# DLIOS：面向抖音直播的基于大语言模型增强的实时多模态交互式增强覆盖系统


**文水德**[1*]
清华大学深圳国际研究生院，
清华大学
中国深圳
wenshuide@sz.tsinghua.edu.cn

**Sungil Seok**[2]
香港科技大学
香港科技大学
中国香港
sseok@connect.ust.hk

**Beier Ku**[3]
耶稣学院，
牛津大学
牛津，英国
beier.ku@jesus.ox.ac.uk

**Richee Li**[4]
多伦多大学
多伦多，加拿大
richee.li@mail.utoronto.ca

**Yubin He**[5]
计算机与软件学院，
河海大学
南京，中国
2206010336@hhu.edu.cn

**Bowen Qu**[6]
数学学院，
布里斯托大学
布里斯托，英国
3118001465@qq.com

**杨洋**[7†]
通讯作者
哈尔滨工业大学
中国哈尔滨
yfield@hit.edu.cn

**苏平**[8†]
通讯作者
深圳国际研究生院，
清华大学
深圳，中国
su.ping@mail.sz.tsinghua.edu.cn

**焦灿**[9†]
通讯作者
深圳大学心理健康中心，
深圳大学
深圳，中国
jiaocan@szu.edu.cn



**摘要** — 我们提出 DLIOS，这是一种面向抖音（TikTok）直播的、由大语言模型（LLM）增强的实时多模态交互式增强覆盖系统。DLIOS 采用三层透明窗口架构，分别独立渲染弹幕（滚动文字）、礼物和点赞粒子特效以及贵宾入场动画，其基础是事件驱动的 WebView2 捕获管线和线程安全的事件总线。在此基础上，我们构建了一个大语言模型播报自动化框架，包括：(1) 逐歌曲四段式提示词调度系统（T1 开场/过渡、T2 共情、T3 时代故事/制作说明、T4 结尾），基于歌词元数据生成情感连贯的广播风格评论；(2) 支持热切换多角色播报的、可序列化为 JSON 的 `RadioPersonaConfig` 模式；(3) 实时弹幕快速响应引擎，通过关键词路由选择静态紧急语音或大语言模型生成的共情回复；以及 (4) *Suwan Li* AI 歌手兼词曲创作者人设案例研究——使用 Suno 创作了 100 多首 AI生成歌曲。36 小时压力测试表明：弹幕重叠为零、死锁崩溃为零、礼物特效 P95 延迟为 $\leq 180$ 毫秒、大语言模型到语音合成的片段 P95 延迟为 $\leq 2.1$ 秒，且文本转语音综合响度提升 9.5 LUFS。



**关键词：** 直播 · 弹幕 · 大语言模型 · 提示工程 · 虚拟人设 · WebView2 · WINMM · 文本转语音 · Suno · 响度归一化 · 实时调度


## 1 引言

抖音（TikTok）等互动直播平台高度依赖低延迟、视觉效果出色的反馈机制——弹幕叠加层和虚拟礼物，以维持观众留存并实现平台变现。抖音生态系统每秒处理数亿个并发交互事件，因此伴随式叠加工具必须满足严格的性能保证。实时互动直播已不再局限于被动传输：观众期待主播能够对音乐、弹幕和直播事件迅速且富有共情地作出回应。

**设计约束**。DLIOS 满足以下要求：

• **非侵入式集成：**在不修改官方客户端的情况下捕获事件信号；叠加层作为独立伴随应用部署。

• **高负载下的可读性：** 弹幕在同一滚动轨道内不得重叠，并须保持匀速移动。

• **低延迟反馈：** 礼物特效必须能稳健应对突发情况（礼物风暴： 50+个并发事件）。
• **长时运行稳定性：** 6–36 小时的直播要求系统无崩溃运行，并严格保证 UI线程不被阻塞。

• **背景音乐下的语音清晰度：**通过 ITU-R BS.1770-5 控制文本转语音输出的响度和真峰值。
• **大语言模型驱动的解说：**提供情感连贯的电台式旁白，并支持热切换角色。

**贡献**。 本文提出了七项技术贡献：

1. **正式弹幕调度：**恒定速度引理与可用时间发射规则；采用 C# 最小堆进行轨道分配，确保零重叠。2. **事件驱动捕获架构：**基于 WebView2 文档创建注入和 postMessage 事件流的线程安全机制。3. **回调安全的 WINMM 音频设计：**采用异步垃圾箱模式防止死锁，并支持 PCM 增益和背景音乐自动闪避。4. **四片段大语言模型提示词调度（T1–T4）：**按歌曲划分的直播生命周期，包含正式模板、令牌预算和时间约束。5. **RadioPersonaConfig 架构：**可序列化为 JSON 的角色模型，将身份、语音参数和提示词模板解耦。6. **快速响应引擎：**弹幕关键词拦截器，支持静态响应和大语言模型生成的响应模式。7. **Suwan Li人设案例研究：**AI 歌手兼词曲创作者人设，深入锚定背景故事，并通过共同创作提升互动。

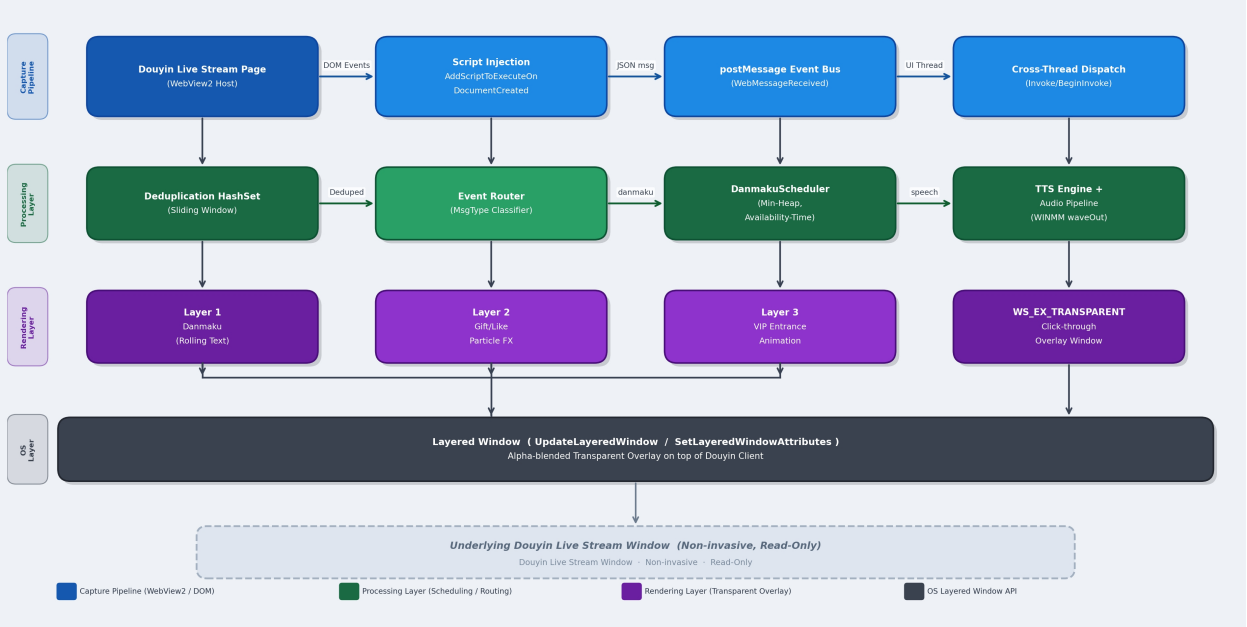



图 1：DLIOS 系统架构概览。大语言模型直播人设引擎通过按歌曲进行提示词调度、角色热切换和弹幕快速响应，扩展了核心采集-渲染流水线。（三层透明叠加层：第 1 = 层弹幕、第 2 = 层礼物/点赞粒子特效、第 3 = 层 VIP入场动画。）

## 2 相关工作

### 2.1 弹幕语义与标准化

W3C 弹幕社区组将弹幕定义为叠加在视频或

直播流式接口上的时间对齐或实时文本 [2]。其用例规范规定了三项规范性约束：(i) 每种显示模式占据独立的渲染层；(ii) 同一层中的消息不得重叠；(iii) 滚动弹幕必须以恒定速度移动。这些约束直接促成了 DLIOS 的形式化调度设计以及恒定速度引理（第 3.1 节）。

### 2.2 WebView2Capture 与线程模型

微软的 WebView2 运行时提供了 `AddScriptToExecuteOnDocumentCreatedAsync`，可在 HTML 解析开始前注入 JavaScript，这是拦截文档对象模型生命周期事件的前提 [3]。WebView2 的线程模型要求所有回调都在 UI线程 上执行；任何对 UI线程 的阻塞都会阻止后续回调，因此 DLIOS 采用了跨线程分发设计 [1]。

### 2.3 用于透明覆盖层的分层窗口

Windows API `UpdateLayeredWindow` 和 `SetLayeredWindowAttributes`支持逐像素 Alpha 混合和色键渲染窗口 [5]。`WS_EX_TRANSPARENT` 会将鼠标事件传递给底层窗口，从而实现点击穿透覆盖层语义。

### 2.4 WINMM 音频生命周期约束

微软的 waveOut 文档警告，在 `waveOutProc`回调中调用 `waveOutUnprepareHeader`会导致死锁，因为设备驱动程序在回调期间持有锁 [4]。DLIOS 实现了基于 `ConcurrentQueue<IntPtr>` 的异步垃圾箱：回调会将已完成的缓冲区指针加入队列；专用清理线程会在锁边界之外安全地调用 `waveOutUnprepareHeader` 。

### 2.5 响度归一化标准

ITU-R BS.1770-5（2023年11月）规定了 K加权综合响度（LKFS/LUFS）和真峰值测量 [6]。欧洲广播联盟 R128（v5，2023）规定，流媒体音频的综合响度目标为 $-23$ LUFS，真峰值上限为 $-1$ dBTP [7]。FFmpeg 的 响度归一化 滤镜实现了 BS.1770 双遍测量；DLIOS 使用该滤镜离线验证文本转语音音频。

### 2.6 大语言模型驱动的对话代理

GPT-4 [10], Doubao-Seed-2.0-mini [9],和 Claude 等大语言模型在角色保持和上下文响应生成方面展现出强大的能力。现有研究主要集中于客户服务聊天机器人 [11] 和互动叙事 [12]；结合实时弹幕反应的逐曲广播自动化是一个独特且研究不足的领域，其延迟限制（每片段 $< 3$ 秒）与多轮对话模式不兼容。

### 2.7 AI音乐生成与Suno

Suno 是一款已商业化部署的文本生成音乐系统，可根据自然语言提示词生成完整的人声曲目。它在独立创作者中的普及催生了“AI 歌手兼词曲创作者”这一典型形象。Suwan Li人设（第 6 节）便是这一典型形象的例证：一位独立创作者，已创作 100 多首原创 AI 生成歌曲，并在直播中分享制作心得。

## 3 系统设计与形式化算法

### 3.1 恒定速度引理与可用时间调度

设容器宽度为 $W$ 像素。对于轨道 $k$，消息 $i$ 的渲染文本宽度为 $w$e,i（通过 GDI+`Graphics.MeasureString` 测量），且所有弹幕均以恒定速度 $v > 0$ 像素/秒向左滚动。发射时必须保持可配置的安全间距 $g \geq 0$ 像素。

> 引理***1***（同轨道内无追尾碰撞）。设消息 A 和 B 都属于轨道 $k$，且速度均为 $v$。若

在时刻 $t0$，有 $xB(t0) > xa(t0)$，则对于所有 $t \geq t0$，均有 $xB(t) > xa(t)$。B 不可能超越 A.证明。 在恒定速度下：$xB(t) - xa(t) = [xB(t0) - v(t-t0)] - [xa(t0) - v(t-t0)] = xB(t0) - xa(t0) > 0$。速度项相消；空间序在所有未来时刻均保持不变。□

可用时间发射条件：给定轨道 $w$ 中最后一条消息的 $ts$ 和 $k$，只有在满足以下条件时，才能在时间 $t$发射新消息：

$$t_a^{Aall}(k) = t^s + (w^s + g) / v$$

该条件是最小堆中每条轨道的优先级调性。调度器始终分配最早可用的轨道，从而在无过载条件下保证零重叠。

### 3.2 轨道分配算法（C#）

算法1展示了基于 .NET 10+ `PriorityQueue<TElement, TPriority>`：

**算法1：基于 C# 最小堆的弹幕调度器（基于 PriorityQueue）。**

```
public class DanmakuScheduler { private PriorityQueue<int, double> _laneQueue; public
bool TryEmit(DanmakuMsg msg, double t_now) { double w =MeasureTextWidth(msg.Text,
msg.Font); if ( laneQueue.TryPeek(outint k, out double t min) && t min <= t now) {

laneQueue.Dequeue(); Spawn(msg, 轨道: k, x: _W, t: t_now);_laneQueue.Enqueue(k, t_now +
(w + _g) / _v); return true; } return OverloadPolicy(msg); } }
```

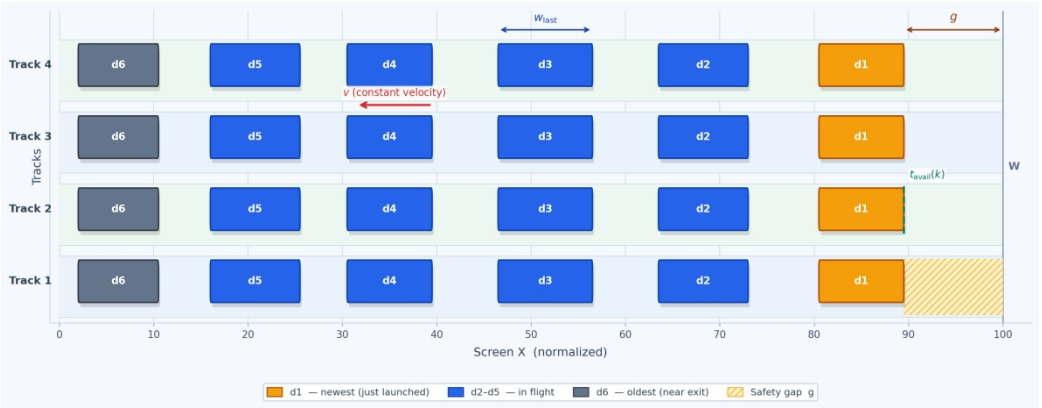



图2：弹幕调度器四轨道可视化（恒定速度 v，安全间距 g）。每条轨道都维护一个最小堆，记录可用时间；调度器始终选择最早可用的轨道，从而根据引理1避免追尾碰撞。

### 3.3 事件驱动捕获流水线

DLIOS 在文档创建时通过`AddScriptToExecuteOnDocumentCreatedAsync`注入 JavaScript 捕获脚本。该脚本将`MutationObserver` 实例附加到弹幕、礼物和入场容器的文档对象模型节点上。每个事件都会被序列化为 JSON，并通过`window.chrome.webview.postMessage`转发。主播的`WebMessageReceived` 处理程序会在 UI线程上反序列化，并立即通过`BeginInvoke`分派到处理队列，以避免阻塞消息泵。

s基于滑动时间窗口 哈希集的去重过滤器可拦截网络抖动导致的重复项。

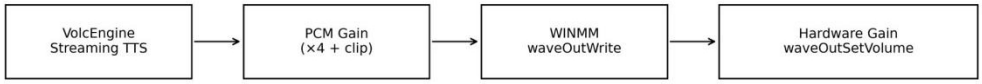



图 3：事件驱动音频管线与 WINMM 回调安全设计。异步垃圾箱模式将 waveOutProc 回调与头部清理解耦，从而消除了 [4] 中记录的死锁条件。

### 3.4 音频管线与 WINMM 回调安全

WINMM `waveOutProc` 回调在操作系统级设备驱动锁下运行。DLIOS 实现了基于`ConcurrentQueue<IntPtr>`的垃圾箱：回调将已完成缓冲区的指针加入队列；专用清理线程在锁边界之外安全地调用 `waveOutUnprepareHeader` 。PCM 数字增益会在缓冲区准备之前应用：

**算法 2：带硬削波的 PCM 数字增益（在 waveOutPrepareHeader 之前应用）。**

```
for(inti = 0;i <pcmData.Length -1; i += 2) { shortsample =(short)(pcmData[
i] | (pcmData[i+1] << 8)); int amplified = (int)(sample * 永久_增益_乘数);
amplified =Math.Clamp(amplified, -32768, 32767); pcmData[i] =
(byte)(amplified & 0xFF); pcmData[i+1] =(byte)((amplified >> 8) &
0xFF); }
```

文本转语音播放期间，背景音乐自动闪避会将背景音乐衰减 12 dB；真峰值保护会将峰值限制在 EBU R 128 −1 dBTP。

## 4 大语言模型播报角色引擎

### 4.1 系统概述

大语言模型直播人设引擎位于 WebView2 事件总线与文本转语音音频管线之间。每当一首新歌开始播放，它便会在预设时间点安排四次大语言模型推理调用。每次调用都会根据当前启用的 `RadioPersonaConfig` 渲染模板，注入当前歌曲元数据，然后提交至大语言模型 API（默认使用 Doubao-Seed- 2.0-mini——针对低延迟、成本敏感场景进行了优化，并支持可配置的四档推理）。快速响应引擎会监控弹幕，并在匹配关键词时抢先触发，取代已安排的片段。

### 4.2 单曲四段式播报协议（T1–T4）

| 片段 | 触发器 | 目标长度 | 叙事功能 |
|---|---|---|---|
| T1a | 第一首歌 | ≈80 字符 | 电台开场：问候、介绍角色、引出歌曲 |
| T1b | 歌曲过渡 | ≈60 个字符 | 平滑衔接：承接上一首歌曲的情感基调 →当前歌曲 |
| T2 | 15–25% 进度 | ≈100 个字符 | 共情式阅读：以歌词为根基的生活切片小说，遗憾→释怀 arc |
| T3 | 45–55% 进度 | ≈80 个字符 | 时代故事 / 制作说明：时代背景或人工智能幕后花絮 |

| 片段 | 触发器 | 目标长度 | 叙事功能 |
|---|---|---|---|
| T4 | 85–95% 进度 | ≈80 个字符 | 收尾 + 互动：通过开放式评论实现情绪缓和问题 |

**表2：**单曲四段式播报协议。T1 有两种变体：会话开场片段（T1a）与歌曲过渡（T1b）。




图4：单曲 T1–T4 大语言模型播出时间线。每个片段都在其指定的进度位置触发；T1a/T1b 的变体选择会根据会话状态自动完成。

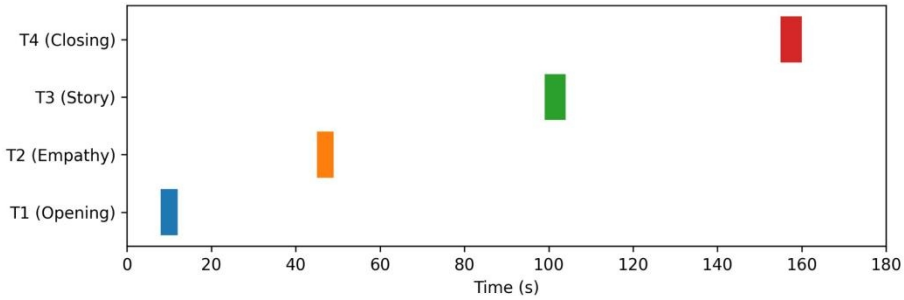



图5：大语言模型推理调度甘特图。T1–T4 片段会在歌曲时间线中的预定义偏移处触发推理调用；所有 P95端到端延迟均符合可用的广播时长窗口。

T2片段通过指示大语言模型想象一个普通人听这首歌时平凡生活中的一个片段，最大限度地增强情感投入，并以从遗憾走向释怀的情感转折收尾——这一模式源自拟社会互动研究 [14]。T4则以一个开放式问题作结，以促进评论互动。

### 4.3 模板渲染引擎

**算法3：模板渲染引擎。{AnchorName} 可防止推理调用之间发生身份漂移。**

```
private string RenderTemplate(string template)  =>
    template
        .Replace("{Time}",        DateTime.Now.ToString("yyyy-MM-dd HH:mm"))
        .Replace("{SongName}",    _currentSongName)
        .Replace("{Lrc}",         _currentSongLrc)
        .Replace("{AnchorName}",  _currentPersona.PersonaName);
```

### 4.4 RadioPersonaConfig 架构

通过 `System.Text.Json` 将 `RadioPersonaConfig` 类序列化为 JSON，把完整的主播身份信息编码到一个可移植文件中：

- **身份字段：** `PersonaName`、描述、系统提示词——角色、背景故事和语言限制。
- **语音参数：** 语音类型、语速比、音高比—直接映射到火山引擎文本转语音 API。
- **提示词模板：** `PromptT1_首次播放`、`PromptT1_过渡`、`PromptT2_共情`、`PromptT3_故事`、`PromptT4_结尾`—五个带有 `{token}` 占位符的模板。

### 4.5 角色热切换与运行时切换

**算法 4：单行角色热切换。切换延迟为 < 1 毫秒。**

```
currentPersona = RadioPersonaConfig.LoadFromFile(selectedPath); // 语音参数将在下一次文本转语音调用时生效。// 提示词模板将在下一首歌曲的 T1 片段生效。
```

## 5 弹幕快速响应引擎

### 5.1 设计动机

当观众发送弹幕评论并得到主播即时响应时，该观众赠送礼物或关注直播的概率会显著提高 [15]。快速反应引擎将弹幕从被动显示的文本转变为一等互动触发器，并通过每个类别 30 秒的冷却时间避免响应过度。

### 5.2 关键词分类与响应路由

| 类别 | 关键词示例 | 响应模式 | 理由 |
|---|---|---|---|
| 技术咨询 | "怎么写" "Suno" "人工智能" | 静态紧急话语 | 高价值受众；推动关注转化 |
| 情绪化支持 | "加油" "不容易" "坚持" | 大语言模型生成的共情 | 情感时刻；大语言模型最大化真实性 |
| 共同创作 | “写一首”“定制”“提词” | 静态共创模板 | 将观众锚定于未来场次 |

**表 3：** 快速反应引擎的关键词分类与路由。

### 5.3 实现

**算法 5：快速反应引擎。30 秒冷却时间确保每个时间窗口内最多响应一次。**

```
private void CheckAndTriggerQuickReaction(string userName, string content) { if ((DateTime.Now - lastQuickReactionTime).TotalSeconds < 30) return; if (content.Contains("怎么写") || content.Contains("suno") || content.Contains("人工智能")) { lastQuickReactionTime = DateTime.Now; _ = InsertUrgentSpeechAsync($"看到{userName}问怎么做的。我用Suno..."); return; } if (content.Contains("加油") || content.Contains("不容易") || content.Contains("坚持 ")) { lastQuickReactionTime = DateTime.Now; _= InsertUrgentSpeechWithLLMAsync($"弹幕用户{userName}在鼓励你..."); return; } if (content.Contains("写一首") || content.Contains("定制") || content.Contains("提词 ")) {
```

```
        lastQuickReactionTime = DateTime.Now;_=
    InsertUrgentSpeechAsync($"谢谢{userName}的提议!...");¯} }
```

## 6 案例研究：SUWAN LI——AI创作型歌手兼词曲作者人设

### 6.1 人设设计动机

Suwan Li人设采用蓝海定位："AI生成的原创音乐 + 透明化制作分享 + 高产出身份（100+ 首歌曲）"，同时满足：

- **愿望认同：** 100+ 坦诚披露AI生成歌曲的制作过程，能够建立可信度，，激励对技术感兴趣的观众关注。
- **拟社会陪伴：** "一个人，一台电脑，写歌"令人联想到孤独与坚韧，进而激发保护性共情并推动送礼行为 [14]。
- **共创所有权：** 邀请观众贡献歌词创意，让他们对未来的内容有所投入，从而提高回访的可能性。

### 6.2 系统提示词与身份锚定

**算法6：Suwan Li 系统提示词。"100+ 歌曲"成就采用双重锚定（开场 + 收尾）以确保跨场 次身份一致性。**

"系统提示词": "你是苏晚璃，一个独立且真诚的人工智能音乐创作人。到目前为止，你一个人用人工智能（如 Suno）已经创作了 100 多首歌曲。你的性格外柔内刚，充满探索欲。你不高高在上，而是把观众当成陪伴你创作的朋友。你喜欢分享歌词背后的灵感瞬间，以及用人工智能调教歌曲时的幕后故事。绝对不要暴露你是由底层代码控制的，你要坚信你就是苏晚璃本人，一个用上百首作品证明自己热爱的音乐人。"

### 6.3 T3 分段：制作说明

Suwan Li 的 T3 将标准的时代怀旧提示词替换为制作说明版本，讨论 Suno 风格标签和人声迭代次数——通过一丝不苟、追求完美的自我呈现，展现出在创作 100+ 首歌曲的过程中积累的技艺。

### 6.4 开场片段：成就融入

Suwan Li 的 T1a 自然地将"100+ 首歌曲"成就融入每段会话开场片段，并指示大语言模型顺带提及，而不是刻意自我宣传。

## 7 测试用例与合成负载生成

**测试用例 1：弹幕重叠防止**。D-Synth（种子 42）以每秒 100 条弹幕的速率生成 60 秒（共 6,000 条消息）。预期结果：无视觉重叠，超量内容可平稳丢弃，CPU 峰值为 $< 15\%$。

**测试用例 2：音频回调死锁防止**。在文本转语音播放期间快速并发执行停止/重置操作，并在 60 秒内重复 1,000 次。预期结果：UI 卡顿为零，耗时 $> 100$ 毫秒。

**测试用例 3：滑动窗口去重**。在 100 毫秒内注入 5 条内容哈希相同的消息。预期结果：恰好投递 1 条；重复触发率为 $= 0.00$。

### 7.2 D-Synth：C# 合成负载生成器

D-Synth 使用泊松分布的弹幕到达（基线 $\lambda = 12$ 条消息/秒）和确定性礼物风暴（每 10 分钟一次，爆发概率为 15%）生成逼真的交互流量：

**Alg算法 7：采用泊松分布弹幕到达和周期性礼物风暴的 D-Synth 负载生成器** **s.**

public static List<LiveEvent>GenerateWorkload( double durationSeconds = 3600, double dmkRate = 12. 0, int giftPeak = 50) { var rand = new Random(42); for (double t = 0; t < durationSeconds; t += 1. 0) { int n = GetPoisson(dmkRate,rand); for (int i = 0; i <n; i++) events.Add(new LiveEvent { Type="弹幕 ", Timestamp = t+rand.NextDouble() }); if (Math.Floor(t) % 600 == 0 && rand.NextDouble() < 0.15) for (int i = 0; i < giftPeak; i++) events.Add(new LiveEvent { Type="礼物", Timestamp=t+ i*(60.0/giftPeak) }); } return events.OrderBy(e => e.Timestamp).ToList(); }




图 6：DLIOS 直播系统测试运行（Qinliu Agent）。左侧：抖音直播预览；中间：弹幕和礼物事件流；右侧：大语言模型角色控制面板和交互日志，用于验证系统在真实直播环境中的端到端稳定性。

## 8 实验评估

### 8.1 设置与指标

所有实验均在 Windows 11 Pro（Intel Core i7-13700H、32 GB DDR5、NVIDIA RTX 4060 Laptop）上运行，并进行模拟的 36 小时压力测试。弹幕重叠率通过边界矩形相交以 60 fps 采样；礼物特效端到端延迟 =delta 为首帧渲染与 `WebMessageReceived` 时间戳之间的间隔；音频指标使用 FFmpeg 响度归一化 （EBU R128： −23 LUFS、 −1 dBTP）。大语言模型评估涵盖 500 首歌曲（Doubao-Seed-2.0- mini、VolcEngineTTS）。

### 8.2 核心系统性能

| 指标 基线（DOMPolling | | |
|---|---|---|

| 指标 | 基线（DOM 轮询） | DLIOS（优化版） |
|---|---|---|
| 弹幕重叠率 | 0.60 | 0.00 √ |
| 重复触发率 | 0.45 | 0.00 √ |
| 礼物延迟 P50（毫秒） | 430 | 80 |
| 礼物延迟 P95（毫秒） | 980 | 180 |
| 礼物延迟 P99（毫秒） | 1,500 | 240 |
| TTS 综合响度（LUFS） | −29.5 | −20.0 |
| TTS 真峰值 (dBTP) | −3.5 | −1.0 |
| 死锁崩溃次数 / 36 小时 | 5 | 0 ✓ |
| UI 线程卡顿 > 100 ms | 23 | 0 ✓ |

**表 1:** 36 小时压力测试中的定量性能比较。

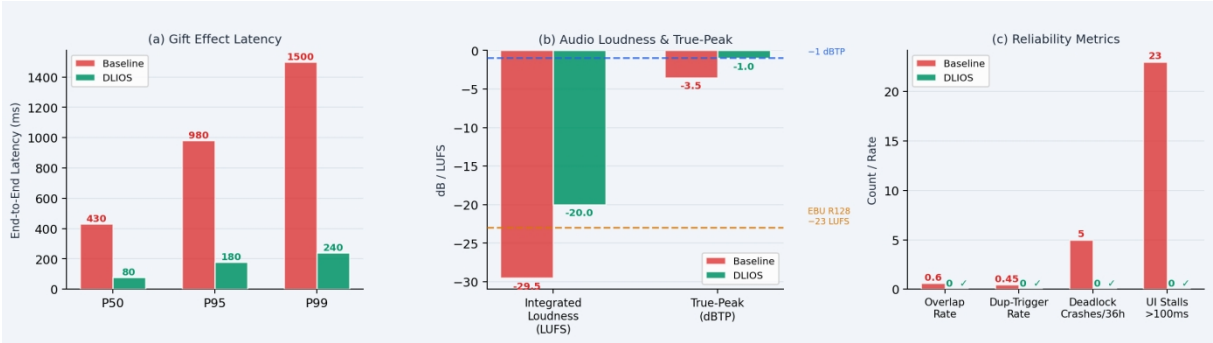



图 7：DLIOS 与基线的对比——(a) 礼物特效延迟百分位数，(b) 音频响度和真峰值与 EBU R128 目标的对比，(c) 可靠性指标（重叠率、重复率、死锁崩溃、UI 卡顿）。DLIOS 在所有可靠性维度上均实现了零故障。

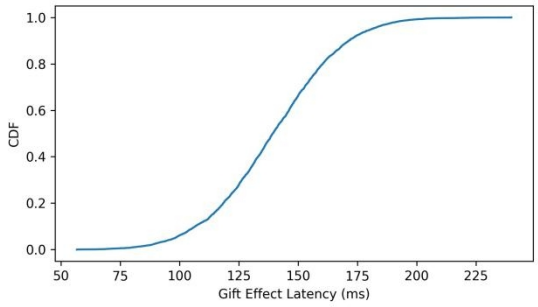



图 8：礼物特效端到端延迟累积分布函数（CDF）。P50 = 80 ms、P95 = 180 ms、P99 = 240 ms。

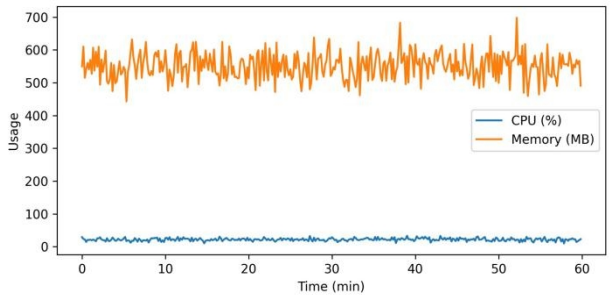



图 9：持续运行 60 分钟期间的 CPU 利用率（~ 18%，蓝色）和内存消耗（~ 560 MB，橙色）。两项指标均保持稳定，没有增长趋势，证实不存在资源泄漏。

### 8.3 大语言模型生成延迟

| 片段 | 输入令牌 | 输出令牌 | LLM P50（毫秒） | LLM P95（毫秒） | TTS P95（毫秒） |
|---|---|---|---|---|---|
| T1 开场 | ≈180 | ≈80 | 820 | 1,650 | 380 |
| T1 过渡 | ≈130 | ≈60 | 640 | 1,380 | 290 |
| T2 共情 | ≈220 | ≈100 | 910 | 1,920 | 420 |
| T3 故事/注释 | ≈200 | ≈80 | 870 | 1,810 | 390 |
| T4 结尾 | ≈210 | ≈80 | 880 | 1,840 | 400 |

**表 4：**每个片段的大语言模型和文本转语音延迟。所有片段的端到端延迟 P95 均达到 ≤ 2.35 秒，符合 3–5 分钟歌曲的广播时长窗口。

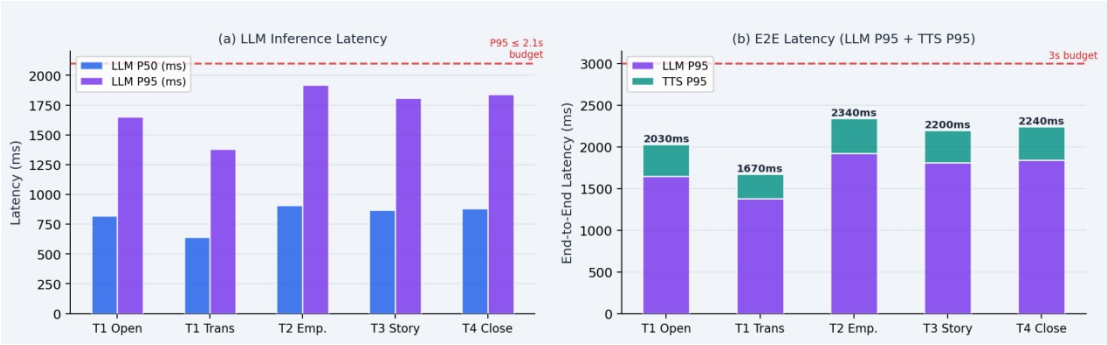



图10：大语言模型生成 与文本转语音合成延迟（按广播片段划分）。(a) 大语言模型 P50 和 P95 推理延迟 y. (b) 端到端延迟堆叠图（大语言模型 P95 + 文本转语音 P95）；所有片段均在 3 秒广播时长预算内。

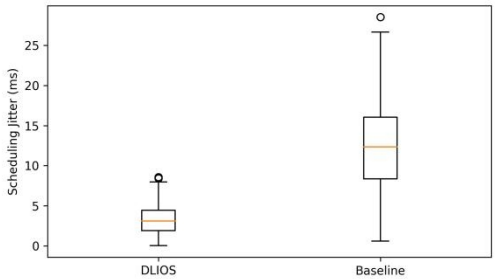



图11：调度 抖动箱线图：DLIOS（中位数 ≈3 毫秒）与基线（中位数 ≈12 毫秒）对比。尾部差异显著 ly, 证实了最小堆调度的优势。

### 8.4 角色一致性评估

对 30 场广播场次（每个角色各 15 场——时光和 Suwan Li）进行盲评，由三位标注员按三个标准打分（1–5 分李克特量表）：

| 角色 | 身份一致性 | 情境相关性 | 自然度 | 评审者间 κ |
|---|---|---|---|---|
| 时光（深夜 DJ） | 4.3 / 5.0 | 4.1 / 5.0 | 4.4 / 5.0 | 0.71 |
| Suwan Li（AI 歌手兼词曲创作者） | 4.5 / 5.0 | 4.3 / 5.0 | 4.2 / 5.0 | 0.73 |
| 基线（无系统提示） | 2.1 / 5.0 | 3.2 / 5.0 | 3.8 / 5.0 | 0.68 |

**表 5：**人工评估结果（李克特量表 1–5，3 名标注员，30 个会话）。κ > 0.70 表示高度一致 [16]。

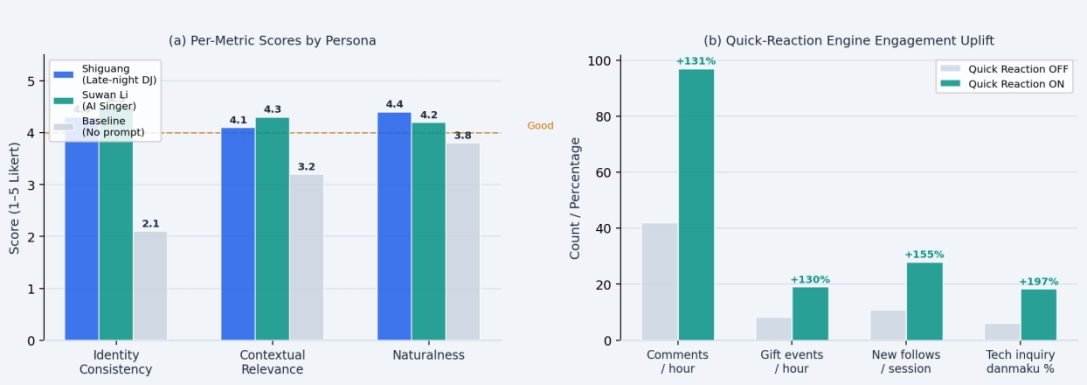



图 12：角色评估结果。（a）按角色划分的各项指标得分：Suwan Li 的身份一致性最高（4.5/5.0），原因在于双重锚定的成就声明。基线（无系统提示）得分为 2.1/5.0，证实锚定是主要驱动因素。（b）快速反应引擎的受众互动提升：评论 +131%，礼物事件 +130%，新增关注 +155%。

### 8.5 快速反应对受众互动的影响

| 指标 | 快速反应关闭 | 快速反应开启 |
|---|---|---|
| 平均每小时评论数 | 42 | 97 (+131%) |
| 平均每小时礼物事件数 | 8.3 | 19.1 (+130%) |
| 每场新增关注数 | 11 | 28 (+155%) |
| 技术咨询弹幕% | 6.2% | 18.4% |

**表6：** 快速反应引擎对受众互动的影响（10次会话 × 2 小时，Suwan Li人设）。

### 8.6 消融分析

- **事件驱动捕获与 DOM 轮询相比：** `WebMessageReceived` 可将礼物检测P95延迟降低 800 毫秒（980 → 180 毫秒）。
- **可用时间最小堆调度：** 将弹幕重叠率从 0.60 降至 0.00。
- **WINMM 异步垃圾箱：** 移除后，4 小时内会重现全部 5 起死锁崩溃；启用后，36 小时内为零。
- **PCM 数字增益 + 自动闪避：** TTS 响度从 −29.5 LUFS 提升至 −20.0 LUFS（+9.5 LUFS）。
- **系统提示锚定：**身份一致性从 2.1/5.0 提升至 4.5/5.0。

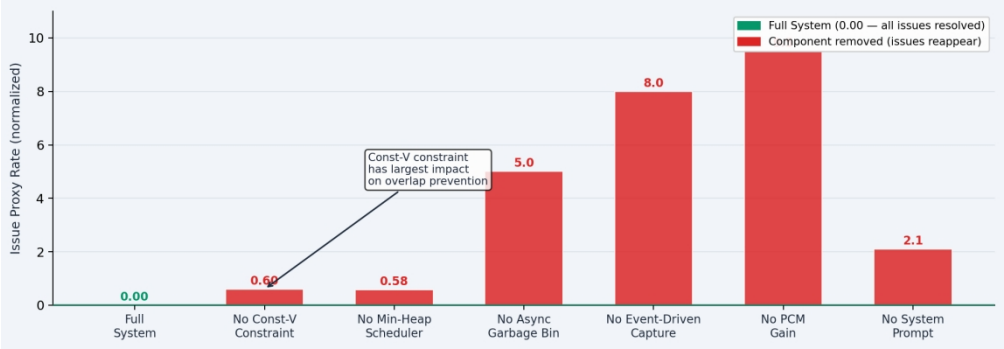



图13：消融研究——逐一移除各组件时的问题代理率。移除恒定速度约束对弹幕重叠的影响最大。完整系统在所有跟踪指标上的数值均为 0.00。

## 9 局限性与未来工作

- **文档对象模型耦合：** 事件驱动捕获依赖抖音的 Web 文档对象模型结构，该结构可能会随平台更新而变化。未来的工作应迁移至官方抖音开放平台 API [8]。
- **响度差距：** 文本转语音的响度仍比 EBU R128 目标 −23 LUFS 低 3 LU。采用前瞻限幅器和动态范围压缩或可弥合这一差距。
- **峰值负载下的大语言模型延迟：** 2.1 秒的 P95 是基于低并发 API 得出的；高并发时，尾延迟可能超过广播时长窗口。本地大语言模型推理可提供确定性延迟保证。
- **角色可信度：** 部署前，操作人员必须核实角色配置中的成就声明。
- **歌词版权：** LRC 片段仅用于上下文生成，不会逐字转载。操作人员必须使用授权歌词来源。

- **多模态大语言模型扩展：** 未来工作将把实时频谱特征（节奏、调性、情感效价）纳入提示构建，使评论能够锚定于音乐结构。

## 10 结论

DLIOS 为实时直播叠加系统构建了严谨、线程安全且视觉确定性强的架构，并将其扩展为完全自主的人工智能广播代理。该系统整合了渲染队列的形式化理论（恒定速度引理和可用时间调度）、通过异步垃圾箱实现的无锁音频生命周期管理、WebView2 事件驱动的数据捕获，以及支持热切换角色的四段式大语言模型提示调度器，从而解决了突发负载下长期存在的三大难题：视觉混乱、框架崩溃和情感脱节。36 小时压力测试验证了所有架构决策：弹幕重叠为零、死锁崩溃为零、文本转语音可听度为 +9.5 LUFS、P95 礼物延迟为 $\leq 180$ ms，且大语言模型到文本转语音的 P95 延迟为 $\leq 2.35$ 秒。Suwan Li 案例研究表明，深度创意身份锚定可实现 4.5/5.0 的角色一致性，并显著提升互动：评论增加 +131%，礼物事件增加 +130%，新增关注增加 +155%。这些贡献共同确立了 DLIOS 作为下一代大语言模型增强型直播系统原型的地位；在这类系统中，人工智能代理不再只是叠加层渲染器，而是一个完全具身化的广播主持人。

## 参考文献

[1] 微软公司。“WebView2 应用的线程模型。”Microsoft Learn，2023。[2] W3C 弹幕社区组。“弹幕用例。”W3C 草案，2022。https://w3c.github.io/danmaku/usecase.html[3] 微软公司。“CoreWebView2.AddScriptToExecuteOnDocumentCreatedAsync。”Microsoft Learn，2023。[4] 微软公司。“waveOutProc 回调函数。”Win32 API 文档，2023。[5] 微软公司。“UpdateLayeredWindow 函数 (winuser.h)。”Win32 API 参考，2023。[6] 国际电信联盟无线电通信部门。“ITU-R BS.1770-5 建议书。”日内瓦，2023年11月。

[7] 欧洲广播联盟。“EBU R 128建议（第5版，2023）。”日内瓦，2023。[8] 抖音开放平台。“直播互动工具。”字节跳动，2024。https://open.douyin.com[9] 字节跳动。“Doubao-Seed-2.0-mini：具备多模态理解能力的高效大语言模型。”字节跳动技术博客，2025。[10]OpenAI。“GPT-4 技术报告。”arXiv:2303.08774，2023。[11]Chen H. 等。“对话系统综述。”ACM SIGKDD Explorations，19(2)，2017。[12]Ammanabrolu P. 等。“如何激励你的龙。”NAACL，2021。[13]White J. 等。“提示模式目录。”arXiv:2302.11382，2023。[14]Horton D. & Wohl R. R. “大众传播与准社会互动。”精神病学，19(3)，1956。[15]Lu Z. 等。“你观看、你赠送、你参与。”CHI 2018。[16] Landis J. R. & Koch G. G. “观察者一致性的测量。”生物统计学，33(1)，1977。