

Tencent 腾讯 |  腾讯云

基于WebAssembly构建 Web端音视频通话引擎

田建华



1 背景

2 WebAssembly引擎

3 方案落地

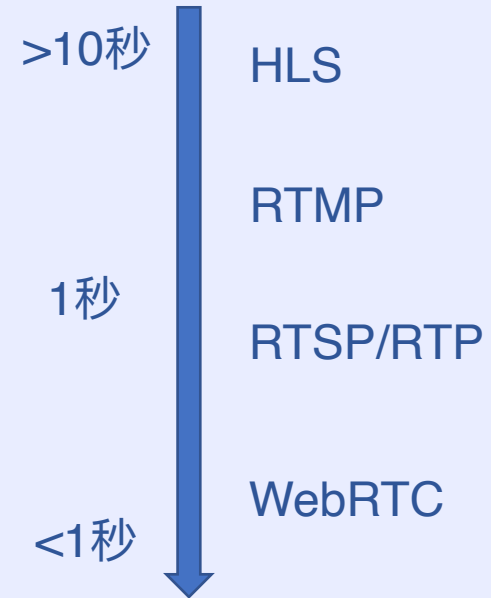
4 问题及展望

1

背景

背景

- 网络基础设施的升级
- 音视频传输技术的迭代
- 音视频消费习惯转变



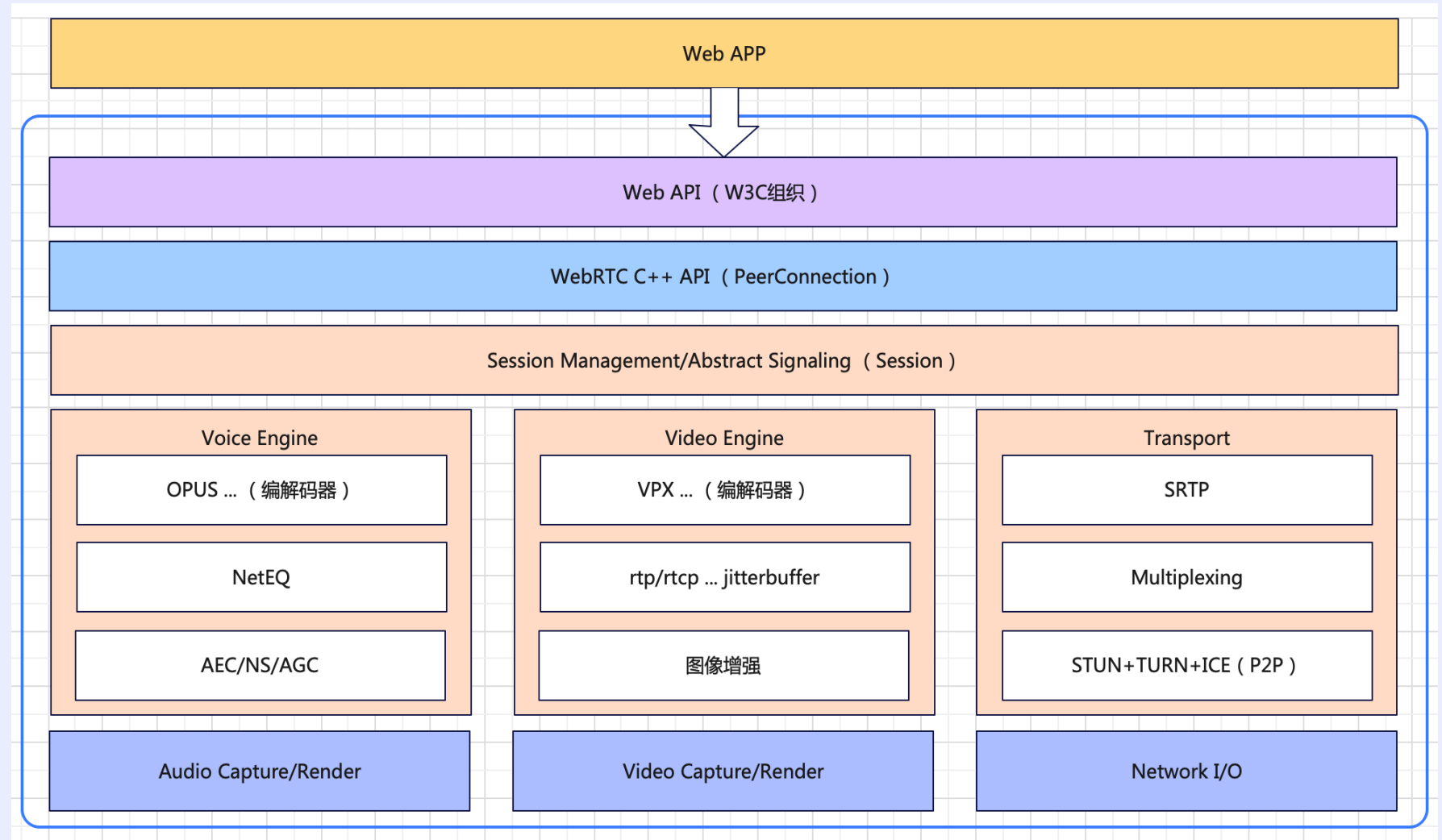
WebRTC架构

有哪些优势?

- 结构简单且开发难度相对低
- 无需插件

有哪些劣势?

- 不能自定义编解码器
- 不能复用现有服务框架以及优化能力
- 可定制化程度低



有没有新的Web技术作为替代来解决WebRTC的问题呢？

WebAssembly

什么是WebAssembly

- WebAssembly是一种运行在现代浏览器中的新型代码，并且提供新的性能特性和效果

目标

- 快速
- 高效
- 可移植
- 可读
- 可调试
- 安全
- 不破坏网络

能解决什么问题

- 解决JavaScript在复杂场景的性能问题——3D 游戏、计算机视觉、图像/视频编辑等以及大量的要求原生性能的其他领域
- 解决下载、解析JavaScript应用程序成本高的问题——WebAssembly体积更小

WebTransport

什么是WebTransport

- WebTransport是一个全新的可插拔的通信协议，支持可靠和非可靠传输

目标

- 更快速
- 更高效
- 安全
- 低延时

能解决什么问题

- 链接迁移
- 灵活的拥塞控制、更好的弱网能力
- 队头阻塞
- 灵活的传输方式（可靠和非可靠）

WebCodecs

什么是WebCodecs

- WebCodecs为开发人员提供了一种使用浏览器中已经存在的媒体组件的方法

能解决什么问题

- 低延时
- 提供更灵活的配置接口

```
dictionary VideoEncoderConfig {  
  required DOMString codec;  
  [EnforceRange] required unsigned long width;  
  [EnforceRange] required unsigned long height;  
  [EnforceRange] unsigned long displayWidth;  
  [EnforceRange] unsigned long displayHeight;  
  [EnforceRange] unsigned long long bitrate;  
  double framerate;  
  HardwareAcceleration hardwareAcceleration = "no-preference";  
  AlphaOption alpha = "discard";  
  DOMString scalabilityMode;  
  VideoEncoderBitrateMode bitrateMode = "variable";  
  LatencyMode latencyMode = "quality";  
};
```

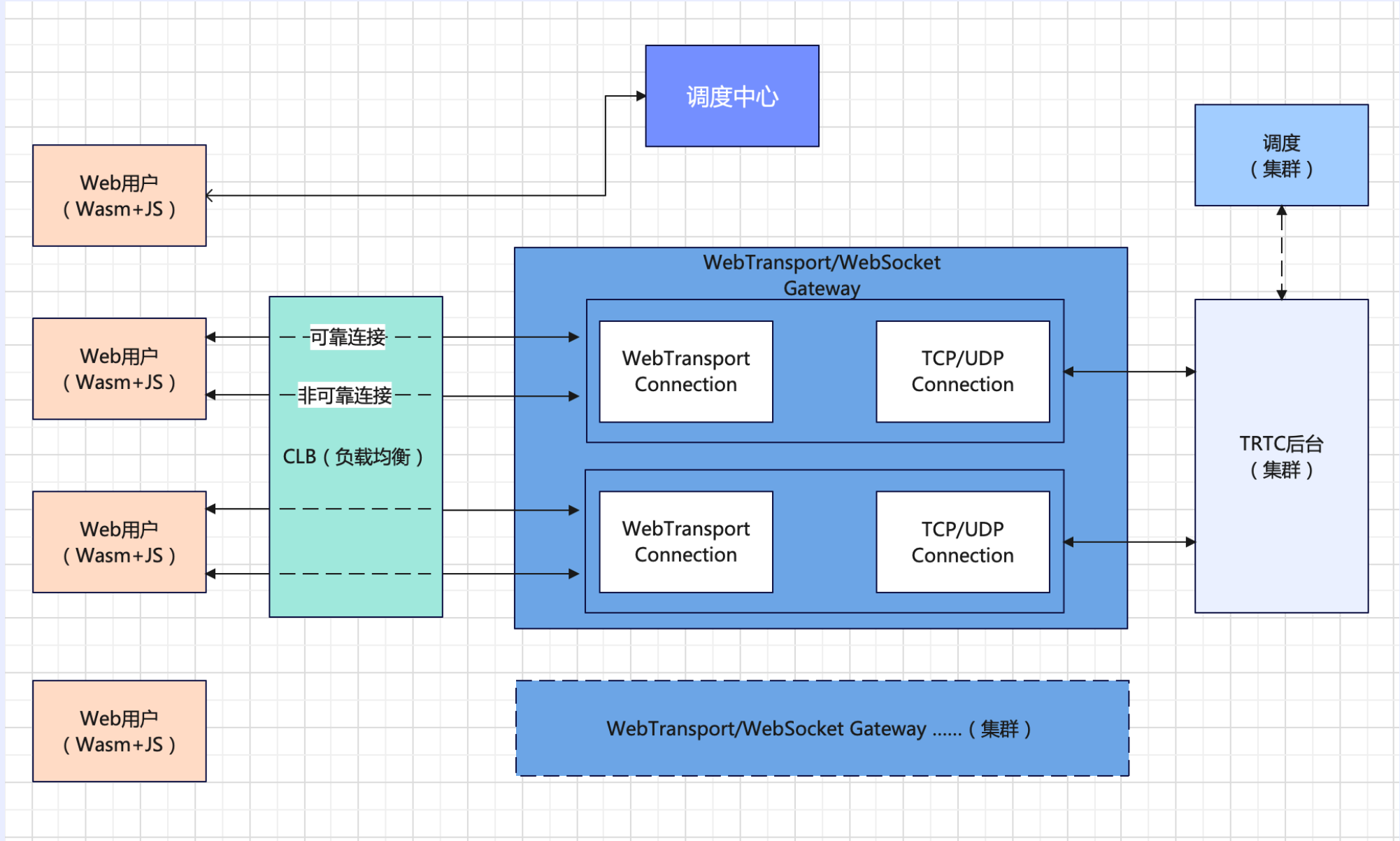
2

WebAssembly引擎

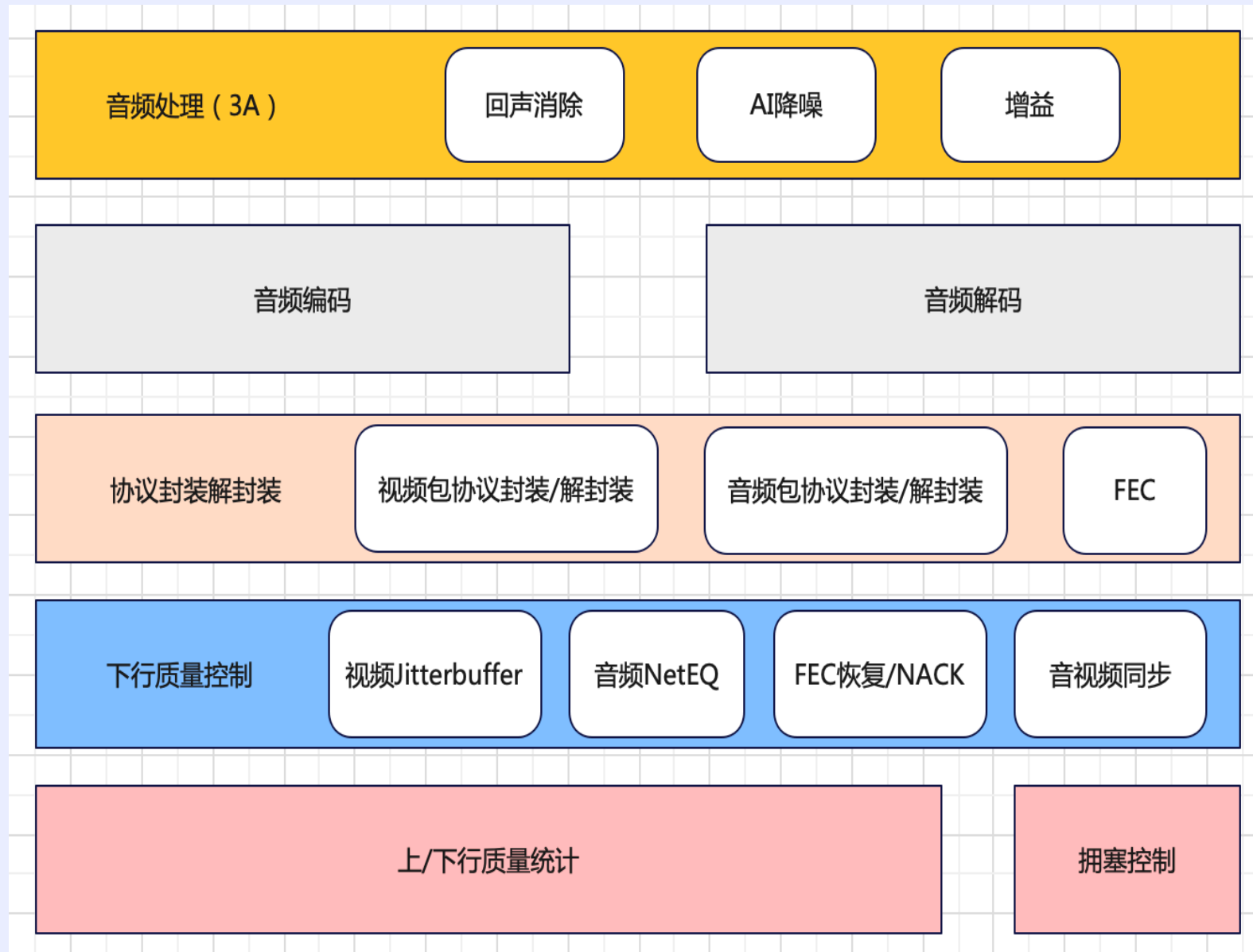
新技术和新架构致力于给用户提供更多的可能性

- 自定义编解码器
- 自定义传输方式
- 自定义数据加密
- 自定义音视频前后处理
- 自定义QOS

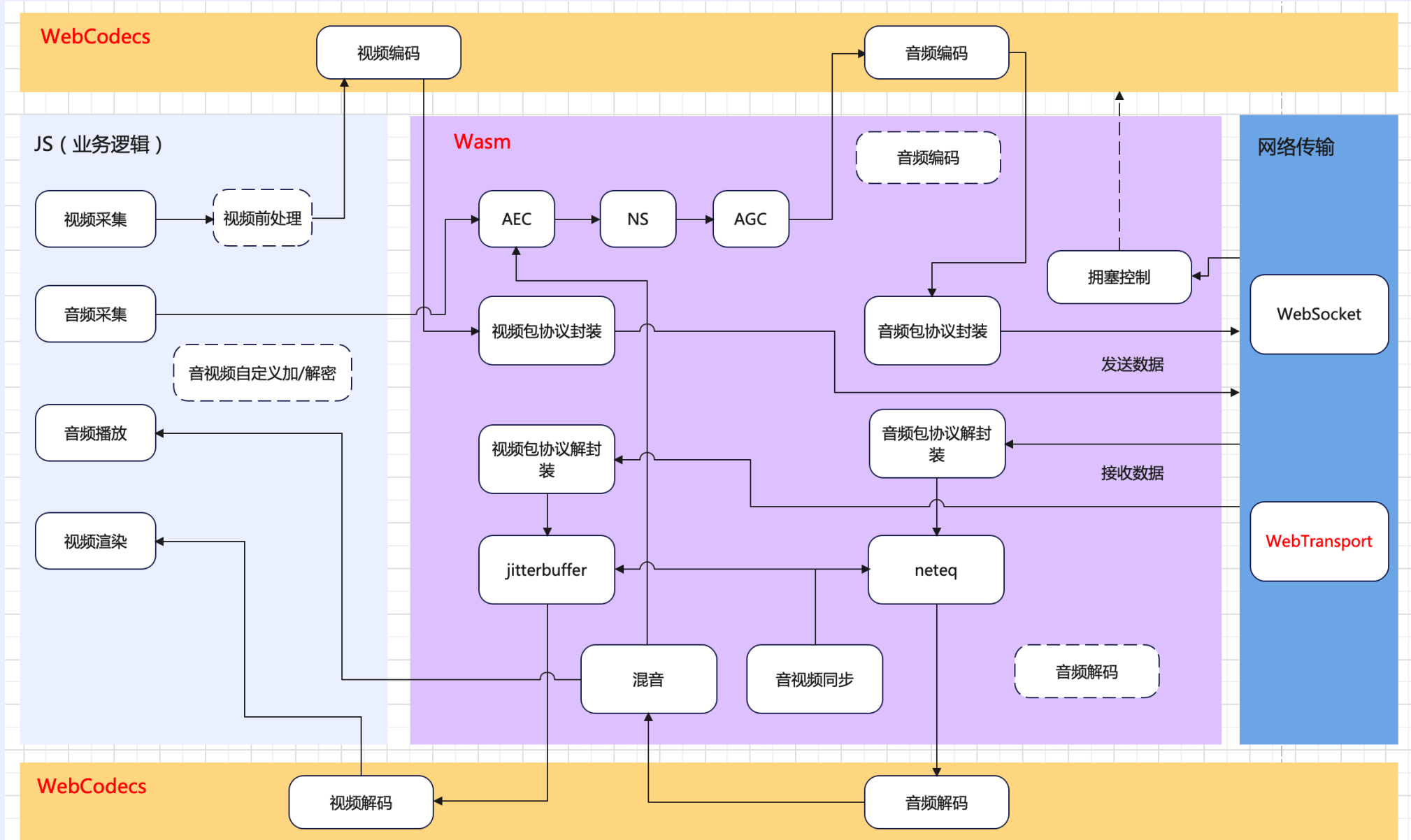
架构图



WebAssembly SDK



Web SDK架构图



3

方案落地

SDK性能

- 6个用户加入同一个房间，Wasm CPU使用率更低

| | | |
|--|--------|------|
| Tab: 带宽/视频码率设置-TRTC 腾讯实时音视频 Dedicated Worker: | 180 MB | 44.4 |
| Tab: 带宽/视频码率设置-TRTC 腾讯实时音视频 Dedicated Worker: | 128 MB | 44.2 |
| Tab: 带宽/视频码率设置-TRTC 腾讯实时音视频 Dedicated Worker: | 138 MB | 44.9 |
| Tab: 带宽/视频码率设置-TRTC 腾讯实时音视频 Dedicated Worker: | 239 MB | 44.7 |
| Tab: 基础音视频通话-TRTC 腾讯实时音视频 | 146 MB | 48.1 |
| Tab: 基础音视频通话-TRTC 腾讯实时音视频 | 146 MB | 48.5 |

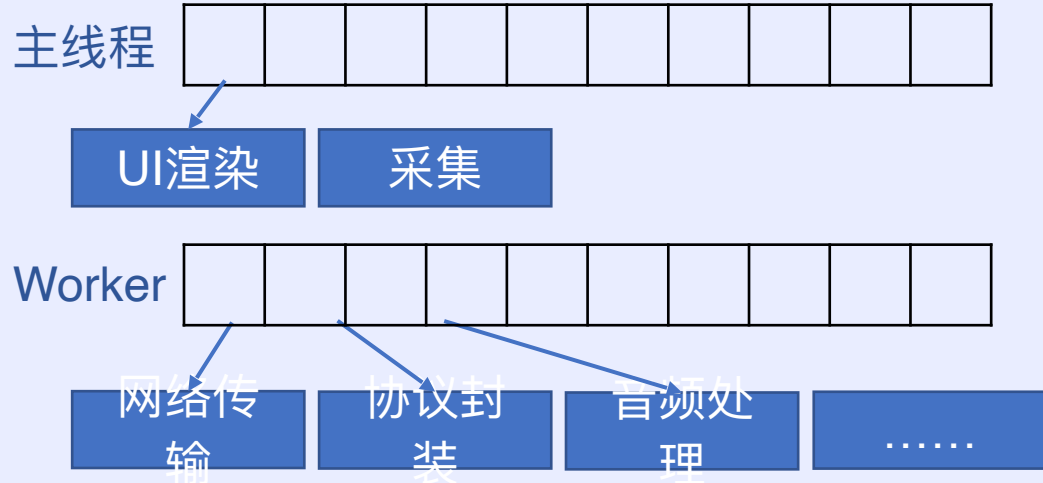
Annotations: Wasm (red arrow), WebRTC (yellow arrow), MEM (yellow arrow), CPU (red arrow)

- 两人进房，编码码率为1Mbps，帧率为30帧，RTT 10ms 截图时差在为延时100ms内

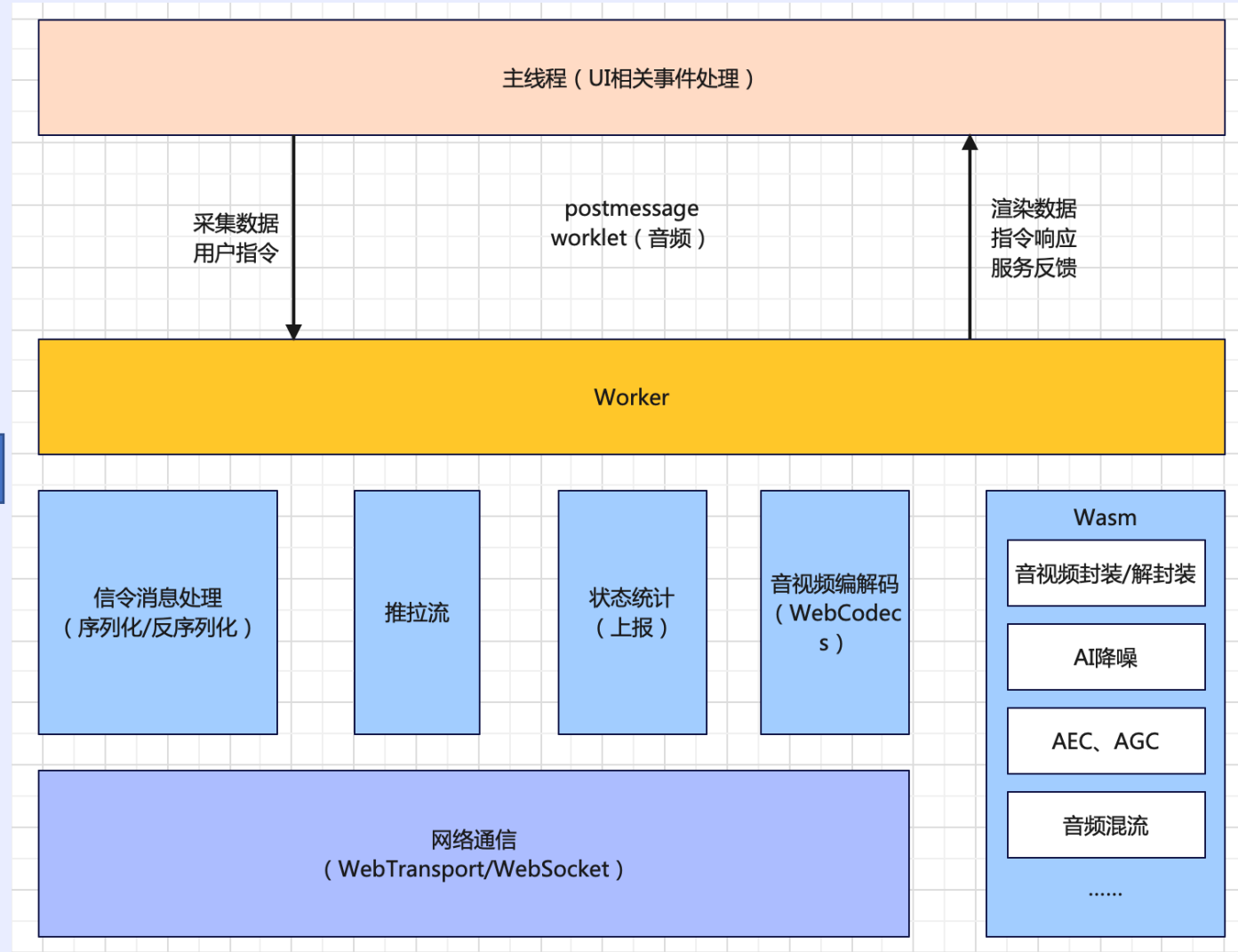


SDK进化

使用Worker——避免单核跑高、精准定时器、拒绝UI阻塞底层逻辑



使用Worklet——提高音频数据传递效率

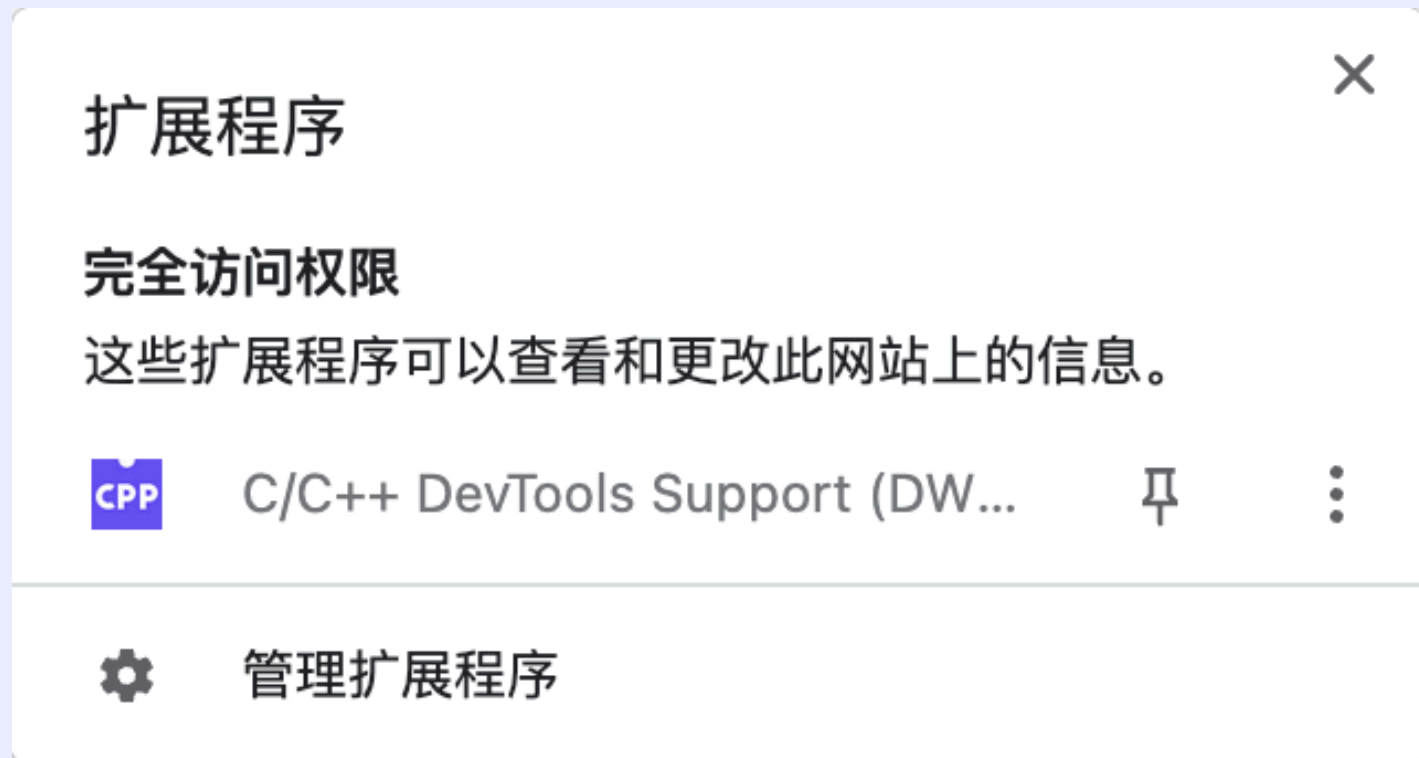


服务器优化

- 服务端采用BBR算法，更激进的拥塞控制
- 根据丢包、Jitter情况，适当调整弱网策略
- 根据网络情况自适应FEC

调试

WebAssembly开发遇到问题怎么办?



C++调试

- 自动加载.wasm
- 打开资源栏
- 设置断点

注意编译wasm时需要设置为Debug (-g)

The screenshot shows a debugger interface with the following components:

- Code Editor:** Displays C++ code for `opus_encoder.c`. A breakpoint is set at line 189, which is highlighted in blue. The code includes a static array `fec_thresholds` and the `opus_encoder_get_size` function.
- Debugger UI:**
 - Paused on breakpoint:** A yellow banner at the top right indicates the current state.
 - Threads:** Shows the `main` thread and a sub-thread `audioEncoder` which is paused.
 - Watch:** An empty section for monitoring variables.
 - Breakpoints:** Lists two active breakpoints:
 - `celt_encoder.c:868` with `trim_index = (int)floor(.5f+trim);`
 - `opus_encoder.c:189` with `if((Fs!=48000&&Fs!=24000&&Fs!=16000&&Fs!=12000&&Fs!=8000&&application != OPUS_APPLICATION_VOIP && application != OPUS_APPLICATION_RESTRICTED_LOWDELAY)) return OPUS_BAD_ARG;`
 - Scope:**
 - Parameter:** Shows `Fs: 48000`, `application: 2048`, and `channels: 1`.
 - Local:** Lists `celt_enc`, `err`, `ret`, `silkEncSizeBytes`, and `silk_enc`, all as `<optimized out>`.
 - Global:** Lists various threshold arrays like `fec_thresholds`, `mono_music_bandwidth_thresholds`, etc.
 - Call Stack:** Shows the call sequence: `opus_encoder_init` (opus_encoder.c:189) → `opus_encoder_create` (opus_encoder.c:535) → `::InitOpusEncoder` (av_process_module.cc:951) → `(anonymous)` (av_processing.js:1676) → `onAudioFrame` (audio-encoder.ts:36) → `(anonymous)` (audio-encoder.ts:18) → `next` (common.js:166) → `(anonymous)` (producer.js:129) → `fulfilled` (producer.js:4) → `Promise.then (async)` → `step` (producer.js:6) → `(anonymous)` (producer.js:7) → `__awaiter` (producer.js:3).

4

问题及展望

WebAssembly引擎能解决什么问题

高度自定义

- 自定义音视频编码方式
- 自定义加解密
- 国密支持
- 自定义3A (AI降噪等)

QOS调优

- 自定义或可复用现有系统的QOS策略

更简单的服务器逻辑

- 可复用后台服务逻辑

更快更安全的网络传输

- WebTransport
- 更好的防火墙穿透能力

WebAssembly引擎会带来什么问题

引入了新的模块

- WebAssembly
- WebCodecs
- WebTransport

更高的复杂性

- 需要更多的技术积累
- 增加开发难度

兼容性问题

- WebCodecs
、
WebTransport不能在Safari浏览器中运行

上行拥塞控制

- WebTransport上行拥塞控制算法暂不支持调整（即将支持）

Wasm引擎踩坑

- 底层逻辑被UI阻塞的问题
- WebCodecs OPUS编码只支持60ms编码（实时性和兼容性？）
- 共享网页标签时出现不采集问题（屏幕共享异常？）
- 有时候回声不能消除的问题（听到回声？）
- H264大小码流编码问题（不能编出不同分辨率？）

展望

- 更开放的Web技术
 - WebTransport更加完善、将提供更灵活的拥塞控制算法。
 - WebGPU开放硬件能力
 - WebAssembly的SIMD的更好支持
 -
- 更复杂的应用场景（高度自定义）
 - 云游戏
 - 自定义加解密
 - 远程桌面
 - 空间音频
 - 音视频前后处理（如AI降噪、美颜、变声等）

```
dictionary WebTransportOptions {  
    boolean allowPooling = false;  
    boolean requireUnreliable = false;  
    sequence<WebTransportHash> serverCertificateHashes;  
    WebTransportCongestionControl congestionControl = "default";  
};  
  
enum WebTransportCongestionControl {  
    "default",  
    "throughput",  
    "low-latency",  
};
```

感谢