



流撕裂者 StreamRipper

一种自适应的分布式局域网缓存

陈文迪, 游灏溢, 陈文浩

2021年12月13日

饮水思源•爱国荣校



01

项目简介

02

设计架构

03

运行流程

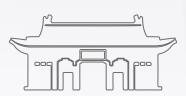
04

实验及结果

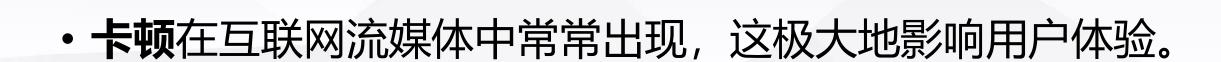










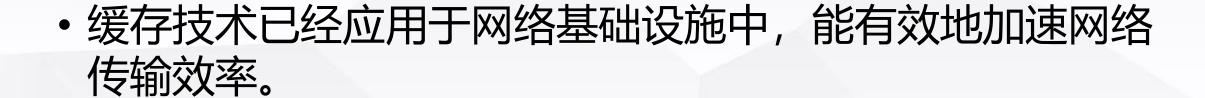


• 相同地区内的人们的网络需求往往类似。

• 用户访问互联网的带宽与局域网内部的带宽有差距。



研究现状



- 但是缓存技术应用具有局限性,目前无法应用于小范围机器集群中,如寝室,教室等。
- 并且使用缓存技术往往需要额外的网络缓存机器,这导致缓存技术应用常常被设备所限制。







• 本项目组设计了一个自适应的分布式局域网缓存。





本方案的优势



• 去中性化的设计,不需要设立一个单独的主机用作缓存服务器。





01

项目简介

02

设计架构

03

运行流程

请输入文字

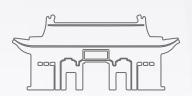
04

实验及结果







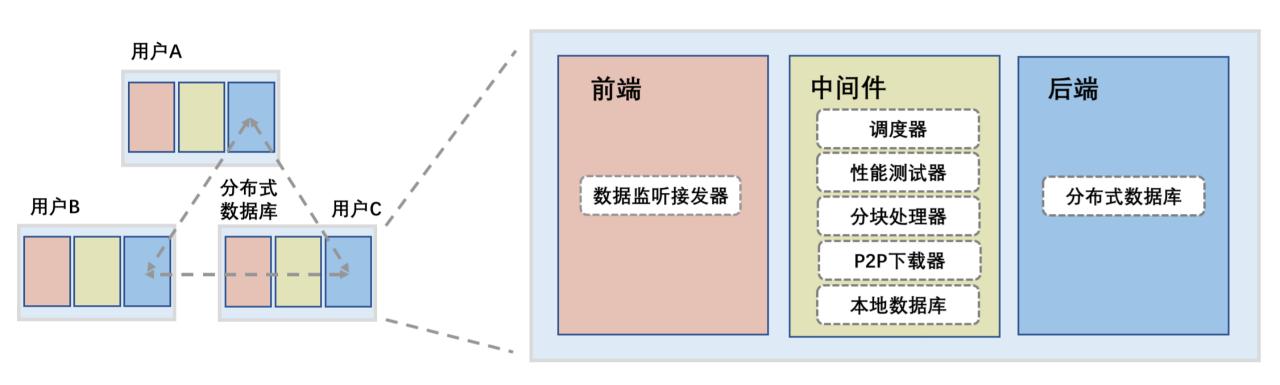




- 请求拦截
- 分布式存储
- 性能调优 (下载节点的选择与流媒体碎片) 以版:

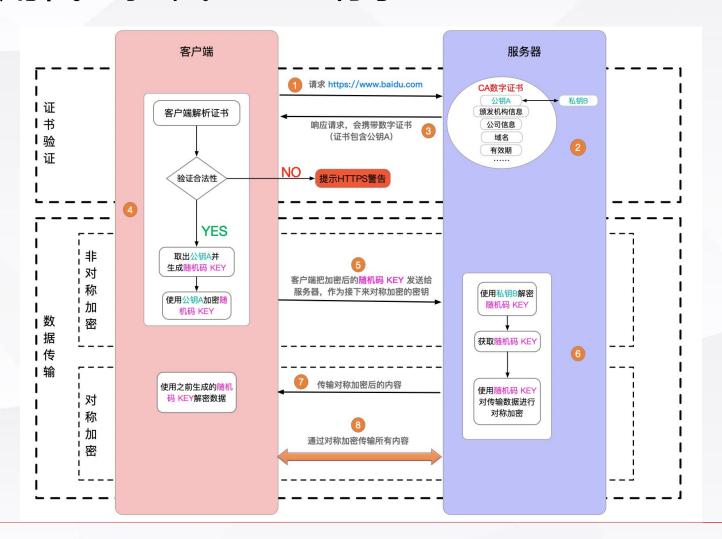


整体采用分层设计思想





方案一: 使用代理拦截HTTP请求



前端设计

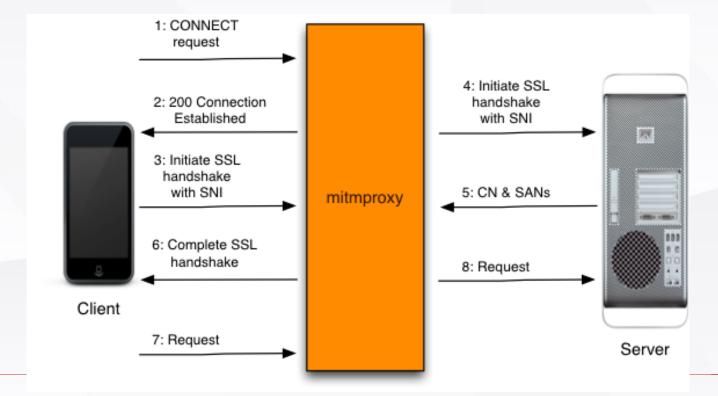
方案二: 类似现有互联网代理的方式, 拦截TCP报文

同样不可行,因为HTTP报文仍然是无法解密的。

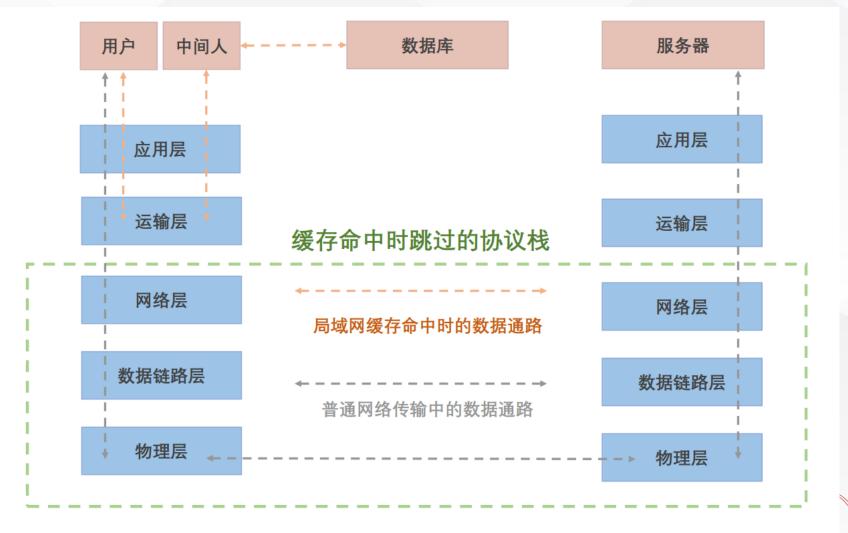


方案三: HTTPS真的无法攻破? 中间人攻击!

通过伪造CA证书来实现中间人攻击,利用中间人攻击的来实现HTTP报文的截取。



最终方案: 使用mitmproxy利用中间人攻击来实现代理





中间件: 调度器与性能测试



委托下载将自己的下载任务委托给别人等待别人下载完成

• 死锁: A->B, B->A 避免死锁: 区分server, client



中间件: 调度器与性能测试



• 性能测试

$$score(i) = speed(i) + \frac{\alpha}{delegation(i)}$$



中间件: 分块处理器

• 服务器分块的不确定性减少缓存命中率

不同截止range的数据大量存在

A: [0, 100]

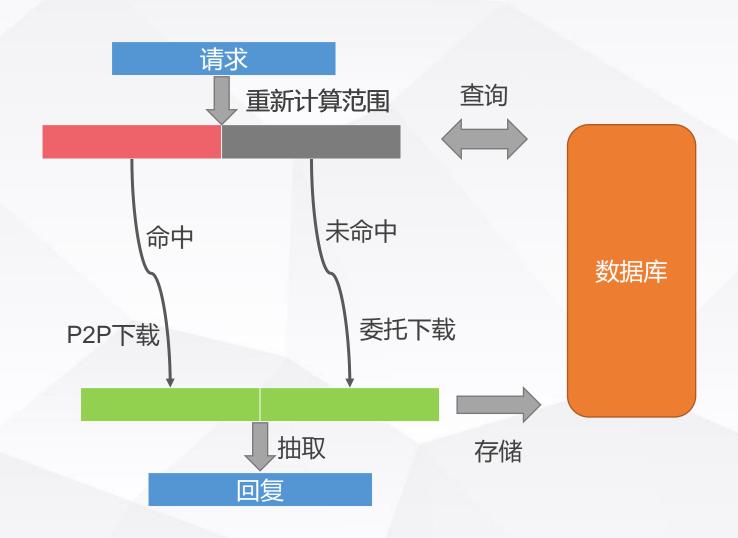
B: [0, 101]





中间件: 分块处理器

- 整流
- 分块处理
- 合并
- 抽取

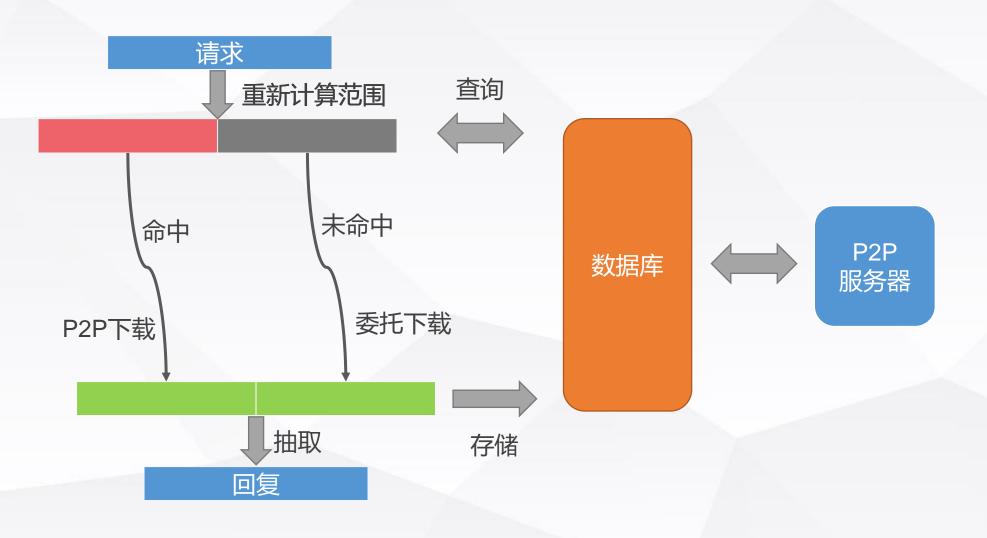






中间件:数据库和P2P下载器



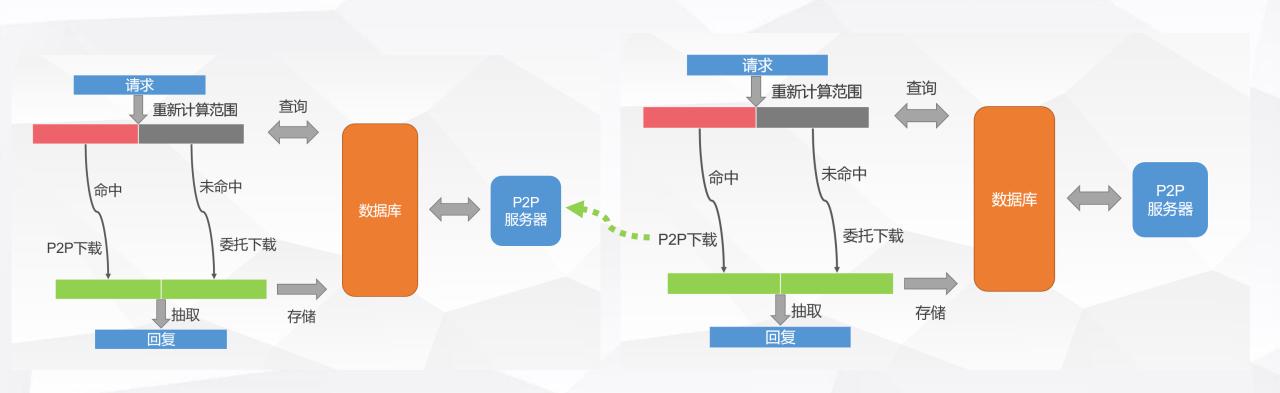






中间件:数据库和P2P下载器









后端设计: 分布式数据库



• 基于Raft的rqlite

• 保持节点数据一致性 + 一定的错误容忍能力

• 根据ID自组织集群, sqlited形式python API





01

项目简介

02

设计架构

03

运行流程

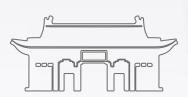
04

实验及结果









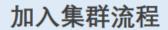








加入与离开集群



程序启动

加入分布式数据库

建立本地数据库

P2P下载器启动

性能测试器启动

调度器初始化

分块处理器初始化

离开集群流程

用户发送中断请求

代理服务器捕获中断

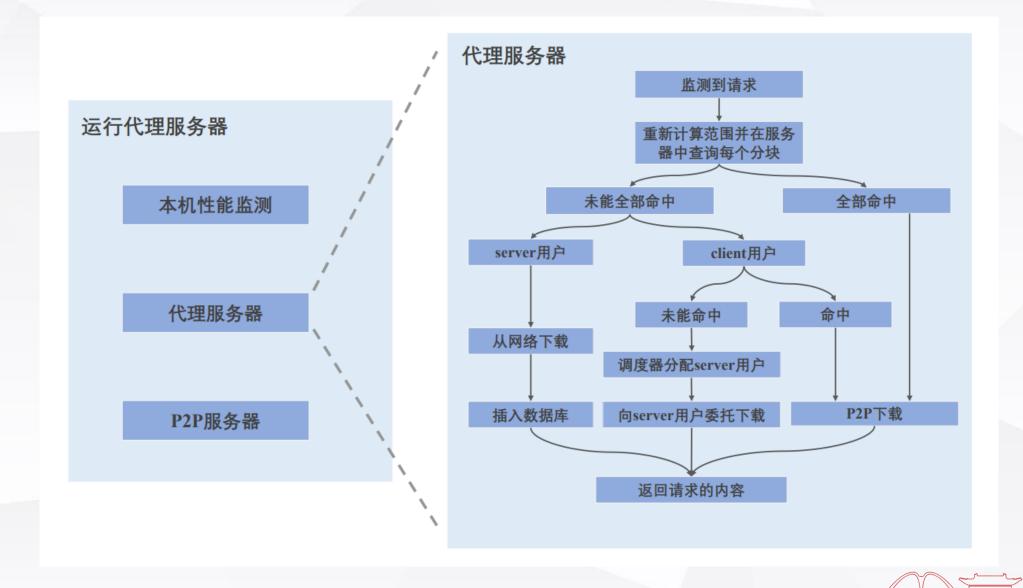
关闭中间件

退出集群

程序运行结束



程序运行过程





01

02

03

项目简介

设计架构

运行流程

04

实验及结果



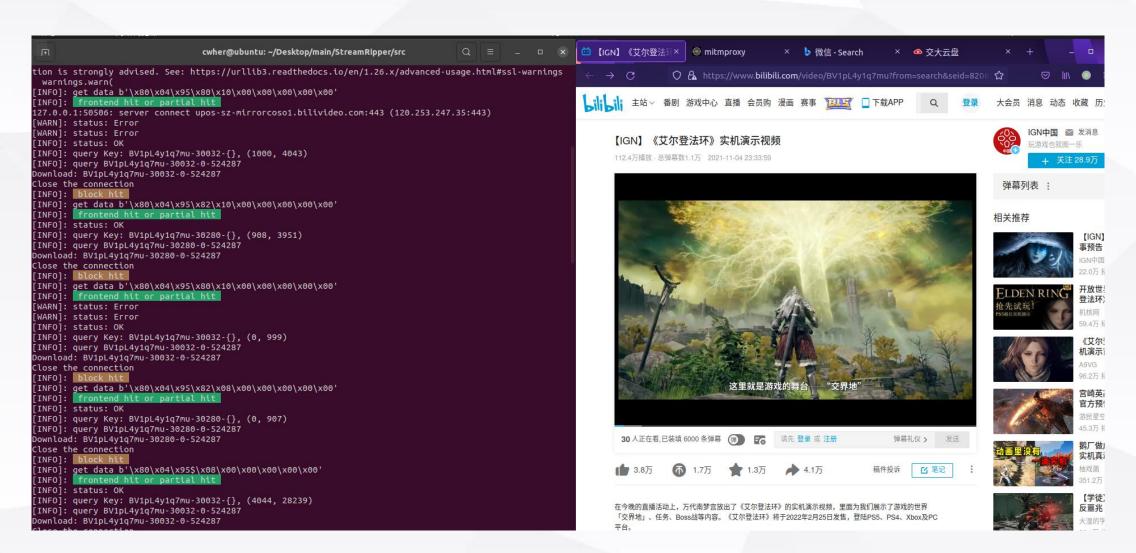








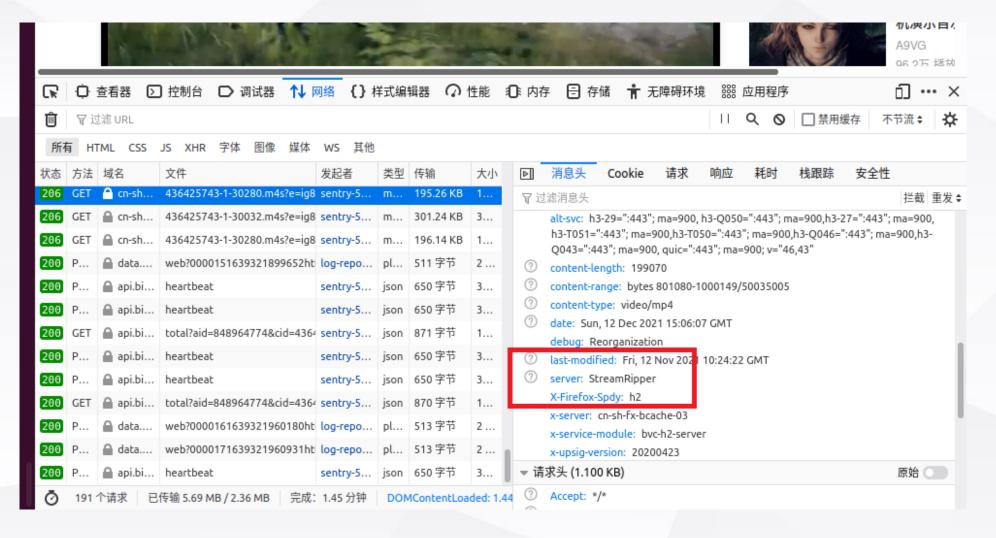
正确性验证:视频正常播放







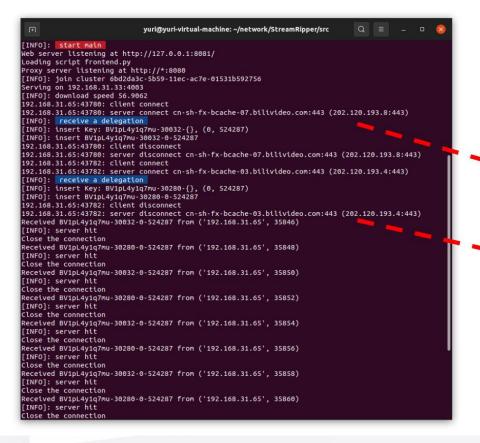
正确性验证: 回复报文被代理所修改

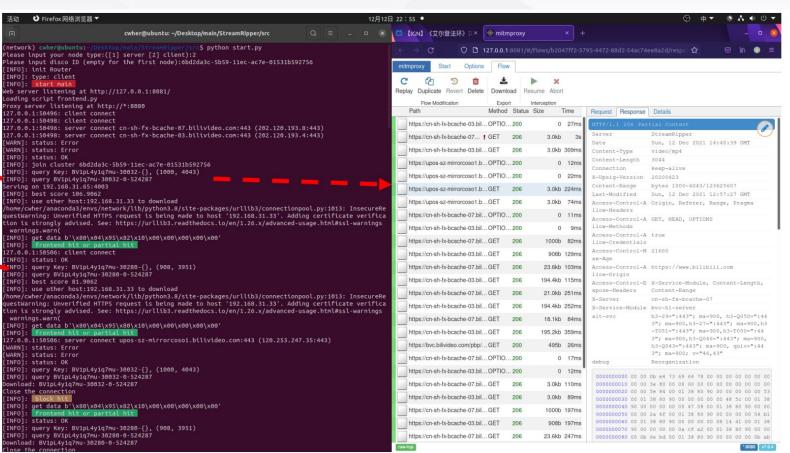










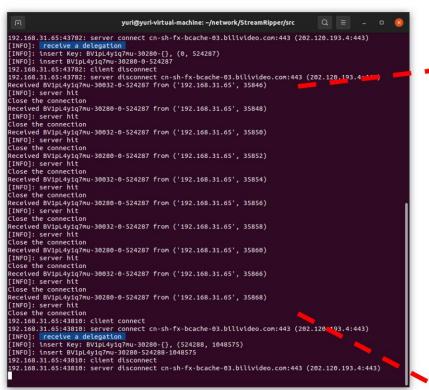


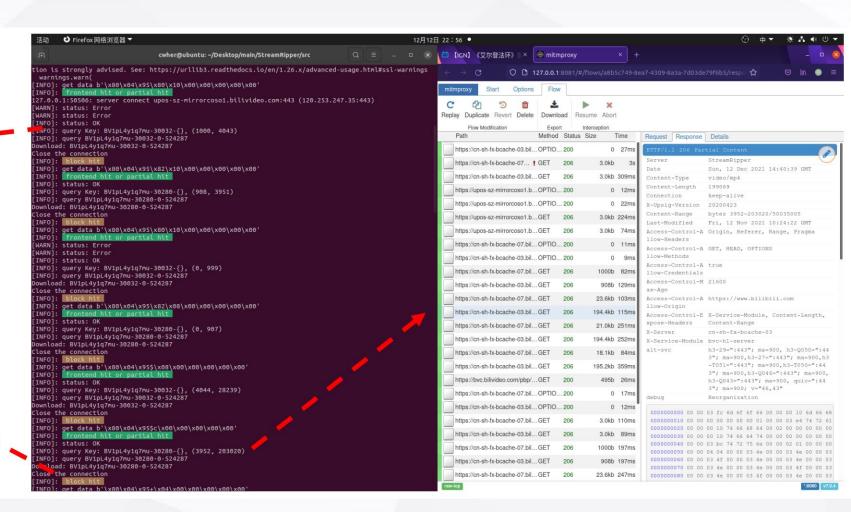




正确性验证: P2P的请求与处理



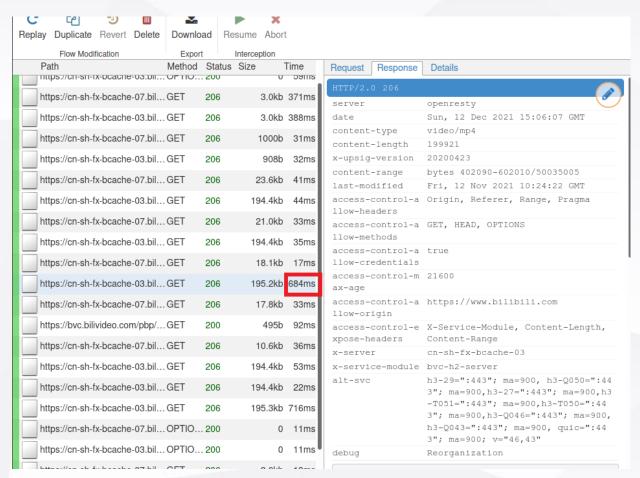


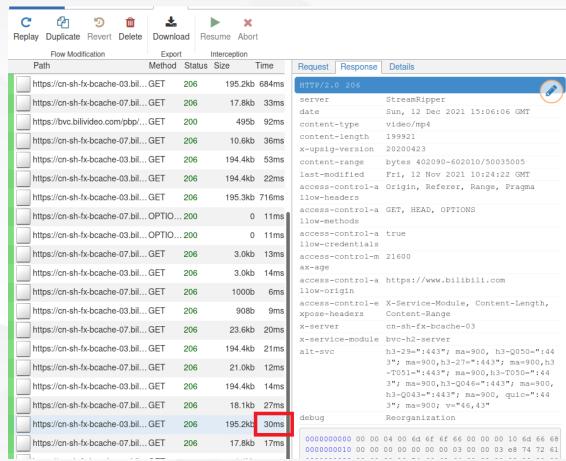






缓存性能测试





来自服务器的请求延迟

来自局域网缓存的延迟





- 设计了一种自适应的分布式局域网缓存。
- 针对分布式存储、请求拦截和性能调优这三个挑战分别给出了我们的解决方案。
- 根据互联网流媒体服务的特点,我们设计了一种分块处理器,有效地减少了数据库冗余并降低了系统整体负载。
- 实验显示, 本项目在对外流量管理上具有明显优势, 可以 很好地解决低质量网络环境下的重复请求问题。



展望:未来的优化方向



- 如定期交换分块的存储位置,可以实现更好的局部性。
- 通过对 P2P 下载请求的合并,我们更好地解决局域网内的性能冗余,提升 P2P 传输性能。
- 可以设计浏览器插件在加密前获取 HTTP 原始报文,从而避免中间人攻击所来带的风险与性能损失



