清风软件测试

tcp粘包与udp丢包的原因

2020-04-30 17:49 清风软件测试 阅读(340) 评论(0) 编辑 收藏

tcp粘包与udp丢包的原因

一，什么是tcp粘包与udp丢包

TCP是面向流的, 流要说明就像河水一样, 只要有水, 就会一直流向低处, 不会间断. TCP为了提高传输效率, 发送数据的时候, 并不是直接发送数据到网路, 而是先暂存到系统缓冲, 超过时间或者缓冲满了, 才把缓冲区的内容发送出去, 这样, 就可以有效提高发送效率. 所以会造成所谓的粘包, 即前一份Send的数据跟后一份Send的数据可能会暂存到缓冲当中, 然后一起发送。

UDP就不同了, 面向报文形式, 系统是不会缓冲的, 也不会做优化的, Send的时候, 就会直接Send到网络上, 对方收不收到也不管, 所以这块数据总是能够能一包一包的形式接收到, 而不会出现前一个包跟后一个包都写到缓冲然后一起Send。

但其实别想得太复杂的, TCP所谓的粘包处理, UDP所谓的丢包处理, 其实都是很简单的.
TCP只要保证自己写入的流是按 长度 + 内容 + 长度 + 内容 这样就可以非常简单的解决粘包问题, 切忌不要采用所谓的 开始标识 + 数据 + 结束标识 来分包, 适用性极低, 错误率极高, 除非数据都是固定有格式, 否则是不能采用这种方式的.
UDP传送当中, 只存在丢包的可能, 收到包的时候, 肯定这个包的内容就是正确的, 很少会有错误的, 因为UDP本身也会用CRC32进行验证, 还有长度验证, 验证不通过, 系统自动就会丢弃, 所以, 只需要去想象一种情况, 把一个苹果切了很多块, 然后要捡起这些块到另外一个容器, 由另外一个人进行重组, 而这捡的过程当中, 有可能切的人捡漏了某些块, 然后重组苹果的人就要求切的人重新把缺失的块捡回来.

TCP在建立连接时有三次握手的过程，这样就保证的连接的有效性。发包时发包完成也有反馈（对方接收完成有标记），所以tcp不存在丢包乱序的问题。但是发包过于频繁时会出现粘包的问题。
UDP建立连接并没有三次握手的过程，而且发送数据只是负责发送，不会有发送成功的反馈，所以当数据量大会被切分为多个小数据发包时会出现丢包现象。

udp是放在数据帧里面传输的 数据帧最多放1500个字节，除去udp的报头 一个数据帧最多发送1472个字节的udp,如果udp过大，就会被拆分成多个数据帧发送到网络，即会造成丢包的现象。

二，什么是TCP粘包？怎么解决这个问题

在socket网络编程中，都是端到端通信，由客户端端口+服务端端口+客户端IP+服务端IP+传输协议组成的五元组可以明确的标识一条连接。在TCP的socket编程中，发送端和接收端都有成对的socket。发送端为了将多个发往接收端的包，更加高效的的发给接收端，于是采用了优化算法（Nagle算法），将多次间隔较小、数据量较小的数据，合并成一个数据量大的数据块，然后进行封包。那么这样一来，接收端就必须使用高效科学的拆包机制来分辨这些数据。

1.Q：什么是TCP粘包问题?

TCP粘包就是指发送方发送的若干包数据到达接收方时粘成了一包，从接收缓冲区来看，后一包数据的头紧接着前一包数据的尾，出现粘包的原因是多方面的，可能是来自发送方，也可能是来自接收方。

2.Q：造成TCP粘包的原因

（1）发送方原因

TCP默认使用Nagle算法（主要作用：减少网络中报文段的数量），而Nagle算法主要做两件事：

只有上一个分组得到确认，才会发送下一个分组
收集多个小分组，在一个确认到来时一起发送
Nagle算法造成了发送方可能会出现粘包问题

（2）接收方原因

TCP接收到数据包时，并不会马上交到应用层进行处理，或者说应用层并不会立即处理。实际上，TCP将接收到的数据包保存在接收缓存里，然后应用程序主动从缓存读取收到的分组。这样一来，如果TCP接收数据包到缓存的速度大于应用程序从缓存中读取数据包的速度，多个包就会被缓存，应用程序就有可能读取到多个首尾相接粘到一起的包。

3.Q：什么时候需要处理粘包现象?

如果发送方发送的多组数据本来就是同一块数据的不同部分，比如说一个文件被分成多个部分发送，这时当然不需要处理粘包现象
如果多个分组毫不相干，甚至是并列关系，那么这个时候就一定要处理粘包现象了

4.Q：如何处理粘包现象?

（1）发送方

对于发送方造成的粘包问题，可以通过关闭Nagle算法来解决，使用TCP_NODELAY选项来关闭算法。

（2）接收方

接收方没有办法来处理粘包现象，只能将问题交给应用层来处理。

（2）应用层

应用层的解决办法简单可行，不仅能解决接收方的粘包问题，还可以解决发送方的粘包问题。

解决办法：循环处理，应用程序从接收缓存中读取分组时，读完一条数据，就应该循环读取下一条数据，直到所有数据都被处理完成，但是如何判断每条数据的长度呢?

1，格式化数据：每条数据有固定的格式（开始符，结束符），这种方法简单易行，但是选择开始符和结束符时一定要确保每条数据的内部不包含开始符和结束符。
2，发送长度：发送每条数据时，将数据的长度一并发送，例如规定数据的前4位是数据的长度，应用层在处理时可以根据长度来判断每个分组的开始和结束位置。

5.Q：UDP会不会产生粘包问题呢?

TCP为了保证可靠传输并减少额外的开销（每次发包都要验证），采用了基于流的传输，基于流的传输不认为消息是一条一条的，是无保护消息边界的（保护消息边界：指传输协议把数据当做一条独立的消息在网上传输，接收端一次只能接受一条独立的消息）。

UDP则是面向消息传输的，是有保护消息边界的，接收方一次只接受一条独立的信息，所以不存在粘包问题。

About

昵称: 清风软件测试
园龄: 4年
粉丝: 546
关注: 7
+加关注

最新评论

Re:git 上传本地代码到远程仓库
给力
-- daofree
Re:Runtime.getRuntime().exec()需要注意的地方
写的非常漂亮!
-- 胖墩哥
Re:接口测试用例设计
密码是什么啦
-- fantasy2010
Re:docker每次都重新拉取远程镜像的问题
什么原理呀?
-- Arthur·Wang
Re:接口测试之接口api文档的重要性
博主可以分享阅读密码嘛
-- 测客

日历

< 2020年10月 >

日	一	二	三	四	五	六
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

我的标签

python(39)
python爬虫(30)
jmeter(22)
性能测试(22)
性能调优(16)
testNG(16)
接口测试(15)
MySQL(13)
testng教程(12)
TestNG入门教程(12)
更多

随笔分类

AI人工智能(5)
app UI 自动化java(18)
app UI 自动化python(7)
app 自动化(42)
appium java(28)
appium python(4)
app测试(13)
docker(17)
Fiddler抓包(6)
flask(5)
ios app自动化测试(4)
java(135)
jenkins(5)
jmeter(67)
junit(4)
linux(38)
locust(4)
maven(2)
mock server(5)
monkey monkeyrunner(7)

随笔档案

2020年9月(25)
2020年8月(12)
2020年7月(11)
2020年6月(13)
2020年5月(16)
2020年4月(9)
2020年3月(10)
2020年2月(1)
2020年1月(5)
2019年12月(16)
2019年11月(21)
2019年10月(14)
2019年9月(17)
2019年8月(31)
2019年7月(33)
2019年6月(19)
2019年5月(16)
2019年4月(2)
2019年3月(20)
2019年2月(14)
2019年1月(41)
2018年12月(8)
2018年11月(14)
2018年10月(2)
2018年9月(38)
2018年8月(8)
2018年7月(3)
2018年6月(18)
2018年5月(10)
2018年4月(16)
2018年3月(26)
2018年2月(4)
2018年1月(17)
2017年12月(28)
2017年11月(25)
2017年10月(16)
2017年9月(11)
2017年8月(2)
2017年7月(51)
2017年6月(24)
2017年5月(6)
2017年4月(13)
2017年3月(33)

举个例子：有三个数据包，大小分别为2k、4k、6k，如果采用UDP发送的话，不管接受方的接收缓存有多大，我们必须要进行至少三次以上的发送才能把数据包发送完，但是使用TCP协议发送的话，我们只需要接受方的接收缓存有12k的大小，就可以一次把这3个数据包全部发送完毕。

三，UDP主要丢包原因及具体问题分析

一、主要丢包原因

1、接收端处理时间过长导致丢包：调用recv方法接收端收到数据后，处理数据花了一些时间，处理完后再次调用recv方法，在这二次调用间隔里,发过来的包可能丢失。对于这种情况可以修改接收端，将包接收后存入一个缓冲区，然后迅速返回继续recv。

2、发送的包巨大丢包：虽然send方法会帮你做大包切割成小包发送的事情，但包太大也不行。例如超过50K的一个udp包，不切割直接通过send方法发送也会导致这个包丢失。这种情况需要切割成小包再逐个send。

3、发送的包较大，超过接受者缓存导致丢包：包超过mtu size数倍，几个大的udp包可能会超过接收者的缓冲，导致丢包。这种情况可以设置socket接收缓冲。以前遇到过这种问题，我把接收缓冲设置成64K就解决了。
int nRecvBuf=32*1024;//设置为32K
setsockopt(s,SOL_SOCKET,SO_RCVBUF,(const char*)&nRecvBuf,sizeof(int));

4、发送的包频率太快：虽然每个包的大小都小于mtu size 但是频率太快，例如40多个mut size的包连续发送中间不sleep，也有可能导致丢包。这种情况也有时可以通过设置socket接收缓冲解决，但有时解决不了。所以在发送频率过快的时候还是考虑sleep一下吧。

5、局域网内不丢包，公网上丢包。这个问题我也是通过切割小包并sleep发送解决的。如果流量太大，这个办法也不灵了。总之udp丢包总是会有的，如果出现了用我的方法解决不了，还有这个几个方法：要么减小流量，要么换tcp协议传输，要么做丢包重传的工作。

二、具体问题分析

1.发送频率过高导致丢包

很多人会不理解发送速度过快为什么会产生丢包，原因就是UDP的SendTo不会造成线程阻塞，也就是说，UDP的SentTo不会像TCP中的SendTo那样，直到数据完全发送才会return回调用函数，它不保证当执行下一条语句时数据是否被发送。（SendTo方法是异步的）这样，如果要发送的数据过多或者过大，那么在缓冲区满的那个瞬间要发送的报文就很有可能被丢失。至于对“过快”的解释，作者这样说：“A few packets a second are not an issue; hundreds or thousands may be an issue.”（一秒钟几个数据包不算什么，但是一秒钟成百上千的数据包就不好办了）。 要解决接收方丢包的问题很简单，首先要保证程序执行后马上开始监听（如果数据包不确定什么时候发过来的话），其次，要在收到一个数据包后最短的时间内重新回到监听状态，其间要尽量避免复杂的操作（比较好的解决办法是使用多线程回调机制）。

2.报文过大丢包

至于报文过大的问题，可以通过控制报文大小来解决，使得每个报文的长度小于MTU。以太网的MTU通常是1500 bytes，其他一些诸如拨号连接的网络MTU值为1280 bytes，如果使用speaking这样很难得到MTU的网络，那么最好将报文长度控制在1280 bytes以下。

3.发送方丢包

发送方丢包：内部缓冲区（internal buffers）已满，并且发送速度过快（即发送两个报文之间的间隔过短）； 接收方丢包：Socket未开始监听； 虽然UDP的报文长度最大可以达到64 kb，但是当报文过大时，稳定性会大大减弱。这是因为当报文过大时会被分割，使得每个分割块（翻译可能有误差，原文是fragmentation）的长度小于MTU，然后分别发送，并在接收方重新组合（reassemble），但是如果其中一个报文丢失，那么其他已收到的报文都无法返回给程序，也就无法得到完整的数据了。

----------下面是一个UDP丢包的例子----------

UDP丢包

我们是后一个包丢掉了

最近在做一个项目,在这之前,做了个验证程序.
发现客户端连续发来1000个1024字节的包,服务器端出现了丢包现象.
纠其原因,是服务端在还未完全处理掉数据,客户端已经数据发送完毕且关闭了.

有没有成熟的解决方案来解决这个问题.
我用过sleep(1),暂时解决这个问题,但是这不是根本解决办法,如果数据量大而多,网络情况不太好的话,还是有可能丢失.

你试着用阻塞模式吧...
select...我开始的时候好像也遇到过..不过改为阻塞模式后就没这个问题了...

采用回包机制,每个发包必须收到回包后再发下一个

UDP丢包是正常现象，因为它是不安全的。

丢包的原因我想并不是“服务端在还未完全处理掉数据,客户端已经数据发送完毕且关闭了”，而是服务器端的socket接收缓存满了(udp没有流量控制，因此发送速度比接收速度快，很容易出现这种情况)，然后系统就会将后来收到的包丢弃。你可以尝试用setsockopt()将接收缓存(SO_RCVBUF)加大看看能不能解决问题。

服务端采用多线程pthread接包处理

UDP是无连接的，面向消息的数据传输协议，与TCP相比，有两个致命的缺点，一是数据包容易丢失，二是数据包无序。
要实现文件的可靠传输，就必须在上层对数据丢包和乱序作特殊处理，必须要有要有丢包重发机制和超时机制。
常见的可靠传输算法有模拟TCP协议，重发请求（ARQ）协议，它又可分为连续ARQ协议、选择重发ARQ协议、滑动窗口协议等等。
如果只是小规模程序，也可以自己实现丢包处理，原理基本上就是给文件分块，每个数据包的头部添加一个唯一标识序号的ID值，当接收的包头部ID不是期望中的ID号，则判定丢包，将丢包ID发回服务端，服务器端接到丢包响应则重发丢失的数据包。
模拟TCP协议也相对简单，3次握手的思想对丢包处理很有帮助。

page object/factory(5)
python(157)
python爬虫(35)
robot framework(1)
RPC(6)
selenium webdriver java(44)
selenium webdriver python(12)
testNG(39)
udp_tcp(17)
UI自动化测试框架(15)
unittest(3)
测试环境搭建(10)
计算机基础知识(2)
架构(10)
兼容性测试(1)
接口测试(79)
接口自动化测试(49)
接口自动化测试框架(21)
软件测试(621)
数据库(56)
数据埋点(5)
网络(9)
性能测试(144)
中间件(8)
自动化测试(311)
自动化测试框架（java）(34)
自动化测试框架（python）(26)

阅读排行榜

1. java判断包含contains方法的使用(58713)
2. python 一个.py文件如何调用另一个.py文件中的类和函数(44371)
3. java基础语法 List(39613)
4. android adb devices offline的解决办法(37642)
5. python写http post请求的四种请求体(35149)

2017年2月(34)
2017年1月(9)
2016年12月(35)
2016年11月(14)

推荐排行榜

1. http webservice socket的区别(4)
2. selenium webdriver模拟鼠标键盘操作(3)
3. 在做自动化测试之前你需要知道的什么是自动化测？(3)
4. gerrit 使用教程（一）(2)
5. chrome正受到自动测试软件的控制-----web自动化测试如何去掉这段提示(2)

udp是不安全的，如果不加任何控制，不仅会丢失包，还可能收到包的顺序和发送包的顺序不一样。这个必须在自己程序中加以控制才行。
收到包后，要返回一个应答，如果发送端在一定时间内没有收到应答，则要重发。

UDP本来存在丢包现象,现在的解决方案暂时考虑双方增加握手.
这样做起来,就是UDP协议里面加上了TCP的实现方法.
程序中采用的是pthread处理,丢包率时大时小,不稳定可靠

我感觉原因可能有两个，一个是客户端发送过快，网络状况不好或者超过服务器接收速度，就会丢包。
第二个原因是服务器收到包后，还要进行一些处理，而这段时间客户端发送的包没有去收，造成丢包。

解决方法：

1，客户端降低发送速度，可以等待回包，或者加一些延迟。
2，服务器部分单独开一个线程，去接收UDP数据，存放在一个缓冲区中，又另外的线程去处理收到的数据，尽量减少因为处理数据延时造成的丢包。

有两种方法解决楼主的问题：
方法一：重新设计一下协议，增加接收确认超时重发。（推荐）
方法二：在接收方，将通信和处理分开，增加个应用缓冲区；如果有需要增加接收socket的系统缓冲区。（本方法不能从根本解决问题，只能改善）

网络丢包，是再正常不过的了。
既然用UDP，就要接受丢包的现实，否则请用TCP。
如果必须使用UDP，而且丢包又是不能接受的，只好自己实现确认和重传，说白了，就是自己实现TCP（当然是部分和有限的简单实现）。

UDP是而向无连接的，用户在实施UDP编程时，必须制定上层的协议，包括流控制，简单的超时和重传机制，如果不要求是实时数据，我想TCP可能会更适合你!

1：什么是丢包率?
你的电脑向目标发送一个数据包，如果对方没有收到.就叫丢包.
比如你发10个，它只收到9个. 那么丢包率就是 10%
数据在网络中是被分成一各个个数据报传输的,每个数据报中有表示数据信息和提供数据路由的桢.而数据报在一般介质中传播是总有一小部分由于两个终端的距离过大会丢失,而大部分数据包会到达目的终端.所谓网络丢包率是数据包丢失部分与所传数据包总数的比值.正常传输时网络丢包率应该控制在一定范围内.

2：什么是吞吐量?
网络中的数据是由一个个数据包组成，防火墙对每个数据包的处理要耗费资源。吞吐量是指在没有帧丢失的情况下，设备能够接受的最大速率。其测试方法是：在测试中以一定速率发送一定数量的帧，并计算待测设备传输的帧，如果发送的帧与接收的帧数量相等，那么就将发送速率提高并重新测试；如果接收帧少于发送帧则降低发送速率重新测试，直至得出最终结果。吞吐量测试结果以比特/秒或字节/秒表示。

吞吐量和报文转发率是关系防火墙应用的主要指标，一般采用FDT(Full Duplex Throughput)来衡量，指64字节数据包的全双工吞吐量，该指标既包括吞吐量指标也涵盖了报文转发率指标。

随着Internet的日益普及，内部网用户访问Internet的需求在不断增加，一些企业也需要对外提供诸如WWW页面浏览、FTP文件传输、DNS域名解析等服务，这些因素会导致网络流量的急剧增加，而防火墙作为内外网之间的唯一数据通道，如果吞吐量太小，就会成为网络瓶颈，给整个网络的传输效率带来负面影响。因此，考察防火墙的吞吐能力有助于我们更好的评价其性能表现。这也是测量防火墙性能的重要指标。

吞吐量的大小主要由防火墙内网卡，及程序算法的效率决定，尤其是程序算法，会使防火墙系统进行大量运算，通信量大打折扣。因此，大多数防火墙虽号称100M防火墙，由于其算法依靠软件实现，通信量远远没有达到100M,实际只有10M-20M。纯硬件防火墙，由于采用硬件进行运算，因此吞吐量可以达到线性90-95M,是真正的100M防火墙。

对于中小型企业来讲，选择吞吐量为百兆级的防火墙即可满足需要，而对于电信、金融、保险等大公司大企业部门就需要采用吞吐量千兆级的防火墙产品。

3：检测丢包率
下载一个世纪前线，在百度可以找到，很小的程序。

NetIQ Chariot 一款网络应用软件性能测试工具

网络吞吐量测试,CHARIOT测试网络吞吐量

原文地址：

https://www.cnblogs.com/zzt-lovelinlin/p/5303884.html
https://blog.csdn.net/weixin_41047704/article/details/85340311
https://www.cnblogs.com/Zhaols/p/6105926.html

清风软件测试
关注 - 7
粉丝 - 546
+加关注

0 0

« 上一篇： python中基于tcp协议与udp的通信
» 下一篇： docker centos8 安装ssh连接xshell并部署javaweb项目

分类 性能测试，软件测试，udp_tcp

支持 Markdown

提交评论 退出 订阅评论 我的博客

[Ctrl+Enter快捷键提交]

相关博文：
· TCP和UDP
· TCP和UDP
· tcp和udp的区别?
· tcp、udp总结
· TCP与UDP笔记
» 更多推荐...

最新 IT 新闻:
· 谷歌Stadia高管：游戏主播应给发行商付版税
· 微软澳大利亚闹乌龙：Xbox Series X提前至10月27日发货
· 苹果大学校长发文详解苹果组织结构
· Chrome正在新标签页测试购物广告功能
· Fedora 33目标将于下周发布 搭载GNOME 3.38桌面环境
» 更多新闻...