在 Chrome、Firefox 等高版本浏览器中

实现低延迟播放海康、大华 RTSP

一、背景

现在到处是摄像头的时代，随着网络带宽的不断提速和智能手机的普及催生出火热的网络直播行业，新冠病毒的大流行又使网络视频会议系统成为商务会议的必然选择，因此 RTSP 实时视频流播放及处理不再局限于安防行业。在如道路、工厂、楼宇、学校、港口、农场、景区等诸多场景实施的信息化系统中，绝大多数都采用的是 B/S 架构，隐藏迫切需要在浏览器中嵌入多路摄像头 RTSP 流低延迟(小于 500 毫秒)播放功能，而在 IE 及 Chrome 45 以下版本等浏览器中，采用 ActiveX 控件或 NPAPI 插件即可实现。然而美好总是短暂的，从 2015 年开始 Chrome 及 Firefox 等浏览器纷纷取消了 NPAPI 插件的支持，而 IE 又在与 Chrome 及 Firefox 等浏览器竞争的过程中不断被用户抛弃，到 2020 年其市场份额已降到可怜的个位数。微软在几经折腾后，索性也拥抱 Chromium 内核推出了新版 Edge 来杀死自己的 IE 和老版 Edge，以挽救自己在浏览器领域岌岌可危的江湖地位。

在 Chrome、Edge、Firefox 等当前主流的浏览器中，即使是 HTML5 标准的 Video 也并未对 RTSP 流播放提供原生支持，从而导致如何在当前主流的浏览器中实现低延迟、低成本播放多路 RTSP 成为了一个重大技术难题。这几年国内外的技术专家经过不断研究总结，形成一些闭源或开源、收费或免费的方案，但多数时候无法完全满足客户的实际需求，要么兼容性和稳定性不好，要么播放延迟高，首屏画面显示慢；尤其是播放高分辨率的 RTSP 流时，卡顿、花屏现象难以根除。而且摄像头比较多时还需要专用服务器高负荷的运转来支持转码转流，带宽的高频占用对系统其它业务的影响不可忽视，系统综合运行成本高，体验差。

二、现有方案

在浏览器中实现播放 RTSP 实时视频流，大体上有如下几个方案：

1、 浏览器插件方案

此方案主要适用于在 IE 及 Chrome 45 以下版本的浏览器，在 2015 年前是绝对主流的选择。使用 ActiveX 播放控件或 NPAPI 播放插件实际调用的是本地原生程序进行直接播放，从而可充分利用本机硬件解码和硬件加速渲染播放，可实现低延迟、低成本多路稳定播放的良好效果。一般使用 VLC 这个最流行的开源跨平台多媒体播放器，IE 及 Chrome、Firefox 低版本浏览器分别有对应的播放插件实现，VLC 对移动端支持也非常好。此方案非常灵活，可以方便的对接各品牌的视频流，也可以很容易实现截图和录像功能。缺点是需要额外安装 VLC 客户端软件，对个别明确要求不能用插件的场景不适用。摄

像头厂家一般也会提供适配的播放插件，比如海康威视提供的播放控件 Web 版，是和自己的 DSS 系统捆绑使用的，但不支持在 Firefox 高版本中运行。

2、 先转码再转流方案

此方案需要架设一个或多个视频流转码服务器，先在服务器上对 RTSP 流用 ffmpeg 进行转码串流成 RTMP，然后前端使用 VideoJS 再调用 Adobe Flash Player 进行播放，然而 2021 年开始基于 Chromium 内核的所有浏览器彻底取消了对 Flash Player 的支持，VideoJS 因此失效。不过幸好还有开源的替代播放方案 flv.js(https://github.com/bilibili/flv.js)工作原理是要求在服务端先把 RTSP 视频流转换为 flv 后用 Web Socket 或 WebRTC 推送到前端，前端收到后再转换为 Video 所支持的 MP4 后播放，这就导致 RTSP 视频流，需要经过 2 次转码才播放，画面延迟时间大幅增加，保守估计延迟至少是 2-3 秒级别了。况且如果有多路视频流时，服务器端转码和转流对 CPU、内存、网络带宽的压力大幅度增加，长期使用综合成本很高，对高分辨率的视频流播放经常出现花屏、卡顿现象。此方案要求浏览器支持流媒体扩展特性(MSE)，且无法利用本机硬件加速实现解码和渲染播放。优点是可兼容移动端网页播放。

此方案在国内有 TSINGSEE 的免插件 EasyPlayer RTSP 播放器，项目地址是 https://github.com/tsingsee/EasyPlayer 和 linkingvision 的免插件播放器 H5stream，项目地址是 https://github.com/linkingvision/h5stream 等，商业用途都是收费的。

3、 先转流再转码方案

此方案的典型代表是 Streamedian 公司的免插件播放器 Html5 RTSP Player，项目地址 https://github.com/Streamedian/html5_rtsp_player。此方案需要架设一个 Web Socket 的视频流转发服务器，前端连接到此服务器后，服务端不断把 RTSP 视频流通过 Web Socket 不断转发给前端的 JS 处理库，JS 处理库再把视频流转换为 Video 所支持的 MP4 后播放。

此方案不支持 IE 浏览器，最大的问题是画面延迟达数秒，首屏内容显示慢，也无法利用本机硬件加速实现解码和渲染播放，CPU 占用高，播放时花屏、卡顿现象，体验比较差。另外无法实现本地自动截图、录像等操作。此方案同样要求浏览器支持流媒体扩展特性(MSE)，对延迟不敏感的单源播放尚可，多路播放就只能洗洗睡了，另外根据一些用户的反馈，对各品牌摄像头的兼容性也不太友好，作为商业用途使用是不可行的。

4、 扩展程序方案

此方案典型代表是基于 Chrome 浏览器的 PPAPI 插件技术实现的开源播放器 VXG RTSP Player，项目地址是 https://github.com/VideoExpertsGroup/Chrome.RTSP.Player。此方案很显然不适用于 IE 和 Firefox 等浏览器，也不适用于低于 45 版的 Chrome 浏览器。VXG RTSP Player 是 Chrome 浏览器的扩展程序，对国内客户来说，由于谷歌服务器在墙外，想要大规模自主可控部署是不现实的。另外最关键的是谷歌已官方宣布，2021 年 6 月终止对 NaCl，PNaCl 和 PPAPI API 的支持，因而此方案也无继续探讨的必要。

5、 双内核方案

此方案典型实现是采用 Chrome 浏览器上的扩展程序 IETab 来实现，官方网站是 https://www.ietab.net，通过在 Chrome 标签页界面覆盖加载显示一个 IE 内核渲染的网页，此网页再调用比如 VLC 的 ActiveX 控件实现。此方案和方案 4 一样，存在大规模自主可控部署难问题。另外和上面的浏览器插件方案类似，需要在播放终端电脑中下载运行 IEHelpTab.exe 程序，对一些高安全要求无插件播放的场景来说不适用。最大的问题是在 Chrome 网页中对播放控件的控制很难实现，只有网页和播放控件都是在 IE 内核环境下才可以，而 IE 对当前一些新技术和前端主流框架的兼容已经不行了，况且 IE 对运行和下载安装 ActiveX 控件经常弹出警告，用户体验很差，维护升级也很麻烦。

6、 Wasm 方案

此方案采用的是 Chrome 等高版本浏览器所支持的一种方便把更复杂的原生应用直接搬进 Web 的标准技术，然而对浏览器的兼容存在很大问题，IE 肯定是不支持的，低版本的 Chrome 及 Firefox 等浏览器也不支持 Wasm，具体兼容性可看这里 https://caniuse.com/wasm。实现的基本思路就是把 RTSP 视频流通过 ffmpeg 的 Wasm 版软解码成 Video 所支持的 MP4 后播放，由于 Wasm 不支持硬件解码，对多路同时播放来说，终端电脑的 CPU 和内存占用会比较高，性能也堪忧。此方案有时应用在需要支持 H265 编码的场景，同样要求浏览器支持流媒体扩展特性(MSE)。由于存在诸多兼容性问题，此方案实际应用的案例较少。

三、改进方案

通过上述总结的现有技术方案可以看出，想要在浏览器中实现低延迟、低成本的多路 RTSP 同时稳定播放，只有不转码并充分利用终端电脑的硬件加速特性这两条才行，这样就只能采用类插件的外接方案。核心就在于如何在各浏览器中实现一个统一的不依赖浏览器自身扩展技术的外接系统，同时必须对各品牌及各版本的浏览器有比较好的兼容能力才具有较大的实用价值。所以改进方案思路就是要在浏览器网页中的指定位置和大小，实现一个内嵌到网页中显示的播放窗口，前端还必须可对这个内嵌播放窗口进行控制，而且播放窗口必须跟随浏览器窗口的移动和缩放、网页滚动、标签页切换、关闭等操作进行自动联动。这就要求播放窗口必须是本地原生程序实现，最好用高性能的 C++语言来开发，还可充分利用终端电脑的硬件加速特性。这个播放窗口同时提供 Web Socket 的服务端和 JSON 打包命令的解析执行模块，前端就可以通过 Web Socket 连接后发送 JSON 打包的控制命令实现控制播放窗口。

目前市场上已经有采用此思路实现的相关软件和实施案例，比如 PluginOK 中间件(https://github.com/wangzuohuai/WebRunLocal)，提供了一个统一的不依赖浏览器本身扩展技术的外接系统，能实现当前主流浏览器的全兼容，包括低版本的 Chrome 和 IE 浏览器；而且小程序的下载和安装提供了类似 ActiveX 控件的机制，去掉了一些影响用户体验的告警并附加了调用方安全验证机制。而这个播放窗口程序也提供了比较好的范例实现，其具体调用方法可以参考这里的说明：https://github.com/wangzuohuai/WebRunLocal/blob/master/Plugins/VlcWebPlayer/VlcPlayerApplet.txt，难能可贵的是在这个播放窗口还直接实现了多路 RTSP 的同时播放支持，可点选切换播放窗口焦点和全屏播放。据了解，此方案已经成功在

机场、地铁站、交管局等客户现场完成实施并取得了良好的效果，获得了客户的一致好评，毕竟能实现低延迟、低成本的多路同时播放是硬道理。下面是播放效果视频展示：https://www.bilibili.com/video/bv1w5411V78i

某视频监控大厂最近也发布了类似的版本，不过经过测试发现，不支持 Firefox 高版本浏览器不说，其播放窗口程序框架采用臃肿的 QT 来实现的，看上去播放窗口只是模拟显示的效果而不是真正内嵌到浏览器中的，导致和浏览器的联动效果比较差，程序包也很大，且未提供前端自动升级和安全调用机制。另外想用此播放组件还必须购买其 DSS 系统，而这套 DSS 系统的售价不菲，对客户来说性价比很低。

对于个别客户提出的免插件播放要求，主要是担心安全问题。其实所谓的免插件实现方案中，也是需要浏览器从服务器端下载 JS 版播放器的，而外接版只不过下载的是本地版播放器，只需要保证下载到本地的播放器程序是安全的即可，必要的话可通过开放播放器源代码来打消客户对安全的顾虑。还有原因就是需要额外下载外接程序导致部署和升级麻烦，但为了超低延迟的稳定播放效果，这个就是必要的代价了，况且前文提到的 PluginOK 中间件提供了播放小程序的自动安装和升级机制，这样就大大降低了部署和升级的压力，效果比 IE 中的 ActiveX 控件还好。

四、总结

一个好的技术实施方案，首先是要满足客户的刚性需求，其次是尽量降低采购、开发、实施及维护的总成本，再次是是良好的兼容性和稳定性，最后需尽量确保技术方案不能因为浏览器的升级而失效。本文基于当前最新的技术信息和实践经验，提供了这样一个稳定可靠、兼容性好、低延迟又可同时稳定播放多路 RTSP 的低成本半开源技术方案，尤其适合播放高分辨率的 RTSP，以供大家选型参考。如还有技术问题或系统开通权限体验，可以加微信 ZorroSoft 沟通。