

Orchestra! : a Distributed Platform for Virtual Musical Groups and Music Distance Learning over the Internet in Java™ Technology

D. Giuli

Dept. of Electronic Engineering
University of Florence
Via S.Marta 3, Florence, Italy
giuli@diefi.die.unifi.it

F. Pirri

Dept. of Electronic Engineering
University of Florence
Via S. Marta 3, Florence, Italy
fpirri@ing.unifi.it

P. Bussotti

Dept. of Electronic Engineering
University of Florence
Via S. Marta 3, Florence, Italy
bussotti@achille.die.unifi.it

Abstract

This paper proposes a platform for musical group sessions over the Internet. It has been conceived as an useful instrument for collaborative music developing and enjoying, both at the professional and at the «amateur» level. It has been designed for music distance learning, as well. The system, that we named «Orchestra!», is based on a distributed software architecture and its services are universally accessible through a simple http-client (a browser) and a sound card. Geographical packet-switched network latencies make it impossible to reach a real musical synchronisation among all members of a virtual group. As an alternative, it has been developed a system of progressive and collaborative elaboration of musical audio streams which is insensitive to delays: this idea permits to reproduce «live» performances as well as recording studio activities.

1. Introduction

Computer music is an important example of how art and technology can co-operate to create new expressive forms. The development of the MIDI interface, the spreading of high-tech electronic instruments as well as popular software like *Cubase* [1] or *CakeWalk* [2] had deeply changed the approach to music of groups, both at the professional and at the *amateur* level.

The idea of Group Music over the Internet is the natural extension of computer music in the world of telematics and distributed multimedia technologies: it's still quite a young field of research. It encounters the critical problem of exchanging synchronous streams of audio data over a geographical packet-switched network, in a type of application that has an extremely low tolerance to delays: the requirements of *musical group synchronisation* gives an upper limit to *tolerable delay*, in agreement to human (imperfect) *perception of synchrony*.

While playing in real life, everyone gets an *immediate* feedback from his own instrument. The feedbacks everyone gets from the others are *negligibly delayed* within a small ambient: this permits to them to experiment group synchrony and to go on playing while trying to maintain this rhythmic equilibrium. The average latencies introduced by the Internet make unpractical the network solution that's more similar to the real situation : multipoint-to-multipoint. In other works about this argument, well documented in A. J. Paradiso [3], like *Spinning Disks* of E. Metois or the *Palette of Brain Opera* at the Massachusetts Institute of Technology (MIT), based on the work of J. Yu, the focus has been put on the aspect of creating interactively something artistically innovative with expert-system generated sounds, assuming less stringent requirements on synchrony. An interesting system using audio and MIDI streams is ResRocket's *DRGN* [4] : tracks are produced , and stored locally, on the basis of previous tracks broadcast in repetitions by a server. The server receives asynchronously each new track and inserts it within the successive loop. This permits true collaboration, but lacks in interactivity, since a song can only be completed after several loops, as in a recording studio.

2. Musical Group Synchronisation

By opposite, Orchestra! tries to give a pleasant *ensemble* feeling and employs real-time delivering of user-produced audio streams. Musical group synchronisation is maintained by forcing configurations of uni-directional audio-stream connections among the participants that do not imply any delayed listening of theoretically synchronous tracks: a musical flow is originated from one source (typically the drums) and adds successively all contributions from remote players in such

a way that no closed path is made. The order of insertion in this multi-level flowing scheme, that in its simplest version is a cascade, corresponds, by default, to the rhythmic hierarchy among instruments. The wide literature about multimedia synchronisation has helped to solve the problems of intramedia and intermedia synchronisation, for effective audio stream delivery and multiple streams managing. In particular, since the Internet is the target support, a solution of sample-level mixing into one channel (named the *guide-stream*) has been adopted among synchronised tracks for reducing bandwidth requirements and further work of synchronisation down the cascade. However, a high-quality version of each track is stored locally in the meanwhile, and opportune time references make it possible to mix them into a high-quality group's *master*: this is made collaboratively by means of Orchestra! tools. So, while playing on the basis of the local incoming *guide-stream* gives an approximation of a live performance, although with a relatively low quality and differentiated richness in musical contributions along the cascade, the second phase of local tracks mixing gives the final product in terms of a high quality master and separately stored individual tracks.

Intermedia synchronisation is required in two fundamental cases:

- At the user host, between the incoming *guide-stream* and the microphone captured audio contribution.
- In configuration that present the confluence of *guide-stream* branches that run separately at higher levels.

The presented solution consists in assigning to each host the reference time of incoming packets and in timestamping outgoing packets accordingly. In this manner, after a initialisation phase that's monitored by a specific server, each player buffers the incoming stream, locally synchronises his contribution to it and then let the enriched version flow to lower rhythmic levels: it's much like one single stream that passes through the hosts and collects synchronised contributions.

While the time arrangement among participant hosts would not be strictly necessary for local synchronisation at the user host, it is required for:

- Intermedia synchronisation of confluent *guide-stream* branches (the mixer node may assume the time reference of the first incoming stream).
- Marking the locally stored tracks for successive mixing.

3. Orchestra! Collaborative Environment

The design of Orchestra! as a highly usable collaborative environment for different typologies of users has taken advantage of direct investigation by means of questionnaires and of the author's personal

experience in amateur musical groups. Three main users' profiles have shown to be the most interested in collaborative distance working with music:

1. the professional user
2. the user for didactic aim (teacher and student)
3. the *amateur* user.

Orchestra! assumes the third typology (the *amateur*) as a sort of basic user and defines accordingly a set of basic functions and tools within a flexible object-oriented platform. Other users' typologies could be well served either by specialising existing functions or by adding new ones. Orchestra! appearance is a Windows95-like desktop where tools are represented by icons. Basic tools can be subdivided in three main categories:

1. Contact and Registration tools : information about groups; contacts with them; registration to a group and login.
2. Music Processing tools : player/recorder, track mixer.
3. Group Session tools : live performance manager, group chat-line, presence panel (indicating group people currently connected), access to group private resources and tracks on its server space.

Obviously , the third category of services is accessible only by people registered to a group.

4. The Prototype

The presented prototype is a Java™ applet that's loaded from Orchestra! website (<http://medialab.die.unifi.it/java>) by a Netscape 4.x browser. The choice of this browser is obliged since only Netscape™ allows the required selectivity in security management for signed applets. In fact, Orchestra! applet needs to load native C local libraries for acceding to the operative system low-level sound functions (JVM 1.1 does not provide equivalent methods) ; moreover it must accede to the local file system, and must be allowed to make socket connections to arbitrary hosts (notice that, by default, an applet can only communicate with the host it comes from).

4. References

1. Cubase (1998) , trademark of Steinberg Vertrieb GmbH. , homepage : www.steinberg.de
2. CakeWalk (1998), trademark of Twelve Tone System, Inc. , homepage : www.cakewalk.com
3. Paradiso, A.J. (1997) Electronic music: new ways to play. IEEE Spectrum, December 1997, volume 34, number 12, 18-30.
4. ResRocket DRGN. homepage: www.resrocket.com