Synchronization And Other LSL Topics

Christian A Kothe, CTO

Tim Mullen, CEO

Outline

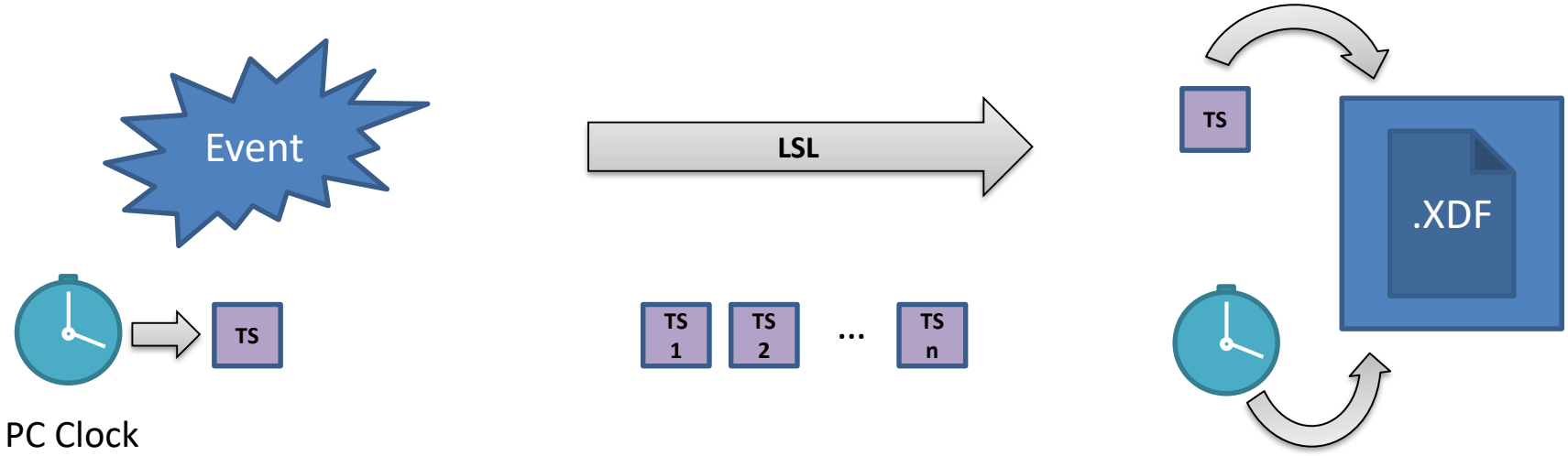
1. Time Synchronization
2. Diagnosing Connectivity Issues
3. More Notes on Metadata

1 Time Synchronization

LSL Time-Synchronization Basics

- Every sample in LSL has its own time stamp
- LSL mainly transports time stamps, but doesn't "tamper" with them
- LSL tracks clock drift across machines and when recording to XDF, that info is stored
- Real-time time synchronization is also available

The Life of A Timestamp

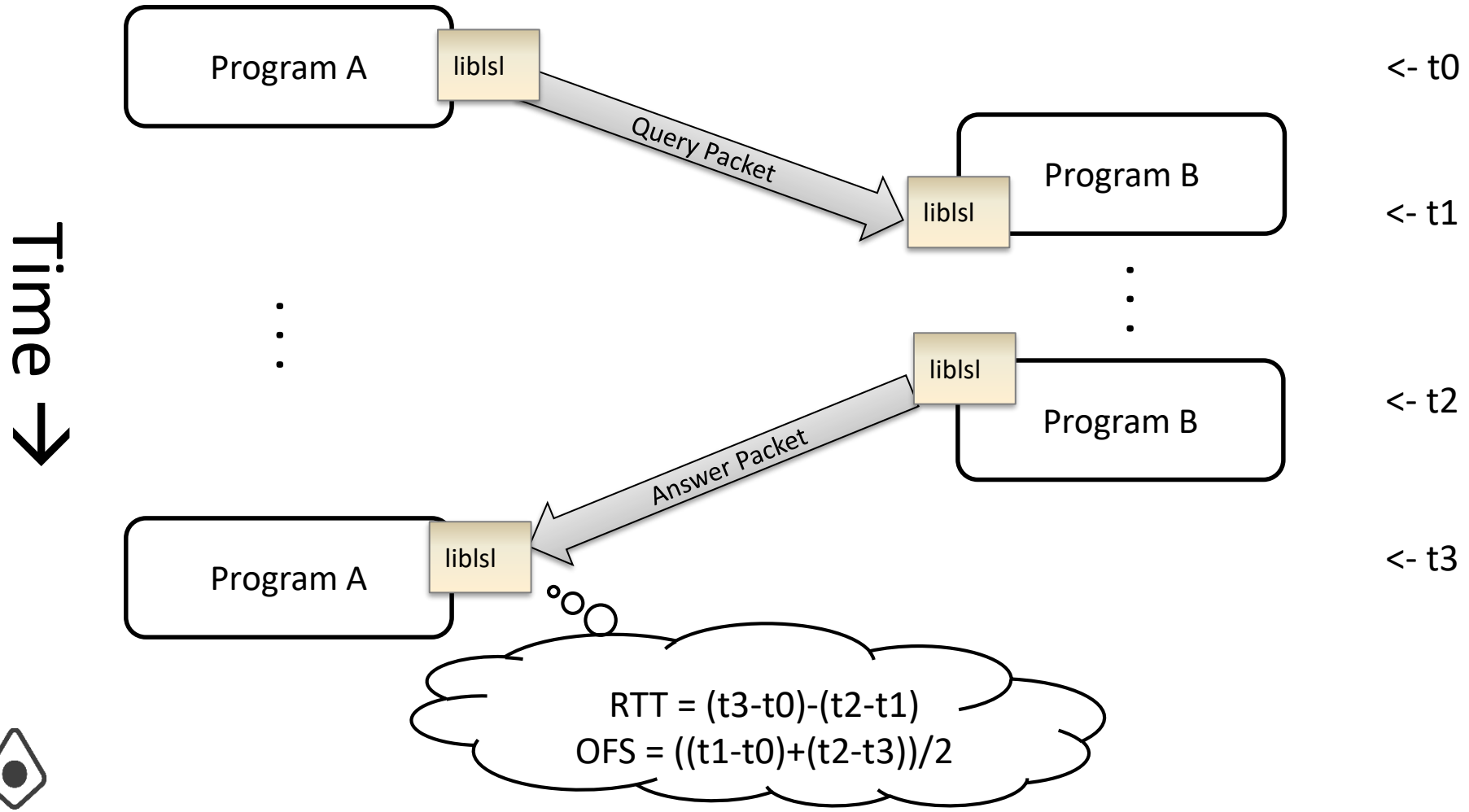


1. Stamp is taken on PC where event (e.g., sample) was observed

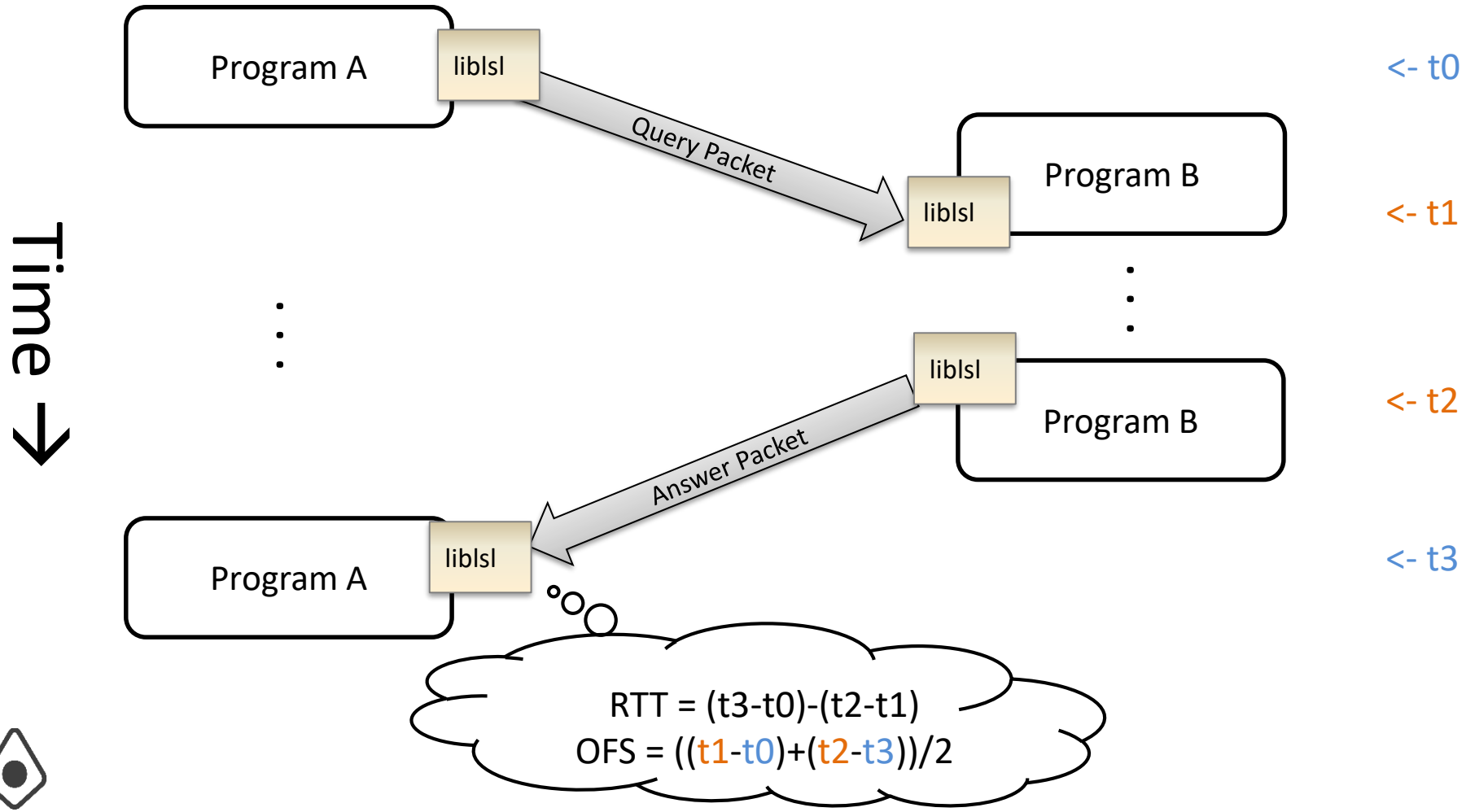
2. Stamp is transported via LSL

3. Stamp is recorded together with clock-sync information

LSL's Built-In Peer-To-Peer Time Synchronization



LSL's Built-In Peer-To-Peer Time Synchronization



Providing Your Own Timestamps

- An LSL data source can provide its own (or modified) timestamps -- otherwise LSL will stamp it at the time of data submission

```
# we happen to know that our sample was actually
measured 25ms ago by our device
stamp = local_clock() - 0.025
outlet.push_sample(mysample, stamp)
```

Performing Online Time Sync

- As a data recipient, you can ask LSL about the current clock offset for the received data
- Adding that number to the received raw timestamps (measured on the remote clock) remaps them to your own PC clock
- If you do that for all received streams, they're all on the same clock (yours) -- i.e., synchronized

```
# get the next sample and its timestamp
sample, timestamp = inlet.pull_sample()
# manually correct the time stamp that we just got
offset = inlet.time_correction()
timestamp += offset
```

Online Time Sync, Simplified

- An even simpler option is to enable automatic time correction upon receiving the data to have LSL do this for you

```
inlet.set_postprocessing(proc_clocksync)
```

Offline (Post-Hoc) Time Sync

- The preferred method for time-sync of recorded data is post-hoc correction (at load time) rather than online sync -- this way, the original time stamps are preserved
- Clock drift over entire experiment is estimated from collected clock-sync info in XDF file, and subtracted
- MATLAB & Python XDF importers do this for you (can be disabled if not desired)

Sources of Device Timing Error

- Clock Drift
- Internal Device Latency
- Time-stamp Jitter

Per-Device Latency

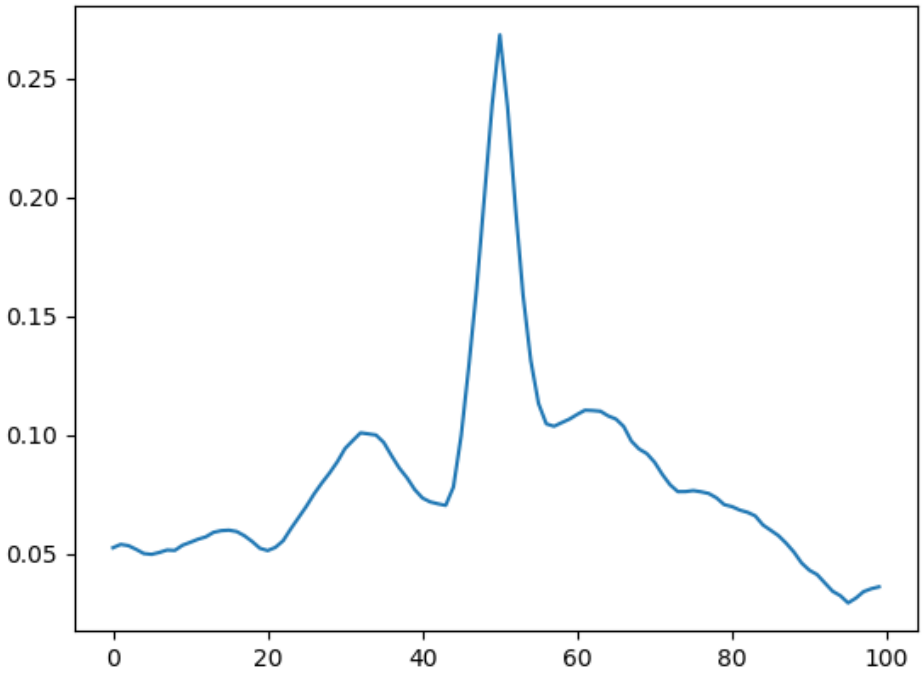
- Most acquisition devices (or driver) have internal latency (0.1-100ms)
- Single-sample latency of a device can vary significantly but average latency in a recording tends to depend only on setup parameters (hardware, drivers, settings)
- True latency of a device can be measured once per device setup (e.g., per study / per lab / under factory settings)
- Every acquisition program has a default assumed latency (many assume 0ms)

Pure Software Techniques for Device Latency Measurement

- Unknown latencies between devices can be estimated using cross-correlation (xcorr in MATLAB) if signals exhibit correlated features
- Example: photo diode vs. on/off event markers
- Can use CCA to do the same for multi-channel signals (e.g., eye tracker vs. EEG)
- Recommendation: when doing this analysis, always check consistency of estimated delays over time (e.g., 5-minute stretches) and across sessions

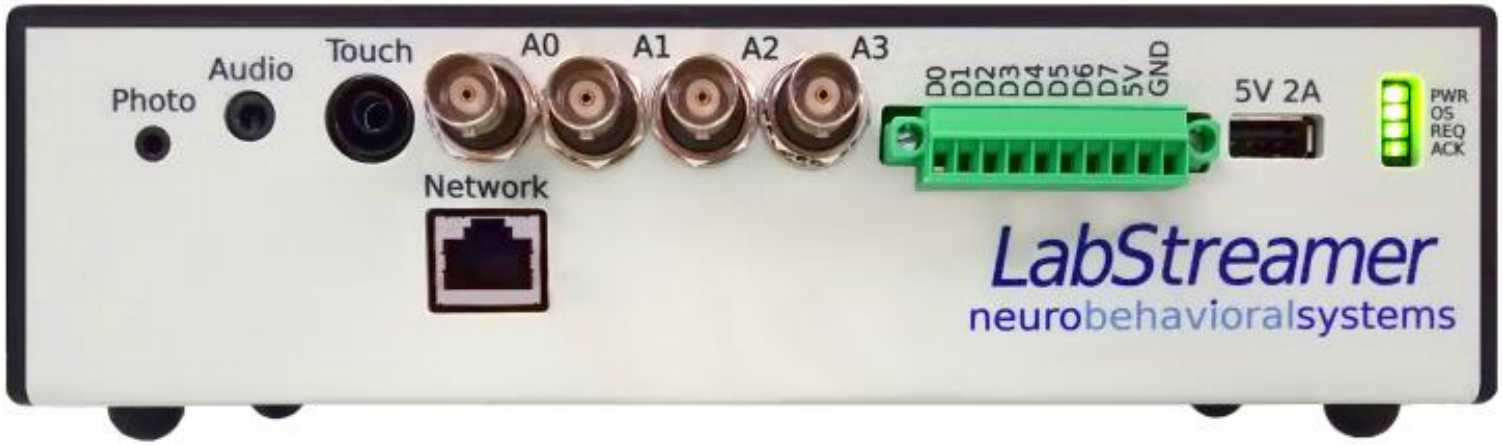
Pure Software Techniques for Device Latency Measurement

- Correlation analysis should yield a clear “spike” at the shift that yields maximum correlation between two signals



HW-Assisted Device Synchronization

- Can use off-the-shelf hardware to test out latencies of devices and/or emit sync data during experiments



- Can use generic DAQ cards and custom hardware solutions to measure latency

Time-Stamp Jitter

- Time-of-arrival of data in PC software is noisy, and so are the raw time stamps
- But actual measurements are typically* regular
- Can “smooth” (linearize) jittery time stamps to estimate the unobserved actual time stamps
- XDF Importers will do this for you post-hoc by default (can be disabled and done manually or skipped)
- LSL can also do it in real time:

```
inlet.set_postprocessing(proc_dejitter)
```

*: see David Medine’s Lecture

The Good News

- Time-stamp errors can be decomposed into The error components (latency + jitter + clock drift) can be separately measured and corrected for
- Odd cases (e.g., non-constant latency, irregular sampling rates, non-linear drift) can occur, and can usually be fixed (if detected)*
- Recommendation: perform pilot run and Actually Look At™ pilot data

*: Also see David Medine's Talk

Synchronization In Practice

- **Approach 1:**
  - Measure device latencies at least once (using trigger cables or a custom timing test experiment) and confirm that variability is acceptable
  - Re-measure after every significant setup change (e.g., driver update, OS/hardware change)
- **Approach 2:**
  - Rely on LSL time synchronization
  - Also record trigger channels as backup (possibly correct device latencies post-hoc)
- **Approach 3:**
  - Rely on trigger cables in every experiment and perform synchronization based on trigger information
  - Use LSL time-stamps as “safety net” when trigger sync fails (measure device latencies post-hoc)

2 Diagnosing Connectivity Issues

Something Doesn't Connect, What Now?

- **Possible reason:** Firewall
- Most likely reason is the OS firewall on either the sending or receiving machine
- Try to disable firewall and see if that resolves the issue – if yes, can try to re-enable more fine-grained application rules
- See also wiki article on "Network Connectivity"

Something Doesn't Connect, What Now?

- **Possible reason:** Additional Network Cards
- Current version of LSL only transmits over the primary network card
- Some software installs additionl "virtual" network hardware (e.g., Docker, Microsoft Hyper-V, Oracle VirtualBox) which can interfere with LSL – try temporarily disabling additional network adapters

Something Doesn't Connect, What Now?

- **Possible reason:** WiFi router blocks p2p packets
- Some WiFi routers in public spaces (e.g., campus/hotel networks) are configured to explicitly disallow packets commonly used for peer-to-peer service discovery (broadcast/multicast packets)
- To troubleshoot, try same applications on different (e.g., wired or home router) network – if they work, the network policy is likely the issue
- Note: some bad routers may seemingly randomly let some applications through and not others
- On these networks, machines do not see each other – only workaround is to set KnownPeers variable in config file (explicit list of IP addresses or hostnames)

Something Doesn’t Connect, What Now?

- **Possible reason:** Using wrong stream name/type/etc.
- Make sure that the stream that you’re transmitting really has the name and/or type that you expect, including capitalization (e.g., check in LabRecorder)

Something Doesn't Connect, What Now?

- **Possible reason:** Non-default session id used
- Every LSL application looks for a config file
- If present, that config file can override settings, including the "session id"
- Only LSL applications that have the same session id can see each other – by default it is "default", and unless you change it, any LSL application can see any stream
- Purpose is to allow for multiple recording activities that are sandboxed from each other
- However, easy to set a config file and later forget…

Troubleshooting Tools

- LSL comes with simple command-line applications that send and/or receive simple streams (e.g., SendStringMarkers/ReceiveStringMarkers, SendData/ReceiveData)
- If unsure whether the bug is in your application, can check if these tools can or cannot transmit – on a properly configured machine, these tools should **always** be able to transmit
- Another useful tool resolves all streams visible on the network and shows their exact metadata (GetAllStreams)

3 More Notes on Metadata

Why Set Metadata On Your Streams?

- LSL can handle any amount of per-stream metadata
- If added to a stream, metadata will be recorded to XDF files
- Available for later analysis
- Impossible to lose (compared to separate notes/files/etc)

How to Set Metadata?

- When declaring a stream (stream info object), metadata can be added using the .desc() field

```
% create a new StreamInfo and declare some meta-data (in accordance with XDF format)
info = lsl_streaminfo(lib,'MetaTester','EEG',8,100,'cf_float32','myuid56872');
chns = info.desc().append_child('channels');
for label = {'C3','C4','Cz','FPz','POz','CPz','O1','O2'}
    ch = chns.append_child('channel');
    ch.append_child_value('label',label{1});
    ch.append_child_value('unit','microvolts');
    ch.append_child_value('type','EEG');
end
info.desc().append_child_value('manufacturer','SCCN');
cap = info.desc().append_child('cap');
cap.append_child_value('name','EasyCap');
cap.append_child_value('size','54');
cap.append_child_value('labelscheme','10-20');
```

What To Set?

- Stimulus presentation program
  - Configuration settings of the session/run (can also write as event marker), random seed, sizes of stimuli, etc.
  - For time series (e.g., on-screen trajectories): channel names, channel unit (e.g., pixels/normalized/…)
- Device/sensor application
  - Channel labels etc., information about device (model, manufacturer, serial no., etc)

Thanks! ☺

Q&A until 5pm