Implementation of the Synchronization Mechanism of Application Instances using SwellRT's Collaborative Objects to Optimize the Communication

Macaire Ngomo

Abstract— We propose to define a protocol which manages the synchronization of two instances of the same application, through a communication network. We are aiming for something simpler: to synchronize instances of the same application on different computers, especially in a context of low-speed Internet connection, with applications for example in the fields of distance education or remote assistance. In this article, we describe the situation using an extended automata asynchronous composition model. Our study is carried out within the framework of a general synchronizable application. In the particular context of this study, the envisaged application corresponds to a SERPOLET environment integrating communication modules, for the synchronous and / or asynchronous follow-up or support of a learner (respectively a group of learners) by his tutor (respectively their tutor). Our early implementations of the synchronization mechanism used socket and RMI communications in Java. In this new implementation, which is the subject of this study, we use the natural collaborative possibilities of SwellRT to optimize the communication modules between application instances.

Index Terms— Synchronization of application instances, synchronization protocol, synchronous or asynchronous monitoring and support, low-speed Internet connection, intelligent systems, learning Technologies for Education systems and intelligent environment, SERPOLET system, SwellRT, Collaborative objects, RMI, Socket.

1 Introduction

As part of our development projects on e-learning management systems, advanced learning technologies, intelligent environments, educational systems, we were faced with the problem of the poor quality of the links in certain geographical areas, due to low-speed Internet connectivity, not allowing us to use the standard software of videoconferencing, remote display, or remote document sharing, typically carrying very high-bandwidth bitmap images. The application instance synchronization mechanism is used in different contexts. For example, the synchronization procedure for ADO.NET is to use session variables for collaborative synchronization [8]. To address this specific need, we propose in this paper to define a protocol that manages the synchronization of two instances or copies of the same application, through a communication network. In this paper, we present the situation using an asynchronous composition model of extended automata [1] [19] [20] [21] [6]. Our study will be done within the framework of a general synchronizable application. In the particular context of this study, the application envisaged will correspond to the environment of the authoring system SERPOLET and its derived [11][12][2][13][17][18] for synchronous and asynchronous monitoring and assistance between a learner and his tutor. In general, it will be a matter of being placed in a situation of assisted learning or tutored learning using a communication mechanism based on the exchange of events based on scripts between the "Master station" and the "Slave station". Tutoring is more a function related to the accompaniment of learners during their learning pathway [10][3][4][5]. The role of tutor is reinforced in the distance education systems thanks to a set of tools resulting from the advances of the new technologies for information and communication. The general model for organizing online tutoring can be summarized in the following figure [5]:

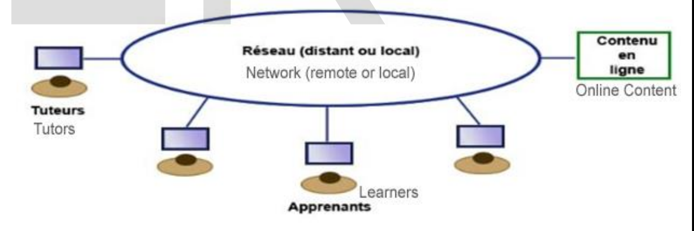


Fig. 1. General Model of online tutoring to online resource.

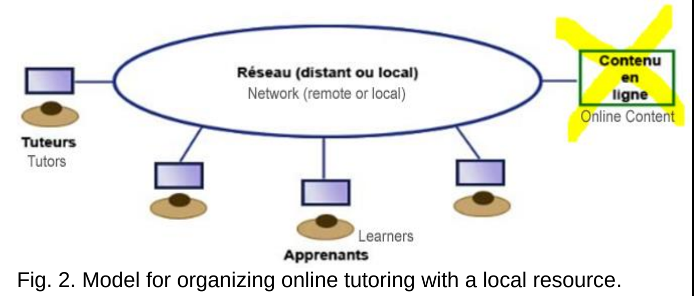


Fig. 2. Model for organizing online tutoring with a local resource.

Our model is inspired by this general model of tutoring and modifies it by removing remote access to educational resources (online content) to obtain a model of tutoring called local resource. [17] makes an inventory of the tools according to the tutoring activities. The palette of the tutor is rich of several families of tools. Each family changes very little in terms of pedagogy because its objectives remain constant but it can evolve very quickly on the technical level according to the contributions of the new technologies.

2 Problematic

A first user, named master (UM), has an application whose localization it manages the evolution or the course. A second user, named slave (UE), has a copy/instance of the same application and, for example for monitoring purposes, assistance, wants to synchronize the state of his copy, and his future evolution on that of UM. We voluntarily chose to detach ourselves from the roles of teacher and pupil, as these roles do not necessarily reflect the sense of synchronization. Indeed, when monitoring the activity of a pupil, the role of the teacher is held by the teacher. In the case of an explanation given by the teacher or guardian on the environment of a student, the role UM is held by the teacher. The local application is called X. It receives the events transmitted by the user, events to which it reacts. Periodically, it saves its current state X as a series of states XS0, XS1, XS2 ... The last saved state will serve as a starting point when requesting to resynchronize the remote copy of the application.

The set of states will be used during an asynchronous exchange (deferred tracking).

The following diagram (Figure 3) describes in a synoptic manner the type of exchanges that one has to consider between the user "Master" and the user.

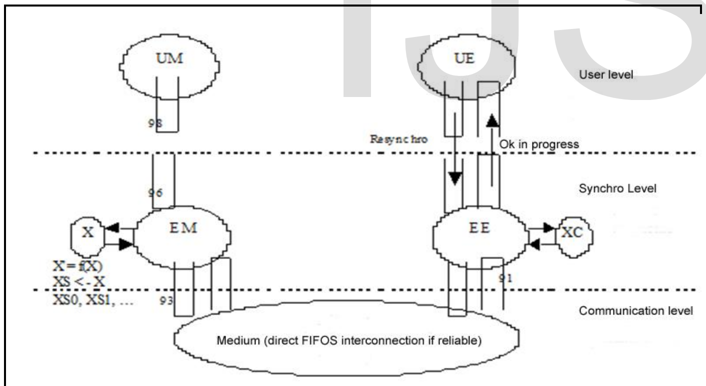


Fig. 3. Diagram of master and slave communication.

Each module is interconnected by queues to the modules with which it communicates. These queues make it possible to model the asynchronism of the behaviours between these entities. The function X '= f (X) describes the sequence of states that the synchronized application saves.

The numbers in the queues represent the numbers of the events that transit between the different entities. Because of the asynchronism, these numbers are not all processed at the same time, on the diagram. The sequence of numbers gives the order of processing of the messages by the entities.

Of course, the notion of the backup state of an application is dependent on the type of application considered. In the same way, events handled by an application depend on the application, both in terms of their choice and their granularity. The main difficulty for a given application will be the possibility of identifying these events and the notion of backup / recovery state. Subsequently, in our study, we assume that the generic application X possesses the three preceding characteristics, implemented elsewhere in the project. This is the case of the authoring system SERPOLET.

3 Synchronization Service Architecture

The model on which the formal specification of the distributed system that we use is based is that of extended automata. In that model, each specified entity has a behaviour represented in the form of a finite state machine.

The architecture of the system respects the OSI three-layer basic design model. In this model, the design of a new communication service is done by means of a software layer accessible to a set of users in the form of a set of service primitives. This software layer consists of a set of distributed entities, which interact with one another by defining a common message format, called PDUs (Protocol Data Units). These PDUs are physically exchanged between entities using a lower level communication service. Figure 5 shows the architecture of the application copy resynchronization protocol.

The "master user" and "slave user" boxes respectively represent the two users of the resynchronization service. This service is physically realized in the form of the two software entities, "protocol entity", which interact with each other using the underlying communication service.

The architecture defined uses two instances of "runtime" SERPOLET, one master (linked to the application) and the other slave (linked to the control module of the application).

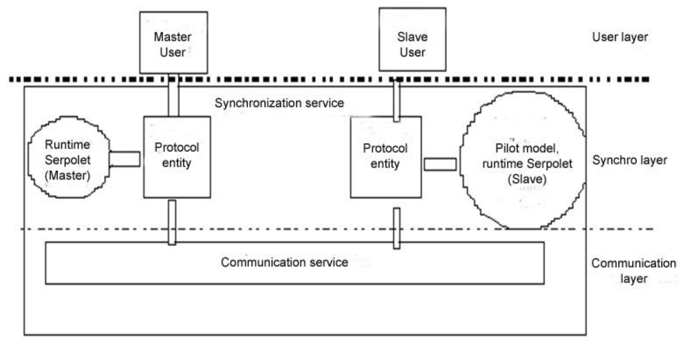
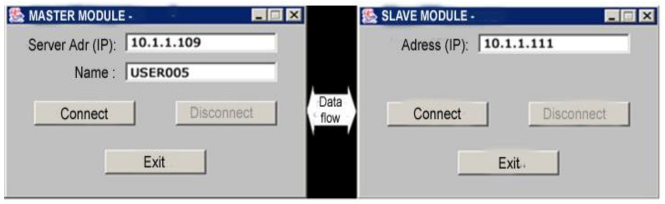


Fig. 4. Diagram of master and slave communication.

Between these two instances, a distributed service, in the sense of communication service, is specified and developed in the form of a communication protocol to effectively ensure the actual synchronization of these two instances. The design architecture (Figure 4) will subsequently be defined in the implementation architecture in which the entities corresponding to each user will be physically distributed on remote machines corresponding to each user. From a conceptual point of view, the synchronization function remains decoupled from the SERPOLET modules. In the final phase of implementation, integration of this entity with this module can be envisaged, but in any case it will remain independent of the other functions of the synchronized software.

4 Synchronization Modules

This part describes the list of interfaces that each user, master and slave, possesses.


MASTER MODULE -
Server Adr (IP): 10.1.1.109
Name : USER005
Connect
Disconnect
Exit

Data flow

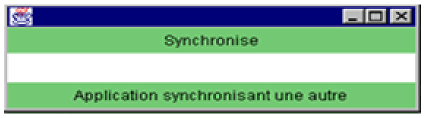
SLAVE MODULE -
Adress (IP): 10.1.1.111
Connect
Disconnect
Exit

Fig. 5. Communication between the "Master" and "Slave" modules.

The state of each interface corresponds to one of the states of the user automata. The change of the contents of each interface is caused by the arrival of an event, modelled by the transitions of the user automata. The two synchronization modules are shown in the diagram in the figure below. Once the two modules are connected, a communication channel allows the exchange of data between the two entities. Each entity receives data from the local "application runtime" that it sends to the other remote module, and data it receives via the channel to the local "runtime". Although it is possible to initiate a communication to several, to optimize the follow-up, in this version the communication is point to point. Indeed, even if our model is not limited and even if there are technical possibilities, we privilege in this study the quality of the exchanges and therefore did not envisage a simultaneous follow-up of several positions.

4.1 Master Module

This user interface has only an indication role: it indicates whether the "master" application is operating autonomously, or whether it slaves the slave.


Synchronise

Application synchronisant une autre

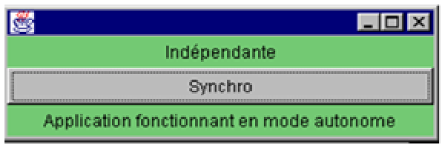
Synchronize

Application synchronizing another

Fig. 6. Application running in standalone mode and synchronizing another.

The proposed interface consists of two text fields. The first one at the top specifies the state of the application. The second, below, is a message that details the meaning of this state. The transition from the "independent" state to the "enslaved" state is caused by the arrival of the primitive "Sbdy_sync_ind". The reverse path occurs when the primitive "Sbdy_end_sync_ind" arrives.

4.2 Slave Module


Indépendante
Synchro
Application fonctionnant en mode autonome

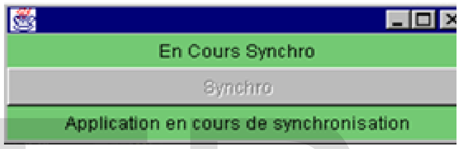
Independent

Synchro

Application running in standalone mode

Fig. 7. Application running in standalone mode and not synchronizing another.

The interface of the slave has two roles: it allows requesting that the local application is enslaved to that of the master and it informs of the synchronization state of the two applications. The requests for enslavement/servo-control and end of enslavement/servo-control are done using a button. The status of the local application is displayed using two text fields. The first gives a brief description of the state; the second explicitly states this state. The interface of the slave only shows that both applications are independent when the user is in the "rest" state. The interface of the slave only shows that both applications are independent when the user is in the "rest" state. The action of the "sync" button causes a synchronization request. This action causes the "Req_sync_req" primitive to be issued, as described by the slave user controller, which changes to the "Wait_conf" state. The interface that corresponds to this state is then as follows:


En Cours Synchro
Synchro
Application en cours de synchronisation

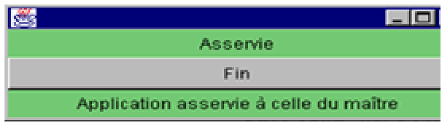
In Sync

Synchro

Application being synchronized

Fig. 8. Application being synchronized.

Once the synchronization is effective, the slave user receives the message "Req_sync_conf" and switches to the "Sync_Work" state. The button becomes active again. It will be used to stop synchronization. The interface that corresponds to this state is then as shown in the following figure.


Asservie
Fin
Application asservie à celle du maître

Enslaved
End
Application slaved to that of the master

Fig. 9. Application enslaved to that of the Master.

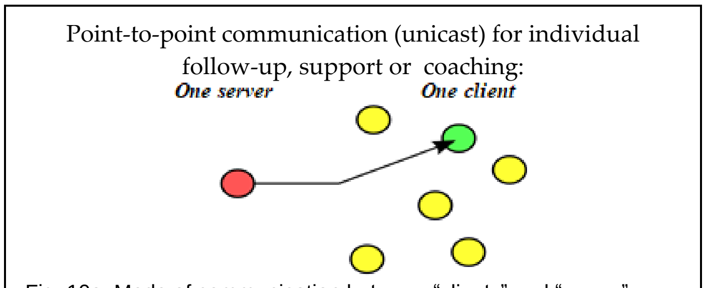
The action of the user on the "End" button corresponds to an end of synchronization of the applications. It results in the issuing of the primitive "End_sync_req", and a return of the slave user to the "Rest" state. The new interface then corresponds to that of the independent applications.

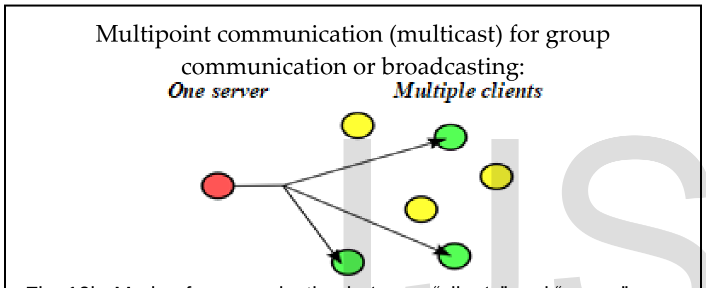
5 Implementation of the Application Instances Synchronization Mechanism Based on SwellRT

In this section, to improve our early implementations of the application instance synchronization mechanism and increase collaboration power, we propose a new collaborative object-based implementation of SwellRT [34] [23] [24] [298] [32] [9].

5.1 Socket Implementation

The first implementations were based on communication by socket, point to point (unicast) or multipoint (multicast) depending on usage [14] [15]. The sockets interface is the most common network programming interface allowing the conventional use of communication by socket according to the usual client-server scheme. Sockets allow building a custom solutions.


Point-to-point communication (unicast) for individual follow-up, support or coaching:
One server
One client
Fig. 10a. Mode of communication between "clients" and "server".


Multipoint communication (multicast) for group communication or broadcasting:
One server
Multiple clients
Fig. 10b. Mode of communication between "clients" and "server".

In the Web version, a WebSocket allows bidirectional and full duplex communication on a single TCP socket between a client and a server. When the server responds, the connection is established and the client and server can send and receive messages. The HTTP protocol is only used to establish the connection of a WebSocket: once the connection is established, the HTTP protocol is no longer used in favour of the WebSocket protocol.

5.2 RMI Implementation in Java

The second implementation is based on RMI (Remote Method Invocation). RMI is a JDK technology in Java for easily implementing distributed objects and remote method calls. The purpose of RMI is to allow the call, execution, and return of the result of a method executed in a different virtual machine than that of the calling object. RMI technology is responsible for making transparent the location of the remote object, its call and the return of the result.

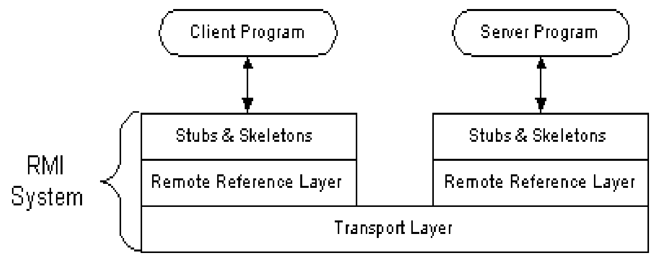


Fig. 11. RMI Layered Structure.

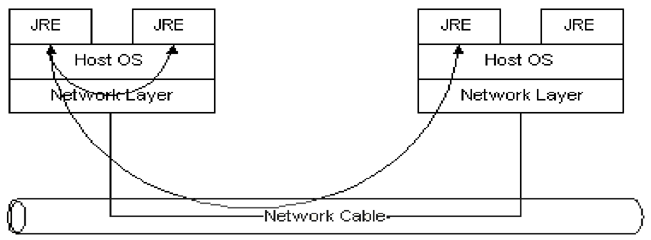


Fig. 12. RMI Transport Layer.

In fact, it uses two special classes, the stub and the skeleton, which must be generated with the rmic tool that comes with the JDK. The stub is a class on the client side and the skeleton is its server side counterpart. These two classes are responsible for ensuring all the mechanisms for calling, communicating, executing, returning and receiving the result.

Communications between client and server are carried out using RMI-IIOP (Internet Inter-Orb Protocol), a protocol standardized by the OMG (Object Management Group) and used in the CORBA architecture. The transmission of data is done through a system of layers, based on the OSI model in order to guarantee interoperability between programs and versions of Java. The stub (translate stub) and the skeleton (translate skeleton), respectively on the client and the server, ensure the conversion of communications with the remote object.

The Remote Reference Layer (RRL) is responsible for the location system to provide a means for objects to obtain a reference to the remote object. It is provided by the java.rmi.Naming package. It is generally called the RMI register because it references the objects.

The transport layer is used to listen to incoming calls as well as to establish connections and transport data on the network via the TCP protocol. The java.net.Socket and java.net.SocketServer packages implicitly provide this function.

5.3 SwellRT Implementation

In this section, to improve ours first implementations of the application instances synchronization mechanism, we propose an implementation based on collaborative objects de SwellRT. First implementations were based on the use of sockets and RMI in Java. For this new implementation, we used collaborative objects by exploiting the natural collaborative possibilities and development facilities offered by SwellRT, an open source framework for the development of decentralized collaborative Web applications naturally offering storage, sharing and collaboration services.

SwellRT is a BaaS technology capable of working in a decentralized manner thanks to the use of an open protocol to interconnect services (a federated network) and to exchange data in

real time.
Development frameworks are built thinking in centralized apps, moreover when thinking of collaborative apps. SwellRT (http://swellrt.org) is a development framework for building decentralized real-time collaborative apps, easily and avoiding extra code to the developer. SwellRT provides a server side (storage, sharing, identity, federation) and an API to build apps in JavaScript, Java or Android. You may think of Google Drive Real-Time API or Firebase but decentralized & open source.
Initially developed as part of the P2PVALUE [31][30][16][25][9][33] project which was stopped in September 2016, SwellRT joined the fold of the Apache foundation. SwellRT is a real-time storage platform. Its API makes it possible to manipulate and share objects in real time on a decentralized network. SwellRT enables real-time collaboration within web applications [26]. Besides code sharing, SwellRT can be used to build chats, survey platforms, or document management platforms, for example.
SwellRT is a free and open-source backend-as-a-service and API focused to ease development of apps featuring real-time collaboration. It supports the building of mobile and web apps, and aims to facilitate interoperability and federation. It provides prebuilt features to speed up development of collaborative Web applications:
- Real-time storage (eventual consistency) (NoSQL)
- Extensible text collaborative editor
- User management and authentication
- Server federation with Matrix (XMPP or Matrix.org [28])
- Integration of third party systems based on events (in development)

Concurrent access and storage control is based on the Apache Wave Operational Transformation System, which ensures the integrity of shared data. Apache Wave is a discontinued software framework for online real-time collaborative editing. Google originally developed it under the name Google Wave. It was announced at the Google I / O conference on May 28, 2009. Business Transformation: Business Transformation (OT) is a technology to support a range of collaboration features in advanced collaborative software systems. In 2009, OT was adopted as the core technique behind collaboration features in Apache Wave and Google Docs.
The main feature of SwellRT is real-time storage based in objects. They can be shared among participants that can mutate them in real-time. All changes are persisted and propagated transparently. Object's state is eventually consistent.

Applications using SwellRT

SwellRT facilitates the development of mobile/web apps. Several apps are built using this technology. Apart from to the demos provided by SwellRT and many applications developed by third parties, two full-fledged applications that are benchmarks currently use SwellRT technology:
- JetPad (jetpad.net) [27], a GoogleDoc-like collaborative editor, free/open source and federated,
- Teem (http://teem.works/) [35], a free/open source web/mobile app for the management of communities and collectives.

Collaborative object-based implementation with SwellRT technology

SwellRT is a fork from Apache Wave, inherits some of its architecture and technology stack. However, it grew beyond the limits of Wave, first presenting itself as a web framework and nowadays growing to a backend-as-a-service platform. Its current technical approach covers the following:
- It is fully free/open source software. It is developed in Java. GWT with JSInterop is used to generate JavaScript API reusing the same source code. Android client is also built from the same Java sources.
- It provides an extensible and pluggable rich-text editor component for Web (only) supporting custom annotations and widgets.
- Real-time data storage is based on Wave's Operational Transformations model, thus it is eventually consistent.
- It is designed to maximize interoperability, and follows a federation approach similar to Apache Wave, using XMPP or Matrix.org communication protocol. It aims to support the creation of apps that are federated, i.e. rely on multiple interoperable servers, and objects shared across servers.

SwellRT provides a programming model based on collaborative objects. A collaborative object is a JSON-like object that can be shared by some users (or groups) that can make changes in real-time. Changes are propagated (and notified) in real-time to any user connected to the object. Objects and participants are uniquely identified on the Internet enabling decentralized access from different federated servers.
SwellRT allows to store and share data in real-time using collaborative objects. They can be thought as JSON documents with a special syntax and methods to access and change their properties.

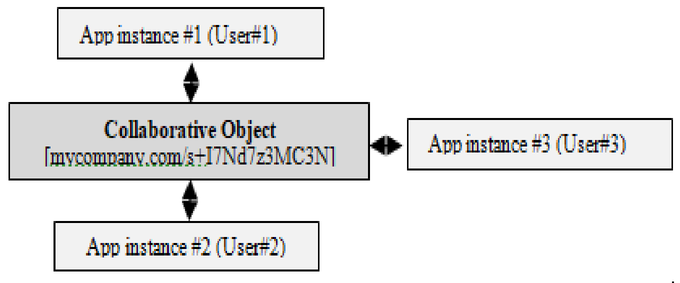


Fig. 13. Communication via collaborative object.

A collaborative object has an unique identifier on Internet, for example mycompany.com/s+I7Nd7z3MC3N, where first part is the domain of the Swell server, and the second part is an id, provided by the client's app or randomly generated by Swell. When different instances of an app have opened same objects, they can share data in real-time through them.

Changes in a collaborative object are persisted in the server and transmitted to all instances in real-time.

A collaborative object can store properties of simple data types (string, integers, etc.) as well as rich-text and references to files or attachments. This approach is suitable to implement any document based collaborative application like text editors or spreadsheets.

Architecture

The architecture of our new implementation is composed of:
- A SwellRT server which manages the storage and real-time sharing of data in the form of collaborative objects (JSON objects)
- Communication modules designed with the SwellRT API in a distributed P2P architecture [16][25][9] playing the roles of master and slave and using collaborative objects. These communication modules ("master" and "slave") ensure data exchange between the master and slave instances of the application.

Several situations are taken into account:
- Point-to-point communication between the "master" instance and the "slave" instance. Thanks to user identification, the "master" instance can exchange simultaneously with several "slave" instances,
- Multicast communication: the "master" instance shares information with several "slave" instances. This is the case when, for example, a user (e.g. a teacher) shares content with other members (e.g. students).

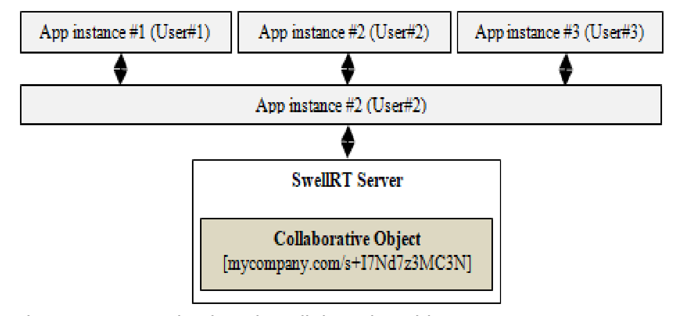


Fig. 14. Communication via collaborative object.

6 Conclusion

This paper presented the design, using the SDL language, of a service and a synchronization protocol for copies of distributed applications, and the remote control of one of them. A set of user interfaces was then deduced from the user behaviour. This protocol has been tested and validated in an environment built around authoring systems SERPOLET. We envision as a continuation of research work to extend it to other environments. This mechanism is an alternative to videoconferencing software, remote screen sharing, or remote document sharing, typically carrying bitmap images that are very bandwidth-intensive. These software are used to communicate, give webinars, web conferencing, remote assistance or trouble-shooting, remote meetings, collaborative online work, etc. which require high-quality links (broadband), and therefore difficult to use in the case of a low-speed network. Compared to these Softwares, the mechanism used here is very simple and economical. It comes down to a very light-based event exchange that does not require a lot of resources or a large bandwidth. This mechanism therefore makes it possible to develop synchronous and / or asynchronous learning tools that are very economical.

Our project on this subject aims to generalize this approach to other types of applications other than online training environments. The idea is to allow any application to receive synchronization services from application instances in point-to-point or multicast mode. The first version of our implementation is based on Java technologies and sockets and RMI. In order to optimize this implementation, we exploited the possibilities of SwellRT. Compared to the first implementation, we noted ease of development as well as a marked improvement in performance. We also plan to experiment with the Linda model [7] [22] (a model of coordination and communication between several parallel processes operating on stored and retrieved objects, associative virtual memory) to implement the exchange of messages low-level between application instances.

From a performance perspective, the implementation of the communication components is done in a naturally collaborative environment, building on SwellRT's collaborative object-based programming model. This has enabled us to free ourselves from the costly specific developments of the first versions and opens up new operating prospects directly and from all the wealth of services offered by SwellRT.

Acknowledgment

The author wishes to thank Habib Abdulrab, Jean-Pierre Pécuchet, Abdenbi Drissi-Talbi, Mohamed Rezrazi, Fabrice Sebbe and all his friends and colleagues for their help and support. He also wishes to thank Olga, Michel, Marielle and Guyriel who have always been a very precious support for the realization of this work.

References

Journal

[1] M.-L. Betbeder, C. Reffay and T. Chanier, "Environnement audio-graphique synchrone : recueil et transcription pour l'analyse des interactions multimodales". In JOCAIR 2006, Premières journées Communication et Apprentissage instrumentés en réseau, Amiens, France, pp. 406-420, July 2006.

[2] M. NGOMO, H. ABDULRAB, "APPLICATION SERVICE PROVIDER SYSTEM : THE NEW WAY TO PROVIDE INTEROPERABILITY BETWEEN LEARNING MANAGEMENT SYSTEMS", Web Based Computer, WBC'2007.

[3] Omeric., "Point sur la FOAD: Rapport de mission. Mission de suivi du projet FOAD-LSN", Coopération Internationale 2012. Collectif

OMERIC, Mars 2013, France.

[4] Omeric., “List des termes courants du domaine de la formation et de l'enseignement”, Collectif OMERIC, Août 2013, France.

[5] P. Nivet, “Un inventaire des outils du tutorat en ligne”. Collectif OMERIC, Avril 2010, France.

[6] A. Sarma, “An introduction to SDL-92”. Computer Networks and ISDN Systems 28(1996), pp. 1603-1615.

[7] N. Carriero, D. Gelernter, T. Mattson, A. Sherman, “The Linda Alternative to Message-Passing systems”. Parallel Computing. doi:10.1016/0167-8191(94)90032-9.

Conference paper or contributed volume

[8] ADO.NET, “Synchronisation de bases de données Scénarios de collaboration Synchronisation d'autres bases de données compatibles ADO.NET”. https://msdn.microsoft.com/fr-fr/library/cc761645(v=sql.105).aspx. retrieved 2020-08-20.

[9] CORDIS - European Commission, “News and Events : A substantial boost for easily and safely producing new online apps”. cordis.europa.eu. retrieved 2020-08-20.

[10] M-C. Monget, T. Kelo, “Work towards automated vendor-neutral certification of ICT skills”. Actes du congrès Ed-Media2008, pp. 37-49, Vienne (au), Juillet 2008.

[11] M. NGOMO, H. ABDULRAB, L. OUBAHSSI, “Application Service Provider System : a new concept to provide interoperability between learning management systems”, Proceedings of E-Learn 2005 World Conference (World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education), Vancouver,(Canada); Research/Technical Showcase; pp. 2763-2769 (2005)

[12] M. NGOMO, H. ABDULRAB, “Application Service Provider System: Using Web Services to Provide Interoperability between Learning Management Systems”, International Conference WTAS 2006 (Web Technologies, Applications, and Services), July 17-19, 2006, Calgary, Alberta, Canada, Editor(s): J.T. Yao; pp 119-125 (2006)

[13] M. NGOMO, H. ABDULRAB, 2007: “Application service provider system: the new way to provide interoperability between learning management systems”, International Conference Applied Computing 2007, IADISI'2007, Salamanca, Spain, 18-20 February 2007; 12 p. (2007)

[14] M. NGOMO, H. ABDULRAB, “Synchronization of distributed application instances as a learning tracking mechanism”, International Journal of Scientific & Engineering Research Volume 9, Issue 3, March-2018, p324, ISSN 2229-5518.

[15] M. NGOMO, “Synchronization of application instances as economical way for E-learning and tele-assistance”, 4th International Conference On Computer Networks and Information Technology Held on23rd-24thMarch 2018, in Pattaya, Thailand, p235, ISBN:9780998900049.

[16] P. Ojanguren-Menendez, A. Tenorio-Fornés, S. Hassan, “Distributed Computing and Artificial Intelligence, (2015]). 12th International Conference”. Advances in Intelligent Systems and Computing. Springer, Cham. pp. 269–276. doi:10.1007/978-3-319-19638-1_31. ISBN 9783319196374.

[17] L. OUBAHSSI, M. Grandbastien, M. Ngomo, G. Claes, “The Activity at the Center of the Global Open and Distance Learning Process” The 12th International Conference on Artificial Intelligence in Education, AIED 2005, Amsterdam.

[18] L. OUBAHSSI, “Conception de plates-formes logicielles pour la formation à distance, présentant des propriétés d'adaptabilité à différentes catégories d'usagers et d'interopérabilité avec d'autres environnements logiciels”, œuvre [Thèse de M. L. OUBAHSSI dans le cadre d'A6/OMERIC, décembre/2005], Thèse de doctorat de l’Université René Descartes – Paris V, Centre Universitaire des Saints Pères, UFR de Mathématiques et Informatique, Paris, 2005.

[19] C. Reffay, T. Chanier, “Mesurer la cohésion d'un groupe d'apprentissage en formation à distance”. In Actes de la conférence Environnements Informatiques pour l'Apprentissage Humain (EIAH'2003), Strasbourg, France, pp. 367-378, April 2003.

[20] C. Reffay, T. Chanier, “How social network analysis can help to measure cohesion in collaborative distance-learning”. In Procs. of Computer Supported Collaborative Learning Conference (CSCL'2003), Bergen, Norway, pp. 343-352, June 2003. Kluwer Academic Publishers : Dordrecht(nl).

[21] C. Reffay, “Réseaux sociaux et analyse de traces des forums d'une communauté d'apprentissage”. In G.-L. Baron, E. Bruillard, and M. Sidir (Dir.), editors, Symposium, formation et nouveaux instruments de communication, Amiens, France, p 13, January 2005.

[22] G. Wells, “Coordination Languages: Back to the Future with Linda” (PDF). Rhodes University. Archived from the original (PDF) on 2009-12-19.

Others references

[23] S. Hassan, “SwellRT: Facilitating Decentralized Real-Time Collaboration”. Harvard's Berkman CRCS (2016-10-06), CRCS Seminar 09/26: crcs.seas.harvard.edu. retrieved 2020-08-20.

[24] S. Hassan, “SwellRT: Facilitating decentralized real-time collaboration”, Harvard Berkman Center, Monday, September 26, 2016, 11:30am to 1:00pm, crcs.seas.harvard.edu. retrieved 2020-08-20.

[25] S. Hassan, “How P2P Will Save The World', with Samer Hassan – STEAL THIS SHOW”. 2018. stealthisshow.com. retrieved 2020-08-20.

[26] Horizon, “Collaboration that doesn't give others a license to distribute your stuff”. Horizon: the EU Research & Innovation magazine. https://horizon-magazine.eu/article/collaboration-doesn-t-give-others-license-distribute-your-stuff_en.html (14 June 2017). Retrieved 2020-08-20.

[27] “JetPad”. jetpad.net. retrieved 2020-08-20.

[28] Matrix (XMPP or Matrix.org), “Matrix is an open standard for interoperable, decentralised, real-time communication over IP”. retrieved 2020-08-20.

[29] OSS|www.opensourceschool.fr. “SwellRT : une technologie open source pour applications”. retrieved 2020-08-20.

[30] P2Pvalue blog (2016). https://p2pvalue.eu/. "Special Announcement: P2Pvalue and Google Summer of Code 2016”. P2Pvalue blog. 2016-04-17. retrieved 2020-08-20.

[31] P2Pvalue (2017). “SwellRT: open source framework for real-time collaboration”, P2Pvalue, 2017-10-20, retrieved 2020-08-20.

[32] Programmez! (French Magazine). Par: fredericmazue, mer, 03/05/2017 - 11:4. “SwellRT : un cadre open source de développement d'applications Web collaboratives décentralisées”. https://www.programmez.com/actualites/swellrt-un-cadre-open-source-de-developpement-dapplications-web-collaboratives-decentralisees-25932. retrieved 2020-08-20.

[33] H. Rough (in RoughHaste, 2017-04-23), “Notes on “How P2P Will Save the World”. https://medium.com/roughhaste/notes-on-how-p2p-will-save-the-world-a12db16d1b47. retrieved 2020-08-20.

[34] “SwellRT”|http://swellrt.org/. retrieved 2020-08-20.

[35] "Teem", http://teem.works/. Retrieved Retrieved 2020-08-20.