在线教育场景下高并发低延迟直播技术实践

分享人：周赵鹏

自我介绍

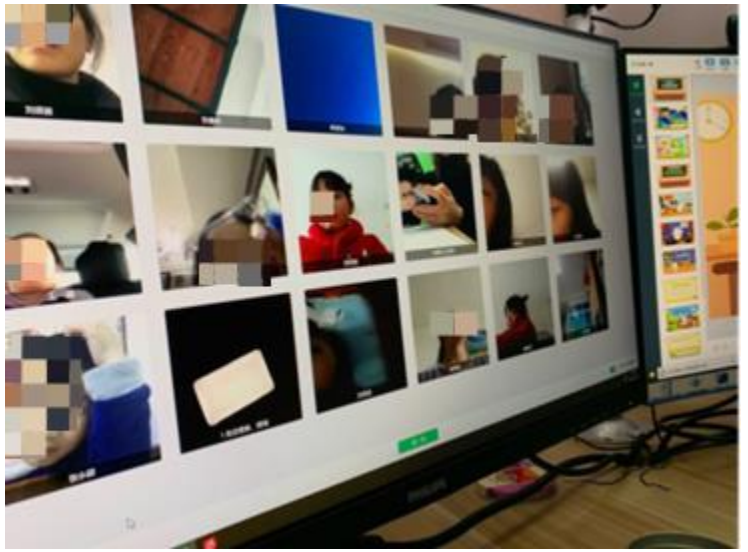
- 毕业后加入百度，在百度知道从事相关业务、基础架构的研发
- 目前在作业帮，主要负责：
  - 长连接&实时消息分发相关的技术研发
  - 大直播、实时音视频相关的技术研发

内容大纲

- 从业务场景谈起
  - 在线教育业务场景分类
  - 技术模型抽象
- 高并发低延迟直播架构核心设计思路
  - 核心技术要点
  - RTC总体技术架构
  - 全局智能调度
  - 分布式IDC
- 精选3个要点剖析
  - RTC协议简化
  - 多核分发模型
  - 上行推流优化
- 总结

在线教育业务场景

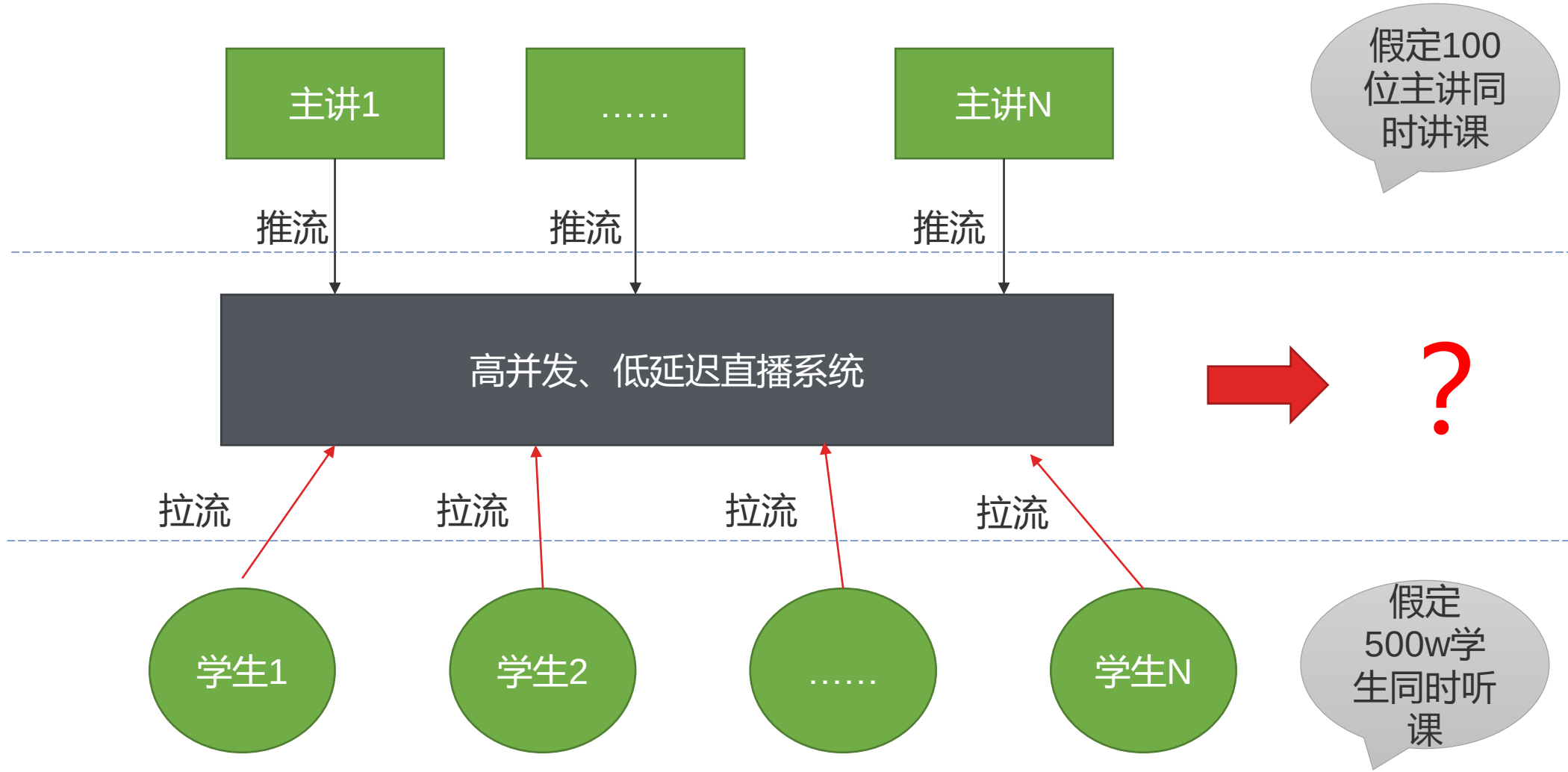
- 1v1辅导业务
  - 1v1双向互动
- 大班课
  - 1v多（允许一定延迟）+ 1v1连麦
- 互动大班课
  - 1v多（低延迟）+ 1v1连麦
  - 学生实时画面
- 超级小班课
  - 1v多（低延迟）+ 6v6小组互动
  - 学生实时画面
- 互动小班课
  - 6v6小组互动



技术抽象

- 1v1实时音视频技术
- 1v多直播技术（不同的延迟）
  - 高延迟，rtmp、hls
  - 低延迟，rtc
- 6v6多人实时音视频技术
  - 类视频会议
- 各类业务场景，无非就是这3大类技术的组合
- 根据具体的业务场景，选择合适的技术模型

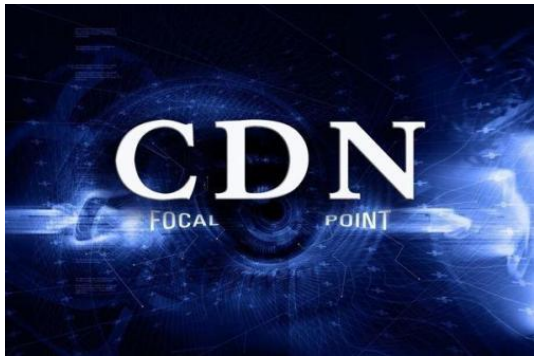
高并发低延迟直播技术


主讲1
……
主讲N
假定100位主讲同时讲课
推流
推流
推流
高并发、低延迟直播系统
?
拉流
拉流
拉流
拉流
学生1
学生2
……
学生N
假定500w学生同时听课

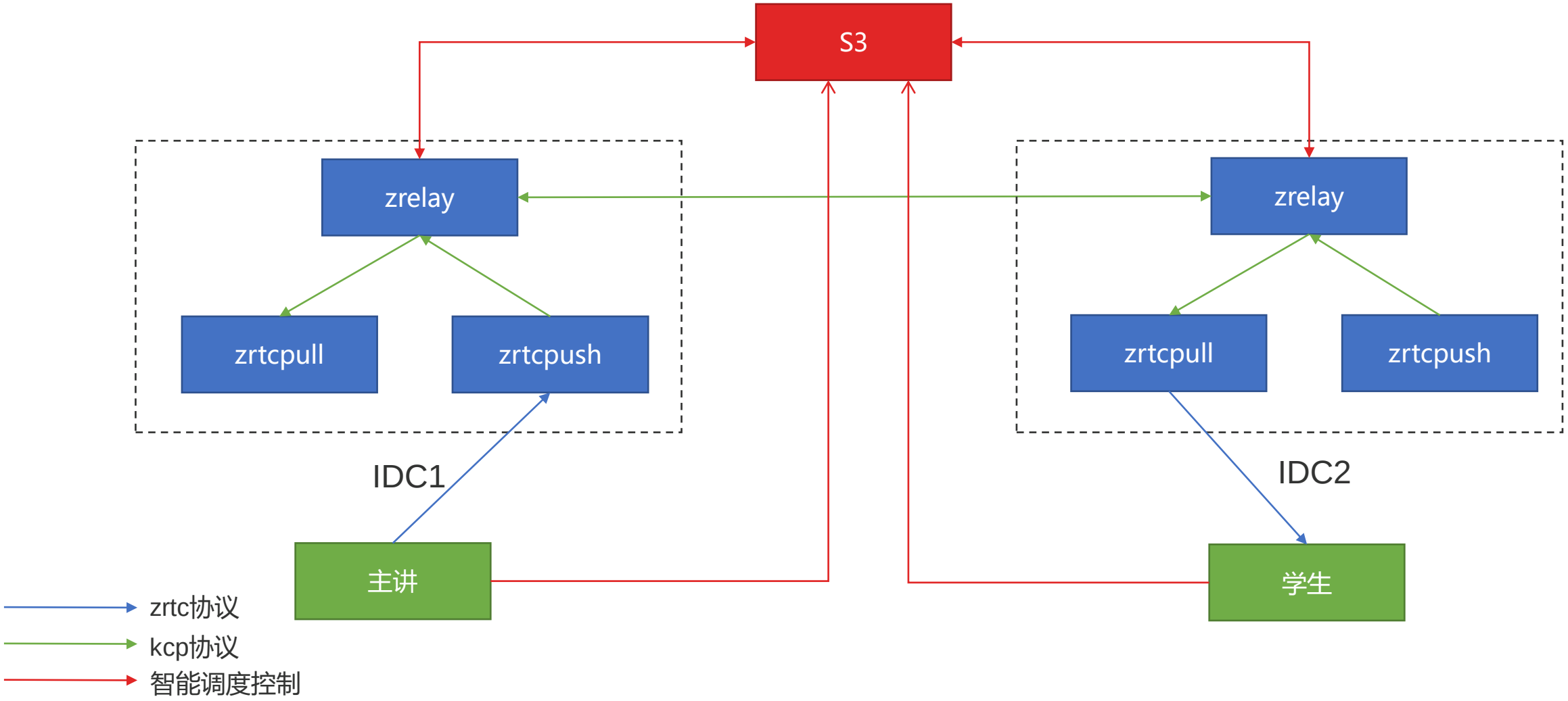
技术要点

- 低延迟
  - 传输层，UDP协议
  - 应用层
    - rtp/rtcp协议
    - 自定义协议
  - 直播全链路
  - buffer的控制
    - neteq
- 高并发
  - 单机性能要高
  - 集群、分布式IDC
  - 分层树状架构
- 要兼顾通信质量和成本
- 参考WebRTC技术

WebRTC



技术架构


S3
zrelay
zrelay
zrtcpull
zrtcpush
zrtcpull
zrtcpush
IDC1
IDC2
主讲
学生
zrtc协议
kcp协议
智能调度控制

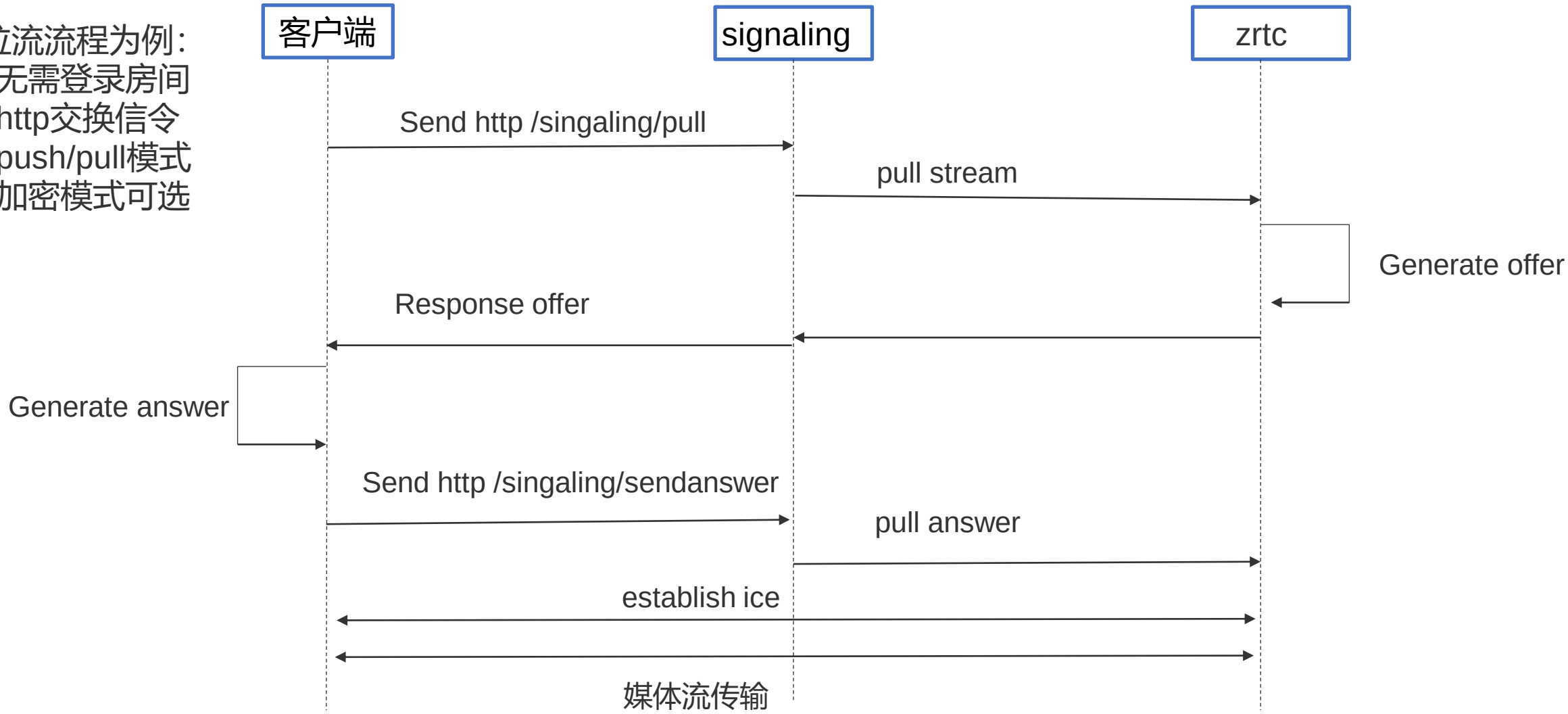
核心模块

- S3，全局智能调度系统
  - 就近接入，选择最佳路由
  - 媒体流全局管理
  - CPU、网络负载控制
- 分布式IDC
  - zrelay，负责IDC内部之间的rtc流中继分发
  - zrtcpush，就近接收用户的推流
  - zrtcpull，就近接收用户的拉流
  - 推流和拉流之所以分开，主要考虑保护源站，防止大并发的拉流影响推流

协议简化

以拉流流程为例：

1. 无需登录房间
2. http交换信令
3. push/pull模式
4. 加密模式可选



协议简化

边缘分发协议

1. http交换信令
2. ICE建立传输通道
3. RTP/RTCP传输媒体数据

没有房间管理，更加轻量级

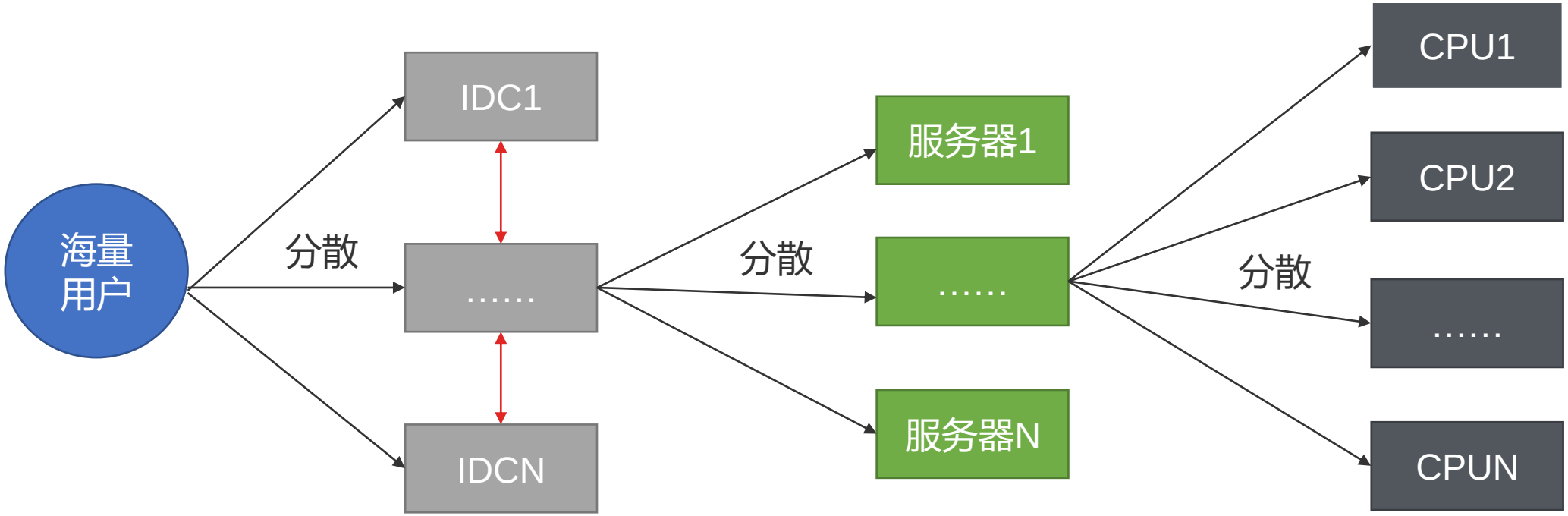
中继分发协议

1. 快速的传输协议，可降低延迟30%~40%
2. 可靠的传输，不用考虑中继转发的丢包问题

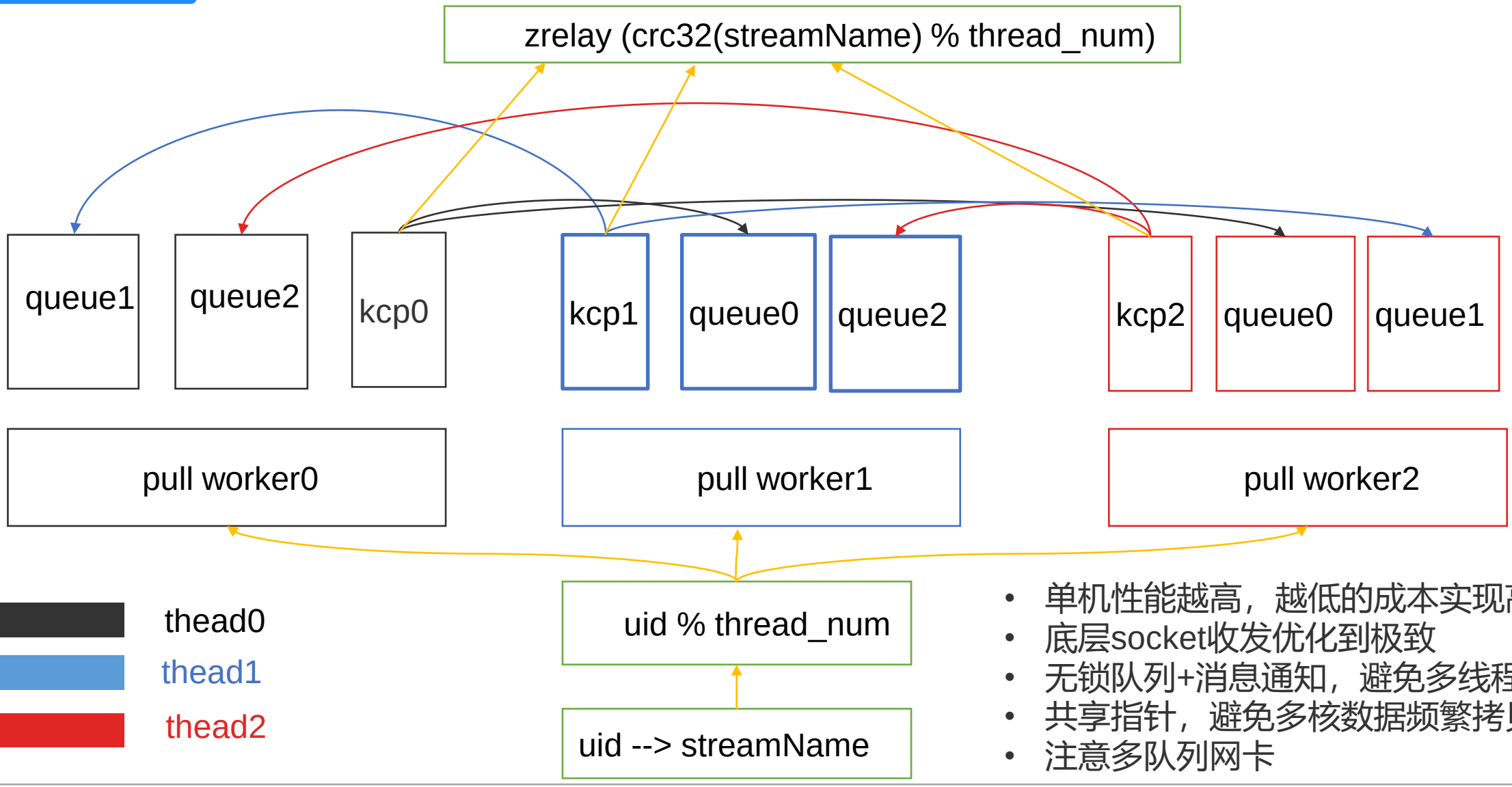
系统越轻量级越容易做到高并发，环节和流程越复杂，按照木桶理论，瓶颈点会更多

如何处理高并发的用户

- 按照地域将用户分散到不同IDC机房
- 同一机房，按照负载分散到不同的服务器
- 同一服务器，按照策略分散到不同的CPU核



单机高性能-多核分发模型



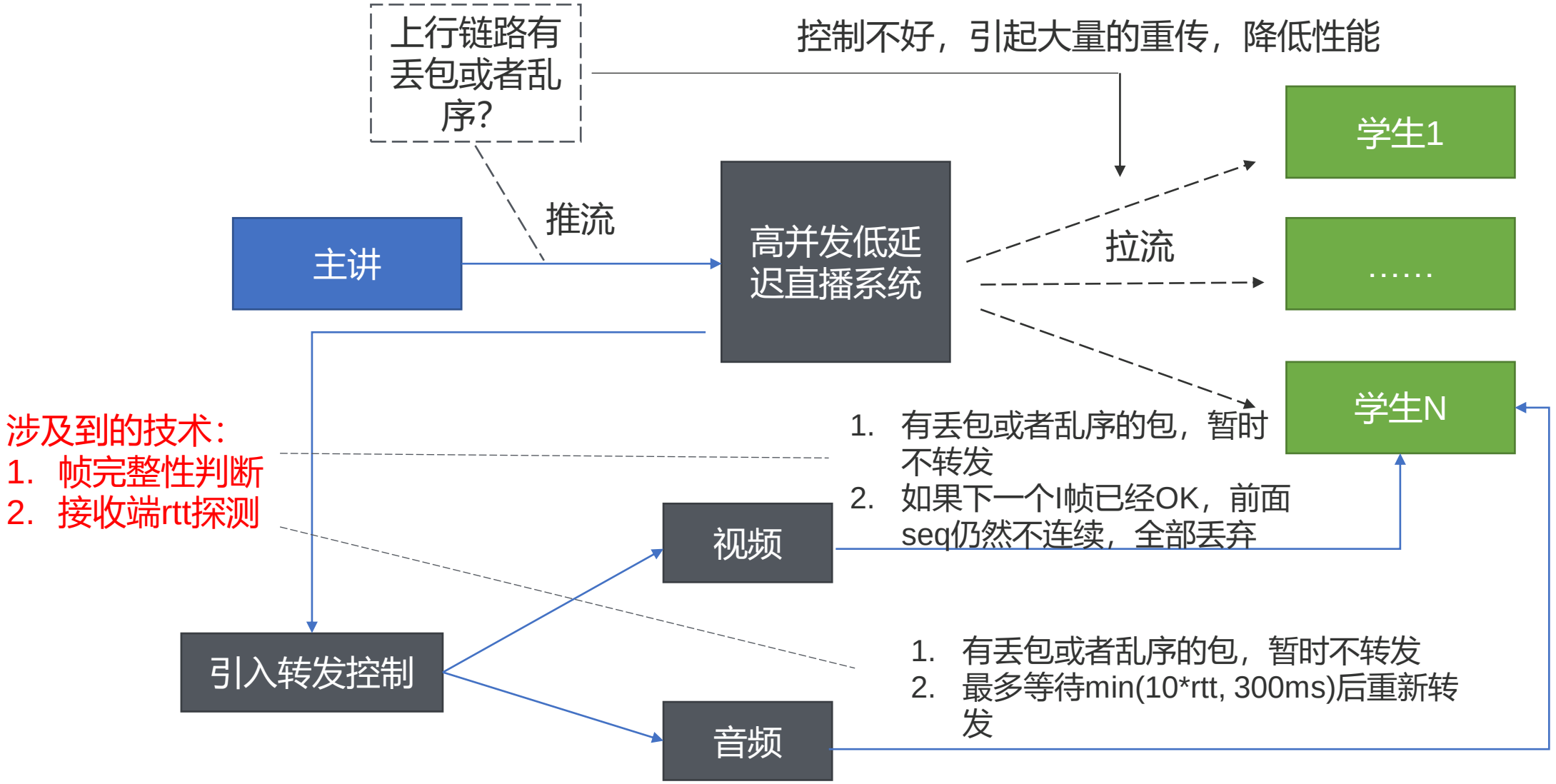
- 单机性能越高，越低的成本实现高并发
- 底层socket收发优化到极致
- 无锁队列+消息通知，避免多线程加锁
- 共享指针，避免多核数据频繁拷贝
- 注意多队列网卡

单机性能评测

- 2020.05.16线上真实的媒体流评测
  - 76路推流
  - 码率300 ~ 400kbps
- 单机配置
  - Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz
  - 32核心、128G内存、2Gbps外网带宽
- 单机稳定性能：4500路拉流

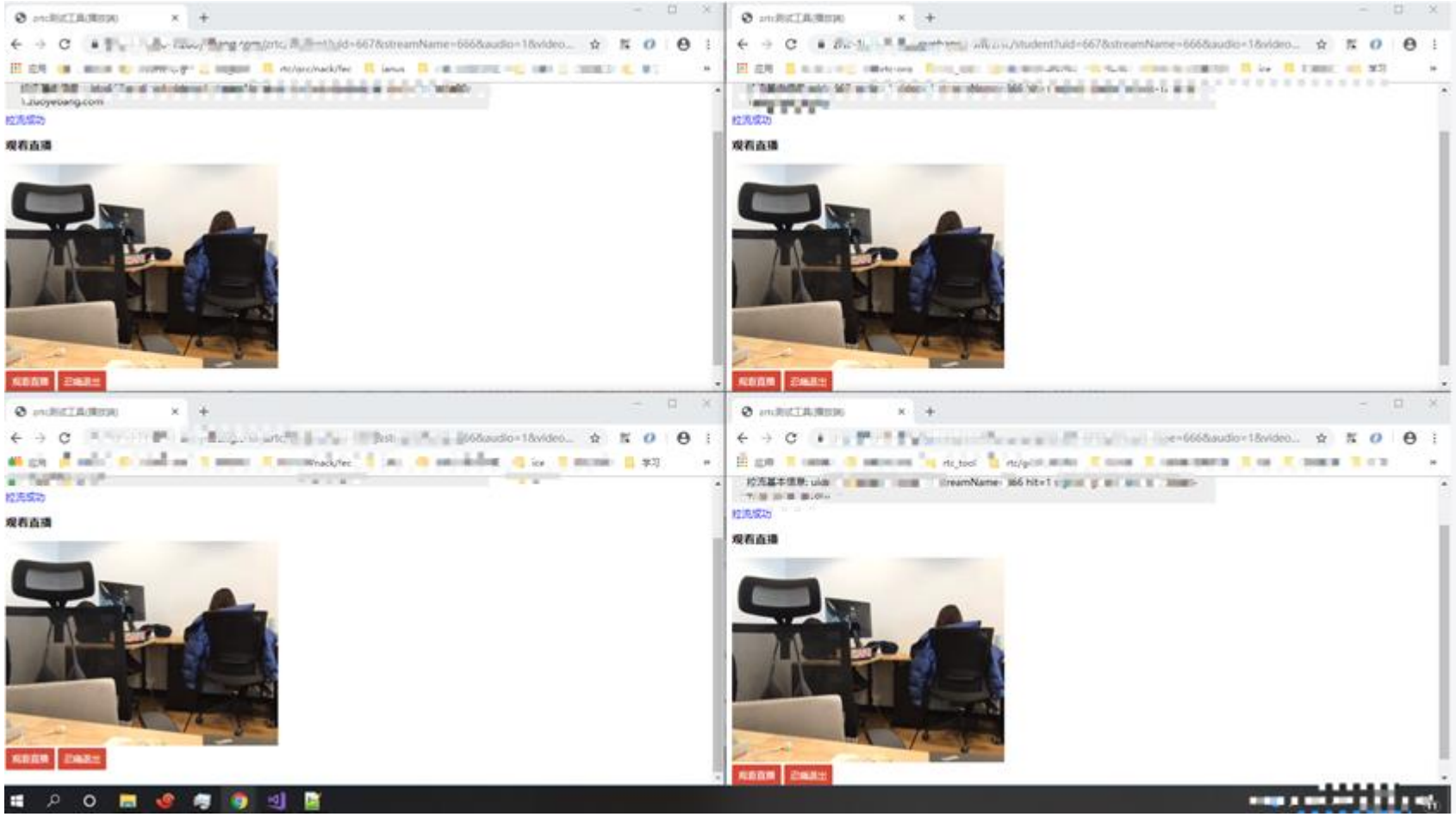
同时拉取流路数	CPU消耗	CPU idle	内存消耗	上行带宽消耗	下行带宽消耗
1140路（76 * 15）	.3.6核	86%	436.4M	34.5 ~ 48.6Mbps	408.6 ~ 429.4Mbps
2280路（76 * 30）	6.6核	75%	669.9M	36.0M ~ 50.8Mbps	805.0 ~ 856.7Mbps
4864路（76 * 64）	17.5核	38%	1.243G	43.5 ~ 61.1Mbps	1717.8 ~ 1837.4Mbps

上行推流优化


上行链路有丢包或者乱序?
控制不好，引起大量的重传，降低性能
推流
主讲
高并发低延迟直播系统
拉流
学生1
……
学生N
涉及到的技术:
1. 帧完整性判断
2. 接收端rtt探测
1. 有丢包或者乱序的包，暂时不转发
2. 如果下一个I帧已经OK，前面seq仍然不连续，全部丢弃
视频
引入转发控制
音频
1. 有丢包或者乱序的包，暂时不转发
2. 最多等待min(10*rtt, 300ms)后重新转发

4个IDC效果测试

- 连续测试24小时，4个IDC未断流、延迟没增大、音视频完全同步



总结

- 根据不同的业务场景，选择合适的架构模型
- 高并发
  - 如无必要，架构尽可能设计轻量级，越重的设计瓶颈点更多
    - 轻量级的协议和流程
  - 单机尽可能高性能
    - 底层socket收发优化到极致
    - 多核分发
    - 利用无锁队列 + 消息通知避免多线程加锁
    - 利用共享指针，降低频繁的数据拷贝
    - 注意多队列网卡
  - 注重上行推流优化，1v多场景，上行不稳会放大下行的问题
- 低延迟
  - UDP协议
  - IDC内部中继分发采用KCP，可降低延迟
  - 全链路优化，buffer合理控制，可参考WebRTC neteq动态自适应抖动缓冲，降低延迟

最后

优秀的系统，绝非一日之功，需要我们保持耐心，以工匠精神，持续的打磨和精进，才能趋于至善！

THANKS