**Tests: Implementing synchronization on Gstreamer using gst-launch script interface**

**Test1:** We run master and slave 1 at first PC and slave 2 on second PC and record each 15 seconds in two channels of each slave (slave1 and slave2). First, as soon as gstreamer master run and started first record name “ after 0 min 15 seconds” , “after 2 min 15 seconds ” and 3 and 5 respectively. (taken approximately the minutes after gstreamer master starts as a reference”). In this test the maximum time duration is 5 minute 15 second and minimum duration is 15 second directly started to record and 4 different records have been conducted. The observation here is when the time increased, in slave 2 each 15 seconds records shifts as seen in the audio editor in compared to slave 1 record.

**Test2:** This test takes 10 minutes records that contains audio record and master and slaves log files. Test includes Unknown PCM default and Unknown parameters error at the log files that made unclear about streaming data is perfectly streaming or not but in audio editor files it look streamed though.

**Test3:** This test presents 15 min record with slave-method=3 configured.  We don’t see any clock skew corrections by the alsasink despite the skew grows constantly over time. This can be cause by the timestamping (pts) of the buffers from the sender side or some pts recalc discontinuity the rtp jitterbuffer component.

**Test4:** Here in test4 the gains on the sound card setted in same scale. Test has been conducted for 10min. Observations: desibel differences both slaves, sometimes disortation occured on signal.

**Test5:** The master and slave PCs are switched and recorded audio during 10 min. PC1 that used to have master and slave1 and PC2 had slave2 privously now it is opposite in the test5 which means PC1 have slave2, PC2 has master and slave1 configurations.

**Test6:** here master and slave PCs are switched and then recorded audio using slave3 method oppositely to test3.

**Test7**: Just the test for rtpjitterbuffer mode=0 for 4 minute.

**Test8**: Here, recording is containing four set of audio recording 1 minute in each set. The following observations are from the last piece of the all 1 minutes recording

At first here observed both channel peace are seem perfectly synchronized, but the skew gets larger when the time duration is increases and at 15 minute the skew reached 14ms.

gstreamer strings: rtpjitterbuffer mode=0 (Control the buffering and timestamping mode used by the jitterbuffer, only using RTP timestamps)

slave-method=1, Different possible clock slaving algorithms used when the internal audio clock is not selected as the pipeline master clock

GST\_AUDIO\_BASE\_SINK\_SLAVE\_SKEW (1) – Adjust playout pointer when master clock drifts too much.)

drift-tolerance=10000 (Controls the amount of time in microseconds that clocks are allowed to drift before resynchronisation happens)

discont-wait=1000 (A window of time in nanoseconds to wait before creating a discontinuity as a result of breaching the drift-tolerance)

We have tested 3 tests in the case only changes of the rtpjitterbuffer with rtpjitterbuffer mode=0,1,4 (test8, test9, test10) based on these records we have observed less skew in only use of RTP timestamps mode=0. Afterward tested using RTP timestamps mode=0 with GstAudioBaseSinkSlaveMethod and by using custom clock slaving algorithm which is slave-method=3 records gave to me less skew compared to other slave methods.

**Test9**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording.

(Audio 1 #1) 0 min - 1 min : slave 2 is 7ms (329 samples) forward from slave1

(Audio 1 #2) 5 min - 6 min : slave 2 is 5ms (245 samples) forward from slave1

(Audio 1 #3) 10 min - 11 min : slave 2 is 13ms (362 samples) forward from slave1

(Audio 1 #4) 15 min - 16 min : slave 2 is 21ms (1031 samples) forward from slave1

rtpjitterbuffer mode=1 (Slave receiver to sender clock) mode made the higher skew in compare to mode =0 Control the buffering and timestamping mode used by the jitterbuffer and we kept the following parameters same as a reference to compare with test8 to see the effect of the rtpjitterbuffer modes slave-method=1, drift-tolerance=10000, discont-wait=1000

**Test10**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording

(Audio 1 #1) 0 min - 1 min : slave 2 is 3ms (151 samples) forward from slave1

(Audio 1 #2) 5 min - 6 min : slave 2 is 11ms (543 samples) forward from slave1

(Audio 1 #3) 10 min - 11 min : slave 2 is 20ms (938 samples) forward from slave1

(Audio 1 #4) 15 min - 16 min : slave 2 is 18ms (850 samples) forward from slave1

**Test11**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording. By the way the following output are conducted from Audio 1 and recorded 1 minute for each set of audios. For example, Audio 1 #1 is recorded 0 to 1 minute’s period of time and Audio 1 #2 is recorded 5 minutes to 6 minute.

Also, in order to have reference for example the skew has been analyzed from last peak of each record of 1 minute records.

(Audio 1 #1) 0 min - 1 min : slave 2 is 2ms (113 samples) forward from slave1

(Audio 1 #2) 5 min - 6 min : slave 2 is 8ms (379 samples) forward from slave1

(Audio 1 #3) 10 min - 11 min : slave 2 is 13ms (632 samples) forward from slave1

(Audio 1 #4) 15 min - 16 min : slave 2 is 19ms (897 samples) forward from slave1

**Test12**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording

(Audio 1 #1) 0 min - 1 min : slave 1 is 6ms (272 samples) forward from slave 2

(Audio 1 #2) 5 min - 6 min : slave 2 is 3ms (241 samples) forward from slave1

(Audio 1 #3) 10 min - 11 min : slave 2 is 11ms (511 samples) forward from slave1

(Audio 1 #4) 15 min - 16 min : slave 2 is 18ms (900 samples) forward from slave1

**Test13**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording

(Audio 1 #1) 0 min - 1min : slave 1 is 2ms (98 samples) forward from slave 2

(Audio 1 #2) 5 min - 6 min : slave 1 is 4ms (187 samples) forward from slave 2

(Audio 1 #3) 10 min - 11 min : slave 1 is 6ms (227 samples) forward from slave 2

(Audio 1 #4) 15 min - 16 min : slave 1 is 8ms (367 samples) forward from slave 2

When playing audio streams, it is important to ensure that the audio remains in sync across multiple devices. One way to measure this synchronization is to use the concept of skew and offset.

Skew refers to the difference in timing between two audio streams, while offset is a measurement of the time difference between the two streams relative to a common reference point. The reference point used can be either the start of the audio playback or some other predetermined point in time.

Tests results are showing that the timestamp vs basetime issues occurred.

In the context of this outcome provided, the measurements indicate that slave 1 is playing audio that is ahead of slave 2 by a certain number of milliseconds. The amount of skew is increasing over time, indicating that the synchronization between the two slaves is becoming worse as time goes on.

The base time and running time are related to the reference point used to measure the offset. The base time is the starting point of the audio playback, while the running time is the time elapsed since the start of playback. By measuring the offset at different points in time, it is possible to calculate the skew and adjust the playback timing of one or both devices to maintain synchronization.

**Test14**: Here, recording is containing four set of audio recording 1 minute in each sets. The following observations are from the last piece of the all 1 minutes recording

(Audio 1 #1) 0 min - : slave 2 is 2ms (95 samples) forward from slave 1

(Audio 1 #2) 5 min - 6 min : slave 2 is 0ms (6 samples) forward from slave 1

(Audio 1 #3) 10 min - 11 min : slave 1 is 1ms (85 samples) forward from slave 2

(Audio 1 #4) 15 min - 16 min : slave 1 is 3ms (172 samples) forward from slave 2

we have identified some gst-launch interface limitations, the gst-launch script interface only provide static configuration options and no control of the actual pipelines. which can only be controlled at C/C++ level as done in the isobel framework.Based on these facts and the simulation needs, it make sense to have an simple C/C++ implementations provide the needed control and also provide an flexible architecture for further experiments

**Test** **drift\_tolerance:** While drift tolerance =10, the skew gets closer after each set of records and after 15-minute slave 1 is 0ms (21 samples) forward from slave2, reached. While drift tolerance=100, the the skew gets started with 4ms skewed and reduced after some time such as slave 2 is 0ms forward from slave1 after 10 minute record (almost perfectly sychronized). While drift tolerance=1 and drift tolerance=1000 both are the skew gradually increases and slave 1 is always forward from slave 2.

Drift tolerance can affect the synchronization of audio streams by defining the acceptable range of deviation between the timebase of the sender and receiver. If the deviation exceeds the drift tolerance the case of 1000, it may result in the audio streams becoming out-of-sync that supports the argument.

**Test** **drift\_tol\_updated:** The test drift\_tolerance is preapeted and the experiment is updated with buffer mode=0 and observed while tolerance increase skew get larger again. The test contains the new test with tol=1,10,100,1000 values respectively. The following observation is conducted, at first with tol=1 the skew seems smaller and while tol value increases the skew gets larger.

**Test discont\_wait:** Here are the tests with 1,10,100 discont wait values and seems it has very different effect to the skew in different value. You can see the skew with millisecond in readme as usual. There is no direct propagation ration with increasing of the discount value. During the record of 15 min - 16 min, slave 2 is 1ms forward from slave1 however it was forward 6ms at the 0 min e- 1 min time duration. For the 1 and 10 values the skew value start very low and increases during the time.

**Test join\_slave\_tests:** Here is making tests regarding the timestamp vs basetime issues, running time of master before join of remote client (audio sync skew offset vs base time and running time). Running first master and slave1 for 5 min, then join slave 2 and record 5 minute more and repated for 10 min same. The skew gets larger same as earlier that we discussed when the time take larger the skew gets larger.

**Test align\_tests:**The alignment threshold tests with respect to 1,10,100 values. While alignment threshold increases the skew increases with direct proportion for all test it seems, for all the test based on the records, it can be conducted after 15 minute the skew in range of 8ms, also for the other time duration the skew values are quite close to each other in milisecond domain, not much difference occuried in compared to the other parameters changing such as drift tolerance. But of course this does not mean the skew always keep same in each time duration of records in each alignment threshold values.

**Test join\_slave\_2\_tests:** Here in this test join slave tests repeated. The first test (test1) is the slave 1 and master run 5 minutes together first and then slave2 start running all together 5 minutes more. The slave 1 and master run 5 minutes together first and then slave2 start running all together 5 minutes more. When slave 2 join master and slave 1 after 5 minutes, slave 1 is forward 11ms than slave 2, at 10th minute slave 1 is forward 13ms than slave 2.

The second test (test2) is the slave 1 and master run 10 minutes together first and then slave2 start running all together 5 minutes more. The slave 1 and master run 10 minutes together first and then slave2 start running all together 5 minutes more. When slave 2 join master and slave 1 after 10 minutes, slave 1 is forward 27ms than slave 2, at 15th minute slave 1 is forward 29ms than slave 2.