New protocol concept for wireless MIDI connections via Bluetooth

CSABA HUSZTY, GÉZA BALÁZS

Budapest University of Technology and Economics, Dept. of Telecommunications and Media Informatics
info@midioverbt.com

Reviewed

Key words: wireless MIDI (Musical Instrument Digital Interface), Bluetooth

This paper describes a new protocol concept for wireless MIDI connections via Bluetooth. For the practical appliance, the protocol design supports nearly arbitrary connection topology. We show the plans of a generic Bluetooth-based MIDI system and describe the main ideas of its data transmission protocol, calculate its latency and investigate its limits of usability, while suggesting a few possible extensions to this system to be further realized.

1. The limits of the MIDI standard

The main function of MIDI is the synchronisation of the system beyond the transmission of control informations required for sound synthesis [1]. Devices utilizing MIDI protocol are equipped with 3 connector types. The IN connector is physically connected to the OUT connector of the adjacent unit, and the THRU output allows the user to form a chain topology of MIDI devices by sending the inbound data right to the device's THRU port. Using this method supports only connection topologies of limited structure without using auxiliary units.

It is also a quite common problem to combine more outputs into one input, which can only be solved with an external device, the MIDI Merger, too. Considering the maximal length of a few meters of the connection cables, the devices to be connected cannot be placed in an arbitrary order and distance, and further considerations have to be made if more than 15-20 devices are connected because of the increasing size, price and delay of the routing and switching units. We will not deal with the limits of the standard concerning the implementation henceforth.

2. Applying Bluetooth for MIDI connections

Among others Bluetooth is one of the inexpensive wireless solutions for the replacement of MIDI connections. The power consumption, the sufficient range and the prosperous noise resistance of the single units also make it applicable for this purpose. MIDI connections formed with Bluetooth can eliminate not only the cable but the other devices needed for the connection, so that they can be integrated into such existing devices. Although presented in a different way in other works [4], this paper describes a new method for realizing MIDI connections beyond the functionalities a MIDI cable offers.

Choosing the proper connection and the packet type for MIDI

The current 1.2 standard of Bluetooth supports one master and seven active slaves per piconet, although much more than seven slaves can be connected to it in parked state [2,3].

The master controls the channel access. All the other participants' clocks in the piconet are synchronized to the clock of the master and every unit is synchronized to the master in hopping frequency, too. The master may start a transmission only in even slots and a reception only in odd slots, while the slaves may do this vice versa: they may start a transmission only in even slots.

The above apply only for the initiation of the transmission: it can take up more than one slot, but the length must be an odd number of slots.

We choose the ACL type transmission for the MIDI application. One piconet supports one such type of data channel. The standard defines seven packet types for ACL connections, the maximum number of transmissible bytes are shown in *Table 1.*

We use the M-type packets for the data transfer, because they utilize 2/3 FEC encoding for error detection and correction, while the H packets feature neither error checking nor correction.

Table 1. Maximum data transfer rate in bytes of BT packets

packet type	max. tr.rate
DM1	18
DH1	28
DM3	123
DH3	185
DM5	226
DH5	341
AUX1	30

Hub-based topology and its advantages

The hub-based topology *(Figure 1.)* of the Bluetooth piconet facilitates the MIDI application. With broadcast type messages and correct settings any connection topology can be implemented, including the connections modifying the data stream, too.

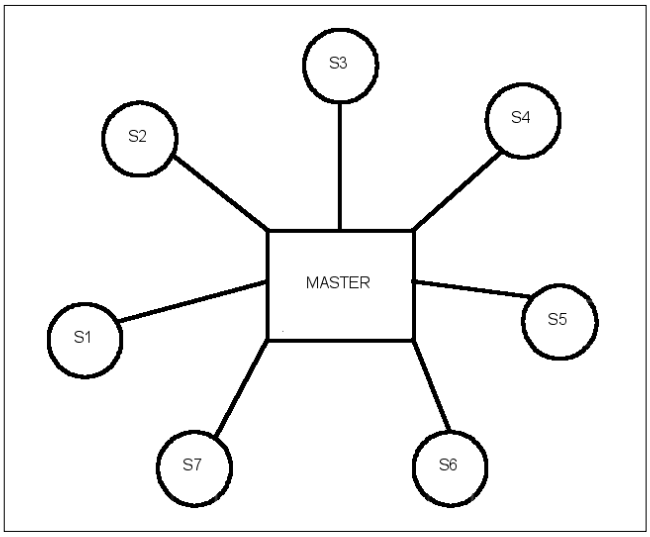


Figure 1. Hub-based topology

When setting up the connection we already know which slave unit will serve as a MIDI In and which as a MIDI Out. The Bluetooth based implementation includes the collateral possibility of looping back one's output to itself as an input – which is usually called 'MIDI Echo'.

Implementing basic MIDI connection types

To be able to create any type of connection topology, we have to implement the logical connections replacing the MIDI In, MIDI Out, the MIDI Thru Box (Hub), MIDI Merge, Echo and Patch Bay, either are they devices or functions.

1. MIDI Out/In (MIDI cable)

The data of the MIDI devices connected to the slave units get to the other device(s) indirectly via the master device. The user assigns the In and Out ports. The master device polls the first device, which sends its MIDI data back to the master. This data will be broadcast by the master to every active slave. Based on the set topology the S1 unit may ignore the incoming data (see *Figure 2*.).

2. MIDI Hub (Thru Box)

It is an easy task to connect one input to multiple outputs with these logical connections. After being polled by the master, the S1 slave sends its MIDI data. The only output port in this system is S1, so the polling ends after this point, and the master passes the data in a broadcast message to every slave. Since S1 is not configured as an input, it will discard the incoming MIDI messages. The constant delay of the arriving data with even more than one slave is a collateral advantage.

3. MIDI Merge Box

To unify two or more inputs in one output the master polls S1, which responds by sending its MIDI data. This data is temporarily stored in the master device for the broadcast packet, which is to be sent later, after polling S2 and getting its data. Slaves not configured as an input will discard the received data. There is need, however, to process the MIDI data streams before they are being merged for keeping its consistency, but the S3 could easily do this locally before sending the data to the MIDI device.

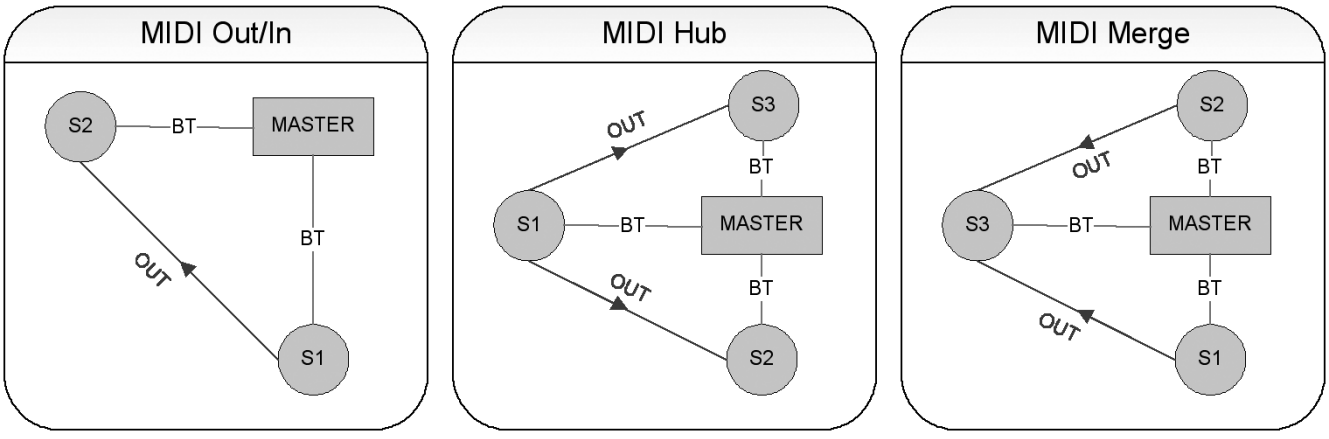
The suggested protocol concept and its timing

The logical units of the MIDI data, the messages have to be collected and split into packets when using the Bluetooth system. The MIDI bytes in the packets transmitted in a time unit (MIDI slot) are always broken on a message boundary. One exception exists: the System Exclusive message (SysEx), which can be of an arbitrary length. The incoming MIDI bytes are read one after the other from the MIDI Out ports by the predefined topology, which then form packets of constant length for each Out ports and are sent to the appropriate In-s. In some cases the packets have to be processed in order not to exceed the MIDI bandwidth (e.g. when merging). Such a cycle is called a MIDI slot from now on, which must have a firmly constant length in time. Considering the possibility of the reception of a variable number of bytes in a MIDI slot, the packet length and the delay time have to be computed for the case of the maximal byte count.

The timing of a MIDI slot (using DM3 packet with single transmission) is as follows:

1. The master polls the first slave that connects to a MIDI device with an Out port in the 0th Bluetooth (BT) slot [5].

Figure 2. Realizing MIDI connections



2. The addressed slave replies with a DM3 packet of constant length containing the received MIDI data. This transmission begins in the 1st and ends in the 3rd Bluetooth slot. If there are more than one Out-s in the system, the described process from step 1. is applied to each of them.
3. 2 empty BT slots follow, and then a DM3, DM5 or DH5 type of packet (depending on the number of the Out-s) is broadcast to every slave. The master device starts to construct the broadcast packet to be sent while still receiving data from the slaves (the UART ports of the BT modules and the Host Controllers are full-duplex), so there is only a little time to wait for the insertion of the data of the last slave. Since an even BT is coming after the data packet of the last slave, the system has to wait for the next even BT slot before it can send the broadcast message. This causes an additional delay of 2 BT slots.
4. Finally one empty BT slot comes, because the next polling sequence may start only in an even BT slot and the broadcast packet may not be responded. The whole cycle repeats from step 1.

To sum up, the amount of the needed BT slots are:

$$N_{BT\_SLOTS} = 4*O + 2 + x + 1$$

where O is the number of Out-s and x is 3 in the case of using DM3 and 5 when using DM5 packets for broadcasting.

Let B be the length of the MIDI slot in bytes:

$$B = S_{MIDI} \cdot T_{BT},$$

where S_{MIDI} = 3125 bytes/s, the transfer speed of the MIDI line and T_{BT} = 625 µs, the length of a BT slot.

In the interest of the planning of the timing let us calculate the maximum need of byte count to be transmitted. In the case of 1 active Out the maximum number of the transmitted MIDI bytes in a MIDI slot is:

$$B_{1\_OUT} = S_{MIDI} \cdot T_{BT} \cdot (1+1+2+1+1) =$$
$$= 6*S_{MIDI}T_{BT} = 11.71875 \Rightarrow 12 \text{ bytes}$$

where we rounded the sum up, considering the worst case. The terms of the sum are: 1 poll from master to slave, 1 response from slave to the master, 2 empty BT slots, 1 broadcast from the master to the slaves and 1 empty slot, which is 6 BT slots altogether, the length of the MIDI slot.

The above calculated value is 2 bytes less than the effective value, because most MIDI messages consist of 2 or 3 bytes, and it can happen that a 3-byte message follows after the 11th byte. The effective maximal length of a BT packet is such:

$$B_{TO\_TRANSMIT} = B_{1\_OUT} + 2 = 14 \text{ bytes.}$$

To be able to keep the timing constant both at high and low loads, let the length of the transmitted packets be constant, regardless of the number of useful bytes in them. The trasmission needs 2 more administrative bytes (a header and a footer). The one-byte header has to contain the number of the Outs, so that the merger can use this to identify the stream in which it has to insert the incoming bytes, while the one-byte footer indicates the end of the packet, which is implemented by using a byte not defined in the MIDI standard. So

$$B_{MERGER} = B_{TO\_TRANSMIT} + 2 = 16 \text{ bytes}$$

is the number of bytes that is to be transmitted during a MIDI slot. Since the DM1 packet can contain exactly 18 bytes of useful data and lasts for 1 BT slot, it is ideal to transmit this amount of information with this type of packet.

It is pretty plausible to use a Bluetooth module with an UART-type *Host Controller Interface* (HCI) because of its flexibility and simplicity, so from now on we show the timing values calculated specifically for UART based systems.

To be fair with the calculations by systems built with UART HCI BT modules it must be considered that the packet formatting needs 5 more bytes (1 byte ACL identifier, 2 bytes of connection handle, the ID of the master-slave physical attachment on-the-air, 2 bytes of flags and a packet length information, which is calculated without the 5 header bytes). These bytes will not be sent on the air, so we do not have to change the packet type. So 21 bytes have to be transmitted to the BT module, and the other module also will transmit this amount of bytes to its host.

With a 1 382 400 bits/s UART this process lasts for 152 µs (1 start bit + 8 data bits + 1 stop bit = 10 bits. T_{UART} = 10*21 / 1382400 = 152 µs), this is 24,3% of the length of the BT slot.

This calculation gives 506 µs for 2 Out-s with DM3 packets, which is still less than the length of a BT slot - which means that the timing will remain accurate.

In the case of 3 Out-s the situation is as follows: $B_{3\_OUT} = 20*S_{MIDI}T_{BT}$ = 40 bytes, because the broadcast packet must last 5 BT slots, as the length of the packet is 132 bytes which already needs the DM5 packet. The transmission lasts for 354 µs for the slaves and 998 µs for the master.

The length of the broadcast is 204 bytes in the case of having 4 Out-s in the system. The transmission times are 405 µs and 1519 µs, respectively.

After all there are no obstacles to use 5 Out-s in a piconet, but the broadcast packet cannot be realized using M packets here because of the 295 bytes to be transmitted. The transmission times are 463 µs and 2177 µs for the slaves and for the master. The latter is 3.48 in terms of BT slots, so the timing is still not vulnerable.

More than 5 Out-s are not applicable in the same piconet because of the bandwidth limits of the current Bluetooth technology.

Unfortunately in the respect of data protection and security the solutions above are pretty bad – nevertheless their delay is the best without doubt – because they transmit everything only once to the recipient. The

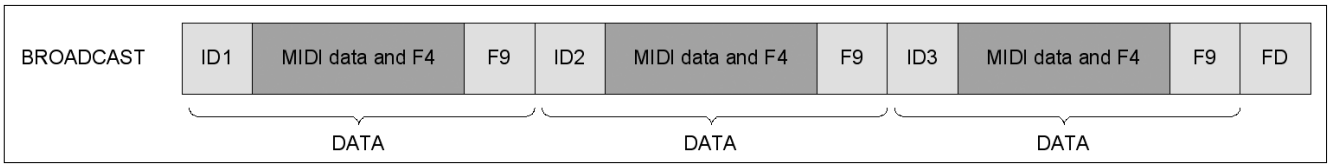


Figure 3. The structure of the broadcast packet

2/3 FEC coding improves the noise margin a bit, but the connection quality is still the function of the spatial placement and distance of the units. The most trivial method to minimize the packet loss probability is to use multiple transmissions.

Using 1 Out we can retransmit the data even 3 times; in this case the delay time is 16 ms, but it is still only 21 ms with 4 times of retransmission.

With 2 Out-s the maximum number of retransmissions is 2 (including the polling, the responses and the broadcasts); the delay time is then 18.75 ms.

Having 3 Out-s enables 2 transmissions for the polling, but does not allow us to retransmit the broadcast messages any more if we want to keep the timing. However, the broadcast can be repeated one more time if we use DH5 packets. With more than 3 Out-s none of the retransmission techniques can be applied with the data speed of the current Bluetooth standard.

The above are summarized in *Table 2*. We remark that using 1 Out port the throughput limit of Bluetooth enables 14 times of retransmissions without affecting the time stability of the MIDI slot.

The structure of the MIDI packets

Assuming maximum 3 Out-s *Figure 3.* shows the structure of the broadcast packet.

The DATA markings stand for the MIDI data of the each individual slave units. The termination mark of the data and the broadcast packet is the 0xF9 and 0xFD byte respectively, which are not defined in the MIDI standard. If the MIDI devices utilize the not defined 0xF4, 0xF5, 0xF9 or 0xFD MIDI bytes which are used for packet formatting, the packet headers, the timing and the structure of the protocol has to be modified to be able to transmit these bytes, too.

The maximum length of the broadcast packet is 238 bytes (DH5 packet, 3 Out-s, double transmission). The contents of the poll message are indifferent.

Reducing the overall latency by creating a scatternet

Using more masters simultaneously the latency can be decreased by distributing the Out units among the different masters. Unfortunately it is not easy to avoid masters transfer on the same frequency, although BT

Table 2. The latency of the data transfer – we suggest using the properties of the highlighted field

	No. of MIDI OUT-s	Response packet type (Slave)	Broadcast packet type (Master)	Latency (slots)	Transmitted slots (in order)	**Latency**	Physically transmitted MIDI bytes	Logically transmitted MIDI bytes
single transmit	1	DM1	DM1	6	1 poll, 1 response, 2 empty, 1 broadcast, 1 empty	**3.75 ms**	12	14
	2	DM3	DM3	14	2x1 poll, 2x3 response, 2 empty, 3 broadcast, 1 empty	**8.75 ms**	28	30
	3	DM3	DM5	20	3x1 poll, 3x3 response, 2 empty, 5 broadcast, 1 empty	**12.50 ms**	40	42
	4	DM3	DM5	24	4x1 poll, 4x3 response, 2 empty, 5 broadcast, 1 empty	**15.00 ms**	47	49
	5	DM3	DH5	28	5x1 poll, 5x3 response, 2 empty, 5 broadcast, 1 empty	**17.50 ms**	55	57
double transmit	1	DM3	DM3	18	2x1 poll, 2x3 response, 2 empty, 2x3 broadcast, 2x1 empty	**11.25 ms**	36	38
	2	DM3	DM5	30	2x2x1 poll, 2x2x3 response, 2 empty, 2x5 broadcast, 2x1 empty	**18.75 ms**	59	61
	3	DM3	DH5	38	2x3x1 poll, 2x3x3 response, 2 empty, 2x5 broadcast, 2x1 empty	**23.75 ms**	75	77
triple transmit	1	DM3	DM3	26	3x1 poll, 3x3 response, 2 empty, 3x3 broadcast, 3x1 empty	**16.25 ms**	51	53
	2	DM3	DH5	44	3x2x1 poll, 3x2x3 response, 2 empty, 3x5 broadcast, 3x1 empty	**27.50 ms**	86	88
4-times transmit	1	DM3	DM3	34	4x1 poll, 4x3 response, 2 empty, 4x3 broadcast, 4x1 empty	**21.25 ms**	67	69
5-times transmit	1	DM3	DM3	42	5x1 poll, 5x3 response, 2 empty, 5x3 broadcast, 5x1 empty	**26.25 ms**	83	85

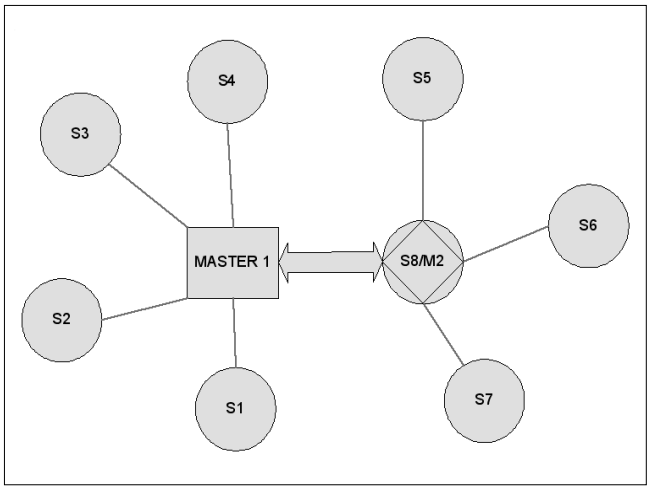


Figure 4. Scatternet with multifunctional device

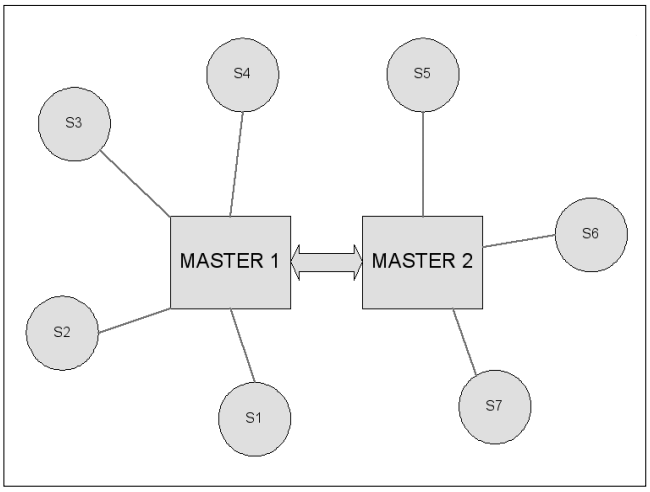


Figure 5. Scatternet by high-speed link

1.2 implements a method to avoid this, so there is even more need for multiple transmissions, which increases the latency at the same time. Creating arbitrary connection topologies is then realized by connecting the piconets, which is known as scatternet. It can be formed in more ways from piconets: a slave unit may operate as a master in the other piconet, or individual masters might be wired via a high-speed link.

Considering the MIDI implementation the first method is nowise adequate, because the multifunctional S8/M2 device (see *Figure 4*.) can only serve one of its functions at a time – synchronism cannot be achieved. However this raises another problem: what happens if the slave units of Out functionality are not balanced equally in the several piconets. As we could see in the discussion of the protocol timing, the latency increases and the parameters of the protocol implementation to be applied (e.g. number of retransmissions, type of BT packets) vary with the increase of the amount of the Out-s.

We find most expedient building a system where the end user does not have to reconfigure the whole system manually when putting a new master in operation and also does not have to set up the new connection topology in an uncomfortable and lengthy manner. When distributing the Out-s we have to strain after that each master gets the least Out-s possible and that the In-s receiving data from the same Out-s get into the same piconet, so that the least data have to be transmitted outside the piconet.

3. Conclusion

This paper made a proposal for a Bluetooth protocol conception and investigated its feasibility for implementing wireless MIDI connections. In spite that the implementations using Bluetooth did not succeed so far, the system described here fully exploits the more increasing throughput of Bluetooth. It allows acceptably safe data transfer while maintaining constant latency.

A system based on the contents of this paper can be easily extended by new units so that the overall latency can be decreased while the number of client units can be increased.

The most important conclusions for the feasibility of the system are:

(1) retransmission – at least for 2 times is needed to achieve a safer connection,
(2) a piconet may contain up to 3 MIDI Out-s, where only two Out-s are suggested to transmit data to the same piconet,
(3) there is no need for use of broadcast messages if there are no Out-s connected to a piconet, the DM3 packet is ideal for 1 or 2 Out-s, delay times of them are 16.25 ms and 18.75 ms, respectively, with a different number of retransmissions for each; while in the case of 3 Out-s, the DH5 packet should be used which results in 23.75 ms latency, and
(4) 3 masters at most can be tied together in the same area while maintaining a sufficient data reception probability if using Bluetooth 1.1. This does not apply for the 1.2 standard, which can implement piconets with arbitrary distribution of the 79 frequency channels.

References

[1] MMA MIDI Specifications, 1983-2003.
[2] Specification of the Bluetooth System, v1.1
[3] Specification of the Bluetooth System, v1.2
[4] J. Keniston, S. Sturdivant (2003):
Wireless MIDI Network Implemented Via Bluetooth
[5] R. Mettala: Bluetooth Protocol Architecture, v1.0
(Bluetooth White Paper Document # 1.C.120/1.0)