WebRTC网络传输控制

李超

音视跳动首席架构师

大纲

- 实时通信的目标
- 存在哪些困难，或存在哪些矛盾点？
- WebRTC是如何解决这些矛盾的

一些思考

问题一：开会时是喜欢在一个办公室里开呢，还是更喜欢在线上开？

问题二：如果有一场演唱会，你愿意去现场呢？还是愿意在线上听？

为什么线上沟通与线下不同?

- 网络延迟过大
- 摄像头与人眼看到的效果不一样，如采集的角度过小
- 采集设备的质量参差不齐
- 现场的气氛无法被摄像头采集到

实时通信的目标

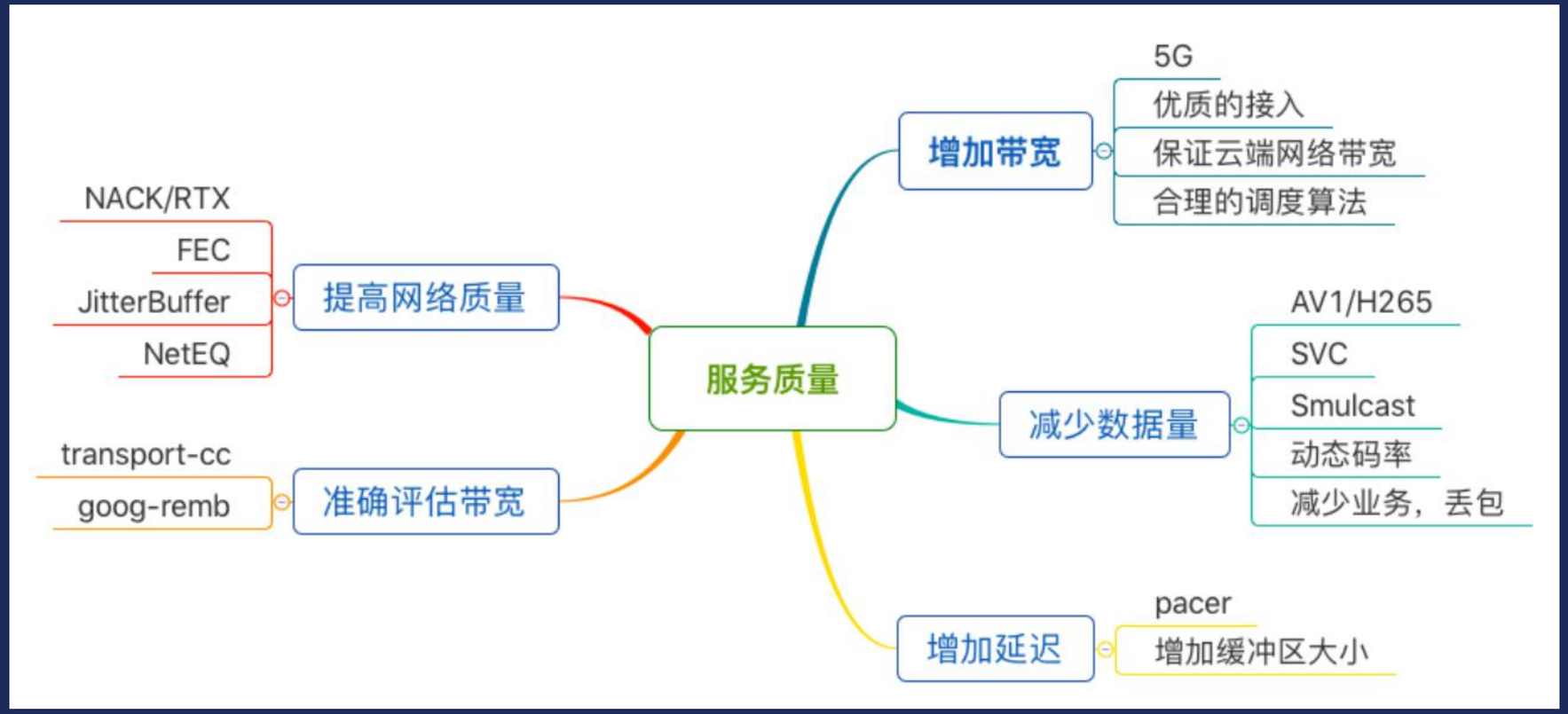
尽可能逼近或
达到面对面交流的效果

主要矛盾

主要矛盾

音视频的服务质量与带宽大小、网络质量、实时性之间的矛盾

WebRTC解决矛盾的方法



网络模块要解决的问题

- 实时性
- 抗丢包性
- 公平性
- 带宽评估的准确性和及时性

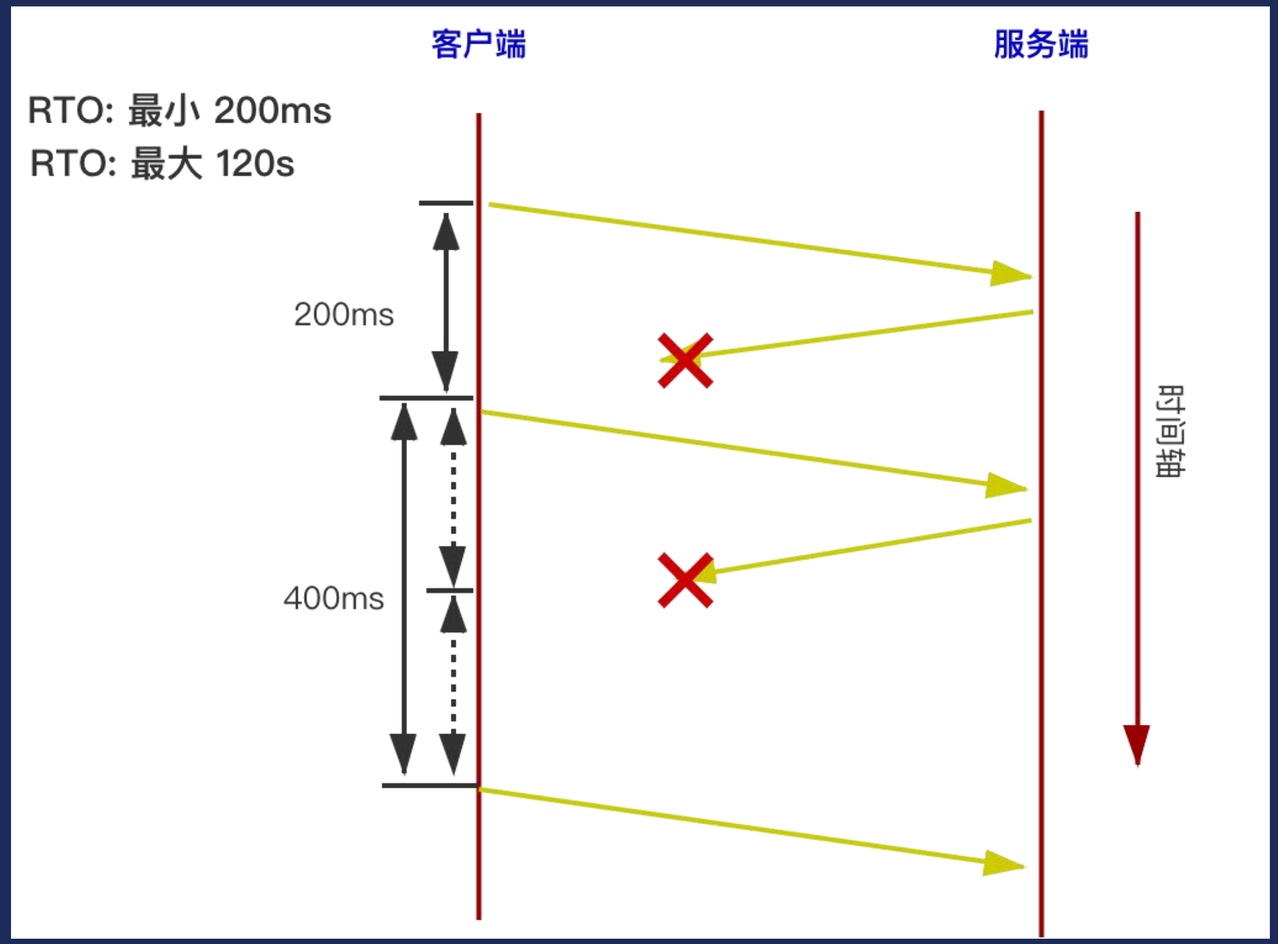
如何能做到实时性?

解决实时性的方法

- TCP/UDP？
- 通信双方是否可以走内网？
- P2P是否可以连通？
- 服务器中转 turn/sfu/mcu

为什么极端网络下不能用TCP?

- 发送->接收->确认

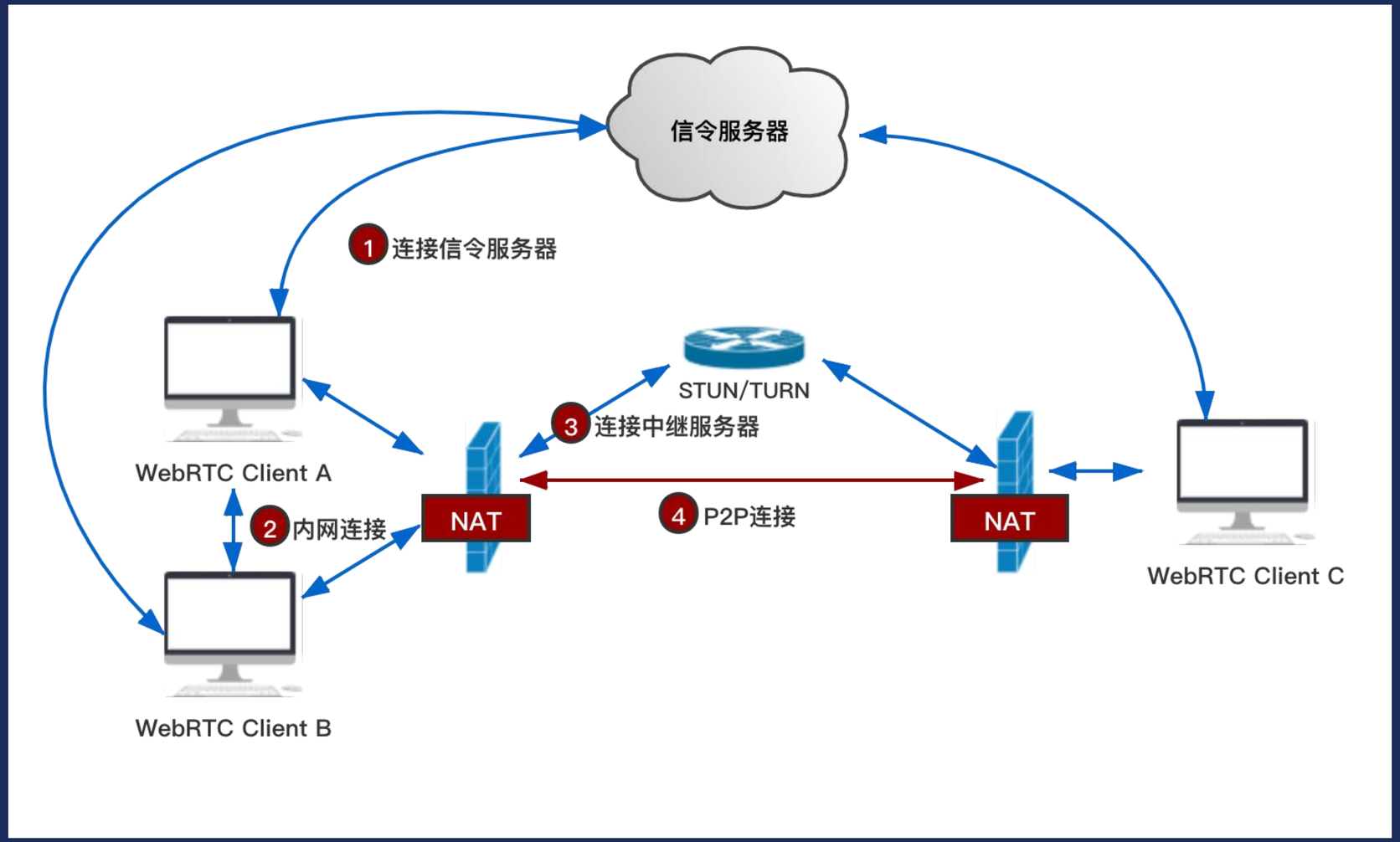


实时性的延时指标

表 3.1: 实时通信延迟指标

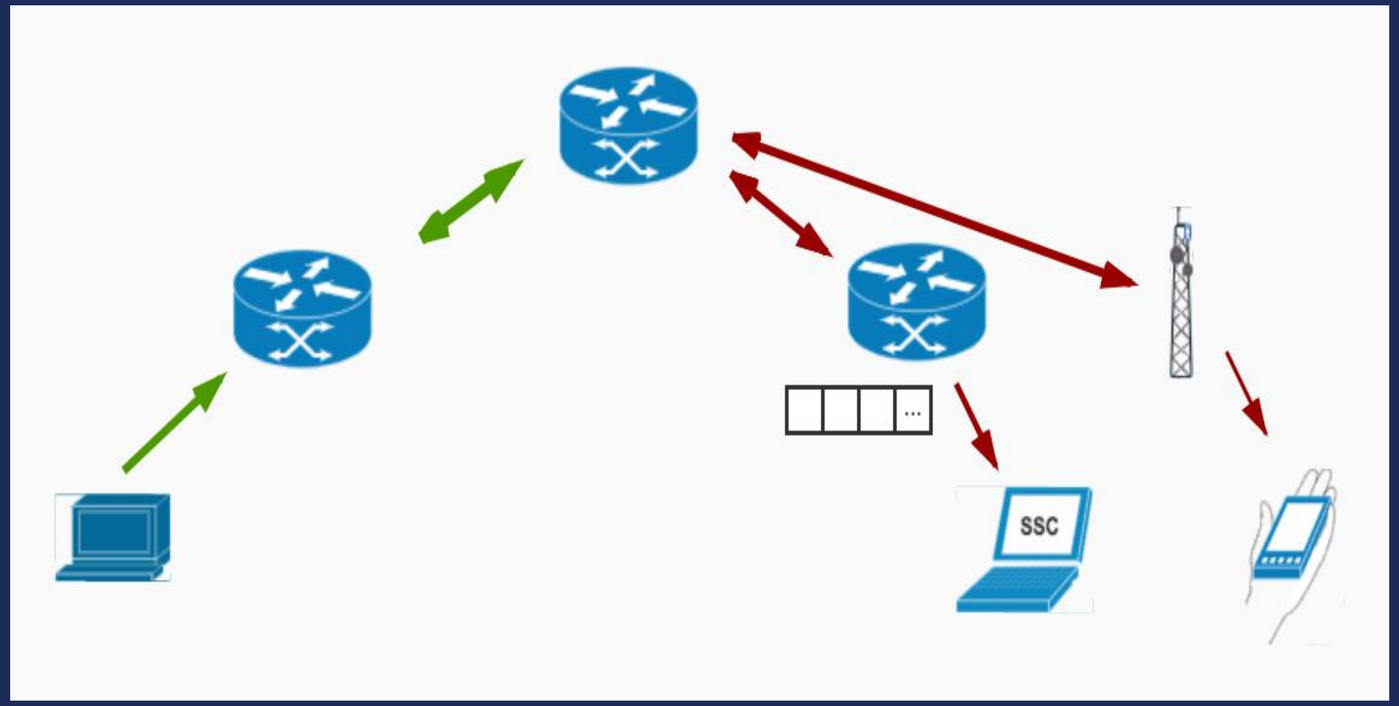
时延	人的感受
200 毫秒	非常优质，如同在一个房间里聊天的感觉
300 毫秒以内	大多数很满意
400 毫秒以内	有小部分人可以感觉到延迟，但还基本可能进行互动
500 毫秒以上	延迟明显，影响互动，大部分人不满意

传输路径的选择


信令服务器
1 连接信令服务器
STUN/TURN
3 连接中继服务器
WebRTC Client A
2 内网连接
NAT
4 P2P连接
NAT
WebRTC Client C
WebRTC Client B

WebRTC抗丢包

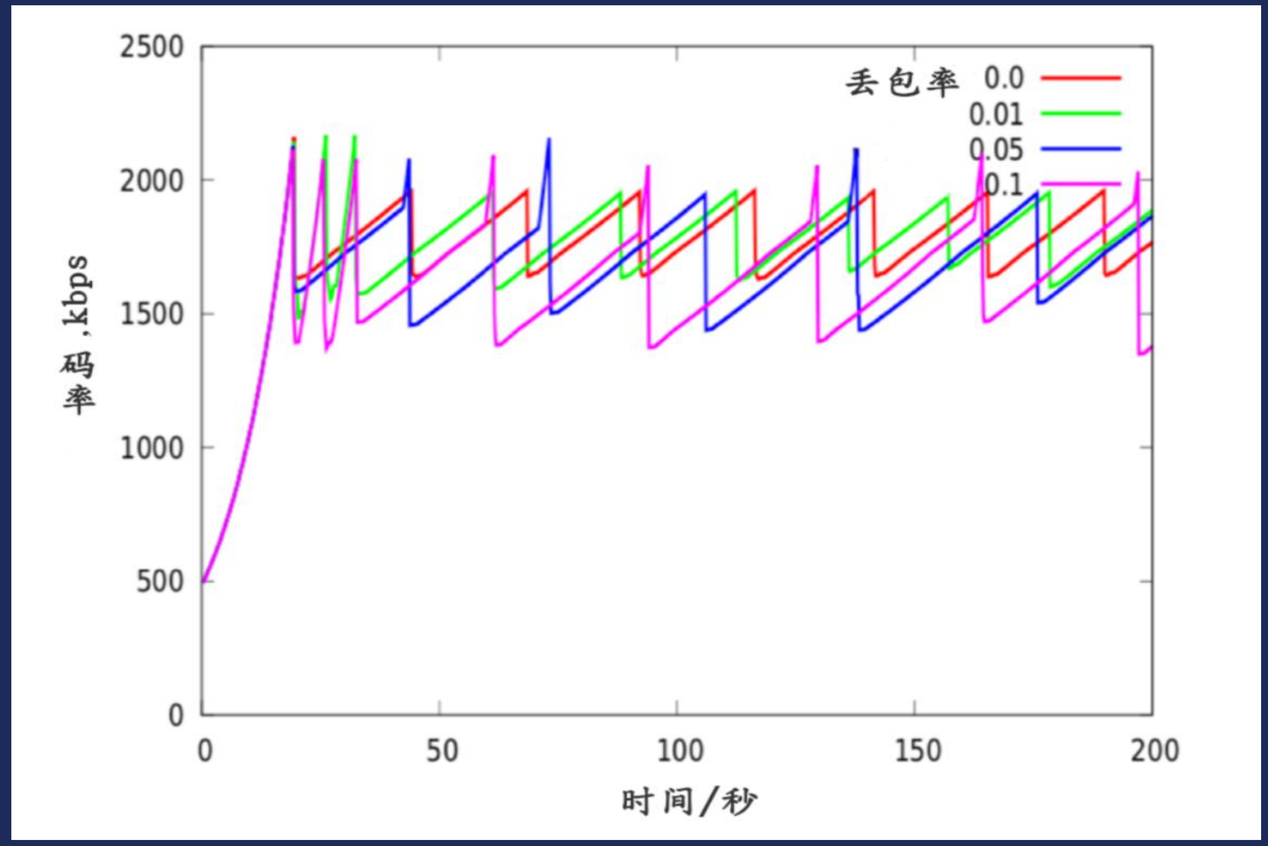
造成丢包的原因



- 链路质量差
- 带宽满
- 主动丢包
- 光纤被挖断
- …

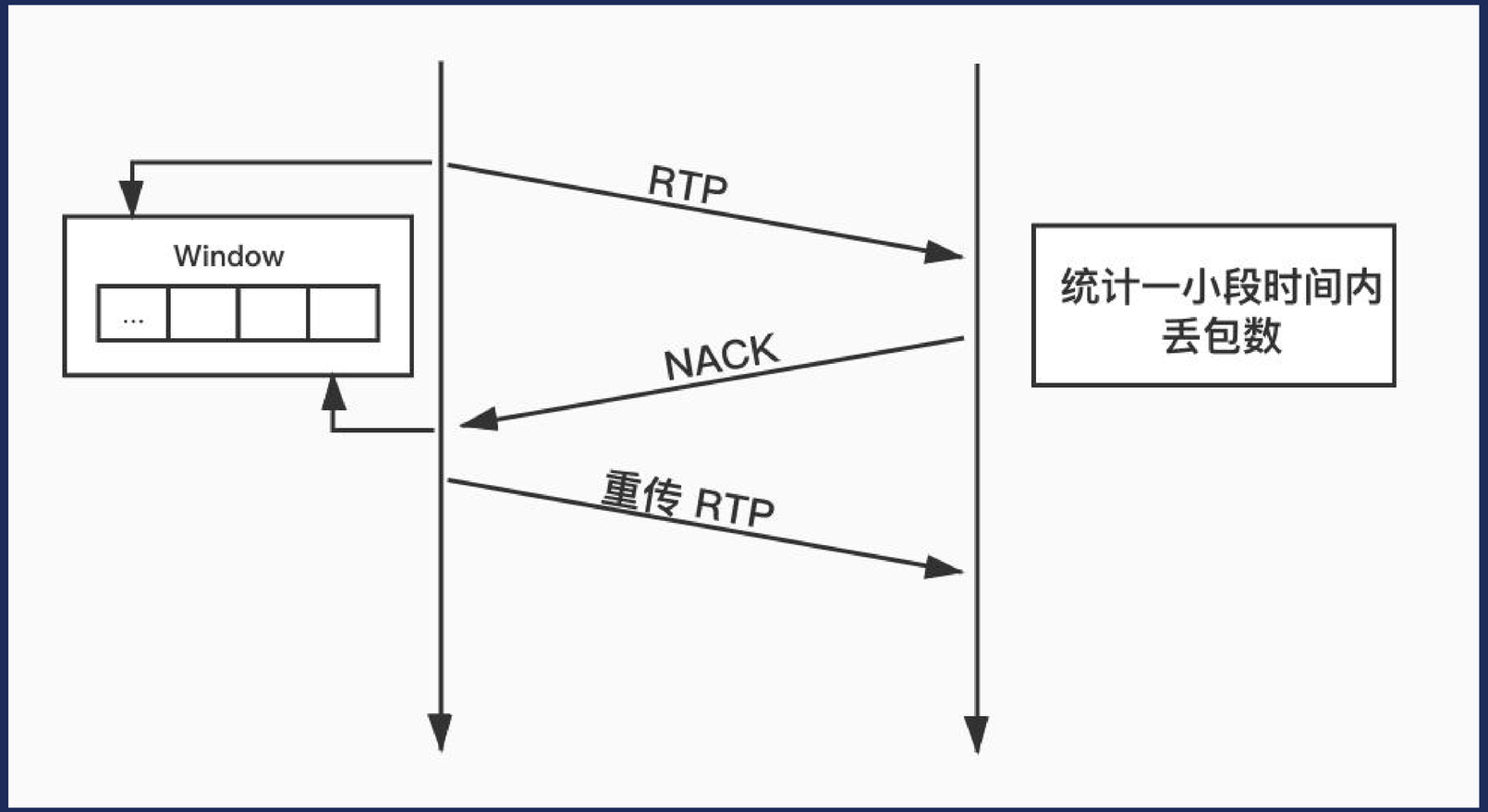
WebRTC抗丢包的方法

- 丢包重传（NACK），可以容忍10%的丢包
- FEC



50%以上丢包靠谱吗?

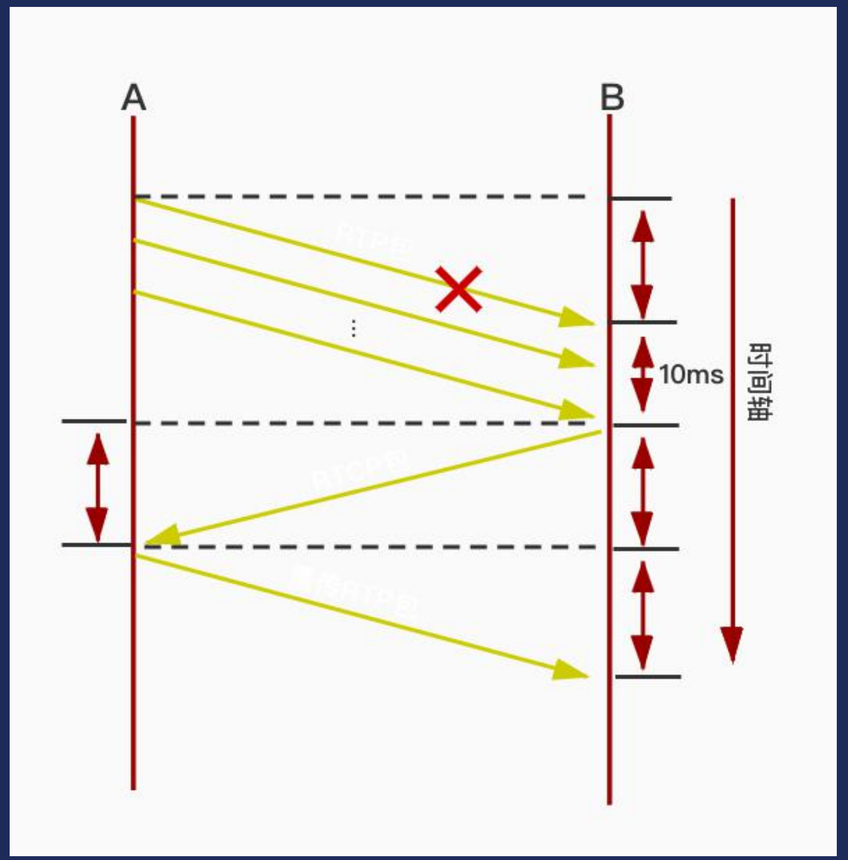
丢包重传



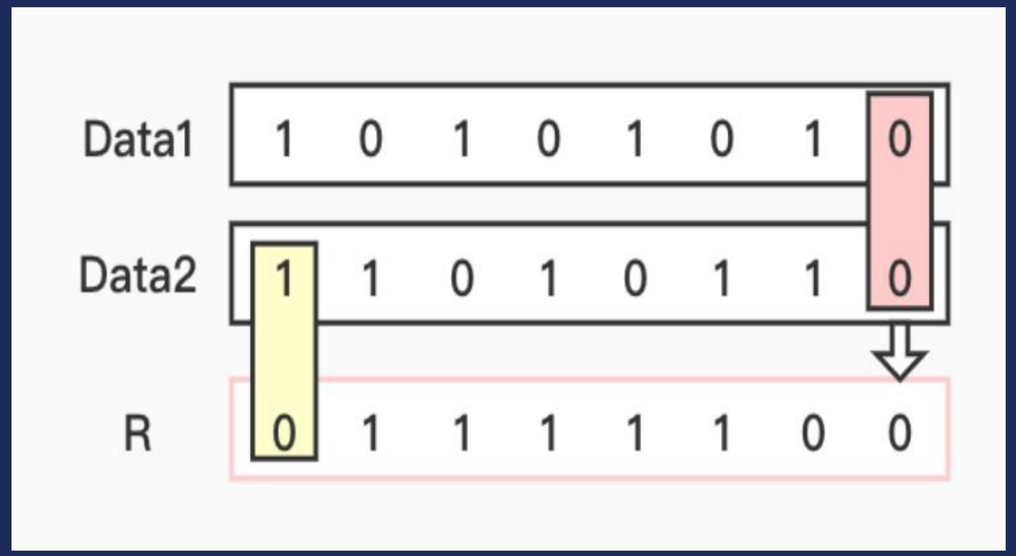
什么时候适合丢包重传?

- 重传用时为:
  1.5RTT+10ms

- 时延比较小的丢包适用于丢包重传

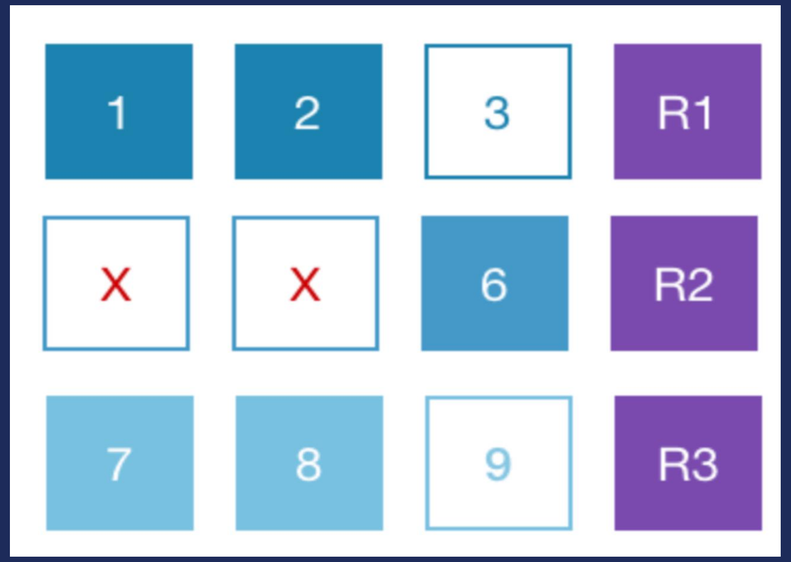


FEC异或操作



- A ^ B = C
- C ^ A = B
- C ^ B = A
- A ^ B ^ C = D
- ...

ULPFEC



- 1-9是正常的数据包
- Rn 是冗余包

FlexFEC

- 先横向进行纠错
- 再进行纵向纠错

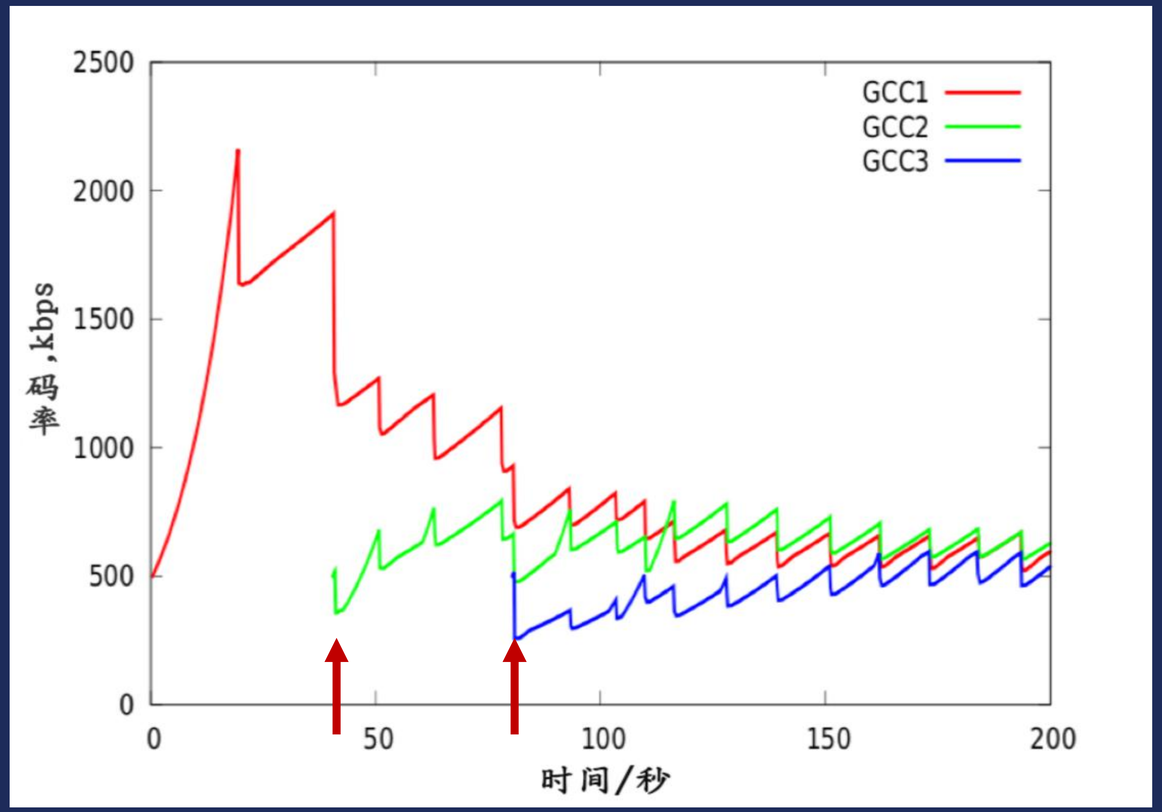
结论

- 丢包重传 NACK不适用于时延大的场景
- FEC不适用于连续丢包的场景
- 50%以上丢包是实验室环境，不是真实场景

网络传输的公平性

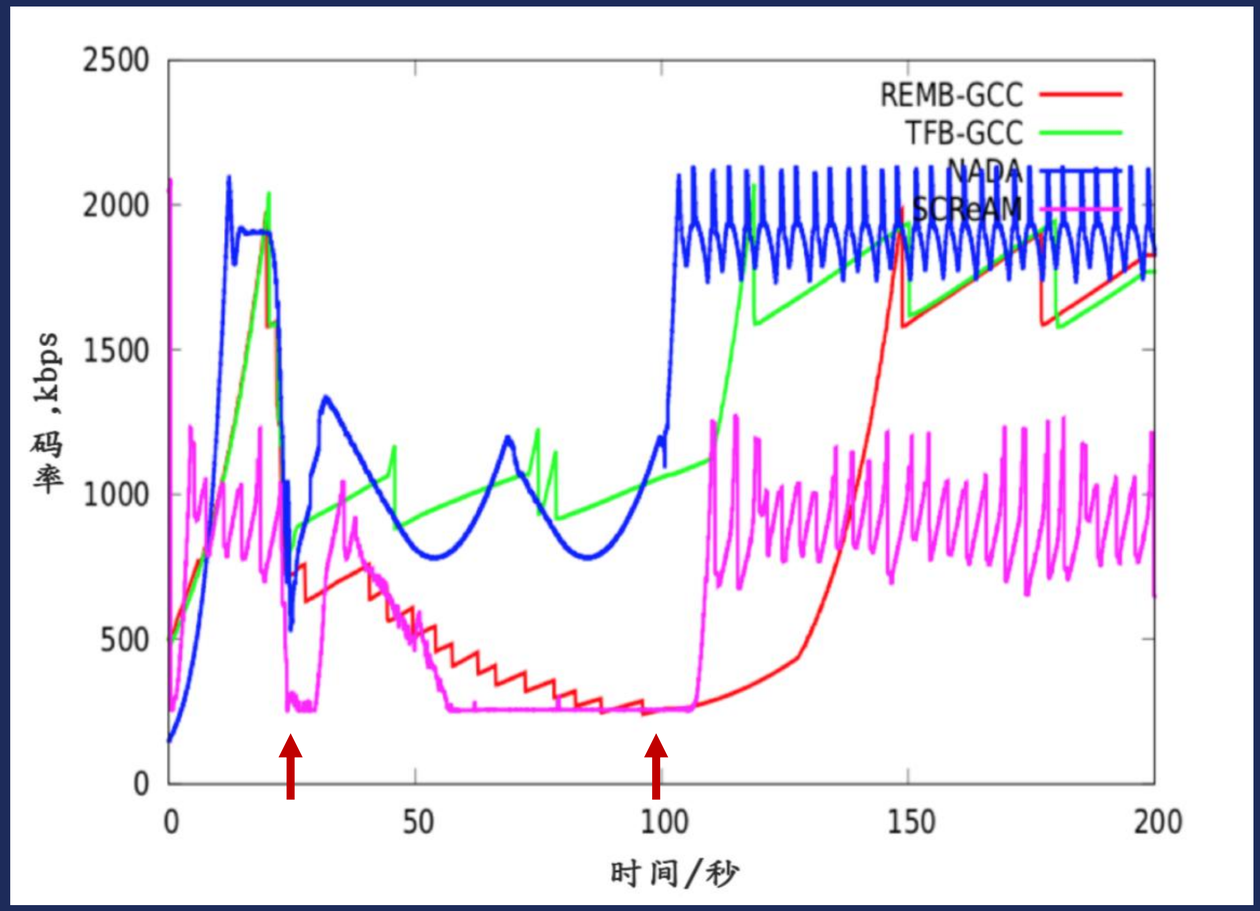
多个GCC连接的公平性

- 40ms加入第二个GCC连接
- 80ms加入第三个GCC连接



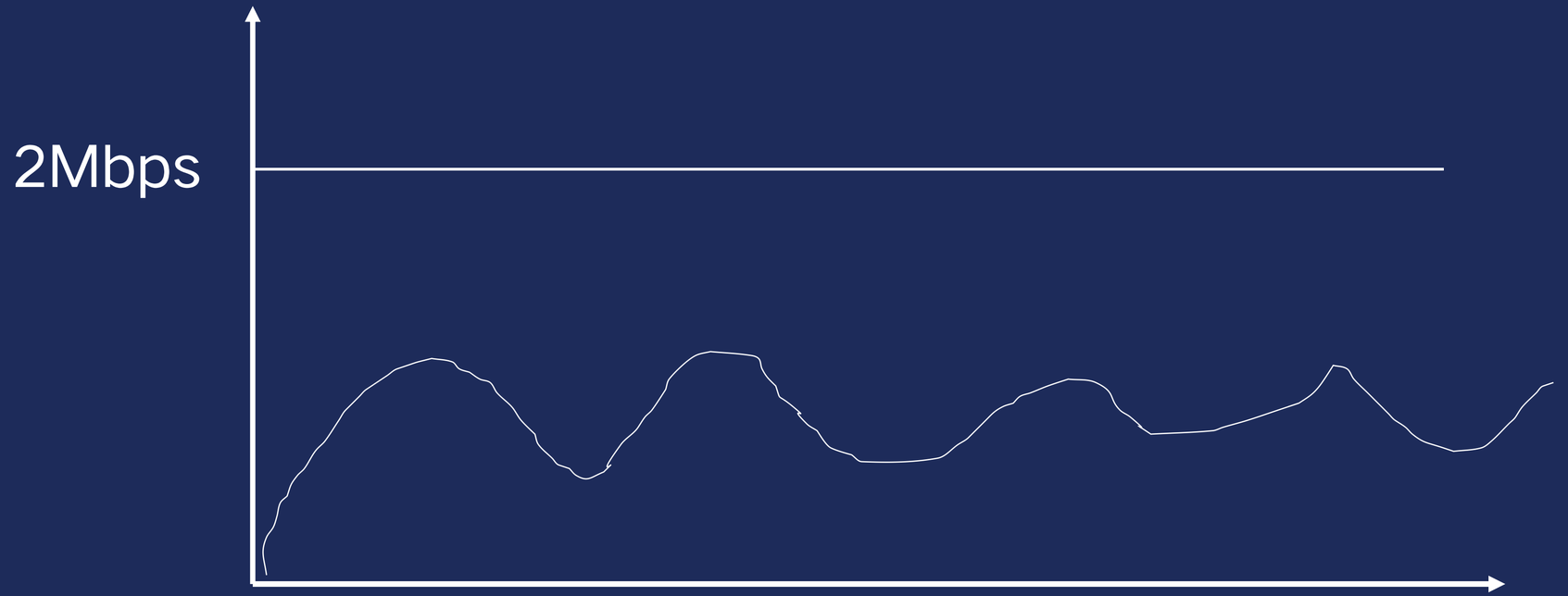
与TCP共存时的公平性

- 20ms~100ms加入TCP
- REMB-GCC和SCReAM表现不佳

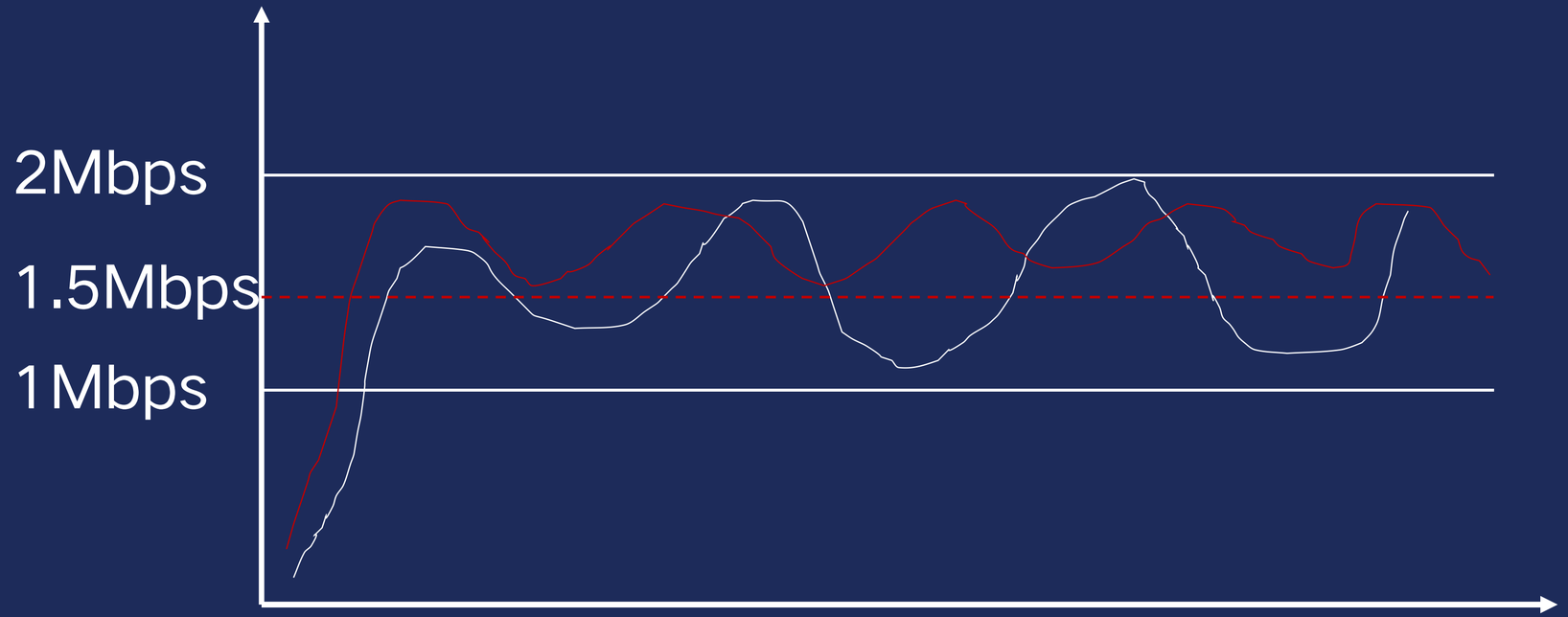


带宽评估的准确性

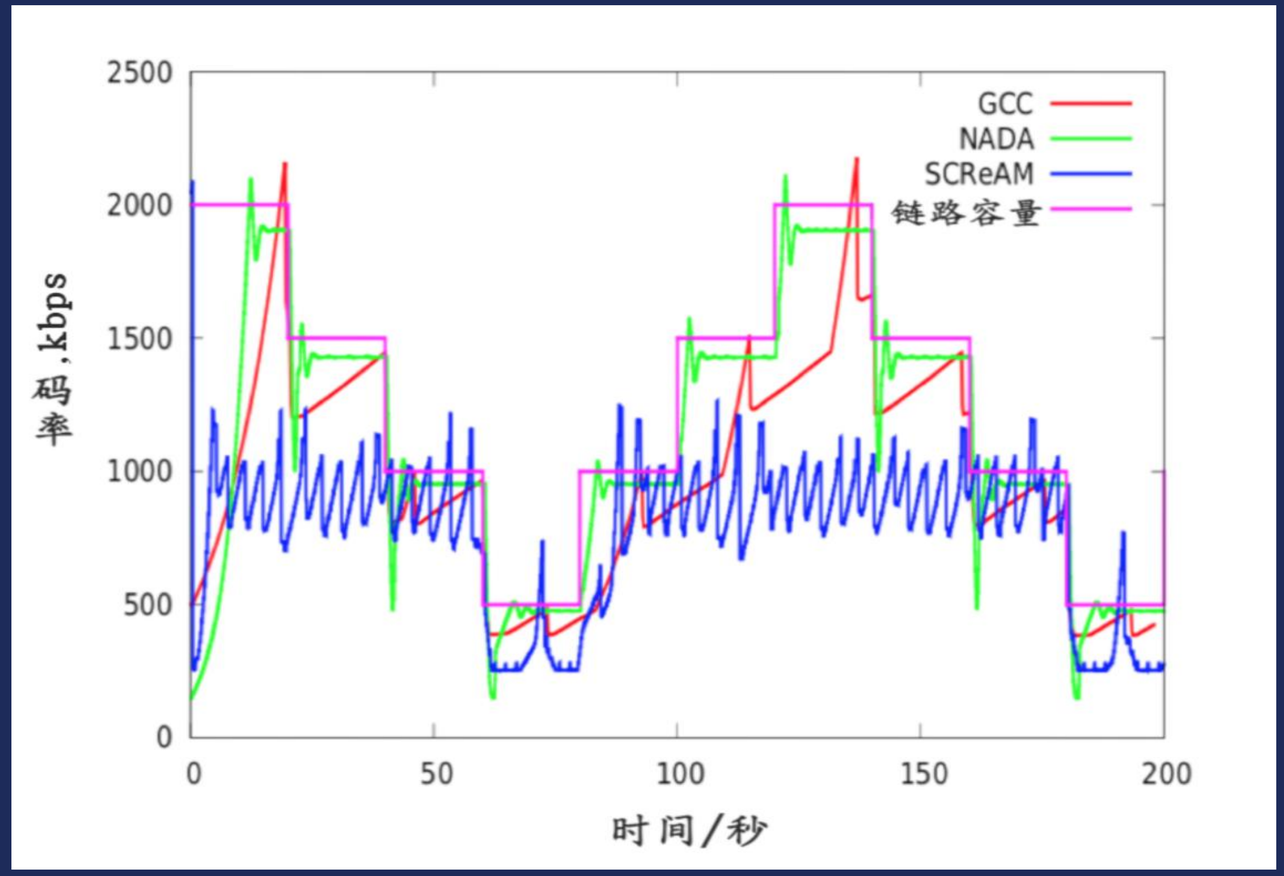
通常带宽的使用



更好的带宽的利用率



带宽评估的准确度



- 基于丢包的带宽评估
- 基于延时的带宽评估
  - Goog-REMB
  - Goog-TCC

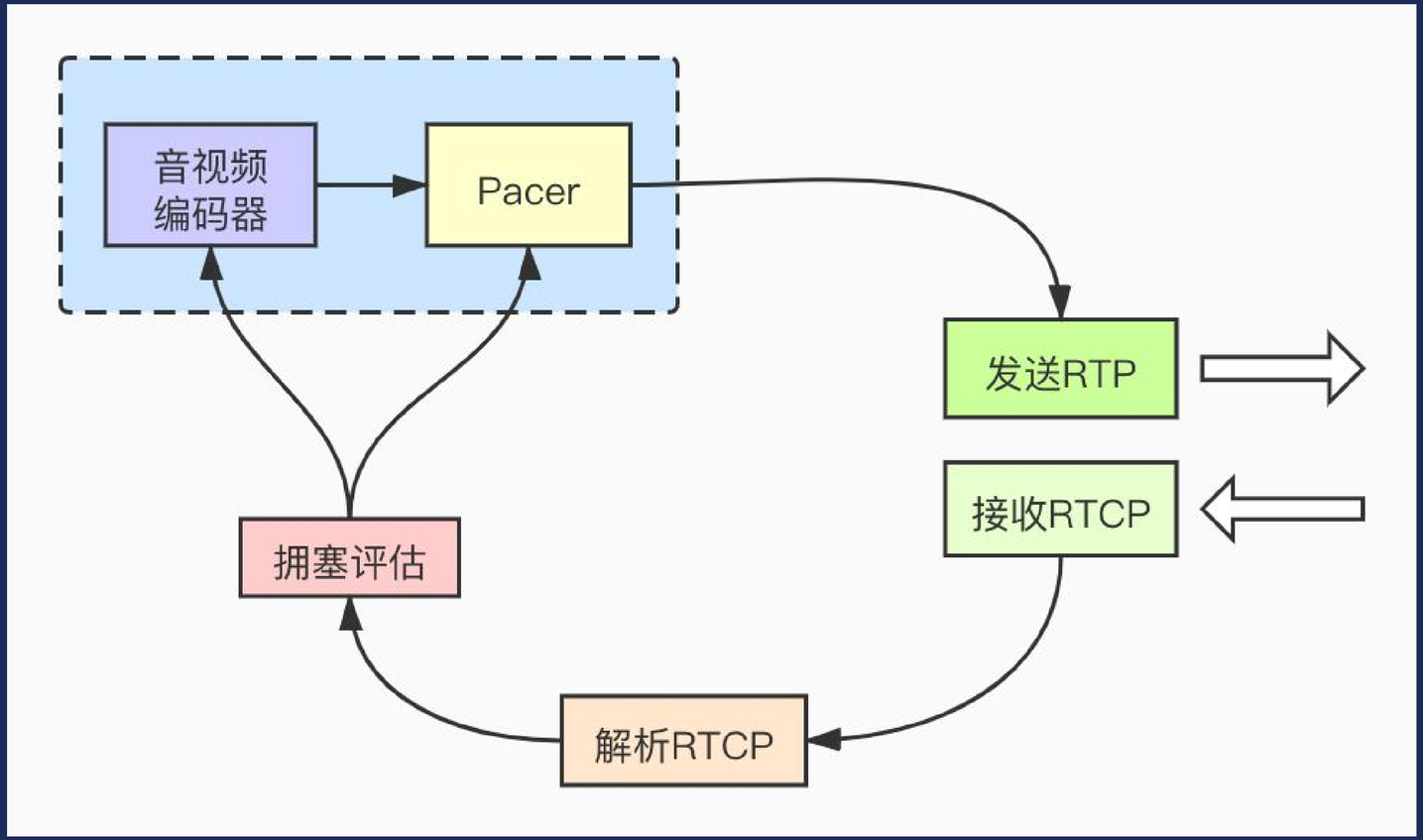
基于丢包的评估方法

$$A_s(t_i) = \begin{cases} 1.08 \cdot A_s(t_{i-1}) & loss < 0.02 \\ A_s(t_i) & loss < 0.1 \\ A_s(t_{i-1}) \cdot (1 - 0.5 \cdot loss) & loss > 0.1 \end{cases}$$

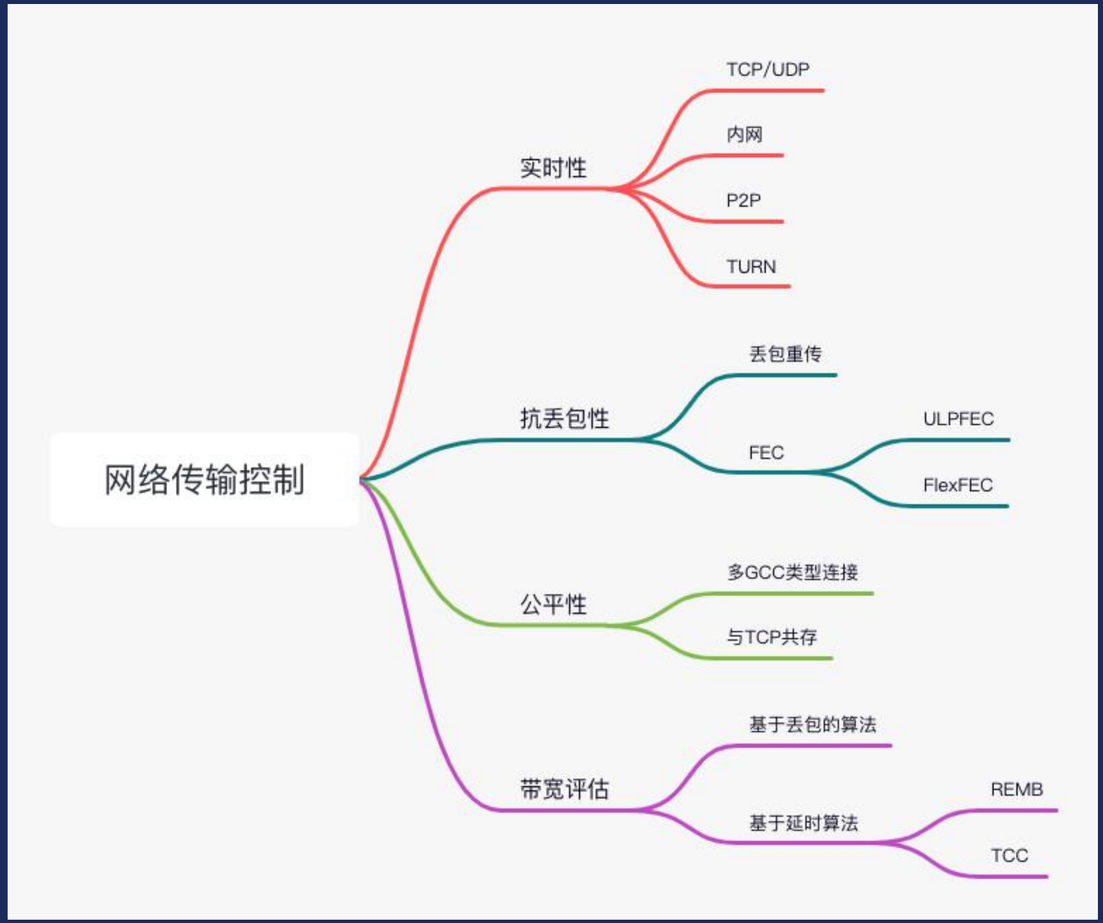
基于延时的评估方法

$$A_r(t_i) = \begin{cases} \alpha A_r(t_{i-1}), & \alpha = 1.08, \sigma = Increase \\ \beta R_r(t_i), & \beta = 0.85, \sigma = Decrease \\ A_r(ti-1), & \sigma = Hold \end{cases}$$

WebRTC拥塞控制流程



总结


网络传输控制
实时性
TCP/UDP
内网
P2P
TURN
抗丢包性
丢包重传
FEC
ULPFEC
FlexFEC
公平性
多GCC类型连接
与TCP共存
带宽评估
基于丢包的算法
基于延时算法
REMB
TCC

深入解析WebRTC开源音视频互动技术原理、架构，并通过开源代码剖析其工作流程和机理。
通过WebRTC互联互通实例详解帮助读者学习WebRTC应用开发方法。

Web开发技术丛书

WebRTC音视频实时互动技术原理、实战与源码分析

PRINCIPLE, PRACTICE AND SOURCE CODE ANALYSIS OF WEBRTC AUDIO AND VIDEO REAL-TIME INTERACTIVE TECHNOLOGY

李超 编著

机械工业出版社
China Machine Press

内容简介

本书深入浅出地对WebRTC技术进行了系统讲解，既有原理又有实战，从WebRTC是如何实现实时音视频通信的，到如何应用WebRTC库实现音视频通信，再到WebRTC源码的剖析，逐步展开讲解。此外，对WebRTC的传输系统进行了重点分析和讲解，相信读者通过本书可以一窥WebRTC传输的奥秘。

书中第1~3章介绍音视频实时通信的由来，WebRTC做了什么，以及它要解决什么问题；第4~10章是实战部分，介绍如何使用WebRTC库实现音视频通信，并对其实现原理进行讲解；第11~13章对WebRTC源码进行分析，让读者对WebRTC有更深层次的认知。

想了解WebRTC实现的专业开发人员可以通过本书了解WebRTC运转机理；学生、老师和音视频爱好者可以通过本书了解WebRTC可以做什么，如何通过WebRTC实现音视频的实时通信。

关于书中内容的建议，可以到 https://rtcdeveloper.com/t/topic/21124 中讨论。书中源码地址为https://avdancedu.com/a0a831a3/。

GMTC
全球大前端技术大会

THANKS

InfoQ

音视跳动 AVDancer

WebRTC_中的网络拥塞控制

扫描二维码 提交议题反馈