Kurento 分享

- 2019.05.20

受众类型

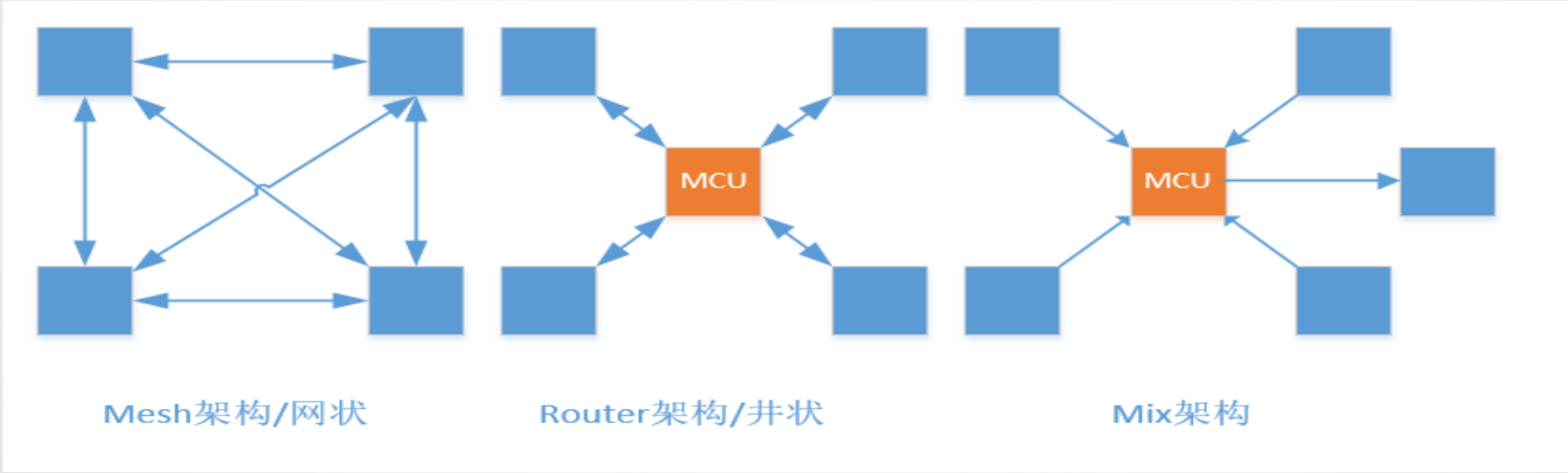
- 有意愿了解 kurento 的人
- 使用者
- 开发者

分享大纲

- kurento是什么
  - 视频群聊架构的演进
  - kurento的底层核心：webrtc
  - kurento基于webrtc的扩展
- 为什么引进kurento
  - 传统方式的3大痛点
  - kurento的优势
- kurento介绍
  - 架构介绍
  - 术语介绍
  - 核心流程介绍
- kurento的使用
  - 分层
  - web端使用示例
  - 服务端使用示例
  - kurento服务端完整流程说明
  - 服务端基本架构推荐
- 深入kurento
  - kurento 设计风格
  - kurento 分层
  - kurento 框架设计依据
  - kurento 框架分层
  - kurento 框架类图分析：接入层、业务层、数据层
  - kurento 协议的报文处理
  - WebRtcEndpoint 分层
  - WebRtcEndpoint 设计目标、 idl 、业务层
  - 事件订阅
- 教学业务场景分析
  - 业务场景如何实现
  - 业务、功能设计
  - kurento 内部细节

Kurento 是什么 - 视频群聊架构的演进

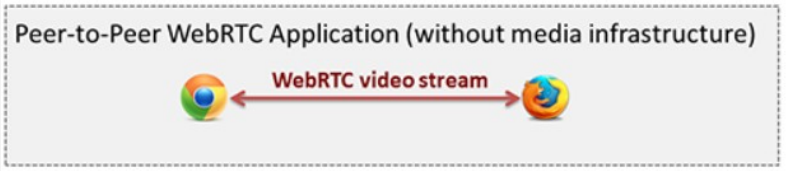
- 目前主流的群聊架构



- 基于 webrtc 视频群聊的开源库有 licode 和 kurento

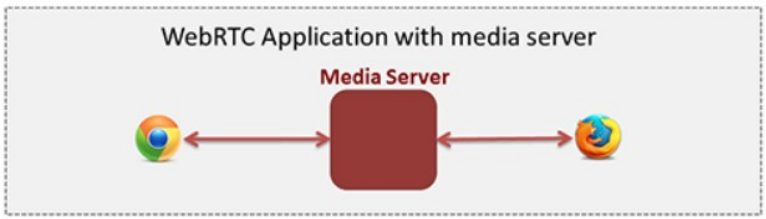
Kurento 是什么 - kurento 的底层核心： webrtc

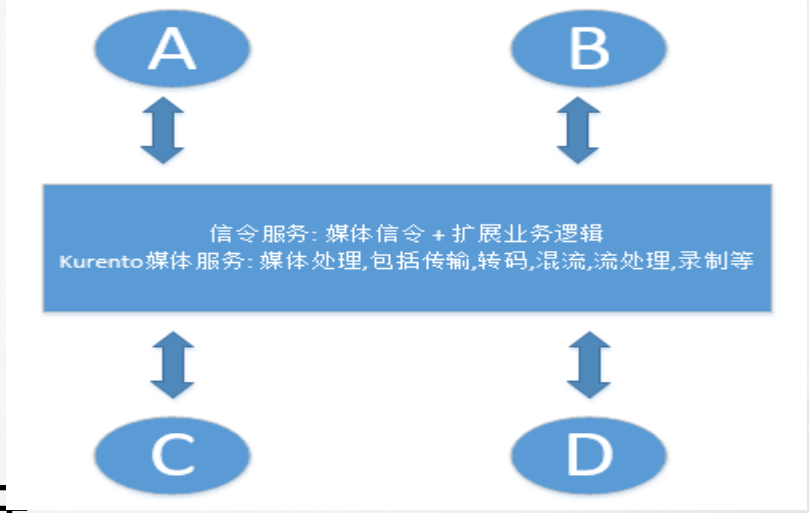
- webrtc 是什么
  - Web Real-Time Communication
  - 协议 / 开源库
  - 下一代视频通话标准
  - 支持网页浏览器进行实时的音视频对话
  - 质量保证：音频处理 / 网络自适应 / 流处理 / 时延保证
  - 从设计上讲 ,webrtc 的目的是将网络两端的数据进行交换



Kurento 是什么 - kurento 基于 webrtc 的扩展

- kurento： 支持 webrtc 的媒体服务器





- 扩展
  - webrtc 服务器
  - 增加了媒体处理可能性

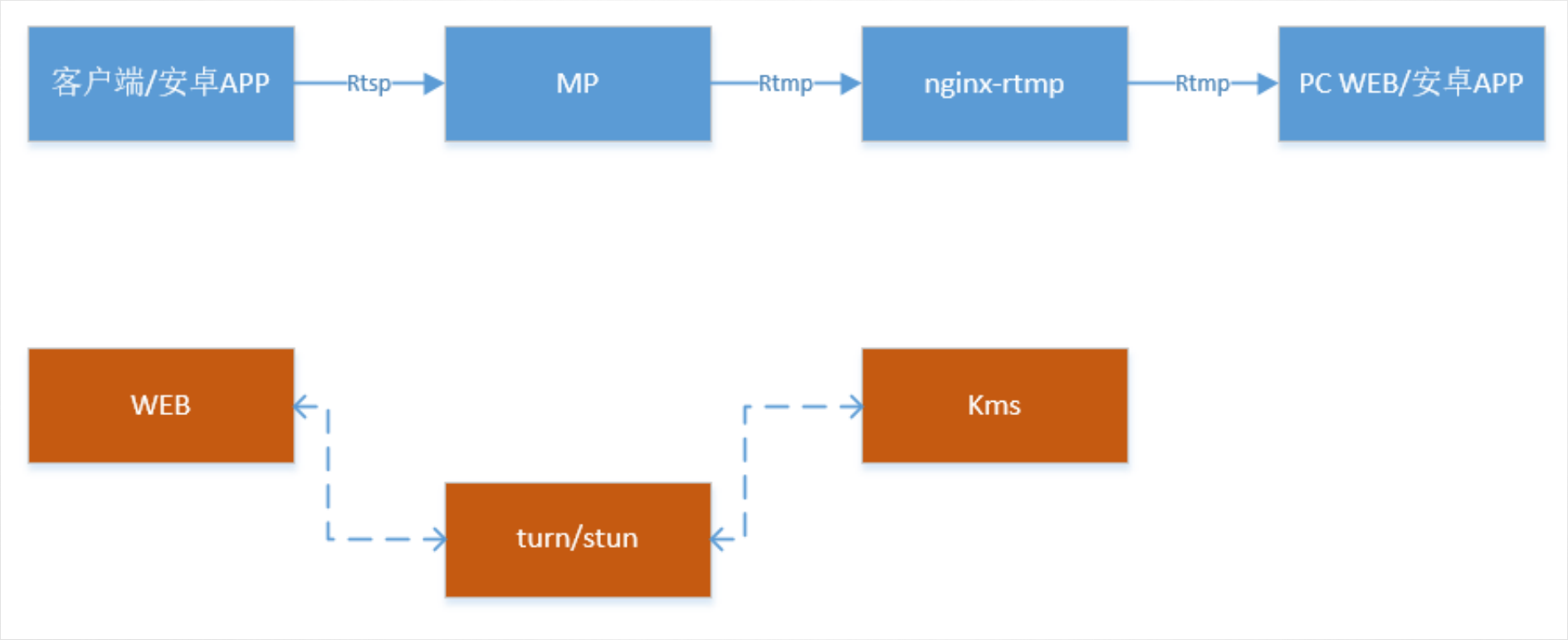
为什么引进 kurento - 传统方式的 3 大痛点

- 基于 rtmp/rtsp 互动 mcu 的痛点：
  - 延时太大无法有效解决；观看端也未做流控
  - 合成画面占用太多资源，一台服务器无法承载预期业务
  - 媒体传输没有流控 (qos) ； pc 端还需要额外的推流工具

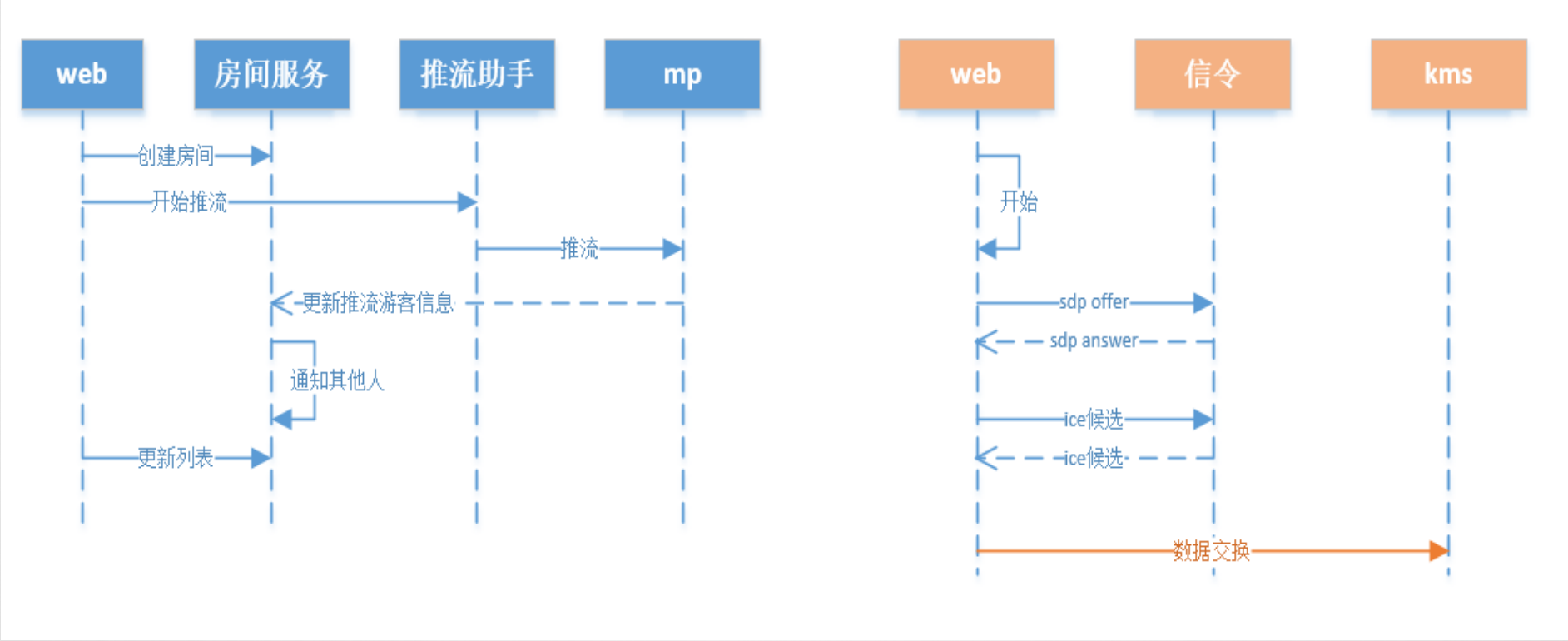
为什么引进 kurento - kurento 的优势

- 低延时：一般网络，没有特殊的媒体处理操作，延时会在 300ms 以内
- 自带流控，这时 webrtc 自带的
- 为媒体处理提供了一些基础功能 (转发、录制)

为什么引进 kurento – kurento 的优势 - 媒体数据流差异

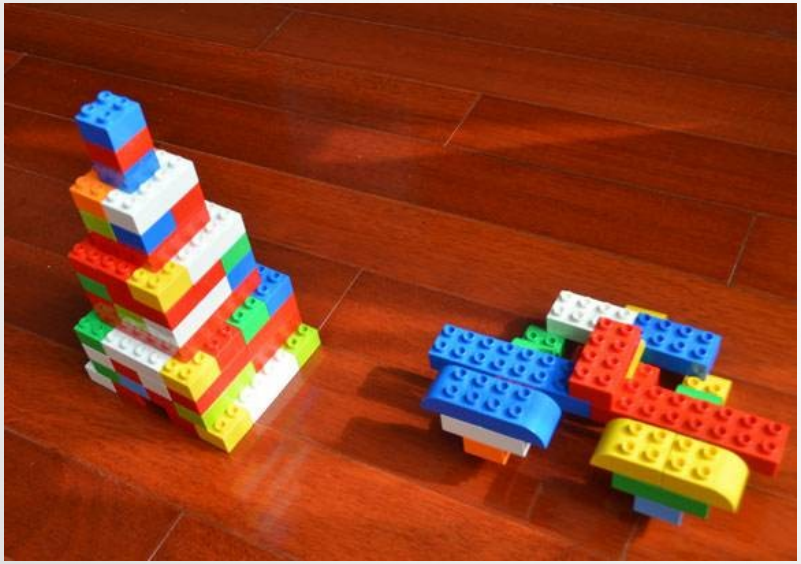


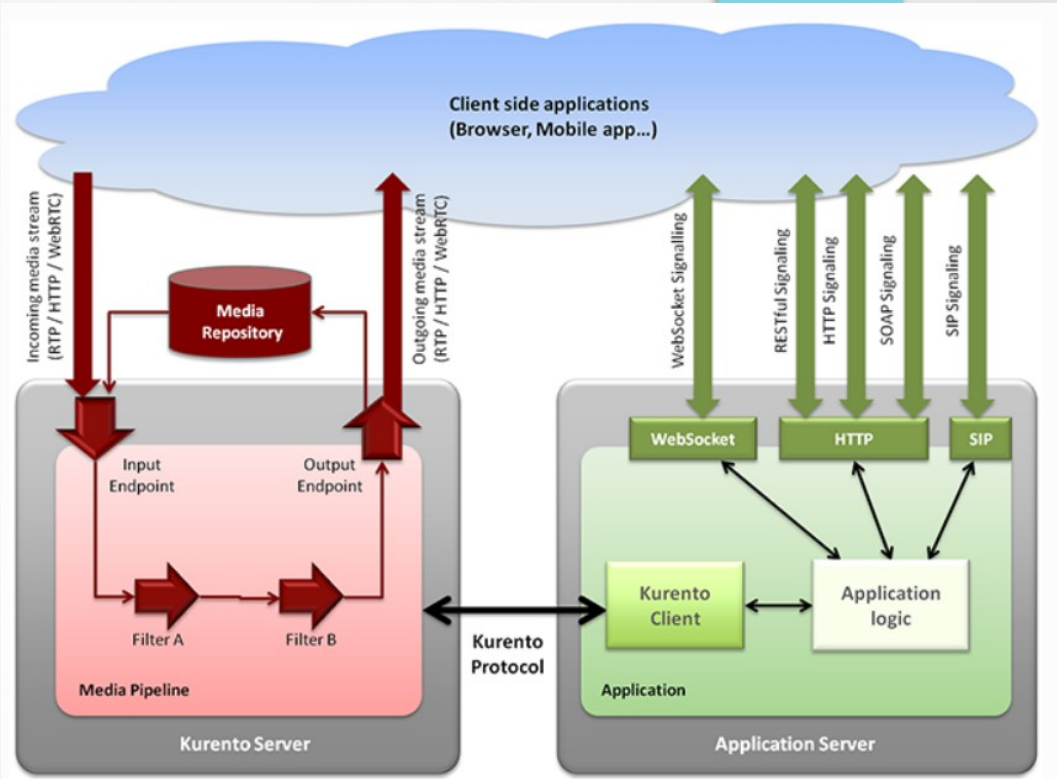
为什么引进 kurento – kurento 的优势 - 进房间



Kurento 介绍 - 架构介绍

- 插件式 (类似 dshow)



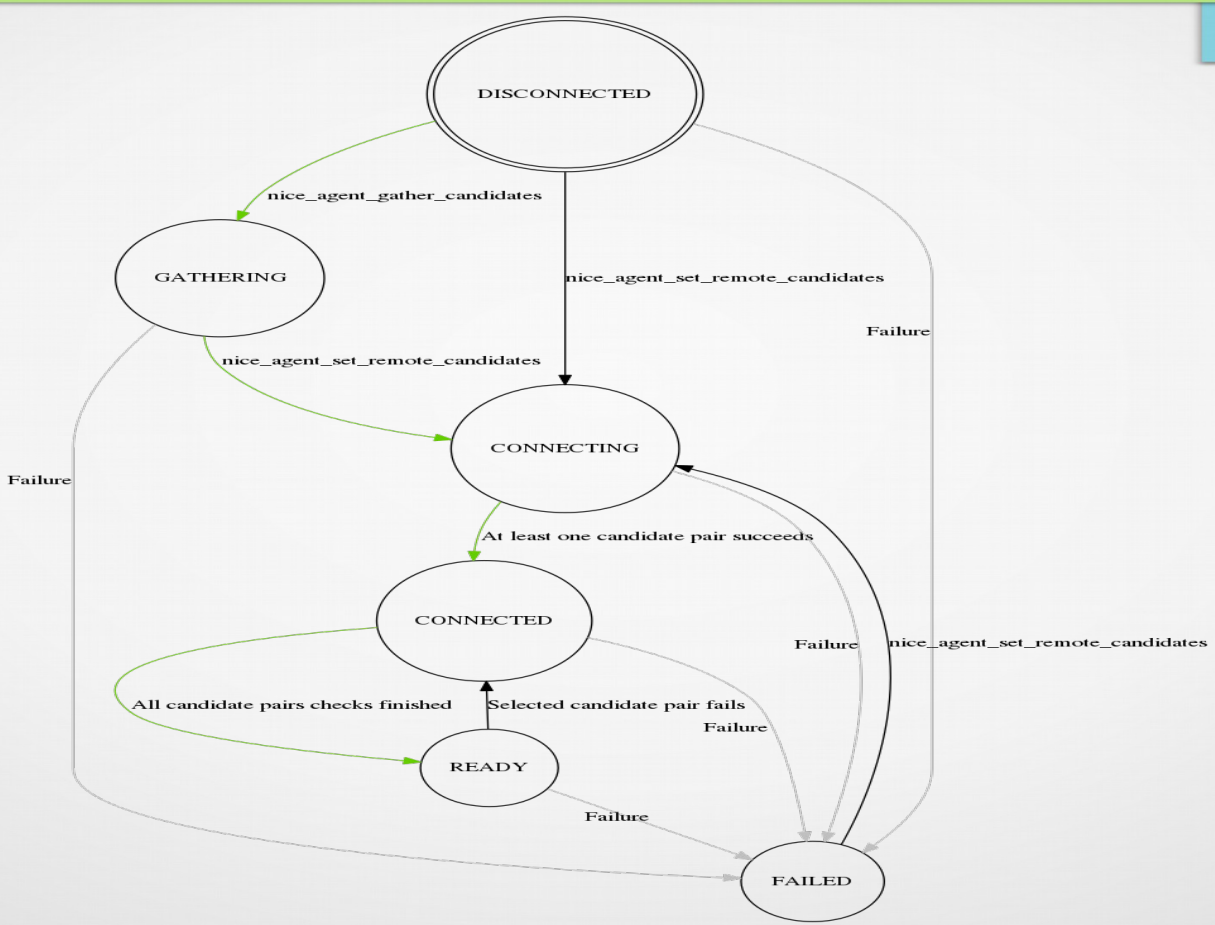


Kurento Architecture. Kurento architecture follows the traditional separation between signaling and media planes.

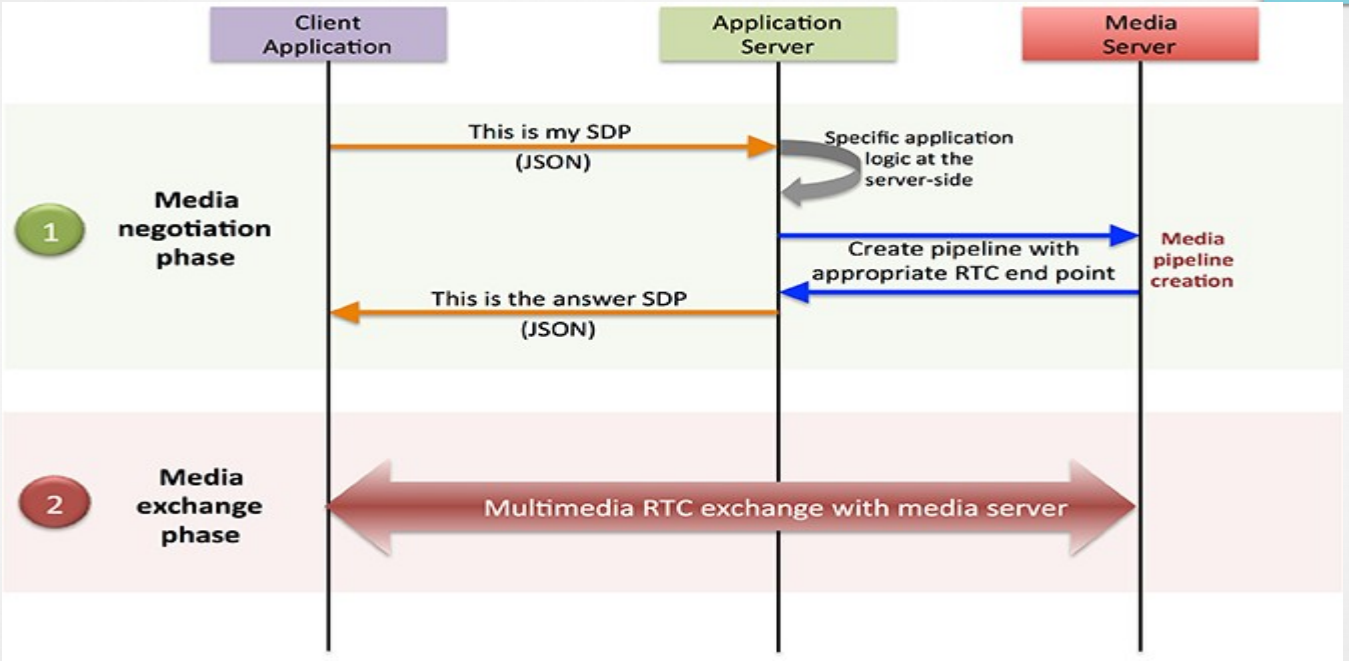
Kurento 介绍 - 术语介绍

- Pipeline 管道
- MediaElement 媒体元素
- Sdp offer/answer 模型
- ice 候选

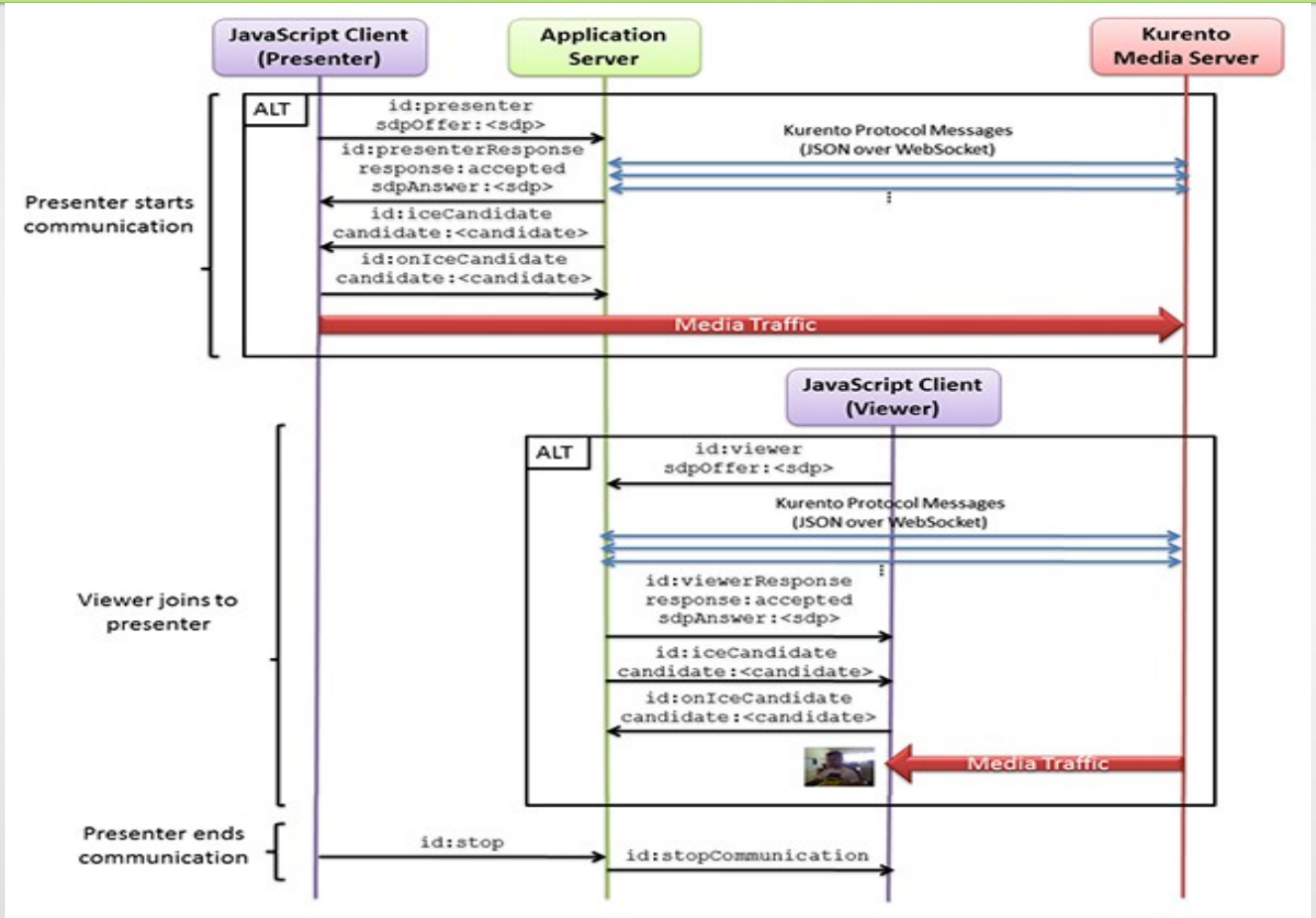
Kurento 介绍 - ice 协商过程



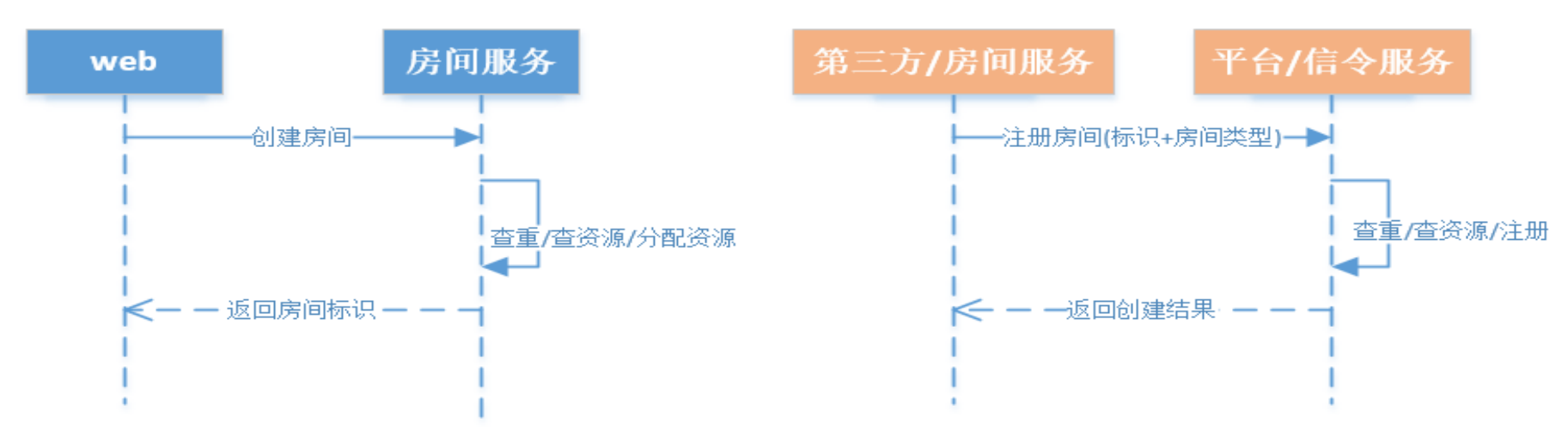
Kurento 介绍 - 核心流程介绍



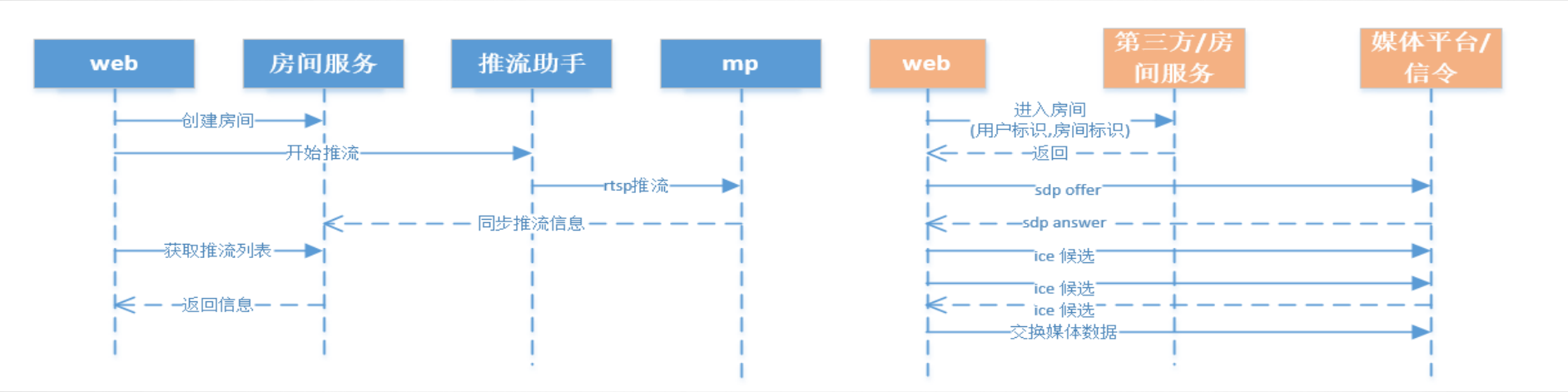
Kurento 介绍 – 核心流程介绍 - 老师学生场景



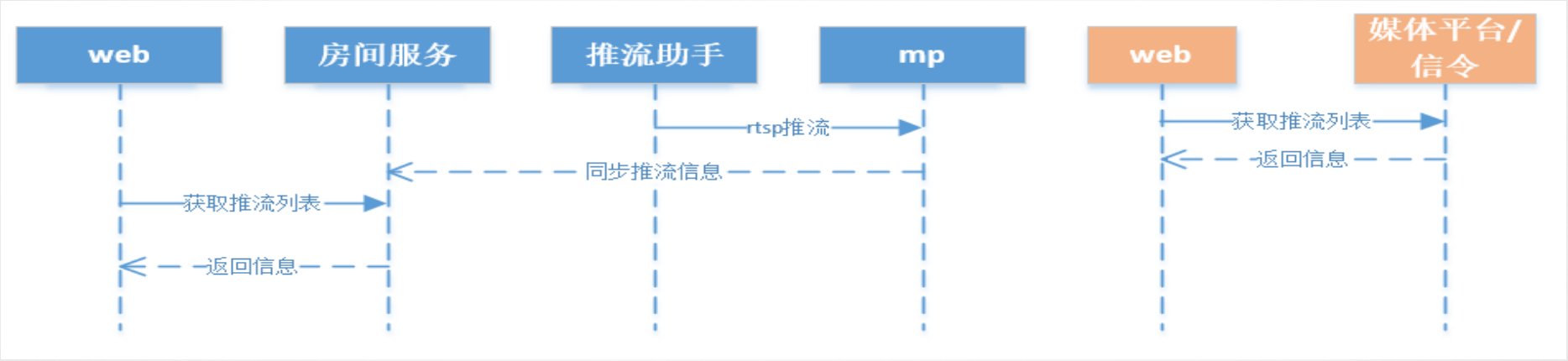
Kurento 介绍 – 核心流程介绍 - 创建房间



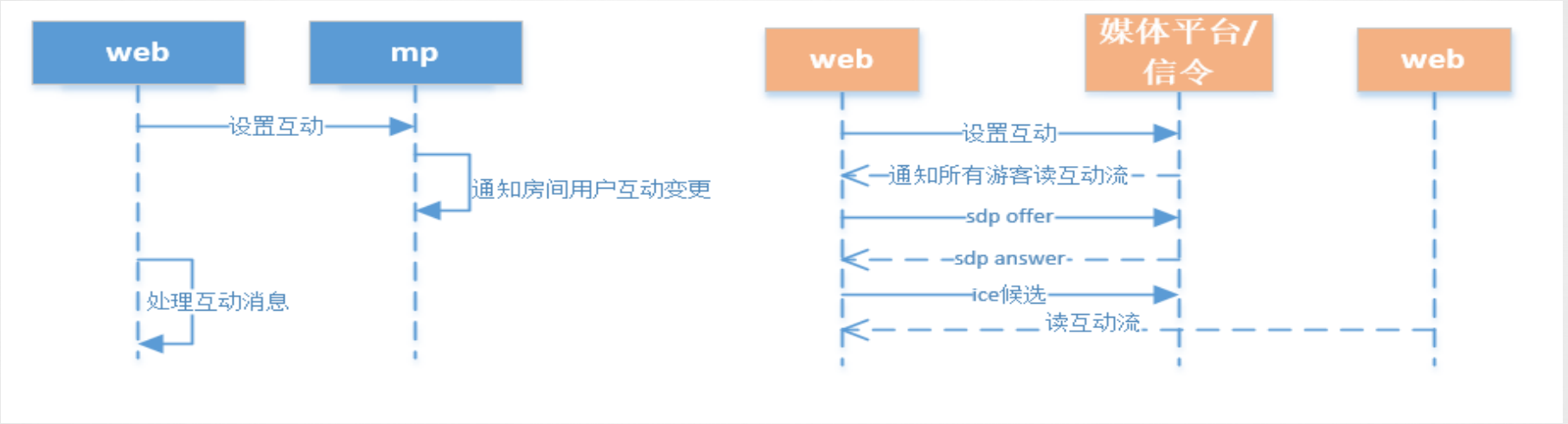
Kurento 介绍 – 核心流程介绍 - 进入房间



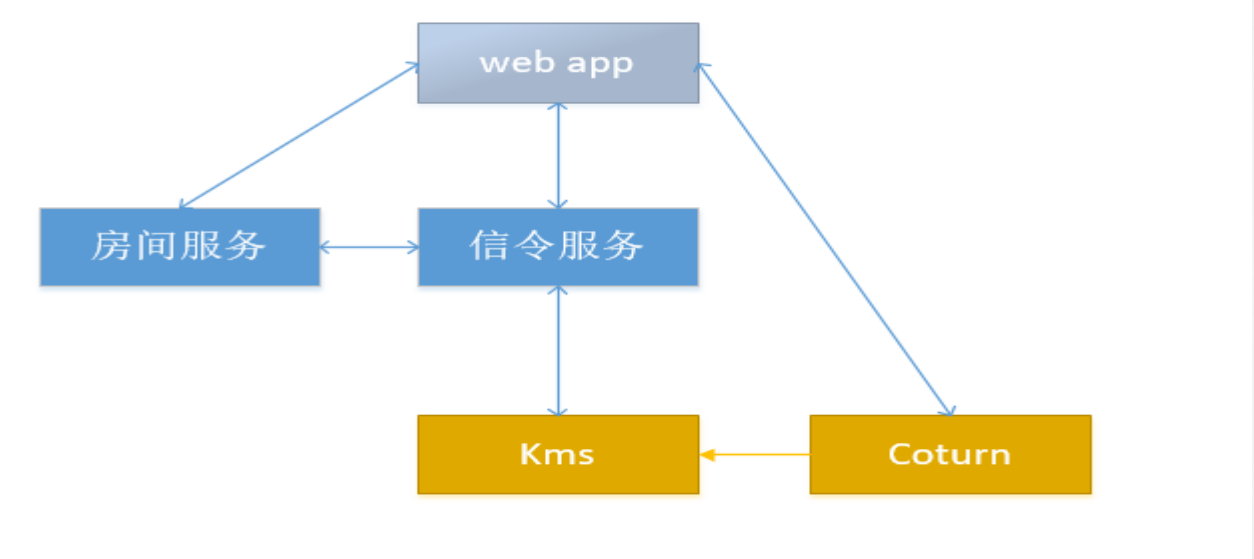
Kurento 介绍 – 核心流程介绍 - 获取当前推流列表



Kurento 介绍 – 核心流程介绍 - 设置互动上麦下麦



Kurento 的使用 - 分层



Kurento 使用 - web 使用示例 - kurento 信令传输协议

```
var ws = new WebSocket('wss://' + location.host + '/helloworld');
function sendMessage(message)
{
  var jsonMessage = JSON.stringify(message);
  ws.send(jsonMessage);
}


{
  var message = {
    id : 'start',
    sdpOffer : offerSdp
  }
  sendMessage(message);
}
```

json + websocket

Kurento 使用 - web 使用示例 - Web app 引用 js

```
<script src="/webjars/webrtc-adapter/release/adapter.js"></script>

<script src="/js/kurento-utils.js"></script>
```

```
function uiStart()
{
  const options = {
    localVideo: uiLocalVideo,
    remoteVideo: uiRemoteVideo,
    mediaConstraints: { audio: true, video: true },
    onicecandidate: (candidate) => sendMessage({
```

```
ws.onmessage = function(message)
{
  const jsonMessage =
   JSON.parse(message.data);


  switch (jsonMessage.id) {
    case 'PROCESS_SDP_ANSWER':
```

Kurento 使用 - 服务端使用示例

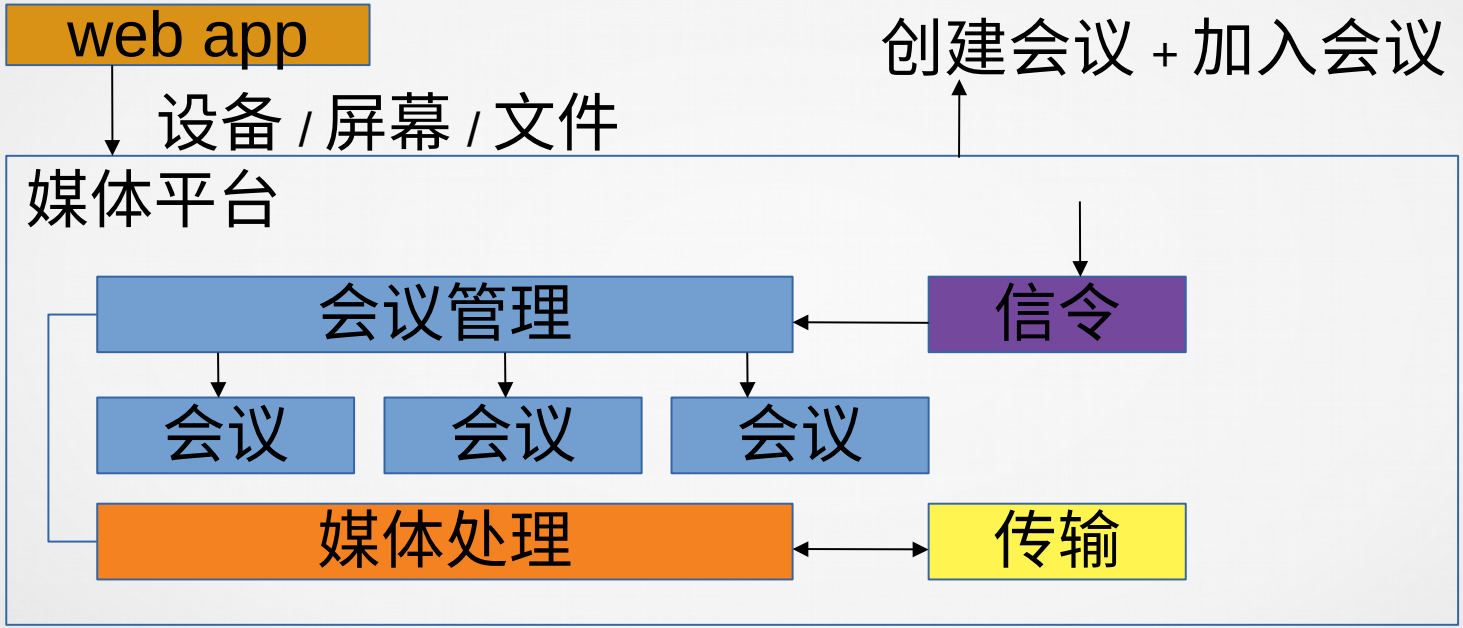
Kurento 使用 - 服务端完整示例说明 1

- 01. 业务服务和 kms 之间创建 websocket 连接 , 业务服务发送 ping 给 kms,kms 回一个 pong
- 02. 浏览器上请求页面数据
- 03. 页面上点击开始按钮 ,web app 发送 start 消息给业务服务 , 参数是 sdp offer
- 04. 业务服务收到 sdp offer 之后 , 通知 kms 创建管道
- 05.kms 返回创建管道的结果信息
- 06. 业务服务通知 kms 创建媒体元素
- 07.kms 返回创建媒体元素的结果信息
- 08. 业务服务通知 kms 把媒体元素进行连接
- 09.kms 返回连接的结果信息
- 10. 业务服务通知 kms, 要订阅媒体元素的错误事件
- 11.kms 返回订阅的结果信息
- 12. 业务服务通知 kms, 要订阅媒体元素的 MeidaFlowInStateChange 事件 ,MeidaFlowOutStateChange 事件 ,ConnectionStateChanged 事件 ,MediaStateChanged 事件 ,MediaTranscodingStateChange 事

Kurento 使用 - 服务端完整示例说明 2

- 13. 业务服务通知 kms 去执行 sdp offer 处理 , kms 返回 sdp answer
- 14. 业务服务发送 startResponse 消息给 web app, 参数是 sdp answer
- 15. 业务服务通知 kms 开始收集 ice 候选
- 16.kms 发现一个本地 ice 候选 , 之前业务服务有订阅这个事件 ,kms 通过 onevent 消息丢给业务服务 , 参数包含了 ice 候选信息
- 17. 业务服务通知 kms, 开启 kms 调试 . kms 按格式把 graph pipeline 按指定格式发给业务服务
- 18. 业务服务发送 iceCandidate 消息给 web app, 参数是 kms 的 ice 候选 , web app 会将 ice 候选加入到匹配队列 ,
- 19.web app 发送 onIceCandidate 消息给业务服务 , 参数是 web app 的 ice 候选 ; 业务服务通知 kms, 有新的 ice 候选 , 参数是 web app 的候选
- 20.kms 会告诉业务服务 , 第一个远端 ice 候选达到 ; 本地 ice 候选和远端 ice 候选组成一个匹配对 ; 第一对匹配对测试通过 ; 所有的匹配对测试完成
- 21. 19-20 这个过程会重复几次 , 知道所有的 ice 候选全部测试完成 . 选出一对权重最高且测试通过的 ice 匹配对 , 执行 dtls 连接认证 , 之后就是传输媒体数据

Kurento 使用 - 服务端基本架构推荐



深入分析 Kurento6.9.0

- 媒体传输
- 媒体处理

深入 kurento - kurento 设计风格

- 扩展性
  - 大量的工厂模式
  - 大量的分层设计
- 低耦合
  - 大量使用 glib 信号
  - 大量使用 signal

插件式的服务在实现上一般分两层：

- 主体
  - 对外提供统一接口来调用插件实现的功能
  - 定下插件的约束规则
- 众多插件
  - 插件只负责功能的实现

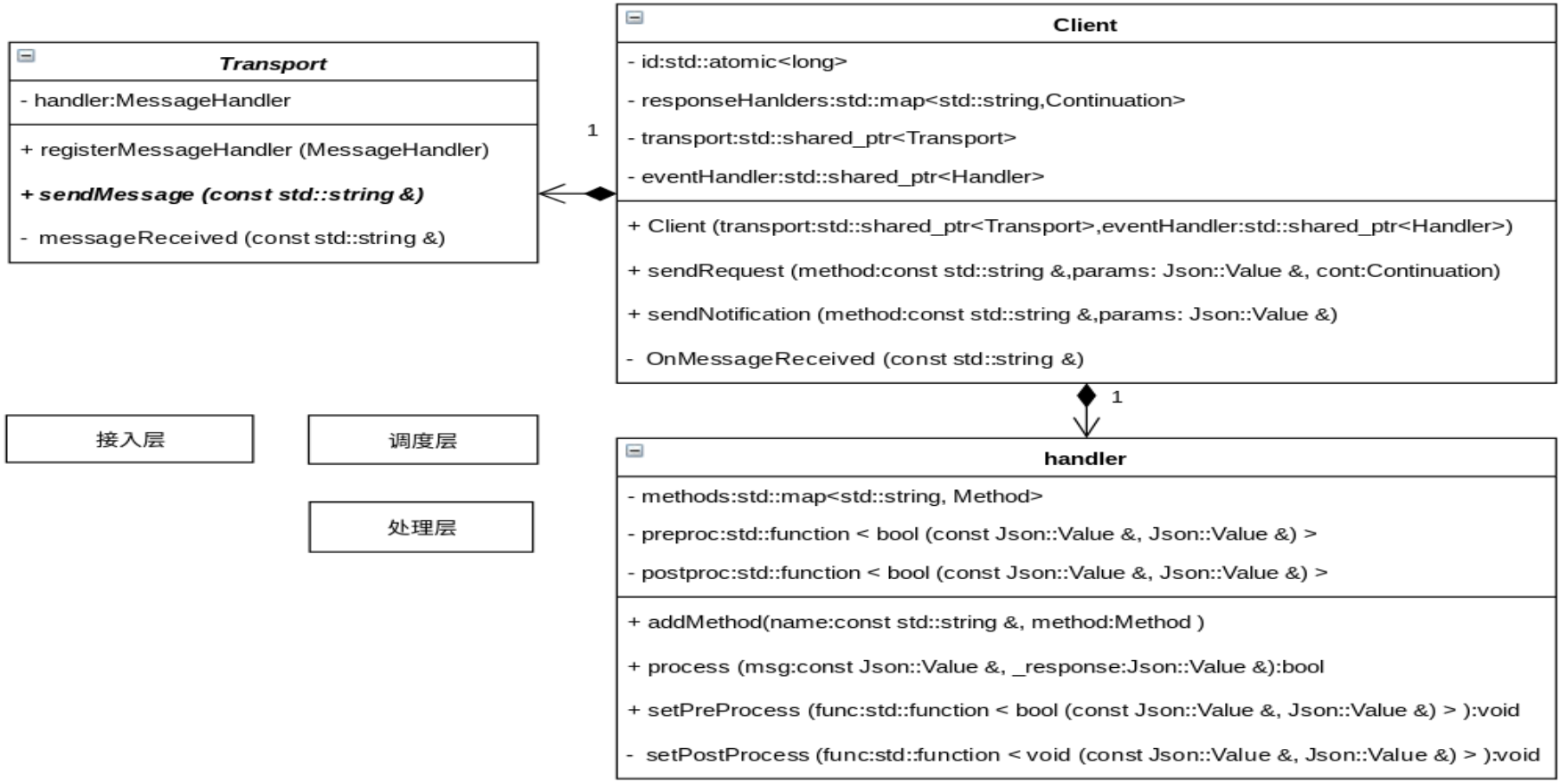
深入 kurento - kurento 框架设计依据

- 对外支持统一的信令格式
  - kurento 协议格式
    - 基于 Jsonrpc2.0，支持 10 个左右的信令
    - 传输内置支持 websocket，支持其他的需要扩展
  - 协议格式要满足各个插件
- 管理各个插件的资源，但不管理各个插件使用场景的业务逻辑

深入 kurento - kurento 框架分层

- 和传统设计类似， kurento 也分 3 层
  - 接入层 - 基于 jsonrpc2.0 的 kurento 协议
  - 业务层 - 基于 kurento 协议的内层来控制插件执行操作
  - 数据层 - 单独放在内存的一个固定区域的

深入 kurento - kurento 框架接入层类图



深入 kurento - kurento 框架业务层类图

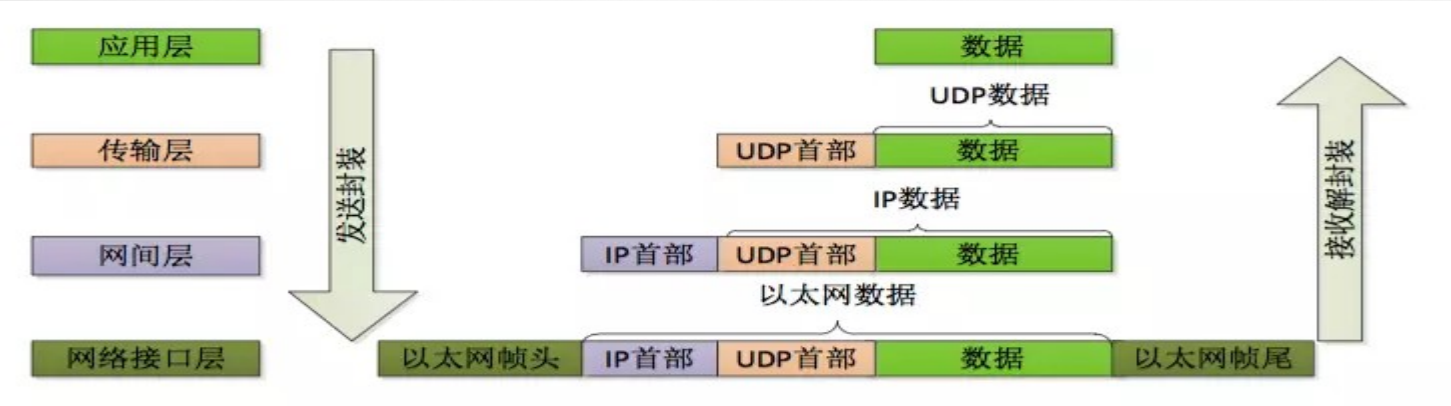
- 上面的接入层体现的更多的是协议上的支持
- kurento 中使用大量的工厂模式，好处是增加了扩展性
- WebSocketTransport 是 kurento 内置的传输方式
- ServerMethods 除了构造函数和从基类继承的函数，其他的都是基于 jsonrpc 协议定义的 kurento 协议；构造中还包含了请求缓冲池；
- kurento 协议中，常用的有资源申请、调用、事件，其他用到的不多
- ModuleManager，保存了所有资源创建的工厂类

深入 kurento – kurento 框架 - 插件的处理

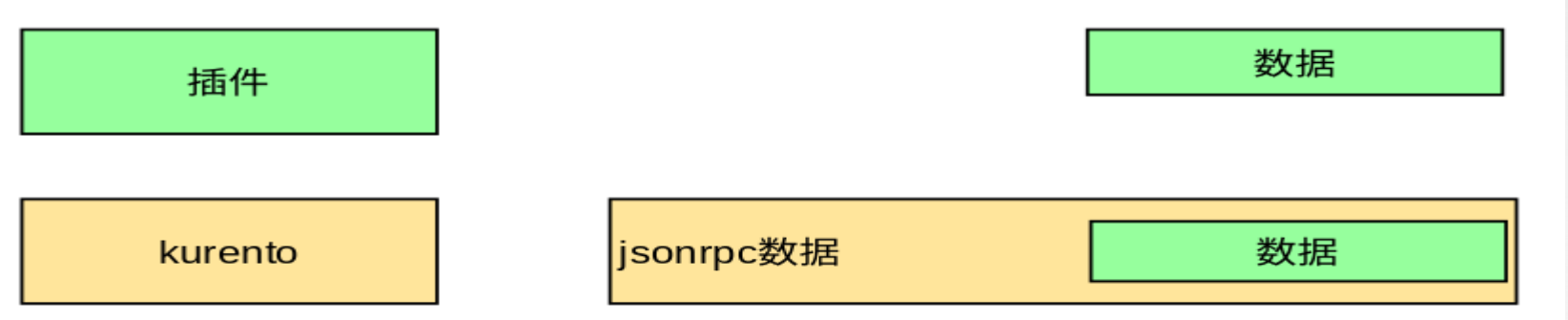
- 整个过程
  - module 加载
  - ws 收 create 信令
  - 信息的存储
  - ws 收 invoke 信令

深入 kurento – kurento 框架 - 信令 1

- 以太网传输 udp 包



深入 kurento – kurento 框架 - 信令 2



深入 kurento – WebRtcEndpoint 插件的分层

- c++ 功能组合层
- c 功能模块层
- gst

深入 kurento – WebRtcEndpoint 深入分析

- 设计目标
- Idl
- 类图

订阅事件

- 订阅：
  - websocket 接收到订阅信令
  - 设定好槽
- 触发：
  - c 功能模块层　glib 信号　c++ 功能组合层
  - 信号和槽 通知主线程
  - 主线程利用 boost 异步库调用 websocket 发送

教学业务场景分析

- 业务场景如何实现 - 准备怎么做
- 业务和功能的设计 - 具体做的细节
- Kurento 内部细节 - 针对每一步， kurento 内部是如何处理的

教学活动业务场景分析 - 业务场景如何实现

和一般的服务类似可分 3 层：

- 应用层 - 客户端，可使用 chrome 浏览器，主要负责媒体渲染和用户操作，核心操作分为进房间、开始活动、结束活动
- 应用服务层 - 业务服务，维护用户会话、资源和信令的路由
- 基础服务层 - 主要是 kms ，提供音视频媒体传输能力的支持

教学活动业务场景分析 - 业务和功能的设计

- 针对老师和学生，业务：
  - 开始教学活动，操作上对应 点击进房间、开始活动按钮
  - 结束教学活动，操作上对应 点击结束活动按钮

- 服务端需提供的功能：
  - 用户会话的维护，区分角色，并判断后续信令的有效性
  - 资源的维护，根据信令，维护 kms 上资源的申请和释放
    - 管道资源的创建
    - WebRtcEndpoint 资源的创建
    - 资源释放
  - 信令路由，接收客户端信令，将之转换成一系列 kms 信令，并维护资源状态
    - WebRtcEndpoint 事件监听
    - sdp 协商 ice 候选协商

教学活动业务场景分析 - kurento 内部细节

- 管道资源的创建
- WebRtcEndpoint 资源的创建
- 资源释放
- WebRtcEndpoint 事件监听
- sdp 协商 ice 候选协商

结尾

我们介绍了以下内容：

- kurento 是什么，适合做什么
- 如何使用 kurento
- kurento 内部细节
- 一个简单完备的例子来讲解从 kurento 应用框架到 kurento 内部实现