1.如何做到可靠性传输



2.如何选择TCp和UDP



image-20240825222416377

不会像tcp一样可能粘在一起

3.arq协议

三种模式:

image-20240825222640246

回退n帧: GBN

image-20240825222823966

选择性重传: SR

image-20240825222922831

4.流量控制与拥塞控制

2.4 RTT和RTO image-20240826091756313

image-20240825222956542

2.5 流量控制

image-20240825223210672

image-20240825223353374

控制方式:接收窗口:

image-20240825223512942

根据丢包情况调整接收窗口,降低发送速率

问题;发送方何时再发送数据?

2.6小结:



2.7拥塞控制



5.UDP编程与优势



2.8UDP并发编程



udp没有liseten和accept概念

image-20240825224819598

3UDP如何可靠传输, KCP协议有哪些方面的优势

image-20240825225020976

image-20240825225118394

优势1: 重传速度快, 定制重传策略

image-20240825225359552

优势2: 选择型重传,只重传丢失包 (tcp丢的包后面全部重传)

优势3: 快速重传

image-20240825225553653

问题: 为什么前面说消耗10%-20%的带宽?

答:因为快速重传机制的存在,其实有时候udp包只是慢了一点来应答,但我们直接触发了快速重传,相当于这部分的包就直接重传了,浪费了带宽

优势3: 非延迟ack

延迟ack减少了ack包的数量,但是加大了RTT时间,导致RTO变大,重传时间变大。

4.UNA vs ACK+UNA

image-20240825230159298

5.非退让流控(抢流量)

image-20240825230416045

可以更快速重传

6.KCP

4.1KCP精讲-名词说明

image-20240826091136958

kcp协议传输示意图:



4.2kcp使用方式

KCP源码流程图

image-20240826094939726

1.创建kcp

ickp_create(conv):

image-20240826092107640

问题: server如何知道每个管道对应哪个client?

答: 会话id。udp包前四个字节conv会话id,检查是否已存在,不存在则创建kcp映射该会话id

问题:

image-20240826092332569

答:

方案1:

image-20240826092350822

方案2:

4.3kcp模式配置

image-20240826095353301

image-20240826095506151

4.4kcp协议头

image-20240826095649854

image-20240826095745244

frg分片:

image-20240826095934802

TS时间序列:

image-20240826100219437

4.5kcp发送过程

image-20240826102501059

代码逻辑:

image-20240826103719829

4.6kcp发送窗口

image-20240826103845652

4.7kcp接收数据过程

image-20240826104037381

image-20240826104819180

image-20240826104950835

4.8 kcp确认包处理流程

4.9 kcp快速确认

image-20240826105249007

4.10 kcp_input逻辑

- image-20240826105448456
- image-20240826105514315

4.11 ack list应答列表

- image-20240826110132259
- image-20240826110332039

4.12流量控制和拥塞控制

image-20240826110513798

4.13 探测对方接收窗口

image-20240826110644128

4.14 如何在项目中集成kcp

(-)

- image-20240826110807664
- image-20240826110822793

 (\perp)

- image-20240826110850682
- image-20240826111236372
- image-20240826111517773
- image-20240826111817532
- image-20240826112015964

7.QUIC

image-20240826113055763

image-20240826113113893

7.1框架图:

image-20240826113201653

7.2为什么quic在应用层实现:

image-20240826113301292

7.3 quic协议术语

image-20240826113400244

7.4QUIC和TCP对比

image-20240826113431712

7.5 QUIC报文格式

image-20240826113559963

STream帧1

image-20240826113819709

image-20240826113902785

Stream帧2

image-20240826114200525

QUIC特点:

image-20240826114235075

QUIC开源库和应用

先考虑学习ngtcp2

image-20240826114404139

image-20240826114506900

QUIC面临的挑战