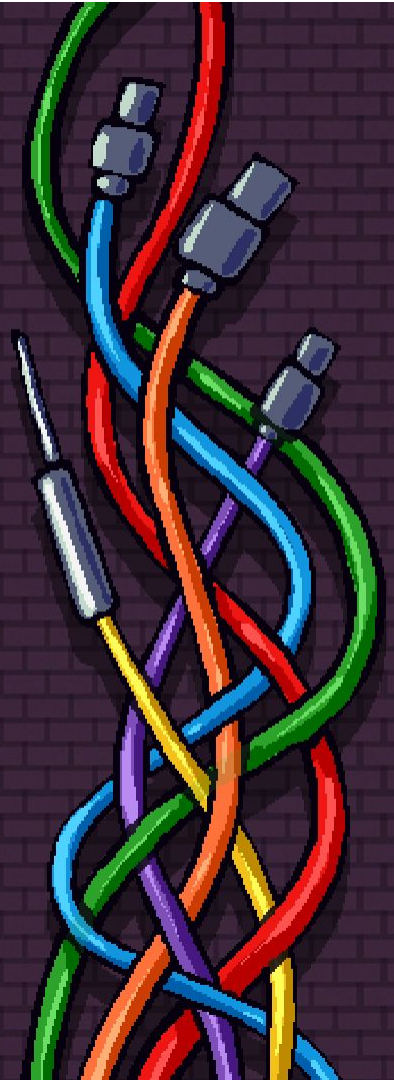


ADCx GATHER

USING KOTLIN/COMPOSE MULTIPLATFORM TO REVIVE A HISTORIC MULTIPLAYER ONLINE DRUM MACHINE

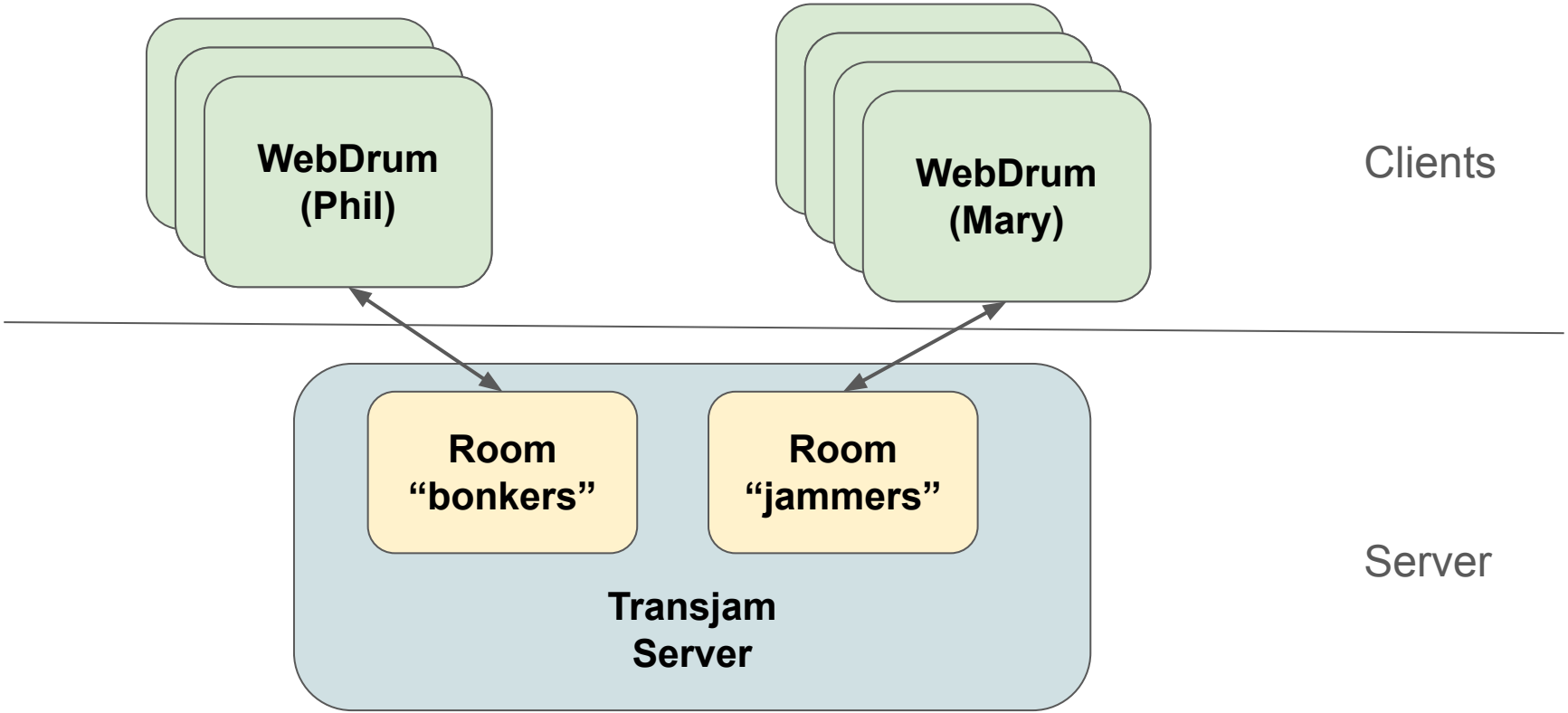
PHIL BURK

Topics

- What is WebDrum?
- Using Kotlin/Compose Multiplatform
- Multiplatform **audio solution - kc-audio-bridge**
- Networking with ktor and WebSocket
- Other problems and solutions
- Online drum jam at https://transjam.com

Updated 2025-09-25

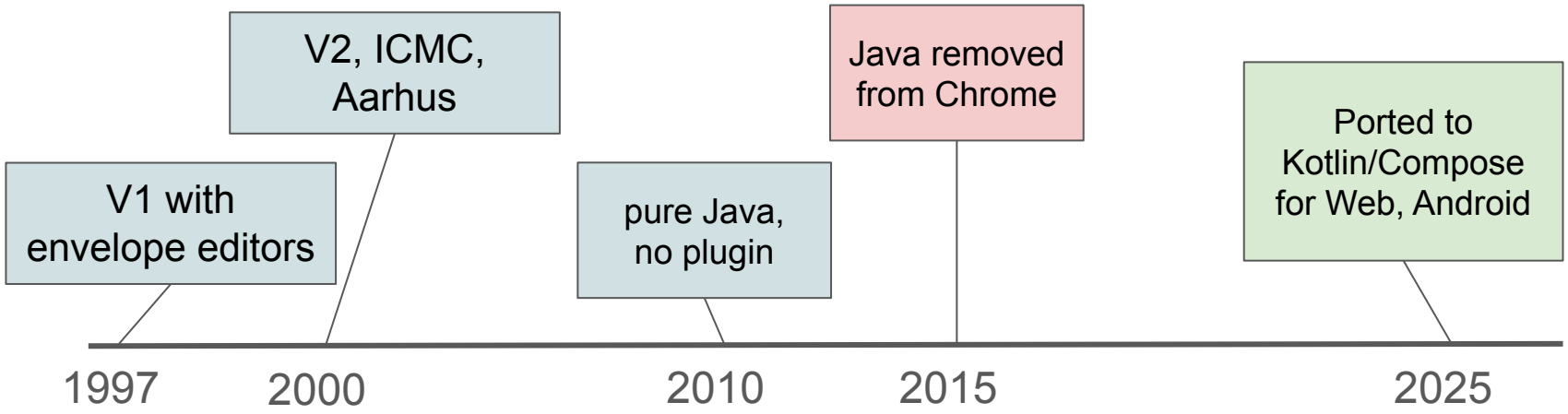
WebDrum is a Multi-Player Drum Machine



V2 UI
in Java

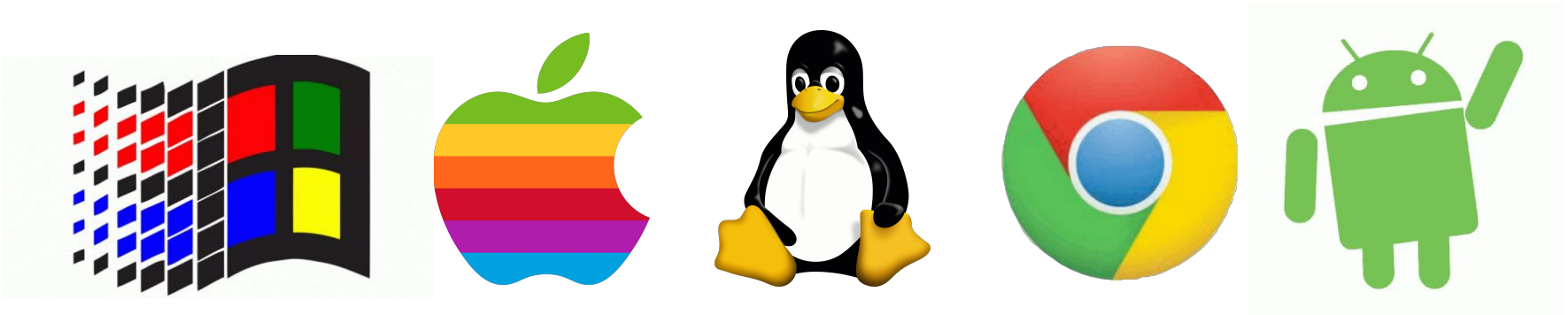
Y2000

WebDrum Timeline



Kotlin/Compose to the rescue

- By JetBrains, with Google, use Android Studio
- Multiplatform system for Web/WASM, desktop, Android, iOS

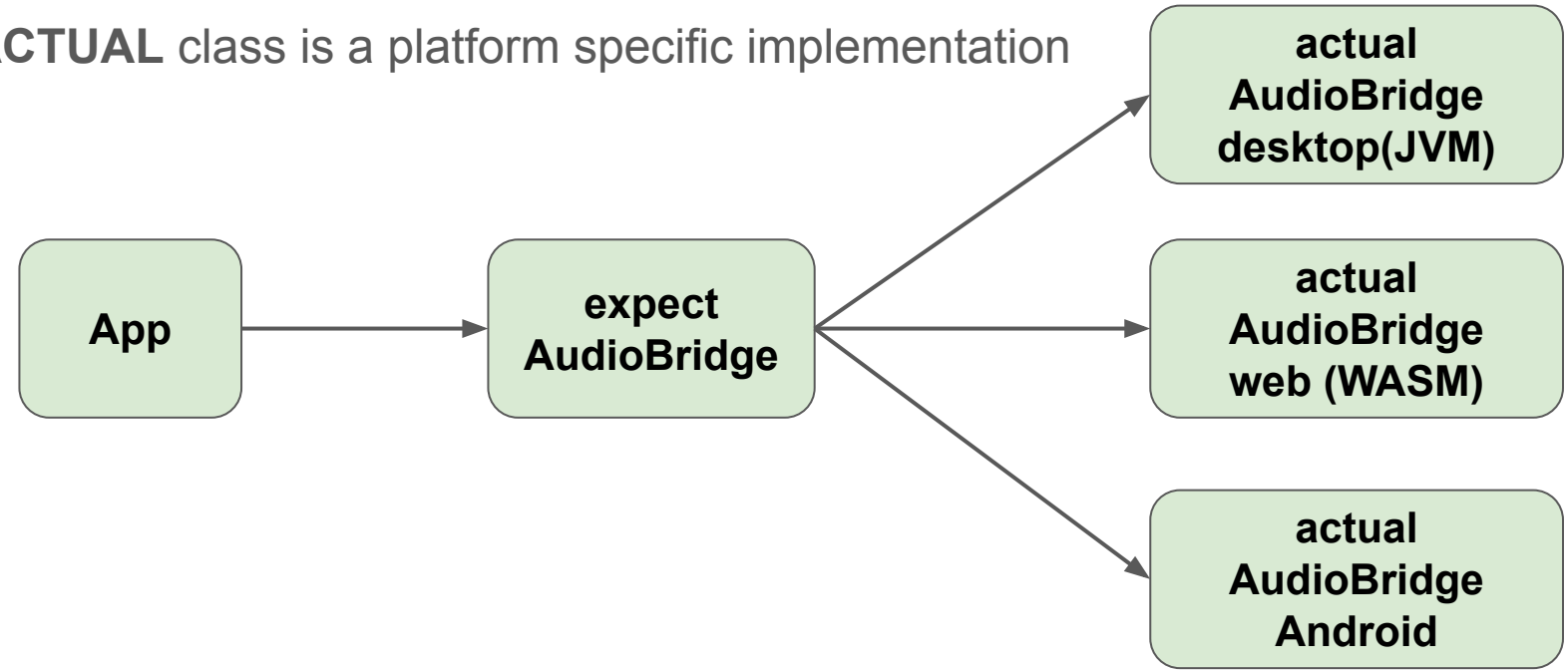


- **Compose** = “functional” language for UI
- But **NO** cross-platform **audio** support!

kc-audio-bridge

Uses Expect/Actual classes for Audio

- **EXPECT** defines an interface
- **ACTUAL** class is a platform specific implementation



AudioBridge Open Source on GitHub

AOSP license

Implementations for Android, desktop (JVM) and Web

https://github.com/philburk/kc-audio-bridge

Web demo (play sine waves)

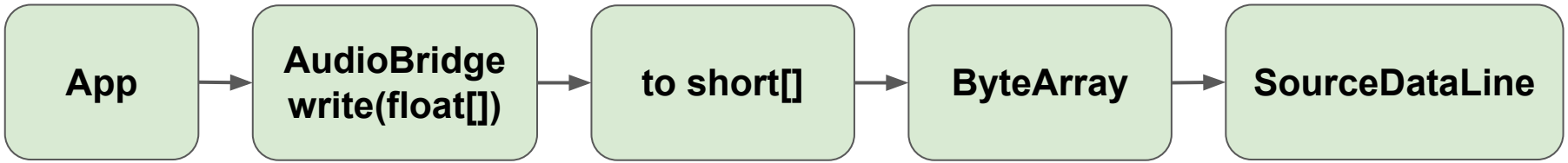
https://transjam.com/kc-audio-bridge/

AudioBridge API

```
val audioBridge = AudioBridge()
val openResult = audioBridge.open() // stereo, float
val sampleRate = audioBridge.getSampleRate() // usually 44100
val job = GlobalScope.launch(Dispatchers.Default) {
  val startResult = audioBridge.start()
  loop {
    renderAudio(stereoBuffer) // your code goes here
    val frameCount = audioBridge.write(stereoBuffer,
                                       offsetFrames, numFrames)
  }
}
```

Actual Audio on Desktop

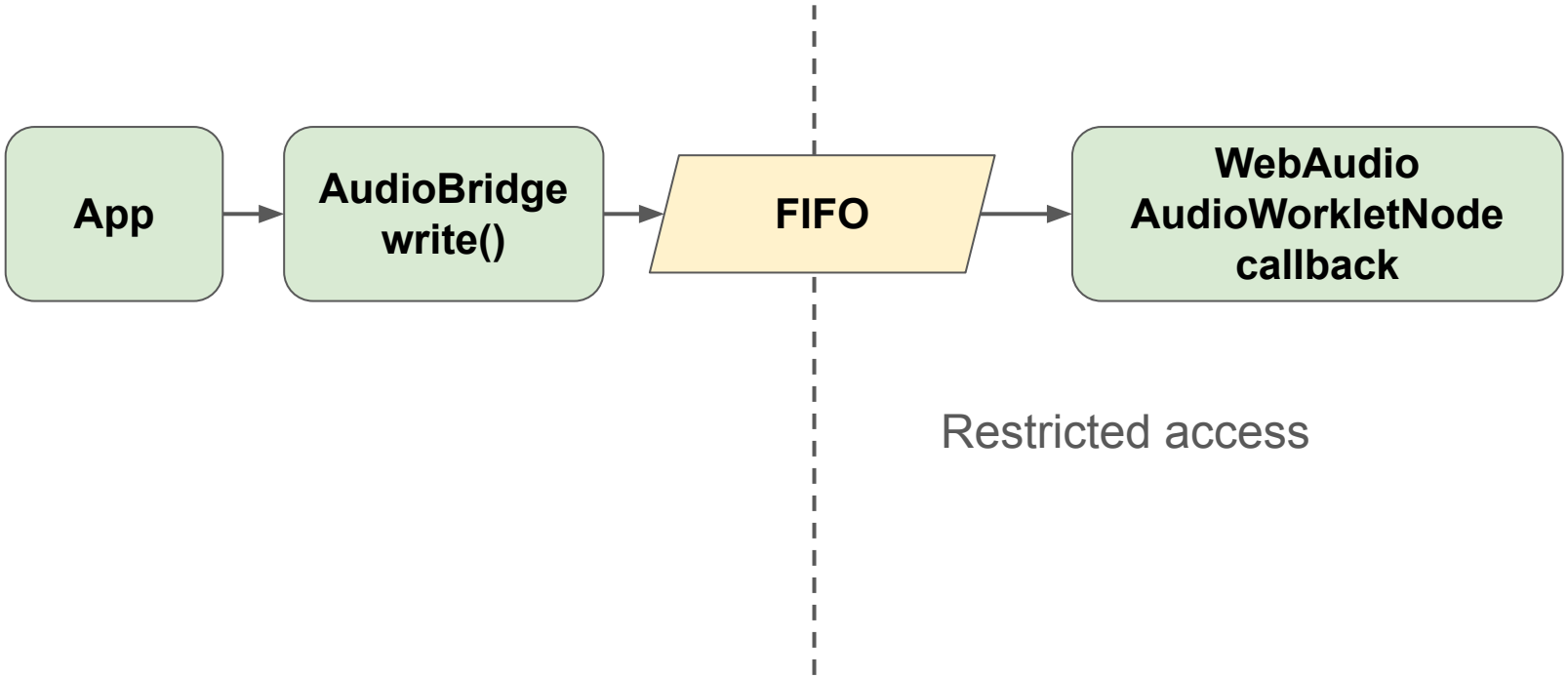
- For JVM, use **JavaSound** API
- Must convert float[] to short[], then to bytes



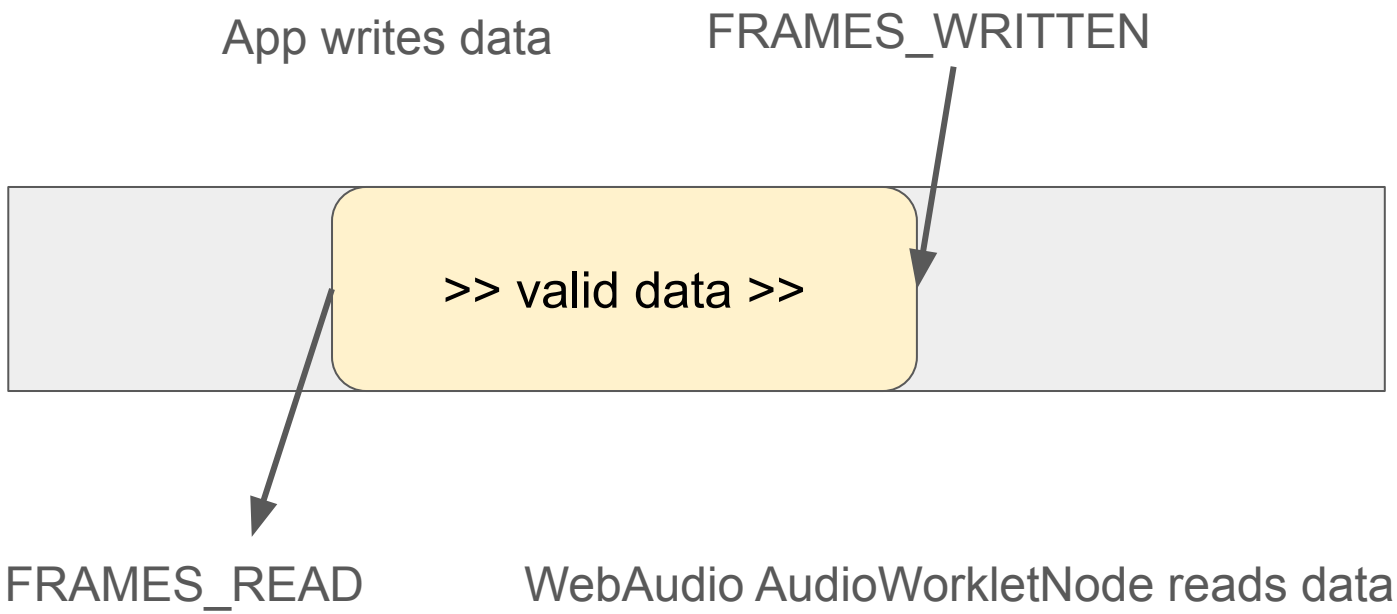
Actual Audio on Android

```
actual fun write(buffer: FloatArray,
                 offsetFrames: Int,
                 numFrames: Int): Int {

    val numFloatsOrError =  mAudioTrack?.write(buffer,
            offsetFrames * mChannelCount,
            numFrames * mChannelCount,       // numFloats
            AudioTrack.WRITE_BLOCKING
        )?: AudioTrack.ERROR_DEAD_OBJECT

   return if (numFloatsOrError < 0) numFloatsOrError else
        (numFloatsOrError / mChannelCount)
}
```

Actual Audio on **Web**



Atomic FIFO


App writes data
FRAMES_WRITTEN
>> valid data >>
FRAMES_READ
WebAudio AudioWorkletNode reads data

WASM uses SharedArrayBuffer

- **Float32Array** for audio samples
- **Int32Array** for atomic cursors and capacity

```
const floatBufferSizeBytes = Float32Array.BYTES_PER_ELEMENT *
                                 capacityInSamples;
const floatSharedBuffer = new SharedArrayBuffer(floatBufferSizeBytes);
const sharedFloatArray = new Float32Array(floatSharedBuffer);


const intBufferSizeBytes = Int32Array.BYTES_PER_ELEMENT * 3;
const intSharedBuffer = new SharedArrayBuffer(intBufferSizeBytes);
const sharedIntArray = new Int32Array(intSharedBuffer);
```

WASM uses Atomics

- Need JavaScript Atomics for memory barriers across threads
- no float[] parameters, single {left,right} frames

- some audio callback:

```
this.sampleMask = this.capacityInSamples - 1;

const framesRead = Atomics.load(this.sharedIntArray, INDEX_FRAMES_READ);

// Audio stream must be deinterleaved
const readIndex = sampleOffset & this.sampleMask;
outputChannel[i] = this.sharedFloatArray[readIndex];

Atomics.store(this.sharedIntArray,
              INDEX_FRAMES_READ, framesRead + framesPerBurst);
```

WASM SharedArrayBuffer requires HTTPS and…

- a **.htaccess** file to enable Cross-Origin access

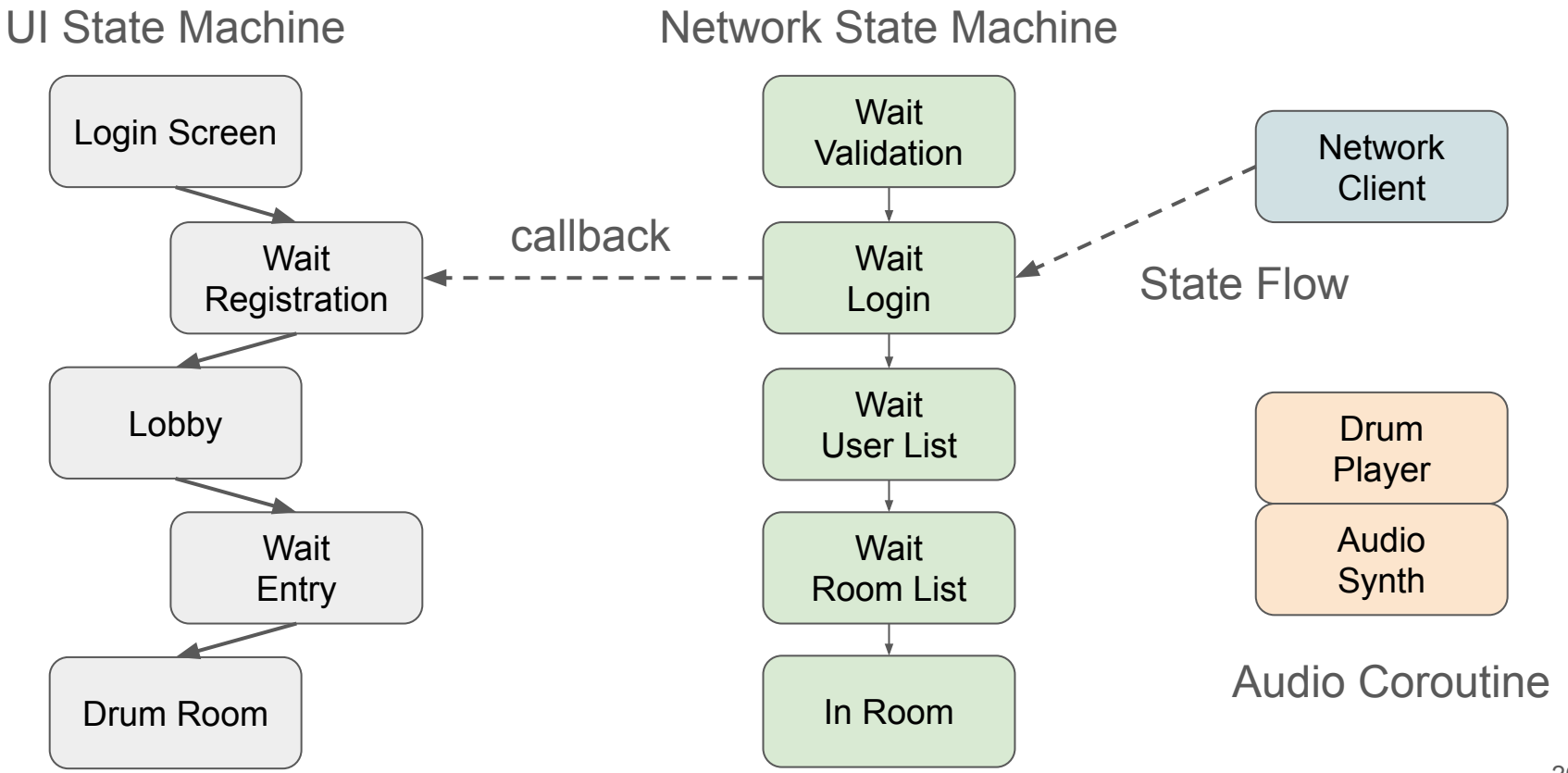
```
# Add the header to all resources served from this directory and below
Header set Cross-Origin-Resource-Policy "same-site"
Header set Cross-Origin-Embedder-Policy "require-corp"
Header set Cross-Origin-Opener-Policy "same-origin"
```

Future for kc-audio-bridge

- API is **experimental**, may change
- **Native** desktop support using **PortAudio**
- **iOS** support? (volunteers?)
- Audio Input?, difficult because of security

Other Changes Needed?

Refactor for State Flow and Coroutines



Original WebDrum used JSyn for instruments

Converted JSyn sounds to samples

Loading multiple samples had high latency, >100 msec per sample

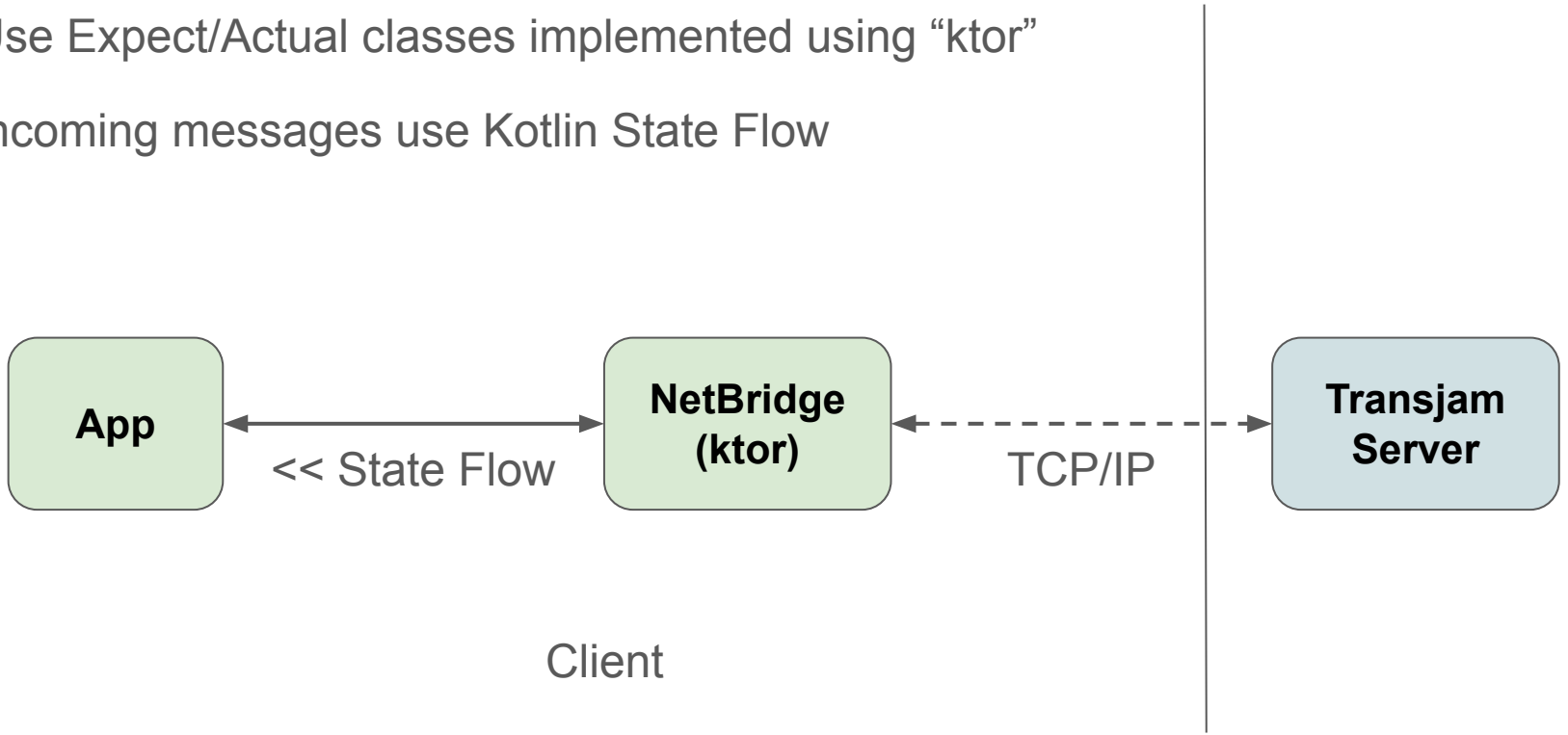
Pack WAV files into a single file for faster loading

Future: convert JSyn to KSyn, pure Kotlin, all math

Network Connection

Use Expect/Actual classes implemented using “ktor”

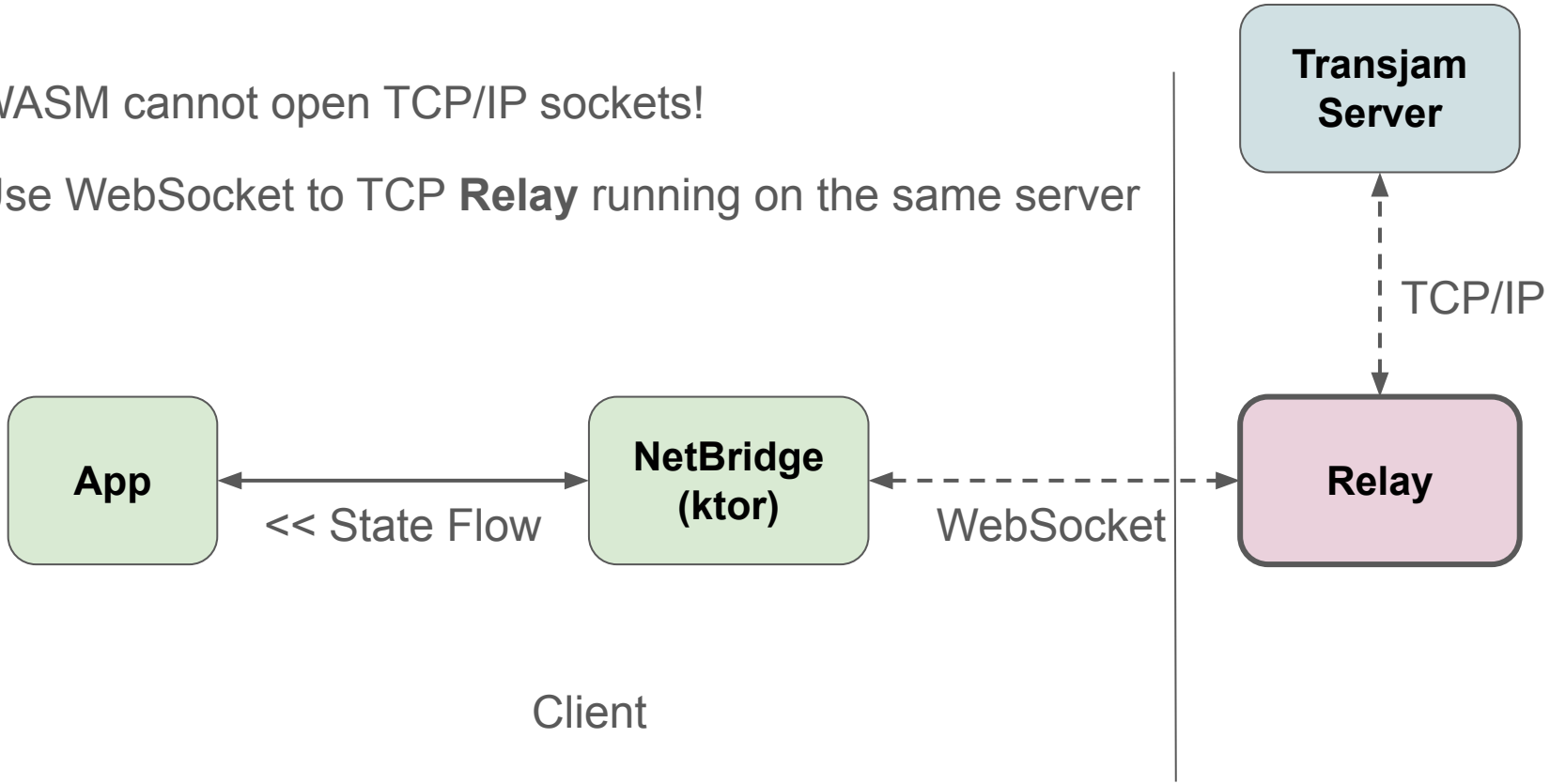
Incoming messages use Kotlin State Flow



Network Connection for Web

WASM cannot open TCP/IP sockets!

Use WebSocket to TCP **Relay** running on the same server



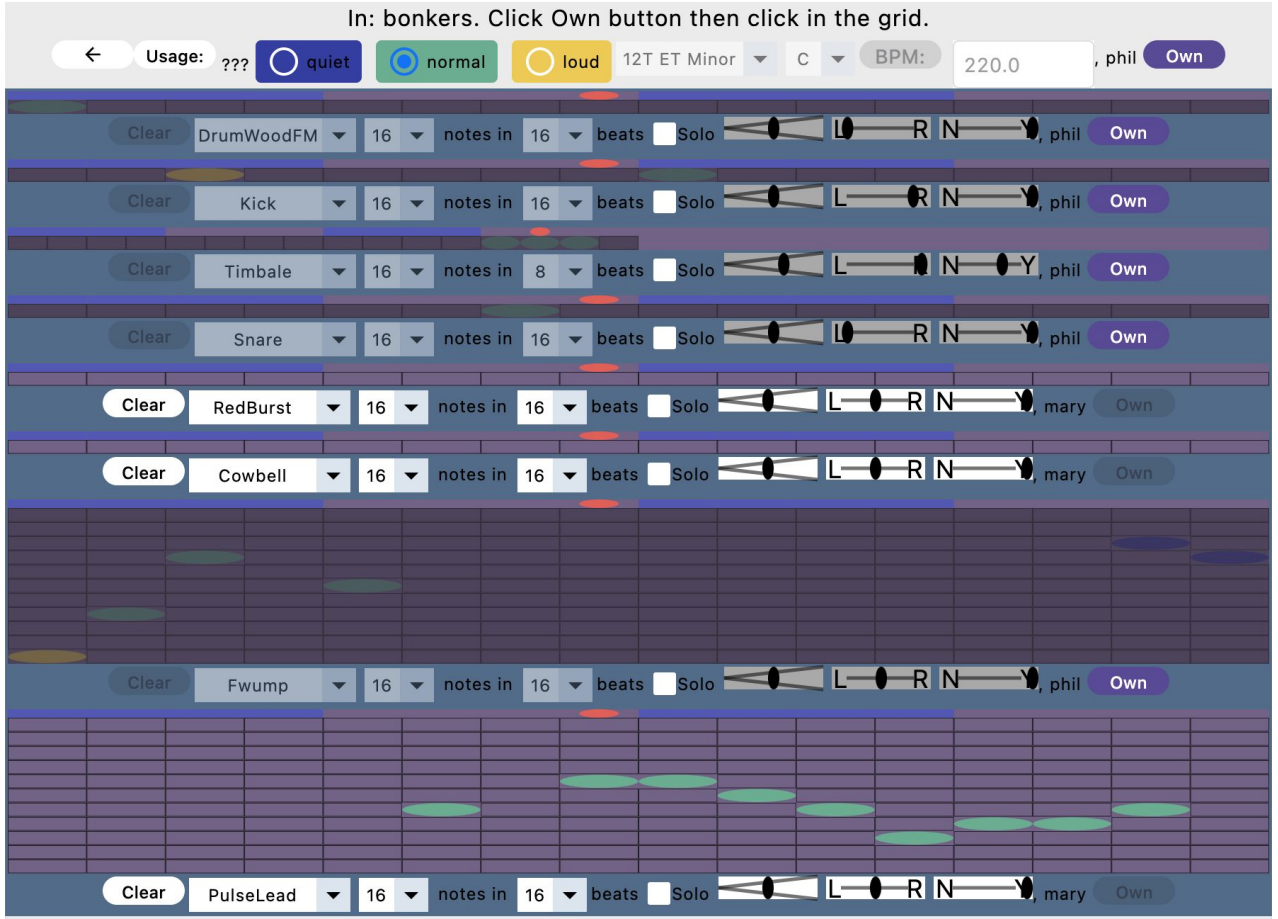
Many Java Exceptions not supported

Need to replace IOExceptions with custom Exception subclasses

```
open class CommonIOException(message: String? = null,
        cause: Throwable? = null) : Exception(message, cause)


@kotlin.Throws(CommonIOException::class, CancellationException::class)
suspend fun handleMessage(client: Client, msg: TransjamMessage): Boolean
```

V3 UI
in
Kotlin/Compose

recreate
Y2000
original



Line Count

Kotlin = 8105

Java = 10584 (more JSyn code)

Conclusions

Use Kotlin/Compose for portable audio and music apps.

Use kc-audio-bridge as an audio solution.
Please contribute. Make it better together.

Let’s jam!

https://transjam.com