2020
北京

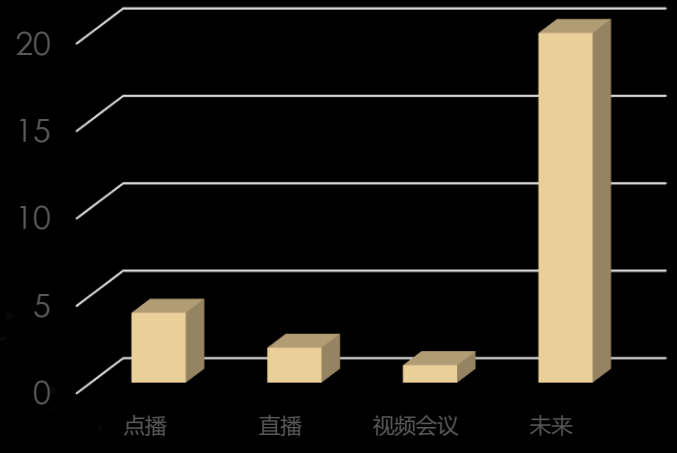
多媒体开启新视界
MULTIMEDIA BRIDGE
TO A WORLD OF VISION

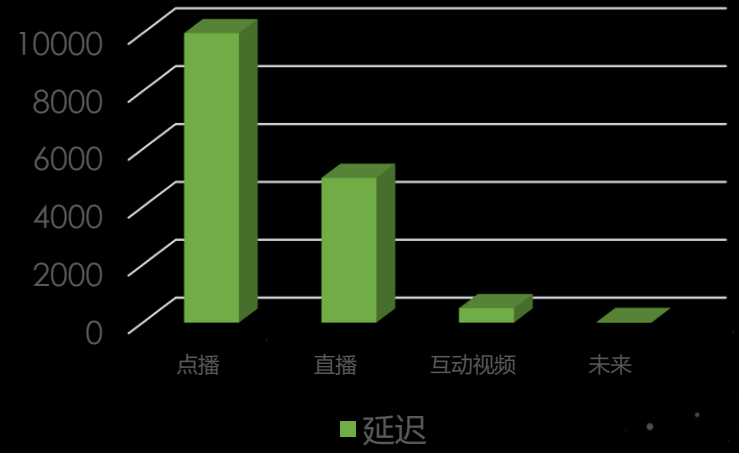
超低延迟实时流媒体传输技术

刘泓昊

网络基础设施的进步正在驱动新的应用场景

从依赖buffer抗抖动变成想办法让他不抖动





零缓冲　超低延迟　大带宽

$$Delay = RTT/2 + \lim_{x \to 0+}$$

协议设计的关键点

2020
北京

- 可靠性
- 流控算法
- 网络传输协议

关于可靠传输机制

观点1： 应用层丢包是应该尽量避免的

观点2： 类FEC和重传

观点3： sack和nack的选择

类FEC和重传的关系

类FEC:
降低丢包对delay的影响
降低码率
CPU利用率高
冗余不够怎么办?

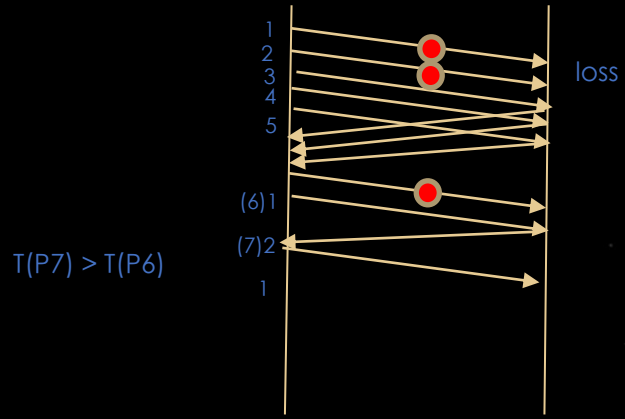
重传:
丢包场景下会增加delay
带宽利用率高

FEC和重传是可以融合的;
即使用FEC，也需要具备重传能力;
低rtt下带宽优先;
高rtt下冗余优先;
先验重传冗余是可能的;

nack和sack的选择

NACK:
首次丢包判断逻辑简单
ACK丢包敏感
重传包丢包判断逻辑不闭环

SACK:
判断逻辑复杂
ACK不丢包敏感
丢包判断更准确，更实时



关于流控

2020
北京

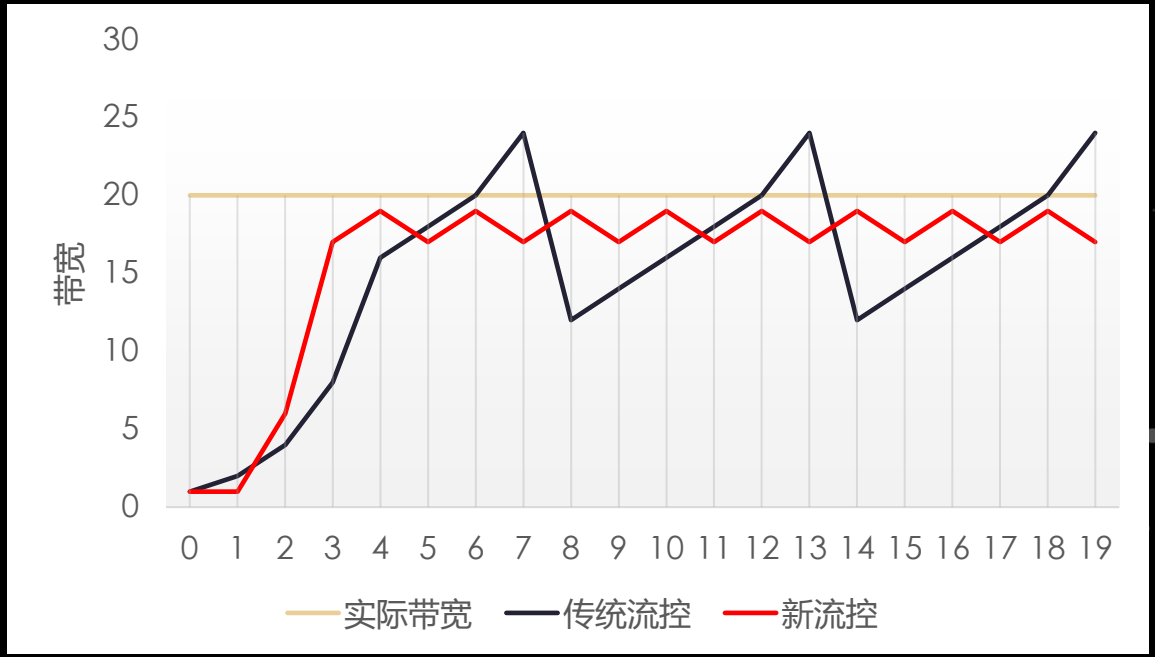
观点1： 需要新的目标模型

观点2： 采集的周期和精度很关键

观点3： 核心是吞吐

流控的新目标

不超出网络瓶颈带宽的条件下，尽可能的充分利用网络带宽



关于采集

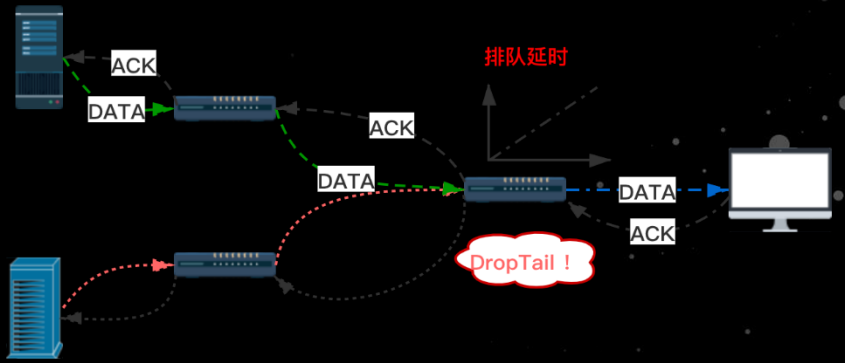
- 帧粒度
- 发端采集为主，收端采集为辅
- 没有数据，也是数据

流控算法

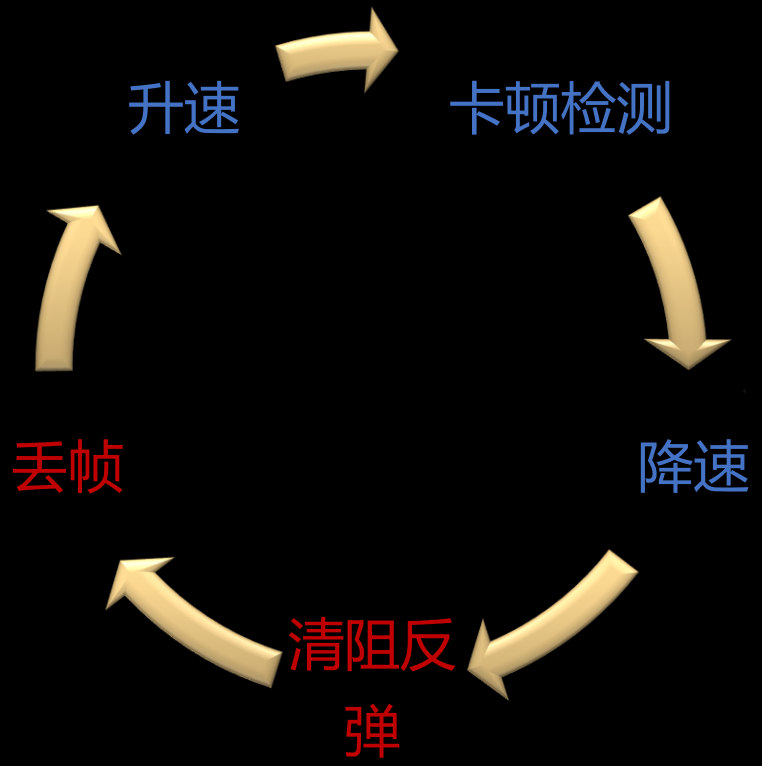
基于buffer：
Buffer受应用的影响，容易产生扰动；
Buffer产生达到阈值就已经晚了

基于丢包：
90%的网络卡顿首先是产生延迟抖动，然后丢包

If （（rate，drop，queue，app_size））
Rate（n） = M * f（rate(n-1)，rate(n-2)，rate(n-3),rate(n-4)……）
Else
Rate（n） = f（rate(n-1)，rate(n-2)，rate(n-3),rate(n-4)……） + AI



流控流程



关于网络传输协议

UDP VS TCP

UDP 比 TCP 更灵活

UDP可以用FEC， TCP不能

客户端上改不了内核代码

内核代码好难改

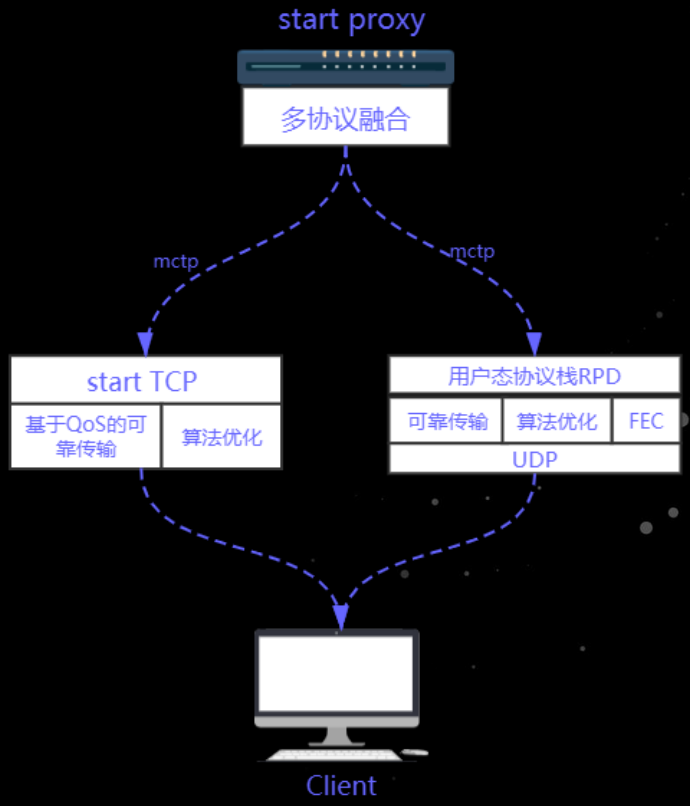
但是
在高码率下 UDP的丢包远高于TCP

关于网络传输协议

使用原则：
- 低延迟用TCP
- 高延迟用UDP
- 高码率用TCP
- 低码率用UDP

应用层流控

可靠传输
PACING
去拥塞控制



一切都是概率

在不可靠的无线链路上，用多链路实现高可靠

多链路是未来趋势

总结

- Sack是更好的重传发现机制
- 帧粒度
- 速率模型
- TCP和UDP混用
- 多链路

Thank you