Ableton Link – A technology to synchronize music software

Florian Goltz
Ableton AG
Schönhauser Allee 6-7
10119 Berlin,
Germany

Abstract

Ableton Link is a technology that synchronizes musical beat, tempo, phase, and start/stop commands across multiple applications running on one or more devices. Unlike conventional musical synchronization technologies, Link does not require master/client roles. Automatic discovery on a local area network enables a peer-to-peer system, which peers can join or leave at any time without disrupting others. Musical information is shared equally among peers, so any peer can start or stop while staying in time, or change the tempo, which is followed by all other peers.

Keywords

Audio, Network, Peer-to-peer, Time, Synchronization

1 Overview of Common Sync Technologies

Synchronizing media devices has been a challenging task for a number of decades. This section provides an overview on existing standards and approaches. No single sync technology has been able to establish itself as a universal standard. Depending on the context and actual requirements of a scenario, one ore more of the existing standards are used.

1.1 SMPTE

In 1967, the Society of Motion Picture and Television Engineers released a standard for the synchronization of media systems [Rees, 1997]. In this standard, time is described as an absolute value separated into hour, minute, second, and frame. A master machine generates the clock signal and sends it to a variable number of clients. The clock signal can be sent across a dedicated channel or embedded as metadata within the media. SMPTE is still widely used today for synchronization of video and audio systems.

1.2 AES/EBU

The Audio Engineering Society and the European Broadcasting Union published the AES/EBU standard in 1985 [Laven, 2004]. It provides the same information as SMPTE but is optimized for audio equipment. AES/EBU can use a wide variety of transports, from XLR cables to S/PDIF.

1.3 MTC

Midi Time Code was released in 1987 and embeds the same data as AES/EBU, but is optimized to be transported via MIDI sysex messages. [Meyer and Brooks, 1987]

1.4 MIDI Beat Clock

Unlike the above standards, MIDI Beat Clock is a tempo-dependent signal. It consists of 24 pulses per quarter note. This is probably the most widely used sync signal in music software and hardware today.

1.5 JACK Transport

The Jack Audio Connection Kit Transport API [JackAudio, 2014] allows sharing sample accurate timecode between its clients. While Jack itself acts as a timecode master for its clients, Jack Transport allows all its clients to start and stop transport or seek the timeline. Using NetJack [Hohn et al., 2009], it is possible to connect multiple clients on a local area network to a master. This way transport controls can be shared among multiple applications running in different computers. NetJack however only allows audio output on the master machine.

1.6 OSC Sync

An OSC-based synchronization scheme has been proposed [Madgwick et al., 2015] which has a master send clock messages on a regular basis. This scheme targets networked use cases such as laptop orchestras.

1.7 Summary

All of the above technologies share the common approach of having a master provide a clock signal to a number of clients, though the representation of time varies. Setting up such systems involves routing the signal from the master to the clients and/or configuring the master and clients to send and receive via the appropriate channels. In a master/client system, the master application is usually the only one that has control over tempo and transport state. As soon as the master fails, or the channel breaks, the clients are in an undefined state.

2 Link Design Criteria

Three criteria drove the development of Link:

- Remove the restrictions of a typical master/client system.
- Remove the requirement for initial setup.
- Scale to a wide variety of music applications.

These goals are achieved by designing a peer-to-peer system that sends multicast messages on a local network. Parameters are controlled mutually and all peers converge to the same shared timing information. The timing information is designed in such a way that peers with different capabilities such as a one-bar-looper or a fully-featured DAW can map the shared information to their specific needs. If peers are connected to a Local Area Network there is no further setup required.

3 Multicast Discovery

Link peers communicate using UDP multicast messages in a local area IP network. Each peer regularly sends messages that contain its unique peer ID and a snapshot of its current musical time. This way all peers and their state is known by each peer on the local network.

The incoming messages are processed by every receiver according to the same set of rules. If a receiver decides to adapt the timing information it has received, it updates its timing information and broadcasts accordingly. As a result of this peer-to-peer messaging, all peers on a network always converge to the same shared description of the current musical time.

Link regularly scans the available network interfaces on the host computer. When a new interface is discovered, multicast messages are sent and received on it as well. As a result, a Link peer that is connected to multiple networks can act as a relay: when the timing information from incoming messages on one interface is adapted, it is sent out on all available interfaces. This way, timing information is shared with peers that are not directly connected.

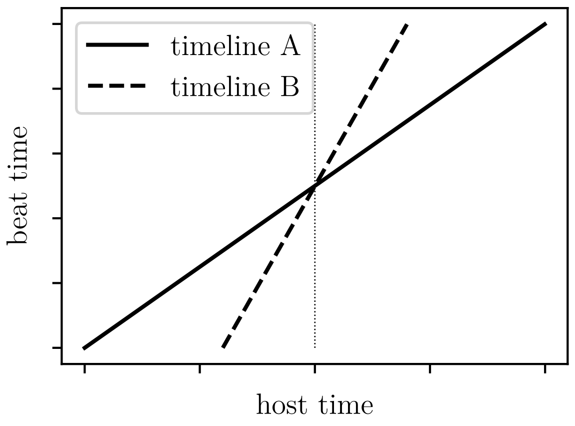

Figure 1: The new timeline B crosses the old timeline A at the host time of the tempo change

4 Timeline

Link describes the timing information of the session at a point in time as a tuple of three values: the *host time* that the hardware provides, a corresponding *beat time*, and a *tempo* that describes the change of beat time over host time. This tuple of values is referred to as a *timeline*. The system's beat time for a given host time and vice-versa can be calculated with a simple linear equation: `BeatTime`/`HostTime` = `Tempo`

When a peer intends to change the tempo at a specific host time, it creates a new timeline, describing a linear equation crossing the desired time point, and shares it with the network. When initializing Link, each peer creates such a timeline and immediately shares it with the network. This timeline then gets either adapted by other peers on the network, or the peer adapts a timeline it is receiving.

5 Host Time

Desktop operating systems usually provide calls that allow applications to ask for the current host time. Examples are `clock_gettime()` [IEEE, 2008], `mach_absolute_time()` [Apple, 2005] or `QueryPerformanceCounter()` [Microsoft, 2001]. The time stamps provided by those calls are based on information that the CPU or specialized hardware provides. Their quality can differ significantly in terms of accuracy and reliability. Additionally, the speed of the clock may depend on factors such as temperature and thus vary

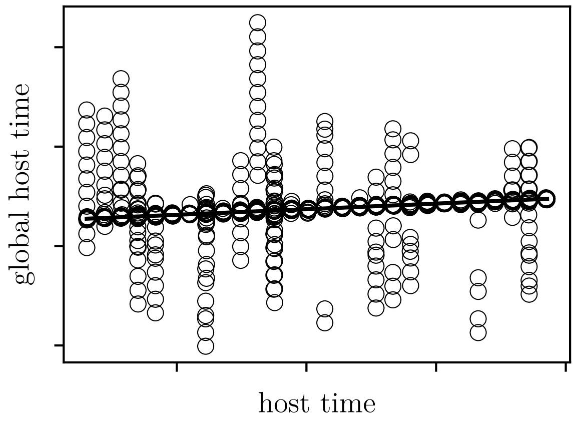

Figure 2: Measuring global host time against local host time in bursts

over time.

To be able to derive the session's beat time from the current host time, it is important that a peer has accurate knowledge of the system's host time. Some audio APIs provide accurate timing information in the audio processing callback. On other systems, it is necessary to query the systems' host time in the audio callback and filter it to get reliable information. The reference time Link uses is the "host time at speaker", which refers to the time the audio is actually perceived by the listener. To calculate this, software and hardware latencies must be incorporated into the host time provided by the system.

6 Global Host Time

Link establishes a reference host time that is shared between all peers in a session. This is referred to as the *global host time.* When a peer initializes Link and starts the initial timeline, its own host time is used as the reference. Every peer joining the session uses ping-pong messaging to calculate the offset of its own host time against this reference time. The result of this measurement, is a function that can convert the local host time of the peer to the global host time and vice versa. `globalHostTime = XForm.hostToGHost(localHostTime)`

As soon as a peer knows the global host time, it can function as a measurement endpoint for other peers. As a result, the peer that originally founded the session can leave, while the global host time is still maintained. Peers regularly measure their host time's offset to the global host time to compensate for speed variations.

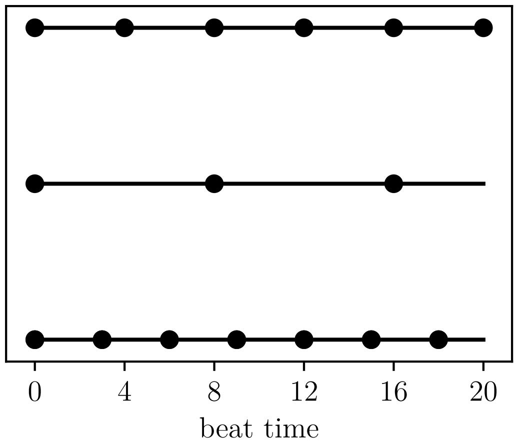

Figure 3: Alignment of timelines with quanta of 4, 8 and 3 beats

7 Quantum

As mentioned above, one of the requirements for Link is to scale to music applications with different capabilities. This means it should work for applications that have different representations of musical time, e.g., loopers that only provide a simple one bar loop, or full featured DAWs that sequence a beat timeline and support different musical measures.

Link takes the approach of allowing each client to map the shared timeline to its own purpose, e.g., a looper can map Link's timeline to a position within its loop by calling `phaseAtTime(localHostTime, quantum)`. The quantum provided by the client describes the alignment grid in beats. A looper with a one bar loop in a 4/4 measure would provide a quantum of 4. Link also provides `beatAtTime(localHostTime, quantum)` which provides a monotonic timeline in a way that would typically be used by a sequencer.

Link guarantees that clients using the same quantum are phase synchronized. Peers with different quanta can form a polyrhythmic Link session, e.g., a peer using a quantum of 3 and another peer using a quantum of 4 would be share a downbeat every 12 beats.

8 Transactional API

Link provides lock-free `capture()` and `commit()` functions to be used in the audio thread, and a similar thread-safe pair of functions to be used in other threads.

The capture functions provide a snapshot of the Link session. This can be used to align the client's audio to the shared timeline. In case the client wants to change the timeline, e.g., to change the tempo, the captured state can be modified and committed back to Link using the commit function. The new state will then be sent to the network and merged with the other peers' states.

9 Resources

Link is available as a header only C++11 library. It is dual licensed under the GNU-GPL and a proprietary license. The source code is currently available at `http://github.com/ableton/link`. Explanation of the concepts used in Link and technical documentation on the API can be found at `http://ableton.github.io/link`.

10 Conclusions

Existing technologies to synchronize music devices, as described in Section 1, are all based upon a master/client communication protocol. It is the master's responsibility to broadcast a signal according to the specification. The clients receiving the signal are dependent on the communication channel not being interrupted.

Link introduces a different approach to synchronize music devices. It creates a peer-to-peer network where all peers share a global time reference and a beat timeline. Any peer can introduce changes to the timeline in order to change the state of the session. To establish and maintain the shared state, it is important that all peers follow the same set of rules. In this sense, Link is not just a communication protocol, but a set of rules for multiple actors to create a shared musical session.

References

Apple. 2005. `https://developer.apple.com/library/content/qa/qa1398/_index.html`. Accessed: 2018-03-06.

Torben Hohn, Alexander Carôt, and Christian Werner. 2009. Netjack - Remote music collaboration with electronic sequencers on the Internet. LAC 2009, `http://lac.linuxaudio.org/2009/cdm/Saturday/22_Hohn/22.pdf`. Accessed: 2018-04-30.

IEEE. 2008. POSIX 1003.1-2008. `http://pubs.opengroup.org/onlinepubs/9699919799/functions/clock_getres.html`. Accessed: 2018-03-06.

JackAudio. 2014. `http://www.jackaudio.org/files/docs/html/transport-design.html`. Accessed: 2018-04-30.

Philip Laven. 2004. Specification of the digital audio interface. `https://tech.ebu.ch/docs/tech/tech3250.pdf`. Accessed: 2018-03-06.

Sebastian Madgwick, Thomas Mitchell, Carlos Barreto, and Adrian Freed. 2015. Simple synchronisation for Open Sound Control. `http://eprints.uwe.ac.uk/26049/1/03FinalSubmission.pdf`. Accessed: 2018-03-06.

Chris Meyer and Evan Brooks. 1987. MIDI Time Code and cueing. `https://web.archive.org/web/20110629053759/http://web.media.mit.edu/~meyers/mcgill/multimedia/senior_project/MTC.html`. Accessed: 2018-03-06.

Microsoft. 2001. Acquiring high-resolution time stamps. `https://msdn.microsoft.com/en-us/library/windows/desktop/dn553408`. Accessed: 2018-03-06.

Philip Rees. 1997. Synchronisation and SMPTE timecode. `http://www.philrees.co.uk/articles/timecode.htm`. Accessed: 2018-03-06.