Proceedings of the International Conference on Live Coding, Valdivia, December 2021
https://iclc.toplap.org/2021/

Synerg(e)ia: A Networked Collaborative Live Coding Environment

Vasilis Agiomyrgianakis
Faculty of Music & Audiovisual Arts, Department of Audiovisual Arts Ionian University
vagiomyr@ionio.gr

Iason Svoronos- Kanavas
Faculty of Music & Audiovisual Arts, Department of Ethnomusicology Ionian University
jason.skk98@gmail.com

Iannis Zannos
Faculty of Music & Audiovisual Arts, Ionian University, Department of Audiovisual Arts, Ionian University
iani@ionio.gr

Giorgos Diapoulis
Interaction Design, Department of Computer Science & Engineering, Chalmers University of Technology, University of Gothenburg
geodia@chalmers.se

ABSTRACT

Synerg(e)ia (Συνέργεια) is an experimental project for networked collaborative live coding practice using Emacs. The goal of this study is to facilitate remote collaboration among live coders and deliver an attractive interface that offers code sharing and audio synthesizing on-the-fly. We build an environment that provides necessary features of networked collaborative live coding namely as time sharing, code sharing, access control, communication facilitation. Synergia relies entirely on sharing code and executing it on each performer's system in order to produce audio. Synergia is based on PIVPN technology for server-client communication that runs on a Raspberry Pi 3. VPN offers an easy and secure access for the users to connect to the server. Additionally, Synergia uses open-source software such as tmux terminal multiplexer, Emacs, and OSCGroups. It also uses a variant version of sc-hacks-redux OscGroups class in SuperCollider to manage and send messages to client's SuperCollider programming environment. While there are numerous environments for collaborative coding, Emacs offers an attractive alternative for an experienced user/programmer. There is a broad variety of customisation features that may be implemented on Emacs. Also, many computer music languages, like SuperCollider, Tidal, Foxdot, Impromptu and more, support Emacs connectivity. Getting advantage of these features and functionality our system may be used for reproducible research.

1. INTRODUCTION

In the latest years, networked collaborative live coding has been used as a mean to perform and practice on-the-fly computer music and computer graphics. In networked live coding users can collaborate in real-time to form shared experiences which may emerge during performance practice and to explore new ways to communicate during performance practices. There are various examples such as, SuperCopair (de Carvalho Junior, Lee & Essl 2015), Troop (Kirkbride, 2017), Extramuros by (Ogborn, Tsabary, Jarvis, Cardenas & McLean 2015), flok[1], and Estuary (Ogborn & Beverley 2017) among others. Like the above examples, Synergia is based on the idea of creating an environment that relies on classification criteria for remote network live coding performances such as those suggested by Barbosa (2003) and Lee & Essl (2014). Networked live coding (NLC) may be either synchronous, for instance, peer-to-peer interaction between several users using a peer-to-peer solution such as Wright's (2005) Open Sound Control (OSC) and OSCGroups by Bencina (2018), or asynchronous, for instance, exchanging MIDI files through e-mail (Barbosa, 2003) in the temporal domain, whereas NLC practices may be either remote or co-located in the spatial domain. Some other features of NLC are code sharing, access control and communication facilitation. In our work we offer a synchronous interaction remotely. Additionally, there is a little research in remote live coding that the users can collaborate using the same live coding interface. For instance, in Troop, flok, Estuary among others, the performers share the same code in the same window and they can make minor alternations to the environment such as changing theme or colour. However, in Synergia it is offered remote collaboration from

1https://flok.clic.cf/

an interface in which any user can make its own customisations. Moreover, Synergia's interface is easily accessible. To access the interface you have to connect to the server via SSH using a VPN client. The server is cross-platform and it is easily accessible from any location (remotely). Our prototype server runs on a Raspberry Pi 3 model under Linux Debian in which an interface for collaborative Live Coding based on Emacs editor (Stallman, 1981) has been implemented. In Synergia all users have access to each others' live coding workspace. Both code and audio synthesizing are produced on each users' local SuperCollider language side (sclang). This creates an alternative method for broadcasting sound that is based on synthesizing the audio locally through OSC communication as opposed to streaming audio between clients. In our opinion this model provides synchronous almost real-time interactions between performers.

2. SYSTEM ARCHITECTURE

Synergia utilizes the PIVPN[2] tool to set up a VPN server with a static IP address on a single Raspberry Pi. It's also critical that it creates VPN profiles with unique IP addresses that can be used by clients to connect to the Synergia interface and consequently to the OSCGroups server. The VPN server listens on the IP address with this format (10.8.0.1) and accepts requests from 10.8.0.0/254. When clients connect to the server, they can access the interface by using SSH from their terminal. Moreover, Synergia runs Tmux terminal multiplexer to deal with concurrent processes such as OSCGroups and Emacs server-client procedures[3]. Any client that can access the Synergia interface can send a code fragment from the server to the connected nodes (clients) of local sclang via Open Sound Control (OSC). Synergia communicates with SuperCollider clients using OSCGroups, which allows the clients to transmit and receive data via peer-to-peer messaging. In Synergia a modified version of the SuperCollider OscGroups class[4] is also running in order to maintain multiple IP addresses from VPN clients connected to the OSCGroups server. The client needs also to install OSCGroups class into their machine so to send and receive messages. Wherever a live coder evaluates a SuperCollider code in Synergia interface, the OSCGroups server sends OSC messages to the clients' local SuperCollider programming language.

3. INTERFACE DESIGN

The concept of Synergia is to motivate people to improvise and to create their layouts using the Emacs editor. Synergia's Emacs configuration comes with SuperCollider (McCartney, 2002), OSCGroups (Bencina, 2018) by default. A future version is expected to include more live coding environments, such as ixi lang (Magnusson, 2009) and Tidal Cycles (McLean, 2010). In Synergia environment all the members of the ensemble can use the same live coding environment where every user can generate music. Furthermore, given that Synergia is configurable and extensible, this may give rise to community engagement which may encourage collaborative practices. For instance, on Emacs it is very easy to make git commits, which may facilitate code maintenance and code sharing. Additionally, Synergia is simple to replicate for reproducible research because it operates on a low budget solution, that is a Raspberry Pi, and its image is open-source and available on a Github repository.

3.1 USER INTERACTION

With Emacs server-client mode it is possible to run many SuperCollider workspaces for collaborative live coding where new files are opened in a running instance of Emacs. Synergia has a default design with SuperCollider workspaces for the users to do live coding and this can be extended to include more live coding Emacs modes. The environment is designed in such a way to provide to all users access and permissions such as read, write and execute code on other users workspaces. In that way, direct communication like having a chat room it is also provided. All users can watch each others' code and activity during the performance. The idea in Synergia is that all performers can share the same text buffer, they can create new or they can use the existed SuperCollider workspace buffers. For instance, any user can create windows and design their appearance onto the screen. They can split windows vertically or horizontally to watch other's users workspaces during the performance. Additionally, users can alternate the

2https://pivpn.io/

3https://www.emacswiki.org/emacs/EmacsClient

4https://github.com/iani/sc-hacks-redux/

appearance of their live coding session by customising emacs variables such as fonts, colours, and other attributes of the text by using control commands (C-x or C-c). Concurrent evaluation with Emacs control commands can result in unforeseen problems and crashes. The control key executes various functions in Emacs such as screen motion, buffer visibility, multiple windows, line execution, etc. This can cause unwanted Emacs behaviour and thus interrupt the live coding session. To address this issue, we made some modifications on the keyboard mapping. We have bind SuperCollider's line and block code evaluation to the Shift key instead of Control key. Thus, one option was to bind Shift-2 keyboard shortcut for line evaluation and Shift-5 for code block evaluation. Those key combinations are not based on Emacs control commands like the default control command C-c/C-c for line evaluation in SuperCollider mode. Figure 1 shows the Synergia interface with two SuperCollider workspaces during a Live Coding session[5]. Figure 2 shows some of the processes running during a live coding session in the Synergia client. The Synergia OscGroups client binary and SuperCollider workspace along with the SuperCollider post window are shown respectively.

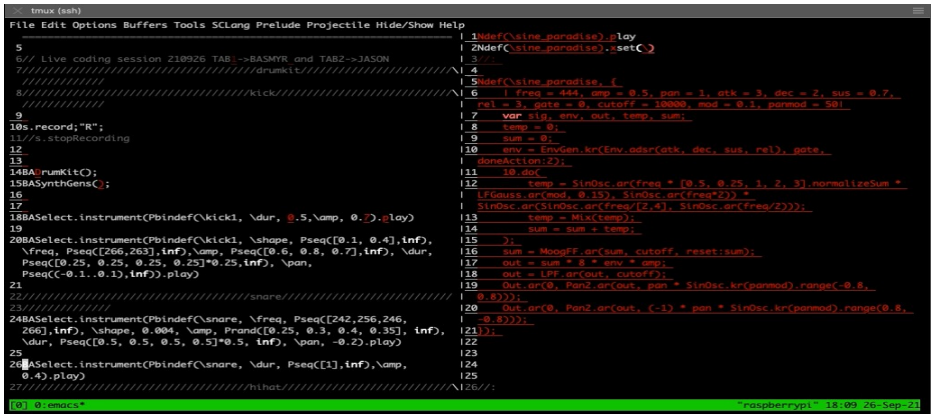

Figure 1. Example of the Synergia interface during a live coding session.

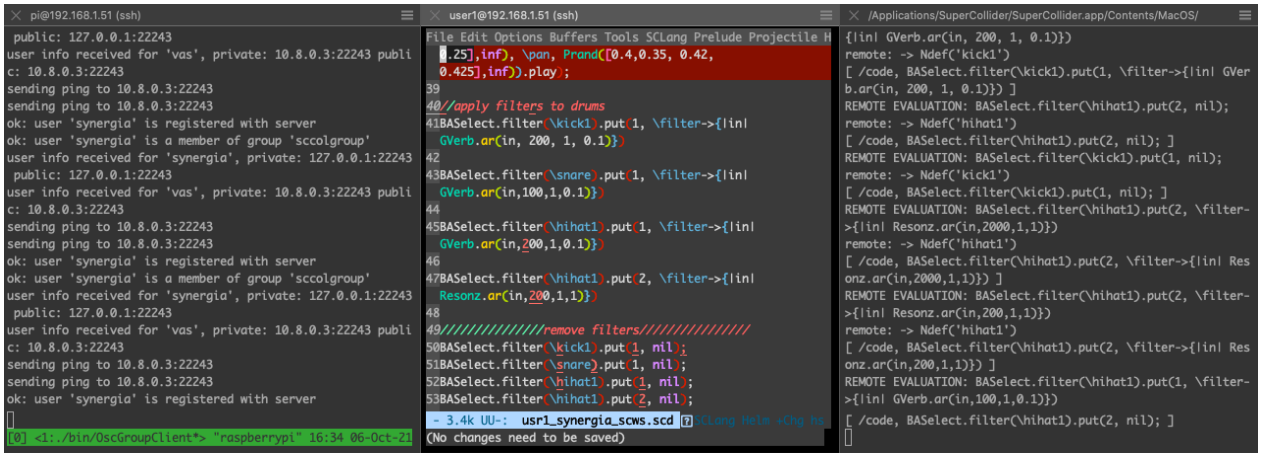

Figure 2. From left to right: Synergia OSCGroups, Synergia SuperCollider workspace, SuperCollider post window.

Moreover, Synergia contains SuperCollider classes for live coding such as BA Live coding classes[6] (Agiomyrgianakis, 2021). Those classes offer an easy way for a beginner to deal with sound and patterns and generally with live coding practices in SuperCollider. BA live coding classes also contain a clock mechanism for synchronisation purposes during live coding. To be able to run BA classes to your SuperCollider it is required to install those classes to your machine. Thus, in Synergia if the members of a live coding session

5Youtube video with Synergia live coding session: https://youtu.be/-0LVr4dkgdE

6https://github.com/Vasileios/BA

equipped with the same SuperCollider setup they can listen to the same audio output during the performance.

3.2 ACCESSING SYNERGIA

To access Synergia you need to have a VPN client service application such Open VPN and to clone the Synergia GitHub repository[7] into your computer. It is also required for the clients having SuperCollider installed into their machine. The Synergia GitHub source code includes the user manual. Automatic and manual connections to the Synergia server are available. Shell scripts can be found in the Synergeia GitHub repository that can automate the procedure. For example, 'usr1_oscgroups_sclang' shell script will run OscGroupClient and SuperCollider on your machine so to receive OSC messages from the server. With this configuration the connected clients can listen to a live coding session without needing to login to the Synergia interface via SSH. That means that there is a way for the audience to listen to live coding sessions with the minimum of effort. Additionally, live coders can login to Synergia interface via SSH by running the command *ssh user1@10.8.0.1* in which 'user1' is a user in Synergia server and *10.8.0.1* the static internet protocol (IP) of the server provided by PIVPN.

4. DISCUSSION

As mentioned in the introduction, Synergia is being developed through the prism of certain basic principles of a remote live coding platform as stated by Lee & Essl (2014). At the same time, we are focused on free software and compatibility. Finally, the ability of the user to customise the platform in any way. Therefore, performing using Synergia can sufficiently cover the limitations of distance in the communication process by means of the network. That applies for both performers and audience. Furthermore, a message in such a communication system has an abstract and subjective form derived through the prism of artistic experimentation. In our case, the transmitter and receiver can be both the performers and audience. More specifically, performers can transmit various messages through their improvisational musical expression, assuming that it is a musical live coding performance. In addition, they also receive intuitive feedback from the audience. Subsequently, the audience receives both musical and optical messages from the performers and it may also motivate emotional responses through their gestural manifestations and nonverbal-communication. On Synergia, that is replicated through textual expression, due to the limitation of remote visual feedback. Our vision aims for a fully interactive live coding performance, where an interactive communication relationship will develop between the performers and the audience. The respective result, in terms of performance, will be derived on this relationship by replicating, in a sense, the physical act of the live coding practice. We suggest that with Synergia performers can strengthen the bonds of interaction to each other. Firstly, by the general concept of collaborative live coding and secondly by giving them the chance to "shape" the environment into a desirable and convenient form where the live coding session will take place. Another important aspect of Synergia is that it can support a live audience. Anyone can attend a live coding session on Synergia by connecting to the server using OSCGroups and starting a local sclang process. A physical performance requires feedback from the audience and this ability is also important. Using Synergia audience can send messages from their local sclang process to the performers in real time by using the VPN configuration. Although, the feature of audience feedback requires improvements in the terms of accessibility and convenience. Taking that even further, one idea would be to introduce the ability of interaction between the performers and audience. Additionally, most studies in the field of remote collaborative live coding have only focused on music. The same applies on Synergia. A future goal is to incorporate visuals and sensor connectivity for interactive performance practices. We strongly believe that a greater focus on including telematic performance and visuals in Synergia could offer a significant number of expressive capabilities to the performers. Potentially, in terms of visuals a limitation could be the necessity of a graphical interface. Since Synergia is accessible by means of the terminal visuals options are limited. For example, one option is to synchronize visuals and audio using Pen[8] class in SuperCollider. In addition, there is a number of implementations of ascii art in openFrameworks[9], that can run in the Synergia environment. At the same time, we are trying to make Synergia as easily accessible as possible for the users.

7https://github.com/Vasileios/Synergia-Collaborative-Live-coding

8https://doc.sccode.org/Classes/Pen.html

9[https://github.com/tgfrerer/ofxNcurses], [https://github.com/naus3a/ofxConsoleRenderer]

Therefore, the question of whether an additional configuration on the user's local machine regarding visuals could increase the complexity should be considered. In our opinion, this work develops and promotes new techniques and new tools for networked collaborative live coding and interactive audiovisual arts.

Acknowledgments

Special thanks to free software community for making our world better.

REFERENCES

Agiomyrgianakis, Vasilis. 2021. "LiveGloving: Experimental schemes for live coding performances". *International Conference DCAC*. Corfu.

Barbosa, Alvaro. 2003. "Displaced Soundscapes: A Survey of Network Systems for Music and Sonic Art Creation". *Leonardo Music Journal 13: 53–59.*

Bencina, Ross. 2018. "OSCgroups: peer-to-peer Internet OSC multicast without the pain". https://www.rossbencina.com/code/oscgroups, accessed:2021-08-12.

de Carvalho, Junior, Lee A.D, Essl G. 2015. "SuperCopair: Collaborative Live Coding on SuperCollider through the Cloud". In: McLean, Alex and Magnusson, Thor and Ng, Kia and Knotts, Shelly and Armitage, J (ed.) *Proceeding of the First International Conference on Live Coding*. pp. 152-158. ICSRiM, University of Leeds.

Kirkbride, Ryan. 2017. "Troop: A Collaborative Tool for Live Coding". *In: Proceedings of the 14th Sound and Music Computing Conference*. pp. 104–109.

Lee, Sang Won, and Georg, Essl. 2014b. "Models and Opportunities for Networked Live Coding." *In Proceedings of the Live Coding and Collaboration Symposium 2014. Birmingham*, United Kingdom.

Magnusson, Thor. 2009. "ixi lang: a live coding programming language for musical performance". Software, UK. http://www.github.com/ixi-thor/ixilang.

McCartney, James. 2002. "Rethinking the Computer Music Language: SuperCollider". *Computer Music Journal 26* (4): 61-68.

McLean, Alex, and Geraint, Wiggins. 2010. "Tidal—Pattern Language for the Live Coding of Music." *In Proceedings of the 7th Sound and Music Computing Conference*. Vol. 2010. http://server.smcnetwork.org/files/proceedings/2010/39.pdf.

Ogborn, David, et. al. 2015. "Extramuros: making music in a browser-based, language-neutral collaborative live coding environment". *First International Conference on Live Coding*. Leeds, UK.

Ogborn, David., & Beverley, J.G. 2017. "Estuary: Browser-based Collaborative Projectional Live Coding of Musical Patterns".

Stallman, Richard, M. 1981. "The Extensible, Customizable, Self-Documenting Display Editor". *ACM Conference on Text Processing.*

Wright, Matthew. 2005. "Open Sound Control: an enabling technology for musical networking". *Organised Sound* 10 (03): 193.