Hierarchical Real Time Interapplication Communications

Yann Orlarey
GRAME
9, rue du Garet BP 1185
FR 69202 LYON CEDEX 01
Tel (33) 72.07.37.00 Fax (33) 72.07.37.01
e-mail : orlarey@rd.grame.fr

Abstract : *Real time interapplication communications are a key feature in musical multi-task operating systems. Independent applications can therefore be connected and collaborate by exchanging messages and data through communication channels. All these collaborating applications define a virtual network the user can dynamically configurate. The topology of such virtual network specifies the way applications can be connected together.*

This paper introduces a new hierarchical topology we recently implemented in our MidiShare multi-task operating system. This approach offers several advantages and particularly when a large number of applications are involved or in a multi-user context.

Keywords : *Interapplication communications, operating systems, real time.*

INTRODUCTION

One of the research themes we study at GRAME deals with the software architecture of musical applications. As part of this theme, we try to set relevant models and tools for the design and the implementation of musical applications. We take a particular interest, and this is our subject here, in the problematic of inter-process communication and collaboration.

Under the impulse of N. Wiener's Cybernetics, of Shannon and Weaver's Mathematical Theory of Communication, but equally of Sociology, Anthropology, Biology and many other fields, Today Communication and Interaction notions set-up a conceptual framework for analysing complex phenomena such as the organisation of human society, the behaviour of ants colonies, or even the morphogenesis in biological systems. Any organised system implies communication and interaction between its elements. According to the richness of its internal couplings, the system displays emergent properties which are not reducible to the sole properties of its elements.

In Computer Science, the notion of communication tends to acquire an increasingly operative part. Even if von Neumann architecture relies on communication, it is mainly with the apparition of parallel and distributed architectures, intended to improve computer performances, that the part of inter-processors and inter-process communications is confirmed. These new hardware architectures will give birth to many researches on programming languages capable of expressing parallelism [3].

But the notion of distributed and communicating systems will rapidly become a conceptual tool for designing complex applications even on von Neumann machines. In the field of Artifical Intelligence, at the beginning of the Seventies, sequential top-down approach, gave place to distributed bottom-up approach in which more or less autonomous agents collaborate, interact and communicate to obtain the global behavior required [2]. Different levels are taken into account from the collaboration of few expert systems [5] to the interaction of large number of formal neurons [9], including Minsky Society of Mind [6]. But other fields resort to this multi-agent model like Simulation, Artificial Life [4] or Human-Computer Interaction [1].

These few examples stress on the double part played by the notions of communication and interaction. On one side, they constitute a valuable framework for analysing emerging properties and behaviours of complex natural phenomena. On the other side, they set up a powerful technique to modelize and design complex applications.

A HIERARCHICAL COMMUNICATION MODEL

The MidiShare musical operating system [7] developed at Grame, allows a design of musical applications based on a hierarchical communication model. In this model, agents are software objects communicating between each other by exchanging messages via a virtual communication network.

Messages exchanged by agents are data structures including all the informations needed for their interpretation. One subset of MidiShare messages correspond to the whole set of Midi messages but in a format simpler and faster to process. Moreover, each message includes a complete physical address describing the midi channel but also the midi port used among the 256 ports managed by MidiShare (figure 1).

The communication system allows to set connections between agents. The connections are temporal. Each transmitted message is time stamped and delivered one the specific date. The agents together with the connections form an oriented graph (figure 2) which can be dynamically configured by the user or by the agents.

Agents are software objects that exchange messages with external musical device and other agents. In order to simplify connections between agents and also to guarantee a better compatibility between them, MidiShare agents have just one input, for receiving messages, and one output for sending messages (figure 3).

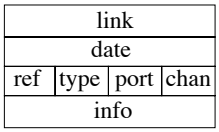

Figure 1 :
A MidiShare message

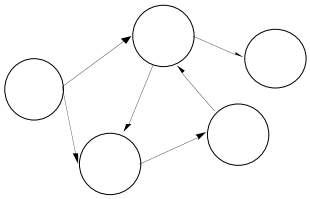

Figure 2 :
The communication graph

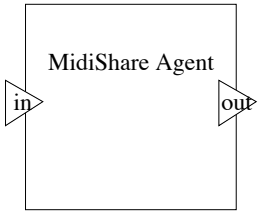

Figure 3 :
A MidiShare agent

In the perspective of a network version of MidiShare we are led to consider a great number of agents, manipulated by several different users. This poses the problem of organising things to ensure the required independence between users. To do that, we use folder-agents (like the folders in a hierarchical file system).

As simple agents, folder agents are equally given an input and an output and can, therefore, be connected to other agents. Any agent is contained in a folder agent, its father. The only exception is the root folder which represent the whole system and is contained in no other.

Simple agents are, a priori, musical applications. But a complex application can itself be organised in term of communicating and cooperating agents grouped together in a folder agent, à la Max [8]. A mechanism allows to keep the content of a folder agent secret and makes the user believe it is a simple agent.

An agent can be connected to itself, to other agents in the same father folder or with the input and output of its father. The input and output of a folder can be connected directly to realize a "THRU".

Actually, the agents together with the connections form a hierarchy of oriented graph, some nodes of which are, in their turn, subgraphs and so on (figure 4). The user, or agents, can dynamically create and delete folders, move agents from one folder to another, and connect agents together (figure 5).

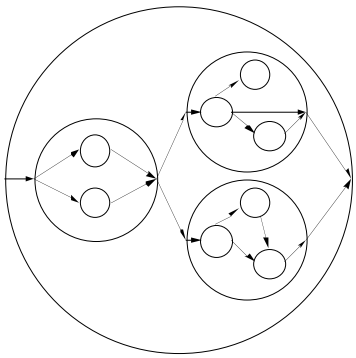

Figure 4 :
A hierarchy of oriented graph

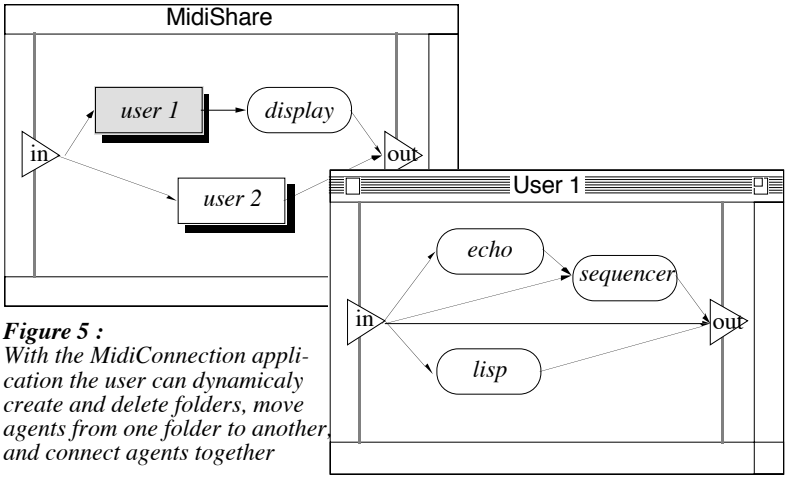

Figure 5 :
With the MidiConnection application the user can dynamicaly create and delete folders, move agents from one folder to another, and connect agents together

The connection mechanisms we have presented here, in particular the "thru" connection of a folder agent bring in two difficulties : a) infinite loops in the graph, b) more than one connection path from an agent to another (figure 6). The algorithm of message distribution we have implemented deals with these cases by detecting loops and ensuring that a message is delivered only once to each destination.

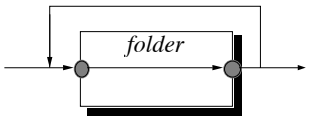

Figure 6a :
An infinite loop

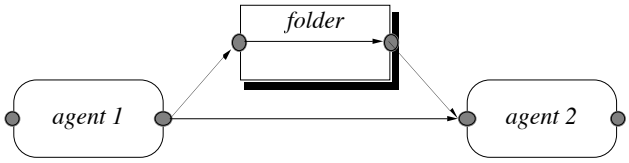

Figure 6b :
Duplicate path between agent 1 and agent 2

On the other hand it allow very flexible connection modes. For instance, the full interconnection of n agents require n^2 connections. It can be brought down to 2n+1 connections using a folder (figure 7). Connecting n agents to m agents require n*m connections. It can also be brought down to n+m+1 connections using and empty folder with a thru connection (figure 8).

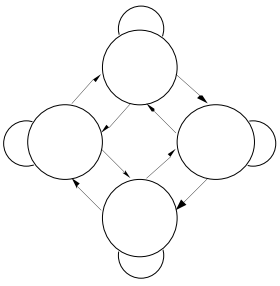

Figure 7a :
*The full interconnection of n agents require n*n connections*

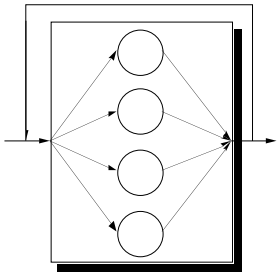

Figure 7b :
*Using a folder the full interconnection of n agents require 2*n+1 conections*

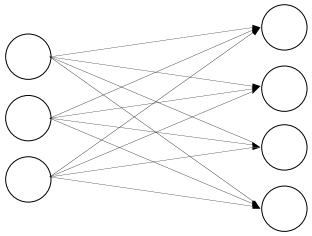

Figure 8a :
*Connecting n agents to m agents require n*m connections*

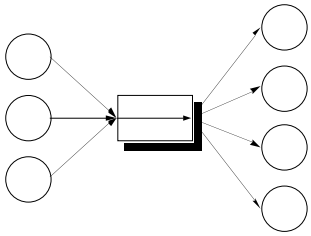

Figure 8b :
Using an empty folder, connecting n agents to m agents require n+m+1 connections

CONCLUSION

The hierarchical communication model we have presented here is well adapted to the musical nature of the considered applications. It can be used efficiently at different levels of organisation from a network of computers to the internal structure of an application.

This model is based on the connection metaphor which is suitable for an operating system. But other communication metaphors are possible (for instance, the blackboard model used in A.I.). As the idea of *socialware* (software with a "social" behavior) will grow, we will need new and powerful communication metaphors.

*
* *

References

[1] COUTAZ J. (1991), *"Interface Homme Machine : un regard critique"*, TSI, Vol 10, n° 1.

[2] HEWITT C.E., BISHOP P., STEIGER R. (1973), *"A Universal Modular ACTOR Formalism for Artificial Intelligence"*, Proc. of the 3th join conf. on A.I. 1973.

[3] HOARE C.A.R. (1978), *"Communicating sequential processes"*, Comm. of the ACM, Vol 21, n° 8.

[4] LANGTON C.G. (1987), *"Artificial Life"*, Proc. of the first workshop on Artificial Life, Los Alamos National Laboratory, Sept. 87 - Santa Fe Institute, Addison-Wesley, 1989.

[5] LENAT D.B. (1975), *"Beings: knowledge as interacting experts"*, Proc. of the 4th join conf. on A.I. 1975.

[6] MINSKY M. (1986), *"The Society of Mind"*, Simon and Schuster 1986.

[7] ORLAREY Y., LEQUAY H. (1989), "*MidiShare : a Real Time multi-tasks software module for Midi applications"* , Proc. of the I.C.M.C. 1989,

[8] PUCKETTE M. (1988). "*The Patcher"* - Proc. of the I.C.M.C. 1988.

[9] RUMELHART E., McCLELLAND J.L. (1986), *"Parallel Distributed Processing"* MIT Press 1986.