Troubleshooting LSL Problems

David Medine 15/06/2021
Troubleshooting Your Setup

• For any questions (especially for beginners) regarding LSL, it is always good to consult the [online documentation](#).

• Check the [connectivity section](#), especially regarding:
  • Firewalls
  • Network adapters
  • Security

• [LabRecorder](#) and the [Brain Vision LSL Viewer](#) will tell you what streams are available on your network.

• Join the [Slack channel](#) and ask the pros!
LSL Synchronization Overview

- LabRecorder records not only data and their timestamps, it also measures and records clock offsets between CPUs and the local timestamps when those measurements are taken.

- Using this information XDF loaders (load_xdf.m/pyxdf/XDF.jl) can correct for 2 things:
  - Clock offsets (slowly drifting differences between the CPU sending the data and the CPU recording the data)
  - Jitter (random noise added to the timestamps due to minutely varying processing times)
LSL Synchronization Overview

• Correcting for clock offsets:
  • LabRecorder uses the LSL function `time_correction()` to measure clock differences between PCs for each stream
  • The default is to do this every 5s
  • This data is stored in the footer portion of an XDF file along with data write times
  • Using this data, the timestamps from each stream can be put on the same timebase as the recording PC
• `time_correction()` uses the clock synchronization algorithm (part of the Network Time Protocol) to compute clock offsets:
  • $t_0$ send time of packet from LabRecorder to stream host
  • $t_1$ receive time of packet from LabRecorder host
  • $t_2$ send time of packet from stream host to LabRecorder
  • $t_3$ receive time of packet from stream host
  • $\text{RTT} = (t_3-t_0) - (t_2-t_1)$
  • $\text{OFS} = ((t_1-t_0) + (t_2-t_3))/2$ for lowest RTT
LSL Synchronization Overview

- Using record times and clock offsets, XDF loaders have all the information they need to remap timestamps from device host PCs to a Recording PC's timebase
- `load_xdf.m` uses a robust fitting procedure, an ADMM (alternating direction method of multipliers) incorporating the Huber loss function ([code/paper](#))
- The result is a simple linear map (DC offset and slope) between the stream host timebase and the LabRecorder host timebase
- This map is then applied to the timestamps from each stream to thus synchronizes the data
LSL Synchronization Overview

- Jitter is also a problem:

- **Linear regression to a line** will map the jittery timestamps to a nice grid, but it will never be exactly the same as the advertised device sampling rate.

- This can lead to confusion:
  - `nominal_srate` vs `effective_srate`
LSL Synchronization Overview
What Can Go Wrong

• A changing sampling rate (or data that drops some, but not many samples) will cause the dejittering algorithm to produce terrible results.

• If there is problems with calls to `time_correction()`, the algorithm that corrects clock offsets will produce terrible results.

• If everything is working properly, your data will not have these problems. Always pilot before doing a study---hearts have been broken by these two issues.
What Can Go Wrong

• The clock offset problem can occur if either of the following is true:
  • The data is coming from a software that imposes its own timestamps.
  • Network insufficiency cause problems with calls to `time_correction()` ---this can happen in faulty or low-bandwidth/highly taxed wireless networks.
What Can Go Wrong

- The ‘external timestamp’ problem can be fixed adding the following code to an LSL config file:

  [tuning]
  ForceDefaultTimestamps=1
What Can Go Wrong

• LSL based experiments often require heavy network bandwidth, so it is best to transmit over a wired LAN.

• But, if wireless is needed, the following tuning may help prevent data loss and time_correction() malfunction:

```plaintext
[tuning]
TimeProbeMaxRtt = 0.100
TimeProbeInterval = 0.010
TimeProbeCount = 10
TimeUpdateInterval = 0.25
MulticastMinRTT = .100
MulticastMaxRTT = 30
```

*Courtesy of Matthew Grivich*
A Note on EEGLAB and XDF

- EEGLAB does not ship with an XDF reader, but it is a downloadable extension (File->Manage EEGLAB extensions->Search for ‘xdfimport’)
- XDFImport only loads EEG and Marker streams and will arbitrarily choose one EEG stream in the case of multiple EEG streams in one file.
- XDFImport keeps the nominal sampling rate, but it uses the effective sampling rate to align Markers with EEG data.
- File->History scripts is your friend! You can go in and tweak the loader if something is amiss.
Links

- LSL on github: [https://github.com/sccn/labstreaminglayer](https://github.com/sccn/labstreaminglayer)
- XDF on github: [https://github.com/sccn/xdf](https://github.com/sccn/xdf)
- All material from this presentation including code, data, and analysis scripts is available on github: [https://github.com/Diademics-Pty-Ltd/EEGLABWorkshop2021](https://github.com/Diademics-Pty-Ltd/EEGLABWorkshop2021)
- Clement is also uploading everyone’s slides---but I don’t know where?
- My website: [www.diademics.com](http://www.diademics.com)
Introduction to Bio-Signals Integration Using the LabStreamingLayer

Virtual Workshop

15th September 2021

http://www.diademics.com/#events
The End

Thank you!