Metodo per il recupero in tempo reale degli errori di trasmissione di segnali audio in formato MIDI

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 May 2022 (27.05.2022)

WIPO | PCT

(10) International Publication Number
WO 2022/107103 A2

(51) International Patent Classification:
G10H 1/00 (2006.01)

(21) International Application Number:
PCT/IB2021/060855

(22) International Filing Date:
23 November 2021 (23.11.2021)

(25) Filing Language: Italian

(26) Publication Language: English

(30) Priority Data:
102020000028085 23 November 2020 (23.11.2020) IT

(71) Applicant: **POLITECNICO DI TORINO** [IT/IT]; C.so Duca degli Abruzzi, 24, 10129 Torino (IT).

(72) Inventors: **ROTTONDI, Cristina Emma Margherita**; c/o POLITECNICO DI TORINO, C.so Duca degli Abruzzi, 24, 10129 Torino (IT). **BIANCO, Andrea**; c/o POLITECNICO DI TORINO, C.so Duca degli Abruzzi, 24, 10129 Torino (IT). **CUCCARESE, Antonio**; c/o POLITECNICO DI TORINO, C.so Duca degli Abruzzi, 24, 10129 Torino (IT).

(74) Agent: **MOLA, Edoardo** et al.; c/o Praxi Intellectual Property S.p.A., Corso Vittorio Emanuele II, 3, 10129 Torino (IT).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

(54) **Title:** METHOD FOR REAL-TIME RECOVERING ERRORS IN TRANSMISSION OF AUDIO SIGNALS IN MIDI FORMAT

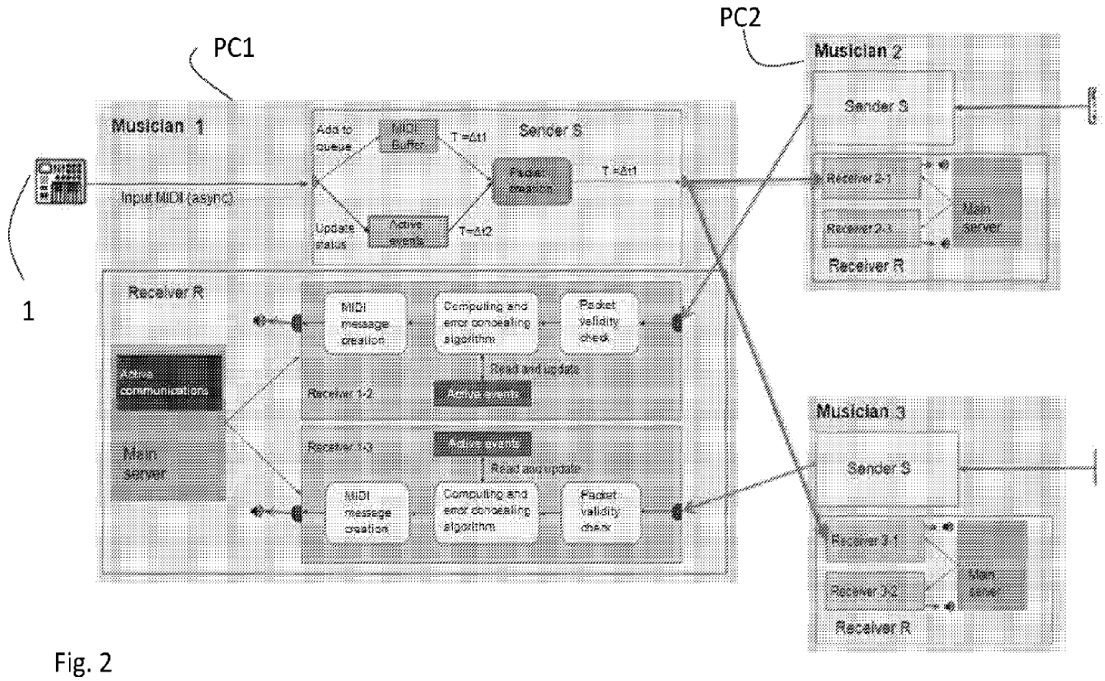

Fig. 2

(57) **Abstract:** A method for so-called "Network Music Performance" for exchanging MIDI data e.g. of a musical instrument is presented, comprising the step of including in data, i.e. in a refresh packet, at successive time intervals, a list of values defining a state of said musical instrument in successive and progressive time instants representative of the action of a musician on an instrument in respective said time instants.

[Continued on next page]

WO 2022/107103 A2

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

— *in black and white; the international application as filed contained color or greyscale and is available for download from PATENTSCOPE*

Method for real-time recovering errors in transmission of audio signals in MIDI format

DESCRIPTION

TECHNICAL FIELD

The present disclosure relates to a method for correcting errors due to packet losses or delays containing audio signals in differential symbolic format (such as MIDI encoding) exchanged between two or more musicians connected to a telecommunications network for Networked Music Performance applications.

BACKGROUND

A differential symbolic audio format, e.g. as defined in Musical Instrument Digital Interface (MIDI) encoding, can be generated and received by different devices, as shown in Figure 1: an electronic musical instrument, also called MIDI instrument, a traditional analogic musical instrument connected to a MIDI converter, an electronic device/computer capable of generating/receiving a MIDI stream, even a control stream. Figure 1 also shows a block called MIDI extender, which is object of this disclosure's method, and which can be implemented both in hardware (HW) or in software (SW), whose functionality will be described later. The MIDI standard allows the generation of digital data such as representations of musical notes to be played and/or control commands that manage the type of sound being played and that can be transmitted to other devices, and/or be used by software applications for recording, playing or

editing the digital data itself.

The MIDI standard thus defines a set of messages for communication between devices. A stream of MIDI messages can be transmitted from a sender device to one (or more) receiving device(s) through a MIDI cable, which directly connects these devices, for example a musical instrument and a sound card. Optionally, data between devices may be transmitted from a sender device over a data network connection, such as a short-range wireless network such as a WiFi network or a geographic telecommunications network such as Internet, by sending MIDI messages encapsulated in data packets.

The primary requirement for most of the data transfer applications in a data network is reliability of packet transmission, i.e., the probability that packets are received correctly. In particular, for packets containing MIDI messages, it is important to manage any packet loss due to network congestion, for example, in order to maintain a good sound quality. Other applications, such as multimedia applications with real-time requirements, latency plays a key role, i.e., the time interval between sending a data packet by a sender and receiving it by a receiver. Low latency becomes crucial, especially for real-time music sessions, in applications called network music performance (NMP).

For standard data application or streaming media application without a real-time requirement, a reliable transport layer protocol such as TCP (Transmission Control Protocol) is preferred to reduce network data loss. This protocol rebroadcasts lost or missing data over a network after a certain amount of time, thus introducing an increase in latency as well as in network traffic.

When a musician produces a MIDI data stream that is transmitted to a remote receiving device to play the sound, a high transmission delay may be noticed by a listener, other musicians contributing to the performance, and a possible audience. Therefore, for sound applications with real-time requirements such as NMP, MIDI messages can be transmitted over a network over unreliable protocols such as User Datagram Protocol (UDP) to avoid retransmission mechanisms of TCP. This choice avoids increased latency due to retransmission of data from reliable transport protocols, but does not provide for any action by UDP in case of data loss.

For example, considering a networked music session among a group of musicians who interact with each other while being in different physical locations, to play together as if they were in the same place. The introduction of silence intervals or delays can compromise the quality of the musical session, causing them to loose rhythm or harmony. In these cases, data loss should be minimized, or better yet, removed, and latencies should be as low as possible to ensure a natural musical interaction.

Several techniques are known in prior art that attempt to mitigate transmission errors through an unreliable network connection.

For example, with reference to a wireless network between mobile devices e.g., smart phones, in U.S. Patent 6,898,729 the MIDI data are analyzed and classified into two categories (critical and non-critical) and further transmitted through a reliable and an unreliable connection, respectively. Typically MIDI commands that specify when a specific note should be played and when it should be stopped are transmitted through the reliable connection. This solution,

however, involves using two connections i.e. reliable and unreliable for a data transmission.

In U.S. Patent 7,447,639, which refers to data sent in streaming, i.e., not in real- time, a system of error concealment is disclosed, at the receiving device end, by extracting rhythm patterns of music signals embedded as auxiliary data at the transmitting device end. This solution involves rhythmic pattern extraction and analysis and is best suited for precise sound reproduction only, for example when streaming music where complex and fast drum rhythm patterns are present. The need to handle MIDI sound sources with low latency is not taken into account. By contrast, rhythm pattern analysis is a complex operation that takes relatively long time and is not particularly suitable for low-latency exchange and mixing of sound tracks generated by two or more users connected to the network.

In US Patent 9,601,097 B2, with reference to data transmission via WiFi between MIDI musical instruments and other MIDI devices, a new UDP-based protocol is disclosed that provides for retransmission of non-received data, implementing a memory used for storing sent data. Every data packet received triggers an acknowledgement transmission. This memory becomes the reference for the retransmission of the entire previously transmitted and unconfirmed data history. This solution involves a mechanism of acknowledgement of sent packets with related traffic generation linked to confirmations and retransmissions. This mechanism increases sound reception latency and is therefore not suitable for real-time MIDI signal transmission for NMP applications.

US patent application 2004/0154460, with reference to short-range radio

frequency data exchange networks, discloses a system for replacing missing or corrupted information in order to avoid silence intervals. This solution, after assessing error severity, triggers a data recovery phase by means of an asynchronous auxiliary channel through a server. The solution therefore assumes both the use of an auxiliary channel and the use of a server and being based on a retransmission mechanism increases sound signal latency.

The Internet Draft "An implementation guide to the MIDI wire protocol packetization (MWPP)", with reference to the data transmission via generic telecommunication network between MIDI musical instruments and other MIDI devices, describes a new protocol based on UDP that provides for the retransmission of non-received data, implementing a mechanism to store a list (called "recovery journal") containing the most recent modification event of the parameter(s) associated to each possible MIDI command, on the basis of a time stamp that identifies the instant in which such modification occurred. Each data packet received triggers an acknowledgement transmission, which contains itself a time stamp. The transmitter compares the value of the most recent time stamp obtained from the receiver with the time stamps associated with each of the events stored in the list and removes the events whose time stamps are earlier than the one received. The remaining events are included in the next packet sent.

This solution involves an acknowledgement mechanism for the packets sent with the related traffic generation linked to the acknowledgements. In addition, the transmission of the time stamp associated with each event in the recovery journal causes a significant increase in the amount of data transmitted.

STATEMENT OF INVENTION

The purpose of present invention is to solve at least partially drawbacks related to MIDI data transmission or other differential symbolic protocol over a data exchange network avoiding i) the introduction of acknowledgement systems conceptually similar to typical mechanisms of reliable connections and ii) the replication of previously received signals, techniques traditionally used to avoid alterations or undesired silences, maintaining a low latency, as required for NMP applications, i.e. to allow real-time reproduction of musical sounds exchanged between two or more network users. For this purpose present invention provides for the use of an HW or SW MIDI extender (shown in Fig. 1)) to the transmitter, which is placed between the musical device source of MIDI messages and the device capable of accessing a data exchange network for sending MIDI messages within data packets. The MIDI extender on transmitter side intercepts MIDI messages coming from a MIDI source (e.g. an electronic or analog sensorized musical instrument, a computer, etc.) and encloses them in packets according to a given format (described later as example) adding periodically to the received MIDI refresh messages that contain a set of values assumed by variables (e.g. active notes), continuously updated, representing the notes and/or commands being executed at time T for a given MIDI device. Hereafter, we will refer to the transmitting MIDI extender as sender, and to the receiving MIDI extender as receiver. The collection of variable values mentioned above defines the state of a sender-side MIDI device.

According to present invention, the purpose is achieved by a method of communicating data exchange representative of musical notes to be played

and/or control commands that regulate the sound being played, implemented in the MIDI extender, comprising the following steps in transmission (and dual steps in reception):

receiving a data stream representative of sounds of a musical instrument through a plurality of data messages formatted according to a differential symbolic format;

repetitively updating values of a plurality of variables encoded in said messages;

storing, for a plurality of variables, a relative list of values defining a state of a musical instrument in successive and progressive time instants representative of the action of a musician on the instrument in the respective said time instants;

generating data packets comprising and a plurality of said messages relating to a given time interval associated with the time stamp;

selectively including in packets said list to generate refresh packets, each referring to a state of the musical instrument at one of said time instants and a progressive sequential or temporal identification code.

Preferably, the code indicating the time instant of the packet creation by the sender. Preferably, the code is included in both refresh packets and data packets; sequentially transmitting packets, including refresh packets, in a data exchange network.

Present invention employs updating mechanisms for data generated by a

musical device and/or computer connected to a sender and transmitted to a receiver, in order to allow it to be played in real-time through e.g., a speaker, a headphone or other suitable device. In particular, sent packets selectively contain, in addition to MIDI messages, values assumed by process and/or status variables of MIDI device related to notes or commands based on said messages at instant T of sender. This allows the receiver to correct the collection of values of its own variables related to notes and/or control commands running on sender at time T and thus, on the one hand to obtain a more natural musical interaction between musicians since data retransmission is avoided, and on the other hand to significantly improve the audio quality played through a modest increase of data transmitted between sender and receiver. The refresh packets can also be sent adaptively depending on network congestion and the percentage of packets lost. With particular reference to MIDI protocols such as MIDI 2.0, MIDI 1.0 (further information can be found at www.midi.org), messages containing notes or control commands that change the values of MIDI device variables can be for example Note ON, Note OFF, System Exclusive, Program Change etc. According to implementation forms, data organized in packets having a time stamp, a plurality of messages and a plurality of variable values, are generated natively on board of MIDI musical instrument integrating the MIDI extender or by a distinct device from the MIDI musical instrument and receiving as input data related to sounds generated by instrument.

The packet generated by the present invention associates a plurality of variables values with a timestamp or other progressive identifier defining the state of a sender-side MIDI device without the need to associate a timestamps

with each variables valuesor by relying on feedback information sent by the receiver to generate it. Therefore, considering the same number of time stamps or other progressive identifiers transmitted, the present invention allows the transmission of more variable values, reducing the amount of data transmitted, and consequently reducing network overhead. At the same time, the time stamp or other progressive identifier associated with the packet allows the receiver to sort the received packets, compreising refresh packets, in a temporal or generally progressive sequence.

According to a preferred embodiment, a receiving MIDI extender, which comprise either a programmable electronic device or a personal computer or a mobile phone or a tablet, receives data packets comprising 'refresh' packets and generates an input signal for an acoustic output device such as a speaker or headphones based on a variables values collection defining the musical instrument state generating input data to MIDI extender. In this way, if due to a transmission error a Note OFF message does not reach a receiving device because a packet has been received late or has been lost, thus generating an undesirably persistent sound, the refresh packet contains appropriate information that will allow the receiving MIDI extender to align variables' values on sender at the time instant referred to by received refresh packet, thus terminating note reproduction.

According to a preferred embodiment, with the aim of reducing the amount of data exchanged, refresh packets contain variable values in progressive time instants and defining time intervals whose width is constant or variable. This width is greater than, for example an integer multiple of, the width of the time

intervals over which packets encapsulating MIDI messages are generated.

Moreover, present invention provides for adaptively adjusting time intervals referring to status data contained in refresh packets based on data network exchange capability.

BRIEF DESCRIPTION OF THE FIGURES

- Fig. 1 shows an example of MIDI extender positioning, on transmitter side, between a MIDI message source music device and a device capable of accessing a data exchange network.
- Fig. 2 shows an example of an interaction between three musicians where only the traffic of musician 1 is highlighted.
- Fig. 3 shows a schematic structure of a MIDI packet generated by present invention.
- Fig. 4 shows section B of the structure of a MIDI packet in detail.
- Fig. 5 shows section C of the structure of a MIDI packet in detail.
- Fig. 6 shows a control routine block diagram carried out by receiver on arrival of each new data packet.
- Fig. 7 shows the effect of the frequency of transmission of refresh packets on the accuracy of reconstructing the original MIDI signal to improve performance quality.

DETAILED DESCRIPTION

Present invention is applicable to a communication system (Fig. 2) preferably for NMP comprising at least two musicians connected with their musical instruments 1 comprising, for each musical instrument, an electronic control device preferably embedded on instrument to generate audio signals according

to a differential symbolic format based on musician's action on digital musical instrument. The system further comprises a MIDI extender, called first computer PC1, or other programmable controller, connected with a data exchange network, in particular an Internet data exchange network, which receives a data stream, preferably encoded messages by means of MIDI protocol, from the digital musical instrument and processes them in packets before transmitting said packets to a second MIDI extender, called computer PC2, connected to the network and programmed to receive and process said data packets and reproducing sound from the digital musical instrument PC1 by means of a speaker or headphones or similar. In order to achieve an NMP, PC2 is also connected to a digital musical instrument and processes its data in order to send it through network to PC1 computer, which processes it in order to reproduce it through its own speaker. Thus, both PC1 and PC2 computers act over the network as MIDI data sender and receiver.

According to present invention, data packets encoded by means of differential symbolic audio format (MIDI) are integrated with an absolute (and not differential) representation in a plurality of known and monitored time instants of the collection of variables values of each sender, each of them referring to a note and/or a command, e.g. which note is generated by a musical instrument or, in case of a MIDI controller connected to internet data exchange network, which effects are applied by controller to musical instruments sounds connected to PC1 and PC2. In this way, if a Note OFF command executed on musical instrument 1 is not received by mistake from receiver PC2, when receiver PC2 receives the list of state variable values of musical instrument 1 at absolute

time T sent by PC1, it will be able to update its own note variables status and/or control commands in execution on PC1 and, more precisely, it will be able to infer that NOTE OFF command has been lost and proceed accordingly i.e. interrupting the relative note. A differential symbolic message is a change-of-state instruction relating to a signal coding parameter of musical instrument 1. For example, a " Note ON " message comprises, properly encoded in bytes: the encoded "Note ON" command itself and the channel on which it is to be applied, a note, a note parameter e.g. a volume or intensity. Said command allows a receiving electronic device e.g. PC2 with its speaker, to start the acoustic signal reproduction corresponding to note 'Note' specified in message itself, while a message "Note OFF" will end the above mentioned reproduction.

Defining as working parameters two time intervals $\Delta t1$ and $\Delta t2$, related to periodicity of sending MIDI messages and the absolute representation of variables value describing notes and/or commands on sender side. To simplify description, we consider $\Delta t2$ as an integer multiple of $\Delta t1$. Computer/electronic device to which the instrument played by the musician is connected in data exchange runs a dedicated NMP MIDI application, which communicates, using for example UDP transport protocol, with remote computer running NMP MIDI application receiving sound data e.g. packets of MIDI messages.

Information transmission through a network is periodic, with period $\Delta t1$, and characterized by a plurality of MIDI messages sent by MIDI extender, grouped in a packet (an example of possible structure of such packet is described below and illustrated in Figg. 2-4). Each packet, sent at the end of each period of duration $\Delta t1$, contains the collection of MIDI messages generated on the basis of

the action applied to the instrument by a musician during that period of time. According to prresent invention, some packets sent by a sender comprise a collection of variable values that, with a periodicity $\Delta t2$, describe note and/or control commands status related to musician's action on musical instrument 1 at time T i.e., refresh packets; $\Delta t2$ identifies frequency or time interval with which variable values at time T are encapsulated e.g., queued within refresh packet. In order to identify variables values on sender at time T, each sender takes successive "photographs" at time T of values of its variables, representing notes and/or control commands representative of musician action on musical instrument 1 at time T, making a copy of the collection of variable values for each subsequent time interval $\Delta t2$. According to a preferred embodiment, subsequent instants of time T at which refresh packets are generated, are defined by a constant time interval and thus T=0, T= $\Delta t2$, T=2 $\Delta t2$, T=3 $\Delta t2$ etc. It is also interesting to notice that time stamp of each packet, comprisng those in refresh packets, together with MIDI message offsets allow synchronization of packets from multiple users. In general, each instant of time T falls within its properly synchronized refresh packet.

Therefore, considering T=0 the time when a NMP starts, the sum of all time intervals $\Delta t2$ returns the current time T. Time intervals $\Delta t2$ can be constant during a NMP and/or adjustable before starting an NMP and/or adaptive during an NMP.

At instant T, each sender checks or sets current variables value (e.g. a note being played may receive a termination command or vice versa a command to start playing another note may be coming from MIDI interface) and this

information is the one that will be put inside a refresh packet. When being received by receiver, these refresh packets comprising additional information about variable values allow the receiving MIDI application/controller to periodically or at monitored time intervals realign values of its own variables which will consequently define commands that are executed to play audio signal through receiver speaker, thus acting as a correction of errors caused by packet loss and thus restoring correct sender situation at time T. With reference to example described above, where a packet containing the message "Note ON" related to a note is not received (or not received in time), without this variable realignment mechanism based on additional information about notes and/or control commands representative of musician's action on musical instrument 1 at time T and existing on relative sender, generic receiver X would no longer have a way to initiate note reproduction if it could not verify through representation of sender state variable values that a given message had been previously generated.

According to a first preferred embodiment of present invention when receiver receives a packet containing a list of state variable values, i.e., a refresh packet, this list contains MIDI messages that directly enable note realignment and/or MIDI commands on receiver. According to another preferred embodiment of present invention, only representative data of variable values will be sent to receiver, possibly not in MIDI format, and it is a task of receiver's MIDI extender, by means of proper coding, to transform such data into MIDI messages allowing realignment between notes and/or control commands on sender side and those on receiver side. The most effective representation of this list, also considering

the amount of additional information added to total size of packet sent, is to insert those variables that represent notes and/or control commands still active at time T, i.e. variables or MIDI commands that cause emission of a sound through receiver speaker. Descriptions of embodiments of the invention in the present application are non intended to limit other possible representations that lead to same result as could be a dual approach to the one described: indicating all variables that have not been subjected to a change in considered instant, from which to obtain notes and/or active commands. Value taken by $\Delta t2$ can be modified during data transmission, depending on network congestion conditions and transmission bandwidth limitations imposed by network infrastructure itself. The specific criterion adopted to define value of $\Delta t2$ also depends on errors ratio per time unit that is considered tolerable for a musician, i.e. ratio between the number of music messages correctly played on receiver side and total number of messages generated by audio source, measured during a given time interval.

Example of practical implementation of invention

For illustrative purposes only, a possible implementation of present invention with reference to MIDI audio standard is described below to better clarify how the invention works.

Data format

A MIDI message is represented as a composition of three bytes:

- The first byte, called Status byte, provides information about the type of message and the MIDI channel to which it relates.

- Second and third bytes, called Data bytes, take on different values, the meaning of which varies depending on the type of message specified in the Status byte.

Two NMP application instances running on remote hosts exchange MIDI messages according to, for example, the following application layer packet format (as described in previous paragraphs and which will be referred simply as " packet "). Packet consists of a string that comprises several fields. Figure 3 shows a schematic representation of packet structure where all three sections are visible:

- A SECTION A contains a sequence number, which univocally identifies the packet number, and a timestamp, which identifies the absolute time on sender side (in UTC format) when packet is sent. In general, other timestamping or sequential marking systems can be employed in order that abreceiver is able to sort received packets from different senders.

- A SECTION B contains the MIDI message sequence generated by instrument played by a musician during the last time window of duration $\Delta t1$. As shown in Fig. 4 each message is characterized by an offset field, a time offset calculated from the instant of the beginning of current window. For example, a message with offset 5ms will be played 5ms after the start of MIDI message reproduction contained in packet itself. There follow 3 bytes to represent the MIDI message: an integer, Message Type, which represents message type and its MIDI channel (e.g. 144 = message type NOTE_ON on MIDI channel 1), and two integers named value 1 and value 2, whose meaning differs depending on

message type (e.g., in case of NOTE_ON value 1 represents the integer relative to the played note tone, value 2 in NOTE_ON case represents the intensity with which the note is played).

- SECTION C as shown in Fig. 3 is optionally valorized and contains the set of variables values representing musical instrument status 1 at time T existing on sender. According to an alternative embodiment, this section is set only in so-called "refresh" packets, otherwise it is empty. As previously explained, the list of variables preferably refers to list of variables that represent notes and/or control commands active on sender side at time T i.e., triggering the generation of a sound at instant T for each sender and are thus representative of the musician's action on musical instrument 1 at time T. As shown in Fig. 4 it is possible to represent variable information with the same triplet that characterizes MIDI messages: Message Type, value1 and value2. For note still active means for example a note whose reproduction has begun at a certain time e.g. belonging to $\Delta t2$ following a message of type NOTE_ON and that is still playing e.g. through the speaker because it has not yet been interrupted by a subsequent message of type NOTE_OFF. This type of representation allows the receiver to compare values of its own state variables related to notes and/or control commands at time T in comparison to the list of variable values in a refresh packet received from each sender. According to an illustrative embodiment, a receiver updates the state of notes and/or sound effects to be played through the speaker based on values of each variable contained in the refresh packet coming from sender. For example, variable values inside a refresh package are converted to notes and/or control commands and executed, or a list of variable values

received from a sender is compared to the list of variable values existing on receiver at time T and these are updated based on parameter values in refresh package.

System architecture

A preferred embodiment is described to explain possible interactions between musicians through an example according to a possible architecture shown in Fig. 2. Considering computer PC1, which plays the role of MIDI extender and identified in the following by S, as sender, which sends a musical signal, and computer PC2 which receives a musical signal (identified by R, as receiver).

Receiver R:

- (i) receives MIDI data from sender S according to format described above;
- (ii) implements, if necessary, error correction mechanism, e.g., updates values of its own varaibles based on those received from sender; and
- (iii) performs message reproduction representing received notes and/or sound effects.

In case of a communication between N musicians, each one will behave both as sender and receiver according to what has been described above. In particular, each receiver will receive MIDI data and independently manage communication with each of N-1 remote sender and each sender will send MIDI data independently to each of N-1 remote receivers (thus adopting a pure peer-to-peer approach). For the purpose of simplicity in Fig. 2, only traffic related to musician

1 is highlighted.

A description of a possible embodiment of present invention refers to communication between a sender and a receiver. Sender S is directly connected to a MIDI interface arranged, for example, on board the instrument played by a musician. MIDI interface sends asynchronous MIDI data to sender S whenever a message is generated by user playing the instrument. Time interval $\Delta t1$ represents the periodicity with which sender sends data e.g. packets to one or more receivers. The sender then groups received messages within each $\Delta t1$ interval and, at the end of the interval, sends to receiver a packet containing messages encoding a set of MIDI data received from MIDI interface generated during the interval considered. Although communication occurs every $\Delta t1$, which can be variable or constant during the NMP, due to offset field it is possible to ensure that individual messages reproduction contained in packet occurs with a finer time granularity than $\Delta t1$, (ideally the same used for message generation, within the limit of clock accuracy and offset field size). When choosing the value to be used for $\Delta t1$, it is important to consider that a smaller time interval reduces the time required for packet creation (and therefore contributes to decreasing the overall latency introduced by the system), but implies sending several packets to receiver with the consequent increase in traffic and amount of data transmitted. In contrast, larger time intervals reduce generated traffic but involve an increase in user-perceived delay. For an NMP application, a reasonable value of $\Delta t1$ is between about 5 and 10 ms, preferably not exceeding 15ms. As far as the choice of granularity relative to the offset of individual messages sent in a packet, it must be lower than chosen $\Delta t1$, but not excessively so, since fluctuations in the instant

of message reproduction lower than 1 ms are imperceptible to human ears. A possible option, which balances the different needs between generated traffic and user perceived delay, can be represented by an offset equal to 1ms for a $\Delta t1=10ms$, generally offset does not exceed 15% of time interval value. Sending of packets is therefore periodical, even in case for some time windows no messages are generated leaving in that case empty packet section B. This approach allows a receiver to distinguish the situation where there is a lack of receiving a packet due to a communication delay. By contrast, in case a sender sends a packet only when MIDI messages exist, a receiver cannot distinguish the difference between the lack of MIDI messages and a transmission delay until it receives a subsequent packet.

Transmission error identification and correction

In order to ensure that user-perceived experience quality meets the characteristics required in NMP applications, it is necessary to handle situations where packets are lost or received with excessive delay, e.g., due to network overload. When a packet is received late and out of sequence, this results in a buffer under-run condition, i.e., when the last received packet is reproduced without new packets arriving. In this case receiver buffer is empty due to delayed arrival of subsequent packet and application will not be able to reproduce messages in it in time.

As previously described, MIDI informations are differential, therefore it is critical for a receiver to play them back in the same order as they were generated by sender, thus ensuring that packets are sorted according to their sequence number before their content messages are reproduced.

In order to allow the receiver to rebuild lost information, each sender keeps track of and stores his own note and/or control command state, i.e., the set of values assumed by variables associated with each active note and/or control command at time T, which with periodicity $\Delta t2$, comprises within its refresh packet.

Similarly, also receiver keeps track of state of set of notes and/or commands of each sender due to refresh packets that each sender sends with periodicity $\Delta t2$ to receiver. Without buffer under-run events, receiver has a representation of sender variable values matching variable values of each sender. When a buffer under-run condition occurs for one or more packets containing at least a message, a receiver has a representation of the list of sender variable values that is mismatched with the list of variable values of sender from which it received data packet. In this case, receiver uses the refresh packet to:

1) ending notes and/or commands in execution whose representations are not contained in the data packet variable list received from sender describing the state of the musical instrument at time T that the sender sent;

2) executing notes and/or commands whose representation is contained in the received refresh packet variable list and that mismatch with value representation of its own variables related to sender from which it received the data packet. In case of receiving the packet, residual error is related to temporal difference in notes and/or commands execution compared to situation that would have been created in absence of buffer under-run event.

Therefore, to handle data packets arriving late and/or out of sequence to

receiver, for each new packet receiver performs a check routine as shown in Fig. 6, where two types of checks are provided:

1) Checking the sequence number or generic time stamp to sort packet messages: if it is lower than any other previously received packet whose content has already been played, then this packet is discarded.

2) Checking for presence of the variables list: if it is not empty (i.e. receiver has received a refresh packet) a comparison is made between the variables values representing sender's notes and/or commands contained in section C of refresh packet sent by sender with variables values corresponding to each sender which receiver keeps track of and implements necessary corrections in case values mismatch.

Finally, it is important to note how the choice of refresh packet sending frequency $\Delta t2$ is relevant to execution quality, as shown in Fig. 7. Choosing $\Delta t2=\Delta t1$ means allowing a receiver to optimally correct errors, but requiring a significant increase in the amount of data transmitted.

According to a preferred embodiment, a user may select a preferred value of $\Delta t2$ based on a target NMP accuracy, defined as the ratio of number of correctly received packets, i.e., not lost or not belatedly received and thus discarded, to total number of transmitted packets. In particular, it is possible to relate, for example by means of a map, a probability that a packet is discarded and the audio reproduction accuracy through data exchange network to variation of refresh ratio defined as a ratio between $\Delta t2$ and $\Delta t1$. A user, on the basis of e.g. historical values or measured before the beginning of NMP of the probability that a packet

is discarded due to the overload of the data exchange network, and the desired accuracy, can obtain by means of tables or interpolating functions the appropriate refresh rate value.

5

10

15

CLAIMS

1. A method of data communication representing musical notes to be played and/or control commands governing the sound being played, comprising the steps of:

receiving a data stream representative of sounds of a musical instrument though a plurality of data messages formatted according to a differential symbolic format;

repetitively updating values of a plurality of variables encoded in said messages;

storing, for a plurality of variables, a relative list of values defining a state of said musical instrument in successive and progressive time instants representing the action of a musician on said instrument in said corresponding time instants;

generating data packets comprising a plurality of said messages corresponding to a given time interval;

selectively including in packets said list to generate refresh packets each referring to a state of said musical instrument in one of said time instants and a sequential or temporal identification code, being said code associated to the list;

sequentially transmitting packets, comprising refresh packets, in a data exchange network.

2. Method according to claim 1, further comprising the steps of:

receiving said data packets and sequentially sorting said data packets based on the sequential or temporal identification code;

updating values of receiver status variables representing notes and/or control commands based on said refresh packets;

generating a sound signal based on updated list of variable values.

3. Method according to claim 2, further comprising the steps of:

comparing a packet identification code received from the receiver based on packets received from a sender over said network;

discarding a first received packet when said first packet identification code is such that said first packet is identified as a subsequent packet to a second packet whose contents have already been played;

executing a note and/or command represented through said values in said refresh packet relating to a subsequent time of said first packet when exsists a difference between sender and receiver variable values.

4. The method of any one of the preceding claims, wherein variables values

are inserted into a refresh packet at adaptively defined progressive instants of time based on a level of accuracy of sound reproduction and a parameter representative of a number of packets lost or undelivered or belatedly delivered by data exchange network in a time interval.

5. The method of any one of the preceding claims, wherein the updated
6. variable list of sender contains MIDI messages executable by the receiver.The method of any one of the preceding claims, wherein said packets

contain data relating to a time interval equal to or less than 15ms to enable a network music performance.

7. The method of any one of the preceding claims, wherein the data exchange network is a telecommunications network.

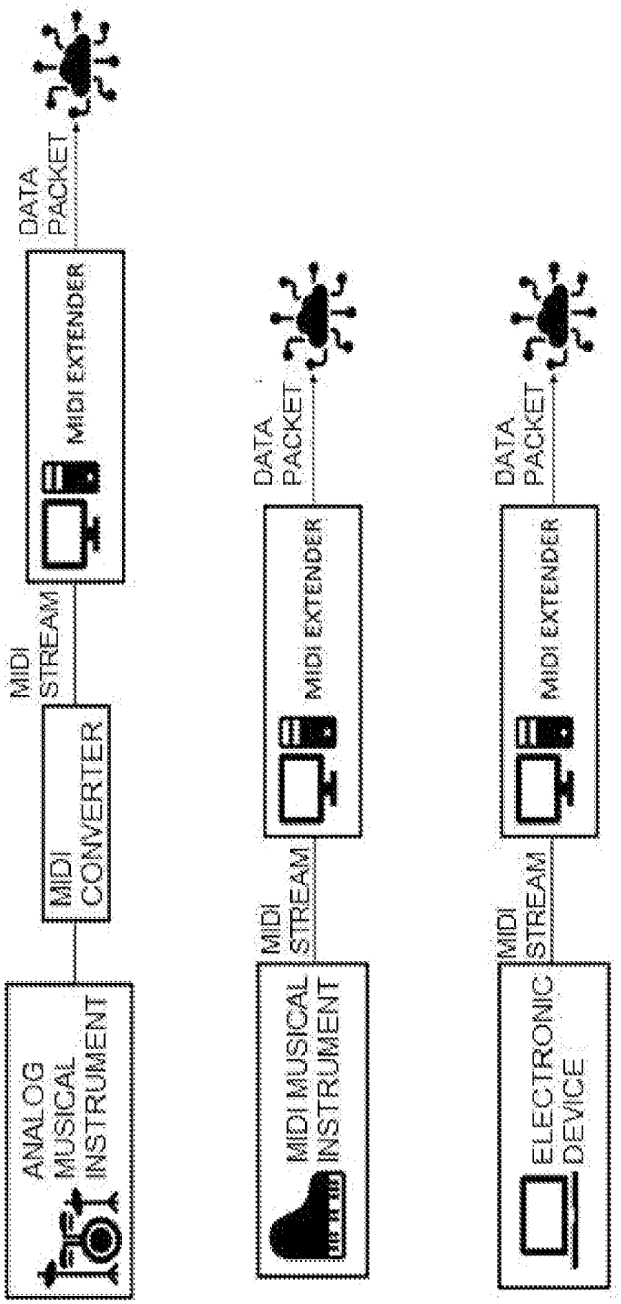
ANALOG MUSICAL INSTRUMENT
MIDI CONVERTER
MIDI STREAM
MIDI EXTENDER
DATA PACKET
MIDI MUSICAL INSTRUMENT
MIDI STREAM
MIDI EXTENDER
DATA PACKET
ELECTRONIC DEVICE
MIDI STREAM
MIDI EXTENDER
DATA PACKET

Fig. 1

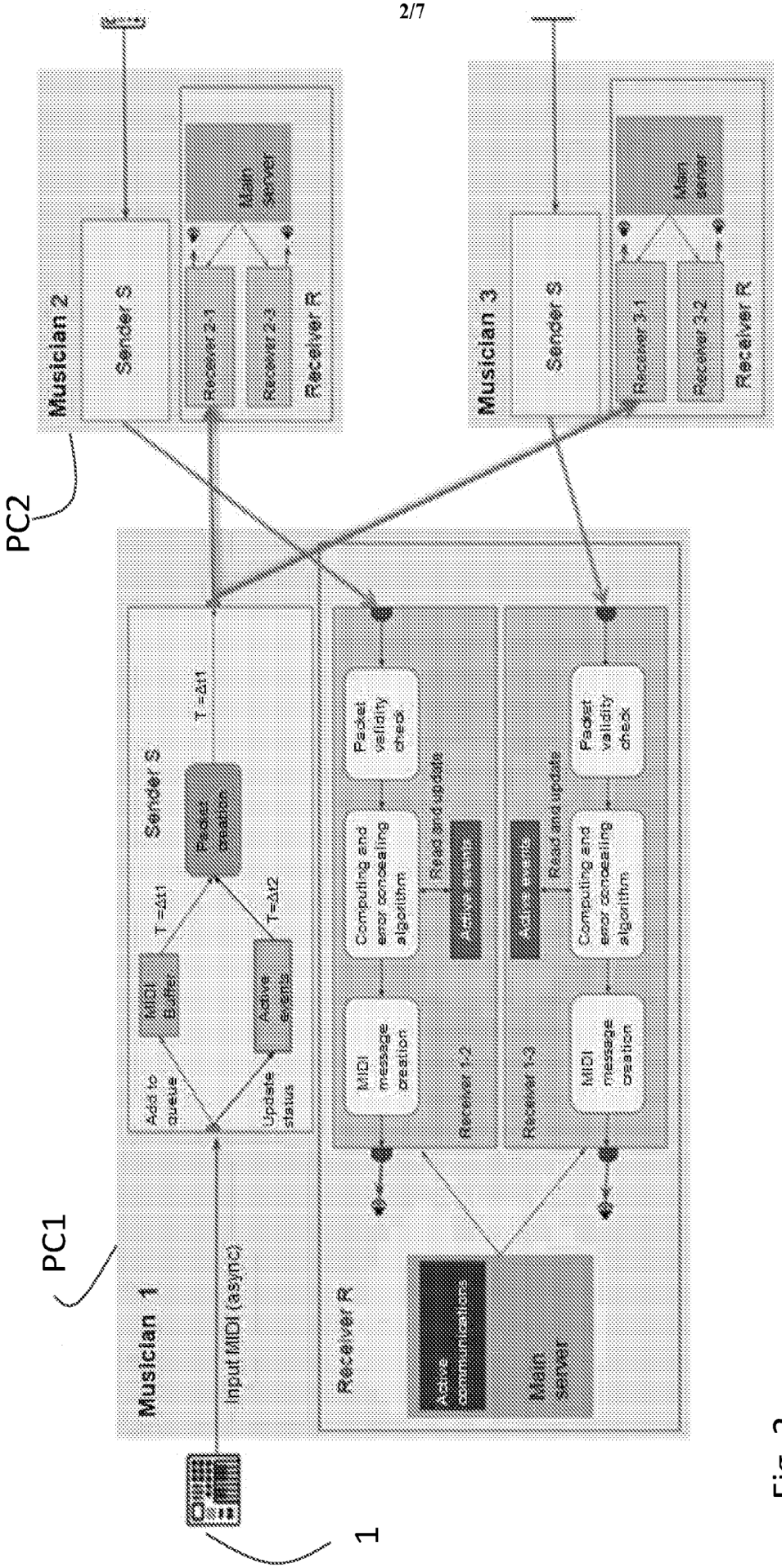
Musician 2
Sender S
Receiver 2-1
Receiver 2-2
Receiver R
Main server
Musician 3
Sender S
Receiver 3-1
Receiver 3-2
Receiver R
Main server
PC2
PC1
Musician 1
Sender S
Add to queue
MIDI Buffer
Update status
Active events
T=Δt1
T=Δt2
Packet creation
T=Δt1
Receiver R
Active communications
Main server
Input MIDI (async)
Receiver 1-2
Computing and error concealing algorithm
MIDI message creation
Packet validity check
Read and update
Receiver 1-3
Computing and error concealing algorithm
MIDI message creation
Packet validity check
Read and update
1

Fig. 2

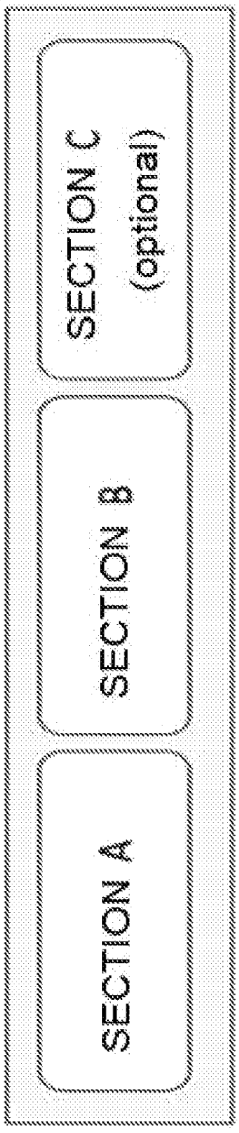

Fig. 3

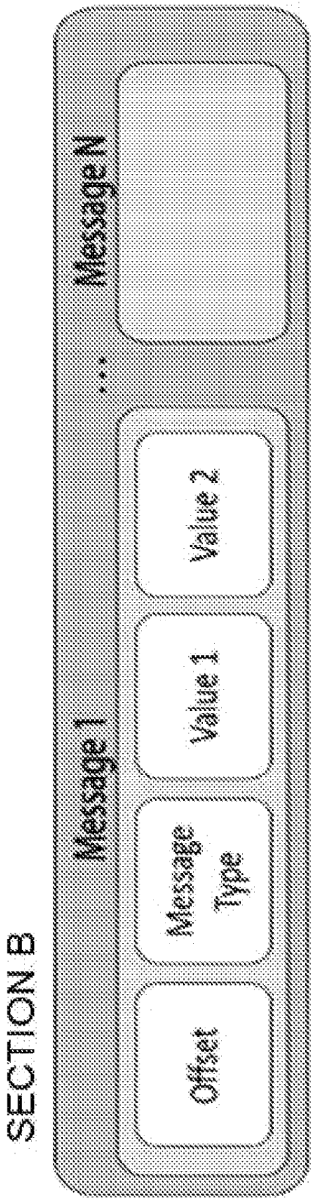
SECTION B
Message 1
Offset
Message Type
Value 1
Value 2
...
Message N

Fig. 4

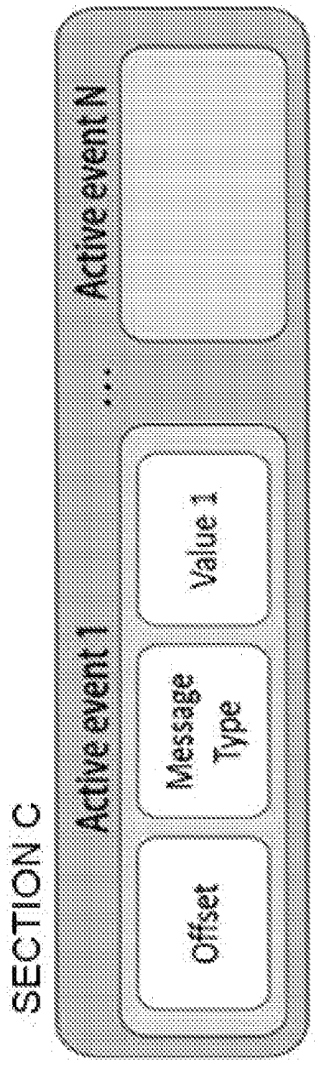
SECTION C
Active event 1
Offset
Message Type
Value 1
...
Active event N

Fig. 5

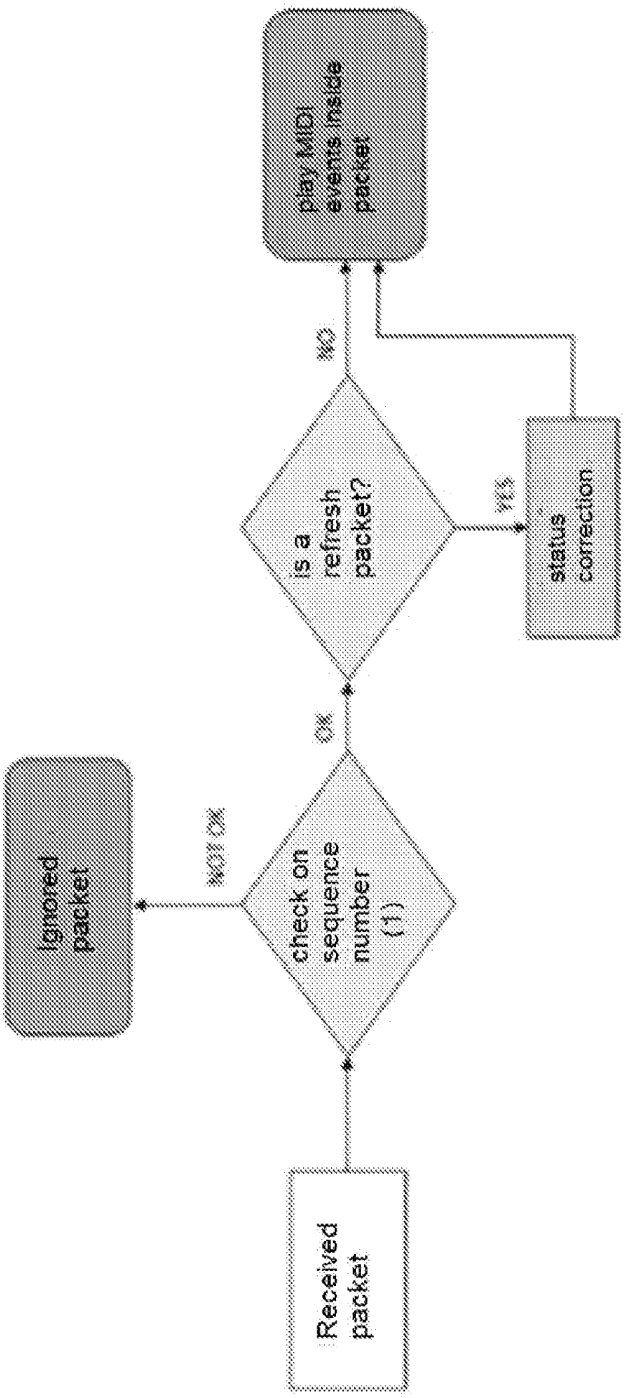
play MIDI events inside packet
NO
is a refresh packet?
YES
status correction
OK
check on sequence number (1)
NOT OK
Ignored packet
Received packet

Fig. 6

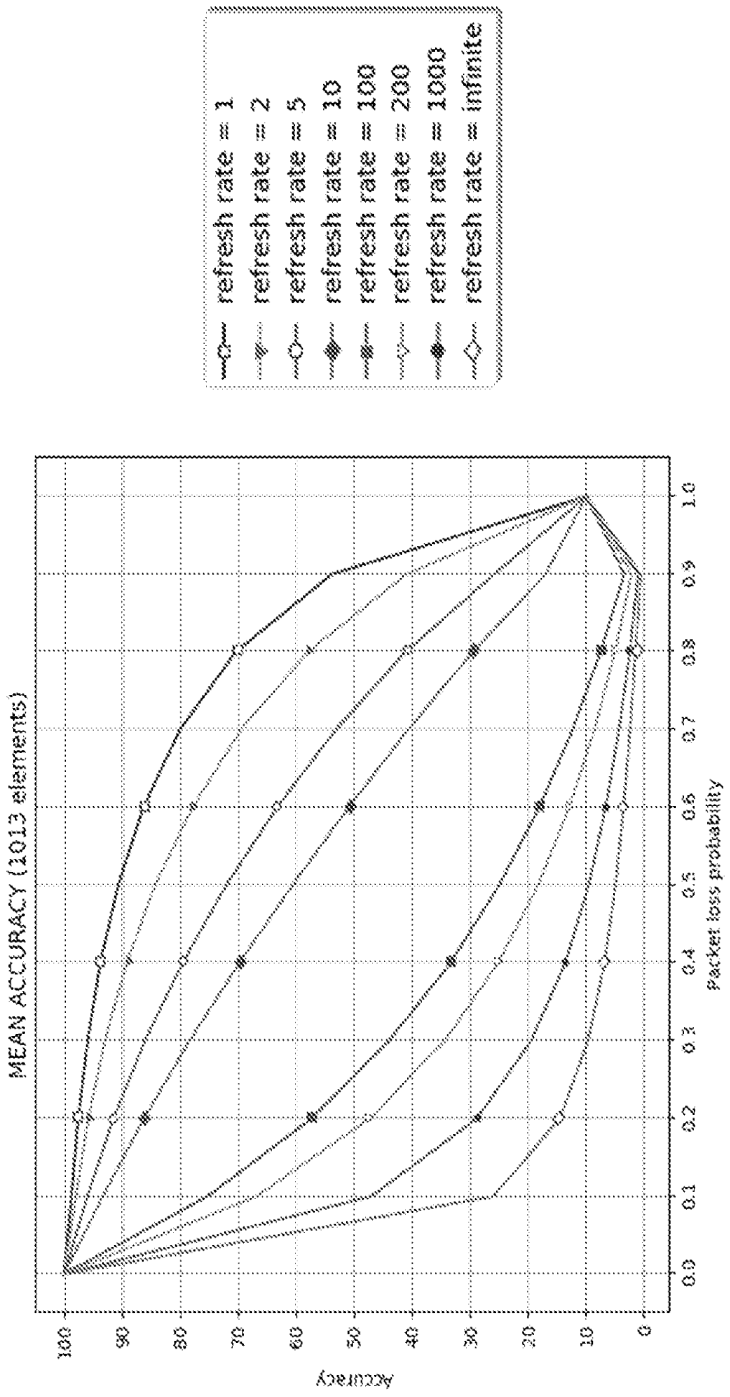
refresh rate = 1
refresh rate = 2
refresh rate = 5
refresh rate = 10
refresh rate = 100
refresh rate = 200
refresh rate = 1000
refresh rate = infinite
MEAN ACCURACY (1013 elements)
Accuracy
Packet loss probability
0
10
20
30
40
50
60
70
80
90
100
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

Fig. 7