires Java environment to work. It consists of the following files:

```
OneBitNetDance$1.class
OneBitNetDance$ColorShowTimerTask.class
OneBitNetDance$CompoundMelodyWindow$1.class
OneBitNetDance$CompoundMelodyWindow$2.class
OneBitNetDance$CompoundMelodyWindow.class
OneBitNetDance$EveryPlayerMelodyWindow.class
OneBitNetDance$MagickSquareWindow.class
OneBitNetDance$NetDanceListener.class
OneBitNetDance$States.class
OneBitNetDance$StringPair.class
OneBitNetDance.class
ClientConfiguration.txt
input0.txt
```

As in the Server program above, the class files contain executable code of the Client for JRE. The ClientConfiguration.txt contains the configuration for the Client program. input0.txt is the input file that will be explained in the next sections

Client Configuration

Typical content of this file is:

```
192.168.87.104:4444
Andreja
Add Muted, Log, Verbose, Show or SoundMax n in separate rows below to
```

```
have additional options
```

The first line contains name or IP address of the Server, followed by a Port Number separated by a semicolon ':'. The port number must match the one stated in the ServerConfiguration file mentioned above. The name or IP address must match that of the computer where the Server program is running. To discover the IP address of the server run the following command on the server:

```
dig @resolver1.opendns.com ANY myip.opendns.com +short
```

On a Windows server you can also run

```
ipconfig
```

For a solo performance without a server, start the server name with a minus sign ('-'). This indicates to the Client that Server is not needed. Also if the server name or IP is stated without the minus sign, but the Server is not found, then the Client program will continue in "solo performance" mode, without the Server.

The second line of the client configuration file contains an alias or nickname. This is free for you to choose. Every performer should have a different nickname. The nicknames will appear in the window that displays the current status of all the performers, as we will show later.

The next rows can contain any or all of the following commands:

`Muted` - this will cause the Client to not emit any sound. This command has to be applied on all the laptops that are not connected to the AP system, otherwise they will also emit sound through their speaker, and that is undesired.

`Verbose` - this is used to have an additional message in the Terminal window appear whenever a note has an overflow. This will be explained in the next section.

`Show` - this causes the Client software to display an additional window with colorful show generated to accompany the music.

`Log` - this causes the Client software to record the sound into a file and the current status into another file so the music can be reproduced both as a performance and as a recording.

`SoundMax n` - Command SoundMax followed by a number between 0 and 32767 sets the maximum volume. If the command is omitted, the value of this is 32767 - the maximum. This command is useful if you rehearse using Skype where the same speaker is shared by both Skype and the Client application. Namely scaling the volume down makes it possible to use both this software and Skype at the same time effectively, because your voice can be heard over the sound of the music. If this is not set, the sound from the Client software is much louder than the voice on Skype.

Input file

Typical input file has a filename input0.txt and looks like this:

```
MainPitch: 110
Duration: 64
Length: 200
DecayRate: 3
ScaleType: 6
PolyphonyType: 2/1 1/1

0 4 13 4 7 4 12 4
```

The file is in plain text format and consists of two parts. First is a list of numerical values, each preceded by a caption, and then another list of numbers without any captions.

`MainPitch` - frequency in Hz of the note 0.

`Duration` - time in milliseconds between two subsequent notes

`Length` - time in milliseconds between the start of a note and its end

`DecayRate` - degree to which a note is decaying after attack.

`ScaleType` - number of harmonics used in constructing the scale. Only 6 and 10 are supported as a values. Value 6 creates a scale described below with 7 notes per octave. Value 10 creates a scale with 13 notes per octve

`PolyphonyType` - a list of usually two fractions, though there could be 1 or 3 or more in rare cases. The number of fractions indicates polyphony level: if there are two fractions then there are two voices. Three fractions create three voices. The music flow is rendered simply in as many voices as there are fractions, in parallel fourths, fifths, or octaves, or in contrary motion (if a fraction is negative), in short, equal rhythmical values.

The remaining numbers after the fractions are the tone phrase stated in numbers 0 to 14. See the Performance guide for the information on how to form a tone phrase out of numbers. This phrase is individual for every performer, while the above 6 arguments including the list of fractions for PolyphonyType are shared. This means that when a performer changes one of the six parameters above, it changes for every performer. When a performer changes a tone in the phrase, other performers are unaffected, but the compound phrase created by the server out of all the phrases will be changed.

Starting the Client program

Open the Terminal window (Mac) or Command Prompt (Windows) and position yourself in the directory with the software by typing cd with the path where the Client program is saved. For example:

```
cd C:\A\things\ElectronicSound\1bit\OneBitMotionMix\programs\net
```

Start the program, as shown in Figure 4 with command

```
java OneBitNetDance
```

Alternatively, you can run *RunClient_PC.cmd* batch (on Windows) or run the executable file OneBitNetDance.class from the context menu (on Mac).

If it is a solo performance without the Server, then you should open the input0.txt in a text editor of your choice (Figure 5).

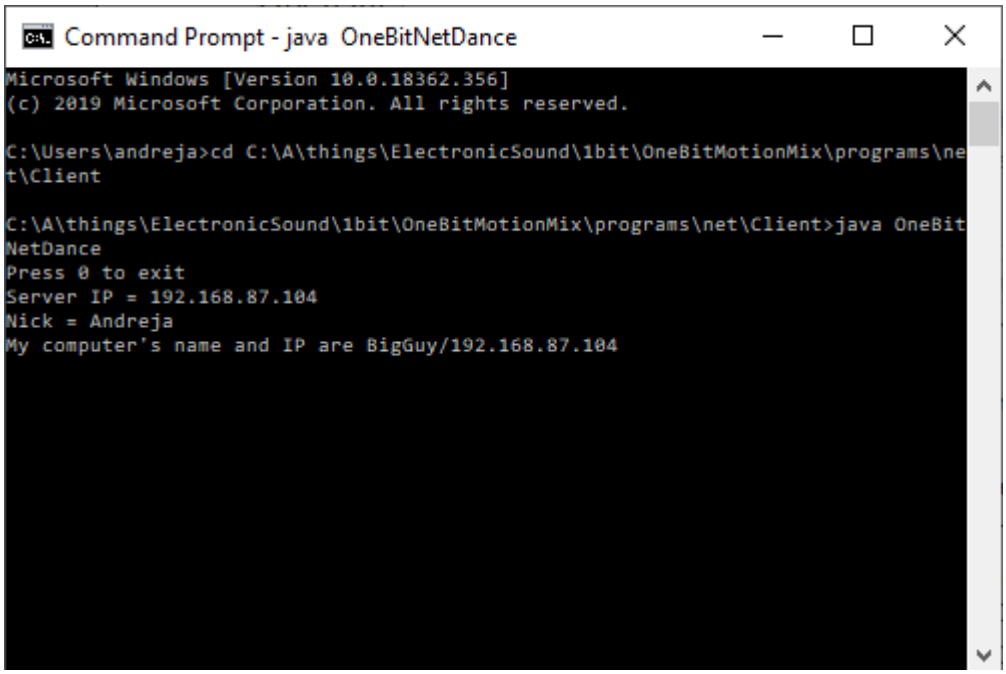

Figure 4. Running the Client program

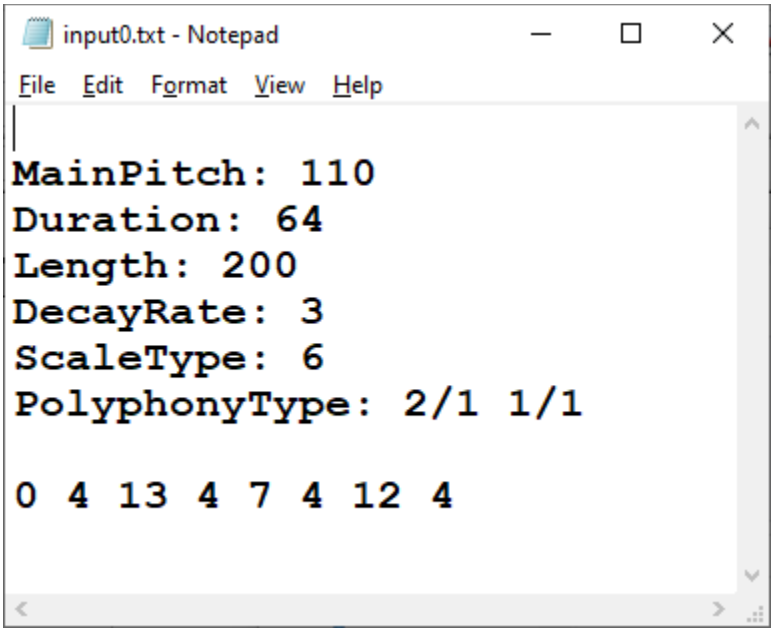

Figure 5: input0.txt in Notepad in solo performance

In a networked performance, two windows will appear on each Client automatically, as soon as you start the Client program. One window is a collaborative, custom built text editor, shown in Figure 6. In this text editor, the global parameters (first six lines on Figure 6 below) can be modified by any performer and the change will take effect immediately in all the Client programs of all the performers.

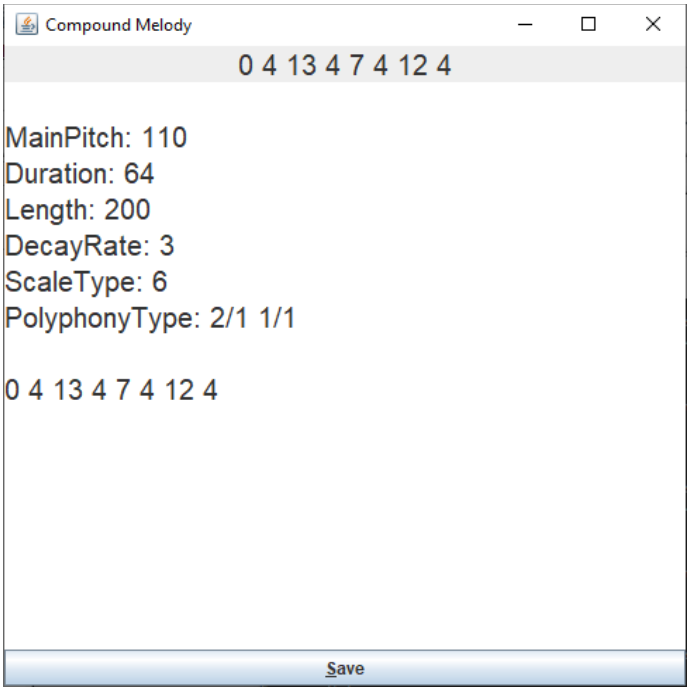

Figure 6 Custom editor for networked performance

The top line shows the compound melody line, assembled from the fragments from all the performers. The bottom line shows the individual melodic fragment for the current performer and this cannot be changed by other performers. This is the reason why you cannot use any other text editor to interact with the program in a networked performance - because this editor is collaborative.

The second window shows a list of individual melodic fragments for each performer followed by the six global parameters (Figure 7). This window only displays information and cannot be edited.

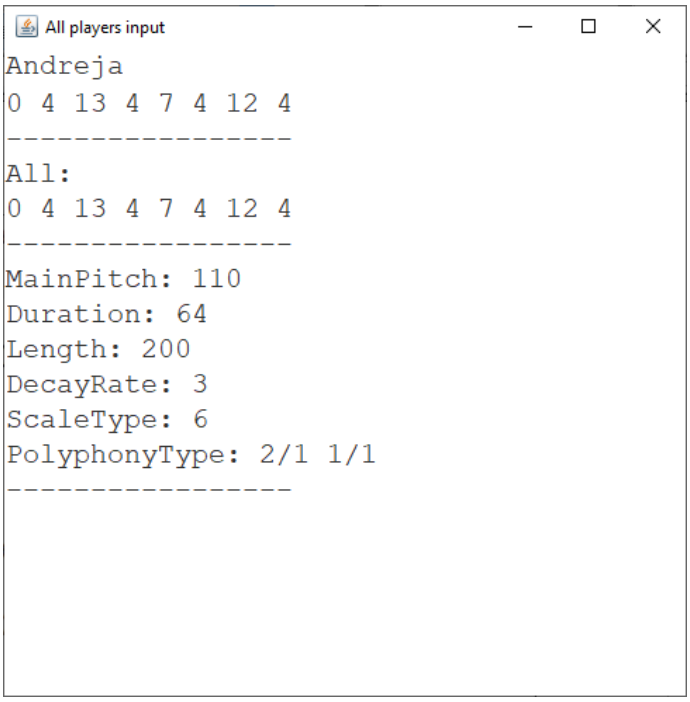

Figure 7. List of performers' inputs

In both solo and networked performance, you can stop the Client program by typing 0 in the Command Prompt / Terminal window (Figure 8), or by pressing the closing button on either the first or the second window described above. If, however, you are recording the session with the Log command (see Client Configuration section above), then do use the first way (pressing 0) because proper closure of the file is not implemented when you close the program with the closing button.

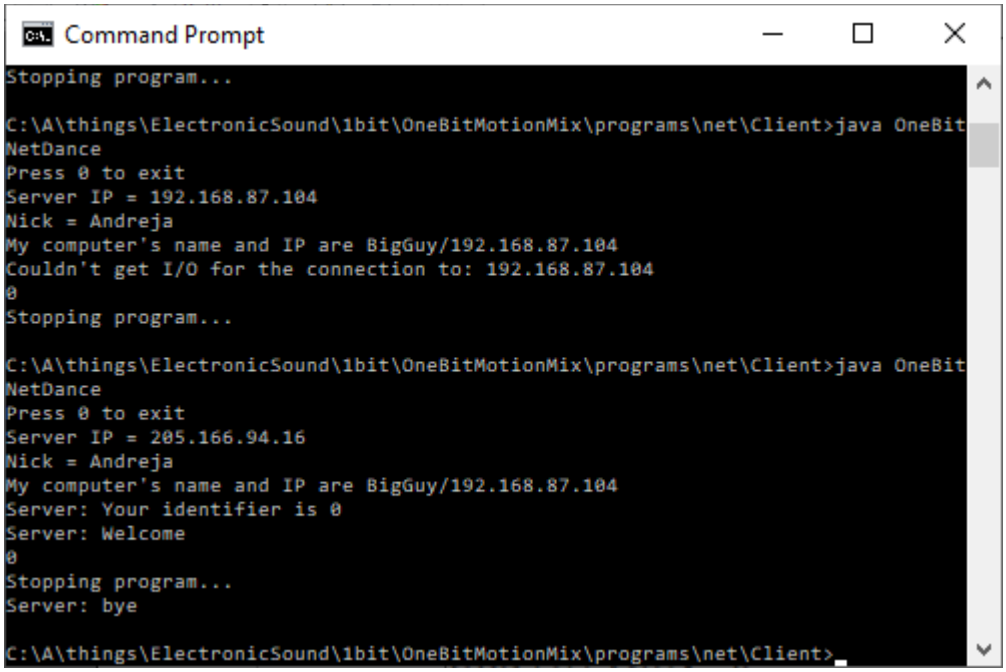

Figure 8. Stopping the Client program

Modifications in the text editor are not taken into account by the Client program until you press Save button or Save menu item. Keyboard shortcuts Command-S on Mac computers and Alt-S and Ctrl-S on Windows can also be used. The modified tone phrase starts playing only after the unmodified one has been played to the end. If the phrase is very short and the network is slow, it may take a couple of additional repetitions of it before the change becomes audible.

The order of Installation and starting the programs

First install and configure the Server program as described above. Then install and configure all the Client programs each on one performer's laptop. Start the Server first, then the Clients. When done performing stop each client with 0 in the Terminal Window / Command Prompt command line. The Server will stop automatically when all the Clients exited, but if that doesn't happen you can stop it manually with Ctrl-C.

For further information contact: andreja.andric@gmail.com