**WireShark网络性能分析**

　　最近生产上出现一个性能问题，表现为：行情延时5s左右。从log一路追查下去，发现是我们自己写的一个行情网关（部署在xx.xx.xx.132）<->第三方的中转网关（部署在xx.xx.xx.133）之间的通信产生的。

　　Who to blame? 这是个问题，是我们的行情网关、网络、还是第三方的中转网关。所以想到用[**WireShark**](javascript:;)抓包进行分析。抓到的包在这里：CaptureData-20160905.pcapng，用WireShark打开后，在统计里有很多有用的信息，有用的是这几个：

　　捕获文件属性：能看到基本信息，抓包起始时间、跨度（01:04，共64秒）、大小（7624KB）、分组（28068个包）等

　　协议分级：能看到各级协议的包的占比

　　IO图表：能看到每秒的包数量，还可以自定义过滤器，对Y轴加一些sum、count、avg等统计方法

　　TCP-时间序列：很能反映通信特征，任何一方的seq不变表示不再发数据，只发长度为0的应答包

　　TCP-吞吐量：反映每秒吞吐的包长度

　　TCP-往返时间：rtt round trip time, 这个图很能说明132处理的快，133发的慢

　　TCP-窗口尺寸：表示一方接收数据的能力，132一直在5万以上，而133到过0

　　首先要设置合适的过滤器，过滤出我们感兴趣的包，试下来最有用的是：ip.addr == xx.xx.xx.132 && ip.addr == xx.xx.xx.133。通过分析，整个60秒的通信过程分为3个部分：

|  |
| --- |
| 132:50012端口与133:7777端口的3次握手，syn->syn ack->ack。建立连接后，状态变为established  　　132:50013端口向133:7777端口发送数据，主要是发送要订阅的标的。在最下面的data窗口能看到ascii码里会显示标的的代码，比如au1609等  　　132:50013端口接收133:7777端口发来的数据，主要是期货行情。 |

　　分析的一些小结论如下：

|  |
| --- |
| 数据的逻辑关系：发送端seq + data bytes = 接收端ack 以及 发送端seq + data bytes = 发送端next seq  　　接收端有时会将若干个ack确认包合并成1个。  　　包15281到15673，注意到133的windowSize一直在不断减少，突然变大就是window update。包16073开始各种window full、zero window，因为133的winSize=0满了，132开始发送0窗口探测报文。订阅完成后133的win就不再减少了，没压力了。换成133给132发送行情数据了。  　　Frame在网线级别，Ethernet在mac级别，IP在IP地址（v4、v6）级别，tcp在端口port级别。  　　注意前两次握手syn和syn ack里有关于Selective Ack、Maximum Segment Size和[**Windows**](javascript:;) Scale（比如：8，表示位移8位，即乘数为256）等选项的协商  　　吞吐量(bytes/sec) = Window Size(bytes) / RTT(round trip time, sec) = 64KB/0.000128s = 500MB/s  　　右键Follow this stream，会产生一个过滤器：tcp.stream eq 604，其实对应于每个ip+port对都会产生一个stream index，不论方向。这个index应该是wireshark在抓取过程中顺序产生的，类似于Frame Number。这个过滤器的效果类似于：ip.addr == xx.xx.xx.132 && ip.addr == xx.xx.xx.133 && tcp.port == 7777 && tcp.port == 50012，还是蛮方便实用的。  　　性能分析：syn、ack在tcp协议级别，不包括任何数据，与应用层无关，握手不卡说明网络没问题。发送行情的过程中，双方的win=65535，表示窗口这块都有富余，双方的网络都不卡。132收到行情处理再回复ack只用了＜0.1ms，说明接收端的速度也很快。所以只能有一个解释，133发送端发的慢。其实从133的日志里也能看出来，“客户端发送队列已满，将丢弃xx报文”。  　　由于中转网关是第三方的程序，我们没法控制，因此只能我们减少订阅的标的数，发送的压力小了，自然就不慢了。 |