MidiShare joins the Open Source Softwares

Dominique Fober, Yann Orlarey, Stéphane Letz

MidiShare joins the Open Source Softwares

Dominique Fober, St phane Letz, Yann Orlarey
GRAME Research Laboratory 9 rue du Garet, BP 1185, 69202 LYON Cedex 01, France
Email: [fober, letz, orlarey]@rd.grame.fr

Abstract

MidiShare is a realtime, multi-tasks operating system dedicated to musical MIDI applications. It was awarded the Apple Trophy (1989), the Paris-City price (1990) and more recently, the Max d'Or at the Bourges International Musical Software Competition (1999). Multi-platform support, powerful inter-applications communication, accurate realtime performances are among the significant services provided by the kernel. Freely available on the Internet to developers since several years, MidiShare is now supported by a growing number of projects. Its developers mailing list count 250 members on average. The MidiShare project itself is reaching a state which requires a different management policy: the MidiShare source code is now publicly available to allow collaborative contributions over the Internet. The poster session will present this new project, including changes in the kernel architecture, the source code portability issues and the organization of the collaborative development over the Internet.

1. Introduction

MidiShare has been designed ten years ago, in response to commonly met problems in the development of realtime musical applications. The main motivation of this work took source in the standard operating systems shortcomings and in their unsuitability to the musical field requirements: generally, existing systems don't allow critical resources sharing, they are also not suitable to take account of time and communication in realtime.

With the development of the multimedia applications, one could expect to see the operating systems design evolving towards a better support of these requirements. On the contrary, today, all the systems are progressively integrating Unix-like characteristics (protected memory space, pre-emptive multi-tasking) which complicates radically the implementation of realtime inter-applications communications and realtime capabilities. For this reason, the works in the field of the software architecture for musical applications are more relevant than ever.

The extent of the MidiShare project exceeds now our simple research purpose: many of the MidiShare related tasks concern porting on new platforms. The number of currently supported systems multiplies of as much any improvement implementation. As the MidiShare based project number regularly increases, we have decided to open the kernel to a collaborative development over Internet. Therefore, the MidiShare source code is now publicly available under the GNU Library General Public License and MidiShare is part of the Open Source Softwares[1].

[1] see at http://www.opensource.org/ for more information about the Open Source Definition

2. The MidiShare revised micro-kernel

The MidiShare architecture was presented several times [Orlarey, Lequay, 1989], [Orlarey, 1990], [Fober & al 1996]. In order to facilitate portings on new platforms and to concentrate on critical services, it has been slightly revised. The figure 1 summarizes its different components:

- a *Scheduler*: in charge of delivering scheduled events and tasks at the right date. It allows events to be sent in the future as well as functions to be called in the future.
- a *Time Manager*: maintains the current date of the system.
- a *Communication Manager*: routes events received from the scheduler to the client applications and the ports manager.
- a *Task Manager*: in charge of calling the tasks delivered by the scheduler.
- a *Memory Manager*: specially designed for real-time operations at interrupt level.
- a *Ports Manager:* it is a new component. It replaces the previous MIDI input and output drivers to provide a more general driver mechanism by the way of plugging. It is intended to facilitate the support of protocols other than MIDI (like Ethernet for example [Fober, 1994]) and to allow developers to provide new drivers without having to recompile the kernel.

For more information about this architecture, you can refer to the MidiShare Developer Documentation [Grame, 1990] or to the MidiShare Kernel Development Guide [Grame, 1999] which details also the source code organization and the key portability issues.

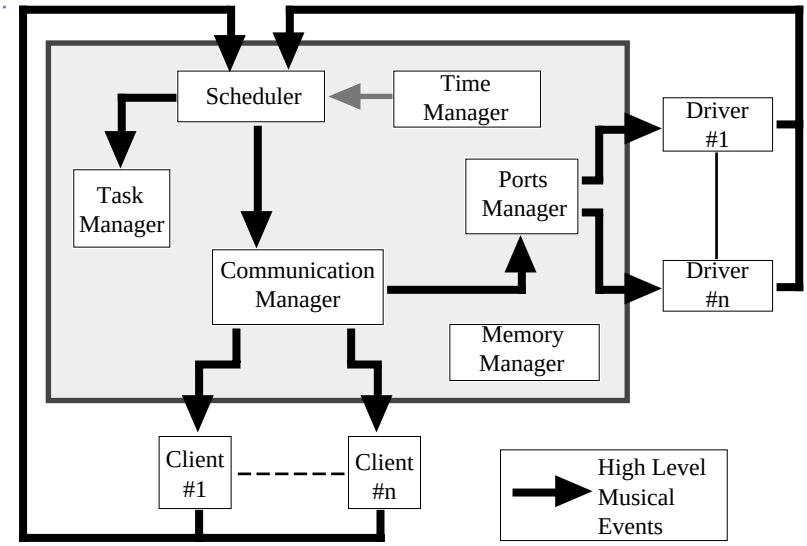

Figure 1 : the new conceptual model of MidiShare

3. Portability issues

Several low level implementation parts of the kernel are platform dependent and generaly concern critical services provided by the host operating system. Among them are: setting up an interrupt service routine, sharing the events memory, switching from the MidiShare kernel to a client process and synchronizing the client processes. Modern operating systems provide high level functions to perform all these tasks but they are generaly not suitable to operate in a realtime context. In particular, accuracy and efficiency issues are critical for a correct operating of the kernel. The implementation problems raised for each point are detailed below.

3.1. The MidiShare process

The MidiShare process represents the entity that owns the necessary resources for the kernel operations. It must ensure of the permanence of these resources all along the kernel active period, from wake up time until sleep time. The MidiShare process does not necessary refer to a process as commonly defined by the operating systems. Its implementation is generally dictated by the host operating system and may or may not be such a process. One of the design issue is to choose the most appropriate system resource. Efficiency and accuracy must be particularly taken into account: for example, if the MidiShare process is implemented as an operating system process and if the process context switching costs are high, the resulting kernel will certainly be inefficient as it will have to pay these costs every millisecond.

3.2. Memory allocation

Memory is allocated and owned by the MidiShare process. It can freely use any way to allocate the kernel internal memory. It is different for events and filters memory:

- events memory must be shared by the MidiShare process and all the client applications. It is the responsibility of the MidiShare process to allocate such a memory at wake up time.
- filters memory must be shared by its application owner and the MidiShare process. It is different from the events memory because a filter is private to one application. Filter memory is allocated by the MidiShare process at application request.

The source code clearly separates these services using an allocation function which takes as argument the type of the desired memory.

3.3. Tasks and Alarms

The MidiShare kernel provides realtime tasks and realtime alarms to its client applications. Realtime tasks are stored in typeProcess events, then inserted in the scheduler and activated at interrupt level by the MidiShare process at falling date. Realtime alarms are activated at application request in two cases:

- when a context change occurs: it is an application alarm, it can be activated at any time, at interrupt or user level.
- when new events are stored in an application fifo: it is a receive alarm, it will always be activated at interrupt level.

The way used to activate a client task is platform dependent. Some operating systems allow a process to directly call the client code, some other prevents such mechanisms by keeping separate address spaces for each application; in this case, activating a task consists generally in waking up the corresponding thread, which raises all the problems of accuracy, priority and context switching costs.

3.4. Processes and synchronization

Concurrent access to critical sections of the MidiShare code raises the synchronization problems. Preemptive operating systems provide the necessary mechanisms to solve these problems but again, these mechanisms are generally not suitable in a realtime context.

The current implementation generally avoids the use of semaphores by the way of lock free shared linked lists [Anderson & al. 1995]. The principle consists in trying to modify a shared linked list using the synchronization mechanisms generally available with modern micro-processors instruction set: for example, the Motorola PowerPC microprocessor family allows to put reservations on a memory zone, then, conditional instructions operates only if the reservation is not altered, on the contrary, the program have to loop until success.

The only semaphore defined by the system is used to synchronize the kernel wake-up and sleep periods.

4. The collaborative development organization

4.1. MidiShare mailing lists

Two MidiShare mailing lists exist for both MidiShare applications and kernel development:

- the "midishare-dev" list is for support and discussion about MidiShare compatible applications development.
- the "midishare-kernel" list is for support and discussion about MidiShare kernel development.

Anyone who plans to contribute to the kernel development should subscribe to the "midishare-kernel" mailing list where are discussed all the changes and the development issues.

4.2. Contributions rules

Many of the free software projects operate as a "meritocracy": the more good code you contribute, the more you will be allowed to contribute. We are setting up such a process for the MidiShare kernel development. Below are the basic rules that we are considering for the project management.

The MidiShare Group is a core group of contributors which is initially formed from the project founders. It may be augmented from time to time when core members nominate outstanding contributors and the rest of the core members agree.

The MidiShare Group is a meritocracy: the more work you have done, the more you are allowed to do. The group founders set the original rules, but they can be changed by vote of the active members. New members of the MidiShare Group will be added when a frequent contributor is nominated by one member and approved by the voting members. In most cases, this new member has been actively contributing to the group's work for over six months.

Primary method of communication is the midishare-kernel mailing list. It is the place to discuss new features to add, bug fixes, user problems, release dates, etc... The actual code development takes place on the developer's local machines, with proposed changes communicated using a patch and committed to the source repository by one of the core developers. Anyone on the mailing list can vote on a particular issue, but we only count those made by active members or people who are known to be experts on that part of the kernel. Changes to the code are proposed on the mailing list and usually voted on by active members.

5. The project Internet locations

The MidiShare kernel source code is located at:

`ftp://ftp.grame.fr/pub/midishare-kernel/`

Contributions to the source code will be done using a version control system. We plan to use CVS but right now, it is not yet in service.

The MidiShare mailing lists can be reached using majordomo at: `majordomo@rd.grame.fr`

We are also publishing the MidiShare libraries source code under a free software license.

For up-to-date information, see the MidiShare web pages at: `http://www.grame.fr/MidiShare/`

References

[Anderson & al. 1995] James H. Anderson, Srikanth Ramamurthy, Kevin Jeffay. *Real-Time Computing with Lock-Free Shared Objects*. Proceedings of the 16th IEEE Real-Time Systems Symposium, Pisa, Italy, 1995, pp.28-37

[Fober & al. 1996] Fober D., Letz S. Orlarey Y. *Recent developments of MidiShare* - Proceedings of the ICMC 1996, ICMA, San Francisco, p.40-42

[Fober 1994] Fober D. *Real time, musical data flow on Ethernet and MidiShare software architecture*. Proceedings of the ICMC, 1994, ICMA, San Francisco, pp. 447-450

[Grame 1990] *MidiShare Developer Documentation*. Grame 1990, Lyon

[Grame 1999] *MidiShare Kernel Development Guide*. Grame 1999, Lyon

[Orlarey 1990] Orlarey Y. *An Efficient Scheduling Algorithm for Real-Time Musical Systems*. Proceedings of the ICMC, 1990, ICMA, San Francisco, pp.194-198

[Orlarey, Lequay 1989] Orlarey Y., Lequay H. *MidiShare: a Real Time multi-tasks software module for Midi applications*. Proceedings of the ICMC, 1989, ICMA, San Francisco, pp.234-237