Computer Science and Application 计算机科学与应用, 2025, 15(3), 129-137
Published Online March 2025 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2025.153065

Hans汉斯

基于HTML5的B/S架构应用中的实时流媒体技术综述

许　哲，谭蓉俊，张增源，杨玉雯，刘升艳，王　恒

中国船舶集团有限公司昆明船舶设备研究试验中心，云南　昆明

收稿日期：2025年2月16日；录用日期：2025年3月14日；发布日期：2025年3月20日

摘　要

在开发基于HTML5的B/S架构应用系统时，在处理实时流媒体文件时，需要考虑因为编码方式、文件容器格式和传输协议的不同而导致的浏览器兼容性、实时性和带宽效率等差异，并结合业务实际需求进行开发。本研究综述了适用于HTML5开发的主流实时流媒体技术，包括流媒体文件编码技术、文件容器格式、流媒体传输协议，综述内容包含技术特点简介、浏览器兼容性、视频延迟情况、带宽效率与视频质量等方面。同时本研究简单介绍了不同视频来源设备常用的实时流媒体传输技术。本研究结果可为开发者根据业务需求选择合适的流媒体传输技术提供参考。

关键词

HTML5开发，实时流媒体技术，B/S架构应用系统

A Review of Real-Time Streaming Media Technology in HTML5-Based B/S Architecture Application

Zhe Xu, Rongjun Tan, Zengyuan Zhang, Yuwen Yang, Shengyan Liu, Heng Wang

Kunming Shipborne Equipment Research and Test Center, China State Shipbuilding Corporation Limited, Kunming Yunnan

Received: Feb. 16th, 2025; accepted: Mar. 14th, 2025; published: Mar. 20th, 2025

Abstract

In developing B/S architecture application systems based on HTML5, it is essential to address

文章引用: 许哲, 谭蓉俊, 张增源, 杨玉雯, 刘升艳, 王恒. 基于 HTML5 的 B/S 架构应用中的实时流媒体技术综述[J]. 计算机科学与应用, 2025, 15(3): 129-137. DOI: 10.12677/csa.2025.153065

differences in browser compatibility, real-time performance, and bandwidth efficiency that arise due to variations in encoding methods, file container formats, and transmission protocols when handling real-time streaming media files. This study reviews mainstream real-time streaming media technologies suitable for HTML5 development, covering file encoding techniques, file container formats, and streaming transmission protocols. The review addresses technical features, browser compatibility, video latency, bandwidth efficiency, and video quality. Additionally, this review briefly introduces common real-time streaming transmission technologies for various video source devices. The findings of this study aim to provide developers with a reference for selecting appropriate streaming transmission technologies based on business requirements.

Keywords

HTML5 Development, Real-Time Streaming Media Technologies, B/S Architecture Application Systems

1. 引言

B/S 架构(Browser/Server 架构)是一种基于浏览器和服务器的网络架构模式，有着跨平台、易部署和易维护的特点，得到了广泛应用。在船舶工程的应用场景中，常会使用到基于 HTML5 开发的 B/S 架构综合显控系统、数据管理系统、运行管理系统等，其中会涉及对来自于不同位置、不同设备的实时视频流的实时查看和存储管理，例如来自监控摄像头的监控画面、下位机工控机图像、计算机共享画面、无人机实时图传画面等；同时观看视频画面的位置会包括局域网内的固定设备、移动设备，或通过互联网实现远程访问。这要求开发人员需要根据不同的流媒体文件格式在加载速度、兼容性、传输延迟和用户体验等的差异，结合业务实际需求，选取合适的技术与格式进行应用系统开发。

本文旨在梳理和分析适用于船舶工程的应用场景中的、基于 HTML5 开发的 B/S 应用的实时流媒体技术，探讨其在技术特点、浏览器兼容性、传输效率、实时性等方面的差异，为开发者在 B/S 架构下的流媒体应用中选择合适技术进行开发提供科学依据。

2. 关键技术对比分析

2.1. 编码技术

编码技术指的是将原始音视频数据压缩成便于传输和存储的格式，在减小数据量的同时尽量保持原始质量的技术。在本节中，我们从技术特点、浏览器兼容性、带宽效率与视频质量三个方面，分别对 H.264、H.265、AV1 和 VP9 共 4 种主流的流媒体文件编码技术进行了综述。

H.264 (AVC-Advanced Video Coding) [1]是一种高效的视频编码方式，采用帧内压缩和帧间预测的组合，在较小的文件体积下提供高质量视频。其压缩比高且能支持多种分辨率(4K 及以下) [2]，成为互联网视频和流媒体的主流选择。H.264 被大多数主流浏览器支持，几乎所有现代浏览器都能够播放 H.264 编码的视频，包括 Chrome、Firefox、Safari 和 Edge。H.264 编码在前端库中有广泛支持，如 Video.js 和 hls.js 等流媒体库均支持 H.264 编码的视频文件。这些库的良好支持使得 H.264 成为大多数开发者的首选。H.264 具有较高的带宽效率和优质的视频质量[3]，尤其在高清和全高清视频传输上表现出色。其编码效

率使其适合低带宽环境，同时还能提供清晰的画面[4] [5]。

H.265 (HEVC-High Efficiency Video Coding)是 H.264 的继任者，支持从 4K 到 8K 等高分辨率视频，采用了更复杂的压缩算法，能够在相同质量下减少约 50%的数据量[4] [6] [7]，进而减少带宽消耗[2]。Safari 浏览器支持 H.265 编码，但 Chrome、Firefox 和 Edge 对其原生支持较少，这限制了 H.265 在跨浏览器视频应用中的普及。前端库对 H.265 的支持相对有限。Video.js 等库可以在 Safari 中播放 H.265 编码视频，但在其他浏览器中需要额外插件或硬件支持。对于需要高兼容性的项目，H.265 可能不是最佳选择。H.265 比 H.264 拥有更高的带宽效率和更优的视频质量，在低带宽条件下可以呈现清晰度更高的画面，非常适合超高清内容的传输。不过其编码过程更复杂，对解码设备的计算能力要求较高[5]。

AV1 (AOMedia Video 1)是由开放媒体联盟开发的开源编码格式，旨在替代 H.265，且不受专利限制。AV1 采用高度优化的压缩算法，支持 4K 及更高分辨率，同时在带宽效率方面进一步提升[8]。AV1 得到了 Chrome、Firefox 和 Edge 的支持，但 Safari 的支持有待完善。由于 AV1 编码在技术上的复杂性编码速度较慢，Video.js 等库对其支持有限，且需要浏览器提供硬件加速或解码支持。但 AV1 比 H.264 和 H.265 有更好的带宽效率，在低比特率下仍可提供较高的视频质量[4] [5]，且解码效率在逐渐优化，适合对带宽敏感的应用场景。

VP9 是 VP8 的继任者，是 Google 开发的开源视频编码格式，主要用于 WebM 容器中，专为网络优化设计，具有较高的压缩效率和较低的计算复杂度，能在保持相近画质的情况下显著降低码率，相较于 H.264，能有效减少带宽消耗[8]。VP9 在 Chrome、Firefox 和 Edge 浏览器中均有良好的支持，然而在 Safari 中支持有限，尤其在较旧版本中。Video.js 和 Plyr.js 等前端库对 VP9 编码的支持较好。VP9 的带宽效率极高，与 H.264 相比，VP9 可以在相同的带宽下提供更高的视频质量，或在同等画质下大幅降低带宽消耗[5]。特别是在高分辨率应用(如 4K 视频)中，VP9 的带宽效率优势明显，能保持较高的视觉质量[4]。

本节中提及的实时流媒体文件编码技术主要特点对比如表 1 所示。在评估带宽效率与视频质量时，根据不同研究资料中对各编码技术的性能对比，我们梳理出了各编码技术之间大致的性能倍数差距。在表格中，我们将带宽效率和视频质量性能最低的编码技术记为一个基准值 1，而其他的编码技术性能记为相应的倍数。

Table 1. Comparison of the main features of real-time streaming media file encoding technology
表 1. 实时流媒体文件编码技术主要特点对比

编码技术	特点	浏览器兼容性	带宽效率	视频质量
H.264	目前最主流的选择	大部分浏览器支持	1	1
H.265	支持大于 4K 分辨率，编解码复杂	Safari 支持，其他大部分浏览器不支持	1.5	1.1
AV1	编解码复杂，开源	Safari 不支持，其他浏览器大多可以支持	1.8	1.2
VP9	VP8 的全面升级	Safari 需要转码播放，其他浏览器大多支持	1.4	1.05

2.2. 文件容器格式

文件容器格式(或封装格式)是用来封装和组织视频、音频、字幕和元数据的文件格式。一个容器可以封装多个流，允许将视频、音频、时间戳、数据索引、字幕轨道等数据同时存储在一个文件中。不同的容器格式支持不同类型的编码技术，使得同一种编码内容在多种平台和设备上都能得到兼容和高效播放。在本节中，我们从编码支持情况、浏览器兼容性、带宽效率与视频质量等三个方面，分别对 MP4、WebM、FLV、MKV、AVI 和 TS，共 6 种适用于网络流媒体传输存储的文件容器格式进行了综述。

MP4 (MPEG-4 Part 14)是最常用的视频封装格式之一，广泛应用于视频流媒体、存储和播放。它支持

多种视频和音频编解码器，因其良好的兼容性和压缩效率而成为流媒体行业的标准[9]。主流浏览器(如Chrome、Firefox、Safari 和 Edge)均对 MP4 提供了良好的原生支持，因此用户在播放 MP4 式时几乎不需要额外的插件[10]。MP4 格式的集成和开发相对简单，如 Video.js、hls.js 等知名前端库，以及 HTML5 中<video>标签均支持 MP4 播放。同时开发者社区资源丰富，使得新手开发者也能够快速上手。MP4 支持多种音视频编码方式，具有良好的压缩率和高效的带宽利用率，适合在有限带宽环境下传输高清内容，在较小的文件大小下保持清晰度，适合高清视频点播和网络分发。

WebM 是由 Google 开发的开源容器格式，专为网络设计，支持高效的视频流，旨在提供高效、低带宽的视频传输体验。WebM 格式的开发相对简单，HTML5 中<video>标签原生支持 WebM 格式，开发者无需额外的解码插件即可实现播放[10]。WebM 格式在 Chrome 和 Firefox 中得到了广泛支持；在 Safari 浏览器中支持有限；Edge 浏览器在较新版本中支持 WebM 格式，但对老版本的浏览器支持不佳。WebM 使用 VP9 和 AVI 格式的编码，在前端库中得到广泛支持，大多数开源播放器(如 Video.js、Plyr、Clappr 等)都兼容 WebM 格式。WebM 格式在带宽效率上表现良好，能够在较低带宽条件下保持较高的视频质量。

FLV (Flash Video) [1]格式基于 Adobe Flash 技术[11]，曾经在流媒体视频传输中被广泛使用，主要依赖 Flash Player 进行播放。然而，随着 Flash 技术逐渐退出主流 Web 标准，几乎所有浏览器都已停止对 Flash 的支持，导致 FLV 格式难以直接在网页上播放。为应对该情况，开发者可以采用 FLV.js 等 JavaScript 库在不依赖 Flash 插件的情况下解析 FLV 文件，并将其转换为 HTML5 兼容的视频流。另一个解决方案是将 FLV 流转换为更现代的格式，如通过服务器端转码将 FLV 转为 HLS 或 DASH，以实现跨平台和多浏览器支持。FLV 格式通常支持 H.264 编码，具有良好的带宽效率，文件体积相对较小，适合在网络带宽较低的情况下传输视频。但 FLV 格式的视频质量一般，在需要高清画质的场景中逐渐被替代。

MKV (Matroska Video)是开源的多媒体容器格式，支持多种音视频编码和字幕、章节等元数据[9]。MKV 常用于高清影视存储和流媒体播放。Chrome 和 Firefox 等主流浏览器对 MKV 的支持良好，但 Safari 的兼容性较差，大部分现代前端库支持 MKV 格式。由于支持多种编码方式，使得其开发难度不高，且应用广泛，适合不同场景。MKV 格式支持 H.264、H.265、VP8、VP9、AV1 等多种编码方式，适合不同需求的视频播放，具有较高的带宽效率，适合高清播放和离线存储，画质优良。

AVI (Audio Video Interleave)是微软推出的多媒体容器格式，历史悠久，广泛用于视频存储。虽然 AVI 格式较老，但仍在某些场合中使用[9]。浏览器对 AVI 的原生支持较少，通常需要额外的插件才能播放。前端库对 AVI 的支持较差，多用于本地播放。AVI 格式开发难度相对较低，但由于其格式限制，可能影响现代应用的灵活性。AVI 格式支持多种编码方式，如 MPEG-4 和 H.264，但由于其设计较早，压缩效率较低，在带宽效率上表现一般，虽然视频质量较高，但文件大小通常较大。

TS (Transport Stream)格式是由 MPEG 组织开发的一种用于存储和传输音视频的容器格式，常见于数字电视广播、实时流媒体和监控应用。TS 格式的原生浏览器支持较少，大多数浏览器并不直接支持 TS 文件的播放。但通过前端流媒体库(如 HLS.js、Video.js 等)，可以实现基于 HTTP Live Streaming (HLS)协议的播放[1] [11]，使 TS 格式能够适应浏览器播放环境，同时令开发者可以简化集成过程，相对轻松地播放 TS 格式文件。TS 格式支持 H.264 和 H.265 等编码，支持高质量的视频传输，适合高带宽的网络环境，也可以在保证质量的同时减小传输带宽的需求。此外，TS 格式的容错性较强，能够在网络传输不稳定的情况下保持较高的视频质量，因此被广泛应用于实时视频和数字电视广播等需要稳定性的场景。

本节中提及的实时流媒体文件容器格式主要特点对比如表 2 所示。在评估带宽效率与视频质量时，根据不同研究资料中对各容器格式的性能对比，我们梳理出了各容器格式之间大致的性能倍数差距。在表格中，我们将带宽效率和视频质量性能最低的容器格式记为一个基准值 1，而其他的容器格式性能记为相应的倍数。

Table 2. Comparison of the main features of real-time streaming media file container format
表 2. 实时流媒体文件容器格式主要特点对比

文件格式	支持编码	浏览器兼容性	带宽效率	视频质量
MP4	H.264、H.265	大部分浏览器原生支持	1.3	1.1
WebM	VP8、VP9、AV1	Safari 不支持，其他大部分浏览器原生支持	1.6	1.2
FLV	H.264	FLASH 停用后的新版浏览器均不支持，依赖前端库转码	1.2	1
MKV	H.264、H.265、VP9、AV1	Safari 不支持，其他大部分浏览器需要依赖前端库转码	1.4	1.1
AVI	H.264	大部分浏览器不支持，需要依赖前端库转码	1	1
TS	H.264、H.265	大部分浏览器原生不支持，但在 HLS 协议内使用，可在大部分浏览器内播放	1.3	1

2.3. 传输协议

传输技术是指将音视频数据从源端传送到目标端的技术方案，包含数据流如何被传输、传输时的协议规范、以及在网络不同状态下如何处理传输流的效率和可靠性。传输技术的核心在于实现流媒体数据的实时性、低延迟和高效传输。在本节中，我们从工作原理与特点、浏览器兼容性和实时性与延迟等三个方面，分别对 HLS、DASH、RTSP、RTMP、HTTP-FLV 和 WebRTC，共 6 种主流的流媒体文件传输协议进行了综述。

HLS (HTTP Live Streaming) [11] [12]是一种基于 HTTP 的流媒体传输协议，由 Apple 推出并广泛应用于直播场景。HLS 将视频内容切割成小的 TS 文件片段，并使用 m3u8 索引文件管理播放列表，播放器可以通过 HTTP 根据索引文件逐片段下载并播放视频，实现流式传输[13]。HLS 在 Safari 浏览器上得到原生支持，但在其他浏览器中需要借助第三方库(如 hls.js)来进行播放。HLS 的实现相对复杂，开发者通常依赖现成的 HLS 播放器库(如 hls.js、videojs-http-streaming 等)来实现流媒体传输，避免繁琐的流切片和播放控制。由于 HLS 分片方式传输，每个分片通常为 2~6 秒，在传输实时流媒体内容时延迟相对较高[1]，通常大于 10 秒。低延迟 HLS (Low-Latency HLS)可将延迟缩短至 2~4 秒，但还需要浏览器和服务器端的全面支持。HLS 支持自适应比特率流，能够根据用户网络情况自动调整视频质量，提升带宽效率，适合大规模直播和高清视频内容分发。

DASH (Dynamic Adaptive Streaming over HTTP) [8]是一种基于 HTTP 的流媒体协议，与 HLS 协议类似，DASH 将视频文件分成小段，客户端根据 MPD 文件索引下载片段。DASH 格式在 Chrome、Firefox、Edge 等主流浏览器中获得良好支持，但在 Safari 浏览器中需要通过插件或转换为 HLS 实现兼容。DASH 的实现较为复杂，开发者通常需要依赖 Dash.js、Video.js 等开源库来简化流程，且服务器配置和播放器管理也需要较高的技术背景，对开发者要求较高。DASH 的实时性与 HLS 相似，一般延迟大于 6 秒，适合于直播延迟容忍度较高的应用。通过启用低延迟 DASH (LL-DASH)模式，DASH 协议的延迟可减少至 2~4 秒，最小延迟可达到 1.5 秒，但依赖于稳定的网络环境和优化的服务器设置。DASH 也采用自适应比特率传输，支持多分辨率，可以根据网络状况自动调整视频质量，带宽利用率高且集成灵活。

RTSP (Real-Time Streaming Protocol) [6] [11] [13]是一种用于控制流媒体服务器的协议，支持音视频的实时传输，支持 TCP、UDP 协议的切换[14]。RTSP 大多用于摄像头等硬件设备的实时视频流观看和推送，一般不在 Web 领域直接使用，未得到 HTML5 原生支持，Chrome、Firefox、Safari 等主流浏览器都不直接支持 RTSP 协议，因此在 Web 应用中实现 RTSP 播放通常需要额外插件或转换工具，例如第三方

库(如 JSMpeg、RTSP to WebRTC)进行协议转换以支持网页播放，需要开发者具备一定的流媒体协议基础。RTSP 协议支持双向传输和快速数据包处理，通常延迟在 0.5~2 秒，最小延迟在 500 毫秒以内[15]。

RTMP (Real-Time Messaging Protocol) [13]是由 Adobe 开发的一种流媒体传输协议，最早用于 Flash 播放器，专门设计用于低延迟的音视频数据传输[11]。RTMP 通信是建立在 TCP 长连接通道上的，在封装音视频数据时会强制切片，限制每个数据包的大小，这在一定程度上保证了实时性，且有一定的弱网抵抗能力。随着 Flash 的逐渐退役，RTMP 在浏览器端失去原生支持。如今，RTMP 通常用于服务器端传输，在浏览器播放时，常用 HTML5 播放器(如 Video.js、hls.js 等)都不兼容 RTMP 格式，需借助中间件将 RTMP 流转换为其他格式，如 HLS、DASH 等格式来实现兼容。这种转换过程对服务器端处理和流媒体管理技术提出了更高要求，拉高了开发难度。然而 RTMP 在现代 Web 环境中仍具有一定应用价值，特别是在需要低延迟的直播或互动场景中，其延迟通常低于 3 秒[1]，最小可达到 1~2 秒[14] [15]。

HTTP-FLV [11]是一种低延迟的流媒体协议，基于 HTTP 长链接传输 FLV 容器格式的视频流数据[1]，支持跨平台播放，适合在直播中实现低延迟传输。与 RTMP 协议类似，HTTP-FLV 会对视频文件会强制切片并限制每个数据包的大小，以此保证实时性和弱网抵抗能力。由于 FLV 格式依赖于 FLASH 播放器，在 FLASH 不受主流浏览器支持后，基于 HTTP-FLV 协议的视频播放也不再被直接支持。开发者可以采用 FLV.js 等前端库将 FLV-HTTP 协议的视频流转换为 HTML5 兼容的视频流进行播放[12]，或通过服务器端转为 HLS 或 DASH 流，以实现跨平台和多浏览器支持。HTTP-FLV 协议的延迟也是比较低的，大概在 1~3 秒左右。

WebRTC (Web Real-Time Communication) [11]基于 UDP 进行通信，用于在浏览器和移动应用之间实现实时音视频和数据的点对点传输。WebRTC 在现代主流浏览器中均原生支持[3]，无需安装插件，跨浏览器兼容性较好。WebRTC 的开发难度相对较高，需要实现多种协议栈来确保流畅、低延迟的数据传输，会话控制和数据加密也需开发者进行精细化配置。然而 WebRTC 得到多个前端库的支持，如 SimpleWebRTC、PeerJS 和 Socket.io，在一定程度上降低了开发难度。WebRTC 以低延迟著称，延迟通常控制在 500 毫秒以内，最小延迟可达 100 毫秒左右，在点对点(P2P)通信中尤其高效。WebRTC 视频质量在高比特率下表现优秀，当网络不稳定或带宽受限时，也可以降低清晰度以保证流畅的实时传输。

本节中提及的实时流媒体传输协议主要特点对比如表 3 所示。

Table 3. Comparison of the main features of real-time streaming media transport protocols
表 3. 实时流媒体传输协议主要特点对比

编码技术	网络协议	浏览器兼容性	特点	播放延时
HLS	TCP 短连接	Safari 浏览器原生支持，其余可通过前端库转码后直接播放	浏览器兼容性强	>10 s
DASH	TCP 短连接	Safari 浏览器需要转码后播放，其余浏览器新版支持较好	浏览器兼容性强	>6 s
RTSP	TCP/UDP 可切换	需要前端或后台转码后才能播放	不适合浏览器播放	<2 s
RTMP	TCP 长连接	需要后台转码后才能播放	不适合浏览器播放	<3 s
HTTP-FLV	TCP 长连接	需要前端或后台转码后才能播放	依赖第三方库	<3 s
WebRTC	UDP	大部分浏览器原生支持	开发难度大	<500 ms

3. 视频源设备特点

不同的实时视频来源设备通常有着不同的流媒体传输方案。在进行信息管理系统的开发建设时，需

要根据所需的应用场景，整理出所需兼容的传输方案，同时对可配置的设备选取较优方案进行设置，从而适应整个系统在使用需求中对视频延迟、质量、兼容性等方面的要求。以下是对几种常见实时视频来源设备特点的分析。

监控摄像头通常用于实时视频监控，通常采用高压缩效率、低延迟的 H.264 或 H.265 编码[1]。对于容器格式，多采用容错率高的 TS，部分设备支持 MP4，适合保存视频回放。由于对延迟要求较高，主要采用 RTSP 协议，且大多数监控设备内置 RTSP 支持，通过局域网或互联网传输实时视频；某些监控设备也支持 HTTP 直播流协议(HLS)，尤其在需要在网页或跨平台查看时。虽然 HLS 延迟稍高(通常 8~10 秒)，但其兼容性较好，适合存档录像和低实时性要求的监控场景。

网络摄像头主要用于视频通话、直播和会议等应用中，通常使用有较高的压缩效率、能在有限带宽下保证视频质量的 H.264 编码。对于容器格式，多采用便于网络传输的 MP4、FLV 格式。WebRTC 是网络摄像头的主要传输协议，支持低延迟和点对点连接，适合实时视频交互。RTMP 和 HTTP-FLV 则通常用于直播平台和网页嵌入应用。

移动设备(如手机、平板等)通常使用 H.264、H.265 和 VP9 编码保证压缩率和视频质量的平衡。由于对格式的兼容性要求较高，通常使用 MP4 和 WebM 作为容器格式，或使用 TS 格式配合 HLS 协议使用。针对浏览器兼容性，移动设备多采用 HLS 传输，也会使用 WebRTC 用于低延迟的实时通信场景，如视频通话和互动直播。

专业摄像机用于新闻直播、赛事转播等场景中，对画质和低延迟要求高，常采用 H.264 和 H.265 编码方式满足高质量的同时也保证了文件的兼容性。在容器格式上，MP4 常用于需要高兼容性的直播和点播场景。在直播过程中，通常使用 DASH 或 HLS 进行流媒体传输，或通过 RTMP 协议传输至服务器，通过服务器推流到不同观看端。

无人机摄像头因其应用于户外实时影像传输，对实时性和带宽要求较高，故通常采用 H.264 或 H.265 编码来压缩视频流[2]。常用的容器格式有高兼容性的 MP4 和广泛用于实时推流的 FLV。RTMP 和 HTTP-FLV 协议因为具有较好的低延迟特性，常用于无人机平台的实时视频传输。如今也有部分无人机平台开始引入 WebRTC 以提供更低的延迟和更高的流畅性。

由于一些数据的处理与可视化工作(如气象数据、雷达数据等)需要依靠计算机实现，而计算机输出的可视化信息有被远程观看的需求，所以需要将计算机输出到显示器的画面编码为流媒体视频。H.264 是计算机显示屏推流的主流编码方式，因其兼容性好、带宽效率高，适合清晰地呈现屏幕细节。VP9 和 H.265 编码在一些高分辨率、高帧率需求下也被采用。MP4、WebM 是最常用的视频容器格式。WebRTC 和 RTMP 是计算机显示屏推流中最广泛使用的传输协议，RTMP 适合向直播平台和内容分发网络(CDN)推送，WebRTC 则用于互动性强的实时应用。对于非实时点播场景，也会使用 DASH 和 HLS 协议进行后续视频回放。

Table 4. Comparison of the main features of real-time streaming media source device
表 4. 实时视频来源设备主要特点对比

视频来源设备	常用编码方式	常用容器格式	常用传输协议
监控摄像头	H.264、H.265	MP4、TS	RTSP、HLS
网络摄像头	H.264	MP4、FLV	WebRTC、RTMP、HTTP-FLV
移动设备	H.264、H.265 和 VP9	MP4、TS、WebM	HLS、WebRTC
专业摄像机	H.264、H.265	MP4、TS	HLS、DASH、RTMP
无人机	H.264、H.265	MP4、FLV	WebRTC、RTMP、HTTP-FLV
计算机	H.264、H.265、VP8 和 VP9	MP4、WebM	WebRTC、RTMP、HLS、DASH

本章中提及的实时视频来源设备主要特点对比如表 4 所示。

4. 技术路线建议

4.1. 选型原则

在基于 HTML5 的 B/S 架构应用中，实时流媒体技术的选型需综合考虑业务需求、设备特性及网络环境。对于开发者来说，首先需要明确核心需求，之后根据场景特性，确定技术组合。根据实际工程中的经验，我们提出了四条技术选型原则建议如下：

原则一：兼容性优先。适用于多终端访问、跨平台分发场景(如视频点播、公共直播等)。在编码上，建议优先选择浏览器支持率高的编码(如 H.264 等)，避免使用专利受限或兼容性差的编码(如 H.265 等)。在容器上，建议优先选择多平台原生支持(如 MP4 格式)或开源且广泛兼容(如 WebM 格式)的容器格式。在协议上，建议选择 HLS 或 DASH，确保分段加载与容错性。

原则二：实时性驱动。适用于强交互场景(如视频会议、无人机操控、远程协作等)。在编码上，建议优先选择低带宽占用和适配超高清场景的编码方式(如 VP9 或 H.265 等)，但需权衡编码复杂度与设备算力。在容器上，建议优先选择低延迟格式(如 WebM 或 FLV 等)。在协议上，建议优先选择点对点传输协议(如 WebRTC 等)。

原则三：带宽效率优化。适用于网络条件受限的场景(如移动端户外直播、物联网设备图传等)。在编码上，建议优先选择压缩率高(如 H.265 等)或低码率(如 AV1 等)的编码方式。在容器上，建议优先选择抗丢包能力强(如 TS 等)或轻量级封装(如 FLV 等)的格式。在协议上，建议优先选择推流稳定(如 RTMP 等)或容错性强(如 HLS、DASH 等)的协议。

原则四：扩展性与未来适配。适用于有长期运维系统、技术迭代需求的场景(如 5G/边缘计算等)。在编码上，建议预留多编码兼容方案，以确保能够逐步支持新兴编码方式(如 AV1 等)。在容器上，建议支持常用行业标准(如 MP4 等)，同时使用支持多轨道、元数据扩展(如 MKV 等)的格式使用。在协议上，建议使用支持模块化设计、易于扩展分辨率/码率层级(如 DASH 等)的协议。

4.2. 典型场景技术栈推荐

针对监控与安防、视频点播、视频会议、实时交互和户外推流五个典型应用场景，我们给出了一份应用场景与设备适配的技术组合推荐，如表 5 所示。

Table 5. Recommended combinations of technologies for application scenarios and device adaptation
表 5. 应用场景与设备适配的技术组合推荐

应用场景	典型设备	编码方式	容器格式	传输协议	核心需求	适配逻辑
监控安防	监控摄像头、无人机	H.264	TS、FLV	HTTP-FLV、HLS	低延迟、高稳定性	H.264 兼容性强；TS 容错性高，HTTP-FLV 替代 RTSP 以适配 HTML5 播放。
视频点播	服务器、专业摄像机	H.264、AV1	MP4、WebM	HLS、DASH	高画质、高兼容性	MP4/WebM 广泛支持；HLS/DASH 自适应码率，适合网络波动。
视频会议	网络摄像头、计算机	VP9、H.265	MP4、WebM	WebRTC	超低延迟	WebRTC 原生支持 P2P；VP9/H.265 优化带宽占用与画质。
实时交互	移动设备、AR/VR 设备	VP9、H.264	WebM、TS	WebRTC、HTTP-FLV	即时反馈、高可靠性	WebRTC 确保低延迟；TS 抗丢包，HTTP-FLV 适配移动端弱网环境。
户外推流	无人机、移动设备	H.264、H.265	FLV、TS	RTMP、WebRTC	抗弱网、低带宽消耗	H.265 压缩率高；RTMP 推流稳定，WebRTC 适配户外实时传输。

5. 结语

本文针对基于 HTML5 的 B/S 架构信息管理系统开发场景，从实时流媒体文件编码技术、文件容器格式和传输协议三个方向综述了多种实时流媒体文件技术，分别对比了它们在技术特点、浏览器兼容性、实时性和带宽效率与视频质量等方面的区别，从而找出其各自的优势和局限性，为开发者应根据应用场景选择适合的 HTML5 流媒体视频技术进行开发提供了参考。

此外，随着 5G 和 AI 技术的发展，流媒体文件格式将进一步优化，更低延迟和更高质量的实时传输将成为未来研究的重点方向，我们也将继续重点关注新技术在流媒体传输和格式优化中的应用，探索更智能化、高效化的解决方案。

参考文献

[1] 张永薪. 基于嵌入式 4G/WIFI 远程移动监控系统设计[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2020.

[2] Benjak, J., Hofman, D., Knezović, J. and Žagar, M. (2022) Performance Comparison of H.264 and H.265 Encoders in a 4K FPV Drone Piloting System. *Applied Sciences*, **12**, Article No. 6386. https://doi.org/10.3390/app12136386

[3] 唐恒博. 云联络中心客户端关键技术的研究和实现[D]: [硕士学位论文]. 西安: 西安电子科技大学, 2020.

[4] Mansri, I., Doghmane, N., Kouadria, N., Harize, S. and Bekhouch, A. (2020). Comparative Evaluation of VVC, HEVC, H.264, AV1, and VP9 Encoders for Low-Delay Video Applications. 2020 4*th International Conference on Multimedia Computing*, *Networking and Applications* (*MCNA*), Valencia, 19-22 October 2020, 38-43. https://doi.org/10.1109/mcna50957.2020.9264275

[5] Layek, M.A., Thai, N.Q., Hossain, M.A., Thu, N.T., Tuyen, L.P., Talukder, A., *et al*. (2017) Performance Analysis of H.264, H.265, VP9 and AV1 Video Encoders. 2017 19*th Asia-Pacific Network Operations and Management Symposium* (*APNOMS*), Seoul, 27-29 September 2017, 322-325. https://doi.org/10.1109/apnoms.2017.8094162

[6] 郭慧. 基于 H.265 流媒体网络实时传输机制的研究[D]: [硕士学位论文]. 呼和浩特: 内蒙古工业大学, 2020.

[7] Perez, S., Facchini, H., Dantiacq, A., Salomon, G.Q., Cangemi, G. and Hidalgo, F. (2020). Comparison of the Behavior of Video Codecs in Wi-Fi IEEE 802.11ac Environment. 2020 *IEEE Congreso Bienal de Argentina* (*ARGENCON*), Resistencia, 1-4 December 2020, 1-8. https://doi.org/10.1109/argencon49523.2020.9505558

[8] Uhl, T., Hoppe, C. and Klink, J.H. (2020). Modern Codecs by Video Streaming under Use DASH Technique: An Objective Comparison Study. 2020 *International Conference on Software*, *Telecommunications and Computer Networks* (*SoftCOM*), Split, 17-19 September 2020, 1-5. https://doi.org/10.23919/softcom50211.2020.9238324

[9] Yi, G., Jianzhong, G. and Dian, C. (2020). Bandwidth Analysis and Research of Automotive Video Transmission Module. 2020 *IEEE 6th International Conference on Control Science and Systems Engineering* (*ICCSSE*), Beijing, 17-19 July 2020, 22-26. https://doi.org/10.1109/iccsse50399.2020.9171936

[10] Jbara, Y.H. (2019) Data Reduction in MMBD Computing. In: Tanwar, S., Tyagi, S. and Kumar, N., Eds., *Multimedia Big Data Computing for IoT Applications*: *Concepts*, *Paradigms and Solutions*, Springer, 217-245. https://doi.org/10.1007/978-981-13-8759-3_8

[11] 陈贺瑶. 基于 HTML5 的 FLV 流媒体播放技术的研究与实现[D]: [硕士学位论文]. 北京: 北京邮电大学, 2019.

[12] 薛粤桂. 支持 Web 直播的视频监控系统的研究与开发[D]: [硕士学位论文]. 广州: 华南理工大学, 2020.

[13] Nunez, L. and Toasa, R.M. (2020). Performance Evaluation of RTMP, RTSP and HLS Protocols for IPTV in Mobile Networks. 2020 15*th Iberian Conference on Information Systems and Technologies* (*CISTI*), Seville, 24-27 June 2020, 1-5. https://doi.org/10.23919/cisti49556.2020.9140848

[14] 王闰琛. 低时延全高清无线视频传输系统的设计与优化[D]: [硕士学位论文]. 杭州: 电子科技大学, 2023.

[15] Aloman, A., Ispas, A.I., Ciotirnae, P., Sanchez-Iborra, R. and Cano, M.D. (2015). Performance Evaluation of Video Streaming Using MPEG DASH, RTSP, and RTMP in Mobile Networks. 2015 8*th IFIP Wireless and Mobile Networking Conference* (*WMNC*), Munich, 5-7 October 2015, 144-151. https://doi.org/10.1109/wmnc.2015.12