RMCP: Remote Music Control Protocol
— Design and Applications —

Masataka Goto Ryo Neyama Yoichi Muraoka

School of Science and Engineering, Waseda University
3-4-1 Ohkubo Shinjuku-ku, Tokyo 169, JAPAN.
{goto, ryo, muraoka}@muraoka.info.waseda.ac.jp

Abstract

This paper describes the design and various applications of a communication protocol in a distributed music system that integrates MIDI and computer networks. We designed this protocol, called RMCP (Remote Music Control Protocol), to share symbolized musical information through the networks. Most previous related protocols were connection-oriented and did not emphasize efficient information sharing among processes. Since the RMCP is a UDP/IP-based connection-less protocol, it supports broadcast-based information sharing without the overhead of multiple transmission. To enable real-time MIDI handling, it also supports time scheduling using time stamps. RMCP has been used in various applications such as a music-controlled virtual dancer, a virtual jazz session system, and a remote session system.

1 Introduction

Computer-network protocols for symbolized musical information such as MIDI information have been taking on importance with the popularity of network computing because of various applications. Integrating MIDI and computer networks such as LANs (local area networks) and WANs (wide area networks), for example, enables network musical applications such as live MIDI broadcasts and remote sessions via network, computer supported sessions such as human-computer improvisation with various interactive-graphics displays, and distributed implementation of music-related software to achieve good load-balancing and to exploit various facilities connected with different computers. Such applications require efficient information sharing over the network.

Most previous MIDI-based network protocols [1, 2, 3, 9] were connection-oriented and did not emphasize low-latency information sharing among multiple distributed processes. MIDI is also weak in efficient information sharing among multiple devices, although it is somehow possible to achieve all-to-all communication among those devices. In addition, its bandwidth is very limited and it was designed just for local communication. Although non-MIDI-based music protocols [8] have been proposed, most of them presuppose the use of special devices.

In this paper we propose a music protocol, called RMCP (Remote Music Control Protocol), which was designed for sharing symbolized musical information through computer networks. This is a communication protocol in a distributed cooperative system that integrates MIDI and computer networks such as the Ethernet. Since it is a connection-less protocol on the UDP/IP, it naturally supports broadcast-based information sharing without the overhead of multiple transmission. Although it was initially designed for use on a reliable LAN, it has been extended to support WANs such as the Internet by using a gateway program that can relay RMCP packets over a WAN. It also supports time scheduling using a time stamp in each packet in order to enable real-time MIDI handling.

Since 1992 we have developed not only the basic MIDI I/O processes but also various RMCP-based applications such as (1) an interactive performance system in which there is a music-controlled computer-graphics dancer that two musicians can choreograph by their improvisation in real time; (2) a virtual jazz session system in which a human pianist can interplay with a virtual bassist and a virtual drummer who have computer-graphics bodies for playing their instruments and making gestures; (3) a networked session in which multiple musicians play together via the Ethernet with the assistance of visualization programs; (4) *Improvision*, with which untrained people can improvise music and interact with each other; and (5) *RemoteGIG*, an innovative remote session over the Internet that has a long delay, in which each musician improvises while listening to other players' performances that are delayed for the constant period of a repetitive chord progression under a fixed tempo.

We have implemented RMCP programming libraries in both C and Java languages and have used the protocol on various operating systems, such as IRIX-5.3, IRIX-6.2, Solaris-2.5, SunOS-4.1.3, HP-UX, Windows-95, Windows-NT, and Linux-2.0. RMCP has already been utilized for various purposes in several laboratories.

2 RMCP

RMCP transmits various musical information in the form of RMCP packets, which are shared by multiple processes on distributed computers. For example, each MIDI message such as note-on and note-off is packed in a packet and sent to other processes via a LAN.

Table 1: The basic RMCP servers and clients.

RMCP Server	Description
RMCP Sound Server	send MIDI messages of received packets to a MIDI instrument
RMCP Display Server	visualize MIDI messages of received packets in the form of a piano keyboard
RMCP Animation Server	generate music-driven real-time computer graphics corresponding to received packets
RMCP Recorder	record all received packets, with the received time stamps, in an RMCP Packet Record File
RMCP Client	**Description**
RMCP MIDI Receiver	receive MIDI messages from MIDI instruments
RMCP MIDI Station	substitute a computer keyboard and mouse for a MIDI keyboard instrument
RMCP SMF Player	play a Standard MIDI File
RMCP Player	play an RMCP Packet Record File

RMCP is based on the server-client model in which multiple servers receive requests from various clients. All RMCP-based processes are therefore categorized into RMCP clients and RMCP servers, which can be considered input and output devices, respectively. An RMCP client generates RMCP packets such as those including MIDI messages received from a MIDI device (MIDI IN) and including messages corresponding to user's interactions, and it broadcasts the packets to all RMCP servers. Since this broadcast is a connection-less one-directional transmission, there is no acknowledgement packet from the servers. Each RMCP server, on the other hand, receives all the broadcast packets and utilizes them in various ways, such as controlling MIDI devices (MIDI OUT) for outputting sounds, visualizing musical information, and producing music-driven computer graphics. RMCP servers can thus share all the broadcast packets without the overhead of multiple transmission. Moreover, it is also possible to add various servers such as a visualizing server without any extra packets.

2.1 RMCP Servers and RMCP Clients

The design policy in developing RMCP servers and clients is to implement necessary functions as small different processes so that they can be reusable. This policy facilitates system implementation and expansion and enables those servers and clients to be allocated on distributed computers to achieve good load-balancing.

We have developed the basic servers and clients listed in Table 1. Figure 1 shows an example of using these servers and clients. A user plays on a MIDI instrument with an RMCP MIDI Receiver or on a computer keyboard with an RMCP MIDI Station. When a MIDI instrument is used, MIDI messages from the instrument are received and broadcast as RMCP packets. When a computer keyboard is used, each key is assigned to a different note (pitch) and the corresponding MIDI message is generated and broadcast. A user can also play a Standard MIDI File using an RMCP SMF Player, which broadcasts the contents of the file as RMCP packets. If RMCP packets broadcast by these clients are received by RMCP Sound Servers, included MIDI messages are sent to MIDI instruments according to their time stamps as described in Section **2.2**. Those packets can also be visualized by RMCP Display Servers in the form of a piano keyboard and can be utilized simultaneously by various RMCP Animation Servers to generate music-driven real-time computer graphics such as virtual dancers and musicians.

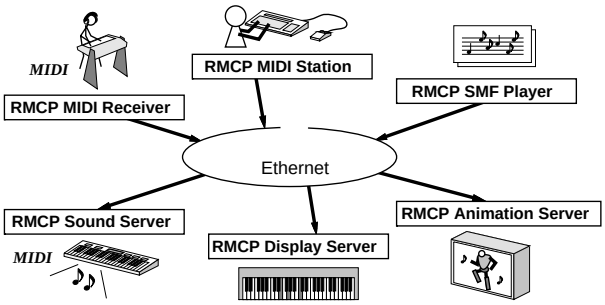

Figure 1: An example of using RMCP servers and clients.

2.2 Time Scheduling

RMCP supports time scheduling using time stamps for real-time packet processing. There are two kinds of RMCP packets: those with a time stamp and those without a time stamp. The time-stamped packets received before their time stamps are kept in each server till the time of their time stamps and are then processed on time and in order. This processing compensates for variations in network latency. The time-stamped packets received after their time stamps and the packets without time stamps are processed immediately.

This time scheduling requires time synchronization among distributed computers because it is necessary for RMCP servers on different computers to process packets with the same time stamp at the same time. We therefore introduced, on each computer, an RMCP Time Synchronization Server (*RMCPtss*) that enables all RMCP servers to handle the time stamps of received packets as if all the internal clocks of the different computers were synchronized.[1] RMCP server developers thus need not care about the time synchronization.

Each RMCPtss makes an offset table of temporal differences between its internal clock and the internal clocks of other computers and periodically broadcasts a packet including the table with the transmitted time stamp (its current internal clock time). When an RMCPtss receives the table from another RMCPtss, it updates the corresponding offset (temporal difference) in its own table by calculating the difference between the transmitted time stamp of the received table and its received time measured

[1]There are other solutions, such as NTP (Network Time Protocol) [RFC-1119] that synchronizes the internal clocks using administrative authority. On the other hand, our solution does not require administrative authority because it does not directly adjust the internal clocks.

by its own internal clock.

Each RMCP server, on the other hand, receives the table from the RMCPtss on the same computer. Every time each server receives an RMCP packet with a time stamp, it adjusts the time stamp by using the sender's offset in the table so that it can compensate for the internal clocks' difference.

3 Implementation of RMCP

We designed the communication protocol RMCP and have implemented RMCP programming libraries and various servers and clients. Under the assumption that the RMCP will be used on a reliable low-latency network such as an Ethernet LAN, RMCP uses the UDP/IP (which is an unreliable connection-less protocol) as its transport protocol and does not ensure the reliability of communication except for remote communication over a WAN such as the Internet.

3.1 RMCP Packets

Each RMCP packet consists of a message header and body. The header specifies the message type, time stamp (with millisecond resolution), target server identifier, sender identifier, and message length. Some of the basic message types are MIDI information for transmitting MIDI messages, beat information for beat synchronization, chord information for broadcasting chord name and voicing, and animation information for controlling computer graphics.

3.2 RMCP over the Internet

Since RMCP was initially designed for use on a reliable LAN, it was hard to use on an unreliable WAN such as the Internet. We have therefore introduced RMCP Gateways that can provide the bidirectional relay of RMCP packets over the Internet by using the TCP/IP, which is a reliable connection-oriented protocol. A pair of RMCP Gateways connects two remote LANs: each gateway relays the RMCP packets broadcast on its LAN to the gateway on the other side of connection and also broadcasts the packets relayed from the other gateway.[2] RMCP servers and clients can thus communicate as if the different LANs connected by the gateways were the same network with the exception of the network latency.

It is impossible to avoid the network latency over the Internet: even signals traveling at the speed of light need more than 66 ms to reach halfway around the globe. We therefore consider that the RMCP should provide a certain constant latency with very small deviation. A pair of gateways measures and reports the network latency at the start so that we can specify enough large constant latency to avoid the latency deviation. The pair then punctually relays the packets with the specified latency. This enables live MIDI broadcasts and an original remote session *RemoteGIG* (described in Section **4.5**).

[2] Note that the relayed packets should be marked for the RMCP Gateways in order to avoid a possible infinite loop of the relay.

3.3 Experimental Results

We have implemented RMCP programming libraries in both C and Java languages and have tested the protocol on various computers and operating systems, such as IRIX-5.3, IRIX-6.2, Solaris-2.5, SunOS-4.1.3, HP-UX, Windows-95, Windows-NT, and Linux-2.0. We found that RMCP facilitated the implementation and expansion of distributed music applications, especially real-time and interactive applications.

We measured the communication delay when RMCP packets traveled from a client to a server on different computers. We tested the C-based RMCP libraries on two different SGI Indigo2s (IRIX-5.3) connected via the 10 Mbps Ethernet, The delay ranged from 0.28 to 1.24 ms, and the average delay was 0.30 ms (standard deviation = 0.06 ms). This result shows that RMCP is fast enough compared with the MIDI data transmission rate (31.25 Kbits/sec).

4 Applications

Various RMCP-based applications have been developed, and this section introduces five of them.

4.1 Virtual Dancer "Cindy"

We have developed an interactive performance system in which two players, a drummer and a guitarist or pianist, can together choreograph a virtual dancer called *Cindy*[3] by their musical improvisation in real time [7] (Figure 2). This system enables the players to communicate using both auditory and visual information. Because the players not only improvise together as in a conventional jam session but also observe the virtual dancer whose motion is changed according to their musical performance, they can interact with each other not only through music but also through 3-D computer animation.

This application consists of an RMCP Animation Server for Cindy, an RMCP Music Analyzer (application-specific server) to analyze the guitarist's improvisation, an RMCP MIDI Receiver, and an RMCP Sound Server.

4.2 VirJa Session

We have proposed a virtual jazz session system called *VirJa Session* in which a human player and computer players can communicate not only by listening to each others' performances but also by seeing each others' bodies and gestures [5]. The human player sees the bodies and gestures of other computer players shown on 3-D real-time computer graphics, and can feel their presence as if they were actually playing together. In addition, each computer player reacts to the gestures of the human player by using a video camera. We can thus achieve multimodal interaction among all players.

The current implementation of this system deals with a jazz piano trio consisting of a human pianist, a computer

[3] Cindy can also dance in time to music by using our beat-tracking system [6].

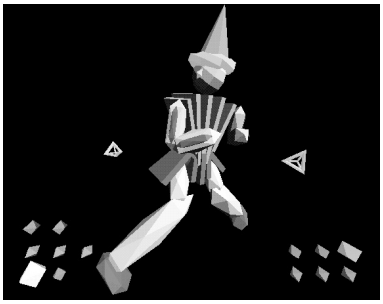

Figure 2: Virtual Dancer "Cindy".

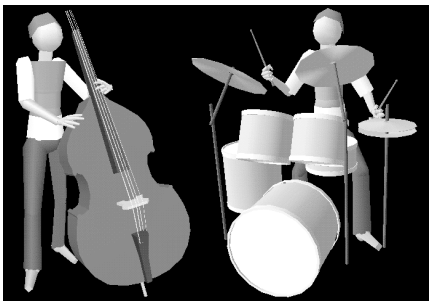

Figure 3: VirJa Session.

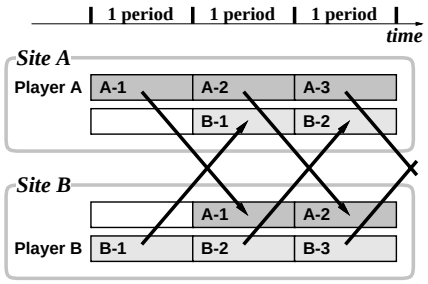

Figure 4: RemoteGIG.

bassist, and a computer drummer (Figure 3). It consists of an RMCP MIDI Receiver, an RMCP Sound Server, and four kinds of application-specific RMCP servers and clients described in [5].

4.3 Networked session

The RMCP has been used in a local networked session in which several players perform music together via the Ethernet [4]. The players can not only listen to other players' performances but can also see visualized performances that facilitate musical cooperation. This application requires RMCP MIDI Receivers or RMCP MIDI Stations, RMCP Sound Servers, and RMCP Display Servers. An RMCP SMF Player is useful for background accompaniment. An RMCP Recorder and an RMCP Player are also available for recording and playing back the performance.

4.4 Improvision

We have developed a musical-instrument interface, called *Improvision*, which enables an untrained novice to improvise unconventional music easily by clicking and dragging a computer mouse [4]. Multiple players can interact with each other by playing RMCP Improvisions (application-specific clients) instead of RMCP MIDI Stations and can see other players' performances on RMCP Improvision Display Servers (application-specific servers) instead of RMCP Display Servers.

4.5 RemoteGIG

RemoteGIG is an innovative remote session over the Internet that has a long delay. RemoteGIG overcomes the network latency and offers a new possibility for future remote sessions. It assumes that the tempo is constant and the chord progression is repetitive, like the 12-bar blues chord progression. Figure 4 illustrates how remote players interact with each other over the network. RemoteGIG turns the network latency to its advantage: each player improvises while listening to other players' performances that are delayed for the constant period of the repetitive chord progression. Because the progression is repetitive, the delayed performances can fit the chords. RemoteGIG requires RMCP Gateways in addition to the servers and clients described in Section **4.3**. In particular, an RMCP SMF Player is needed for playing background drums to keep the constant tempo.

5 Conclusion

We have described a network protocol called *RMCP* that enables multiple distributed processes to share symbolized musical information such as MIDI information. It supports efficient broadcast-based information sharing over a LAN and time-scheduling using time stamps to enable real-time MIDI processing, and it also enables live MIDI transmission over a WAN such as the Internet. RMCP has been implemented on various operating systems and has been utilized for various purposes.

We plan to distribute our RMCP software package on our WWW page "http://www.info.waseda.ac.jp/muraoka /members/goto/RMCP/" and to provide an API to make RMCP and MIDI usable in Java applets.

Acknowledgments

We thank the SALA (Science Art Laboratory), which enabled the first author to originate this project. In particular, we thank Yuji Hashimoto, Shigeru Chiba, and Shin Miyakawa for their support in the early stages of the project. We also thank Yoshiaki Kikuchi, Isao Hidaka, Tetsuya Abe, Hideaki Matsumoto, Yosuke Kuroda, Mitsukazu Washisaka, Keiji Hirata, and Yoichi Nagashima for their helpful comments and cooperation.

References

[1] T. Aoyagi and K. Hirata. Music server system − distributed music system on local area network −. *Journal of Information Processing*, 15(1):1−9, 1992.

[2] D. Fober. Real-time Midi data flow on Ethernet and the software architecture of MidiShare. In *Proc. of ICMC 1994*, pages 447−450, 1994.

[3] D. Fober, S. Letz, and Y. Orlarey. Recent developments of MidiShare. In *Proc. of ICMC 1996*, pages 40−42, 1996.

[4] M. Goto and Y. Hashimoto. A distributed cooperative system to play MIDI instruments − toward a remote session − (*in Japanese*). *IPSJ SIG Notes*, 93(109):1−8, 1993.

[5] M. Goto, I. Hidaka, H. Matsumoto, Y. Kuroda, and Y. Muraoka. A jazz session system for interplay among all players − VirJa Session −. In *Proc. of ICMC 1996*, pages 346−349, 1996.

[6] M. Goto and Y. Muraoka. Beat tracking based on multiple-agent architecture − a real-time beat tracking system for audio signals −. In *Proc. of ICMAS 1996*, pages 103−110, 1996.

[7] M. Goto and Y. Muraoka. A virtual dancer "Cindy" − interactive performance of a music-controlled CG dancer −. In *Proc. of the Lifelike Computer Characters '96*, page 65, 1996.

[8] K. McMillen, D. Simon, and M. Wright. A summary of the ZIPI network. *CMJ*, 18(4):74−80, 1994.

[9] O. Nielsen. MIDI and audio via ISDN. In *Proc. of ICMC 1994*, pages 451−454, 1994.