阿里中间件团队博客

致力于成为中国第一，世界一流的中间件技术团队

就是要你懂 TCP

2017-06-08 | 蛰剑 | 网络

看过太多tcp相关文章，但是看完总是不过瘾，似懂非懂，反复考虑过后，我觉得是那些文章太过理论，看起来没有体感，所以吸收不了。 希望这篇文章能做到言简意赅，帮助大家透过案例来理解原理。

tcp的特点

这个大家基本都能说几句，面试的时候候选人也肯定会告诉你这些：

- 三次握手
- 四次挥手
- 可靠连接
- 丢包重传

但是我只希望大家记住一个核心的：**tcp是可以可靠传输协议，它的所有特点都为这个可靠传输服务**。

那么tcp是怎么样来保障可靠传输呢?

tcp在传输过程中都有一个ack，接收方通过ack告诉发送方收到那些包了。这样发送方能知道有没有丢包，进而确定重传。

tcp建连接的三次握手

来看一个java代码连接数据库的三次握手过程

No.	Source	Destination	Proto	Length	TCP Segment Len	Sequence number	Info
1	client	server	TCP	74	0	1608229138	48287→mysql(3306) [SYN] Seq=1608229138 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=42095070 TSecr=0 WS=128
2	server	client	TCP	60	0	4266193409	mysql(3306)→48287 [SYN, ACK] Seq=4266193409 Ack=1608229139 Win=65535 Len=0 MSS=1460
3	client	server	TCP	54	0	1608229139	48287→mysql(3306) [ACK] Seq=1608229139 Ack=4266193410 Win=29200 Len=0
4	server	client	MyS…	132	78	4266193410	Server Greeting proto=10 version=5.6.28

image.png

三个红框表示建立连接的三次握手：

- 第一步：client 发送 syn 到server 发起握手；
- 第二步：server 收到 syn后回复syn+ack给client；
- 第三步：client 收到syn+ack后，回复server一个ack表示收到了server的syn+ack（此时client的48287端口的连接已经是established）

握手的核心目的是告知对方seq（绿框是client的初始seq，蓝色框是server 的初始seq），对方回复ack（收到的seq+包的大小），这样发送端就知道有没有丢包了。

握手的次要目的是告知和协商一些信息，图中黄框。

- MSS-最大传输包
- SACK_PERM-是否支持Selective ack(用户优化重传效率)
- WS-窗口计算指数（有点复杂的话先不用管）

这就是tcp为什么要握手建立连接，就是为了解决tcp的可靠传输。

tcp断开连接的四次挥手

再来看java连上mysql后，执行了一个SQL： select sleep(2); 然后就断开了连接

No.	Source	Destination	Proto	Length	TCP Segment Len	Sequence number	Info
1	client	server	TCP	74	0	1608229138	48287→mysql(3306) [SYN] Seq=1608229138 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=42095070 TSecr=0 WS=128
2	server	client	TCP	60	0	4266193409	mysql(3306)→48287 [SYN, ACK] Seq=4266193409 Ack=1608229139 Win=65535 Len=0 MSS=1460
3	client	server	TCP	54	0	1608229139	48287→mysql(3306) [ACK] Seq=1608229139 Ack=4266193410 Win=29200 Len=0
4	server	client	MyS…	132	78	4266193410	Server Greeting proto=10 version=5.6.28
5	client	server	TCP	54	0	1608229139	48287→mysql(3306) [ACK] Seq=1608229139 Ack=4266193488 Win=29200 Len=0
6	client	server	MyS…	251	197	1608229139	Login Request user=ren
7	server	client	TCP	60	0	4266193488	mysql(3306)→48287 [ACK] Seq=4266193488 Ack=1608229336 Win=65535 Len=0
8	server	client	MyS…	65	11	4266193488	Response OK
9	client	server	MyS…	91	37	1608229336	Request Query { select @@version_comment limit 1 }
10	server	client	TCP	60	0	4266193499	mysql(3306)→48287 [ACK] Seq=4266193499 Ack=1608229373 Win=65535 Len=0
11	server	client	MyS…	153	99	4266193499	Response
12	client	server	MyS…	74	20	1608229373	Request Query { select sleep(2) }
13	server	client	TCP	60	0	4266193598	mysql(3306)→48287 [ACK] Seq=4266193598 Ack=1608229393 Win=65535 Len=0
18	server	client	MyS…	117	63	4266193598	Response
19	client	server	MyS…	59	5	1608229393	Request Quit
20	client	server	TCP	54	0	1608229398	48287→mysql(3306) [FIN, ACK] Seq=1608229398 Ack=4266193661 Win=29200 Len=0
21	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [ACK] Seq=4266193661 Ack=1608229398 Win=65535 Len=0
22	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [ACK] Seq=4266193661 Ack=1608229399 Win=65535 Len=0
23	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [FIN, ACK] Seq=4266193661 Ack=1608229399 Win=65535 Len=0
24	client	server	TCP	54	0	1608229399	48287→mysql(3306) [ACK] Seq=1608229399 Ack=4266193662 Win=29200 Len=0

image.png

四个红框表示断开连接的四次挥手：

- 第一步： client主动发送fin包给server
- 第二步： server回复ack（对应第一步fin包的ack）给client，表示server知道client要断开了
- 第三步： server发送fin包给client，表示server也可以断开了
- 第四部： client回复ack给server，表示既然双发都发送fin包表示断开，那么就真的断开吧

为什么握手三次、挥手四次

这个问题太恶心，面试官太喜欢问，其实他也许只能背诵：因为……。

我也不知道怎么回答。网上都说tcp是双向的，所以断开要四次。但是我认为建连接也是双向的（双向都协调告知对方自己的seq号），为什么不需要四次握手呢，所以网上说的不一定精准。

你再看三次握手的第二步发 syn+ack，如果拆分成两步先发ack再发syn完全也是可以的（效率略低），这样三次握手也变成四次握手了。

看起来挥手的时候多一次，主要是收到第一个fin包后单独回复了一个ack包，如果能回复fin+ack那么四次挥手也就变成三次了。 来看一个案例：

Proto	Length	TCP Segment Len	Sequence numbe	Info
TCP	66	0	1204583178	35208→http-alt(8080) [ACK] Seq=1204583178 Ack=2051138134 Win=31 Len=0 TSval=3673419830 TSecr=3203003999
TCP	66	0	1204583178	35208→http-alt(8080) [FIN, ACK] Seq=1204583178 Ack=2051138134 Win=31 Len=0 TSval=3673419830 TSecr=3203003999
TCP	66	0	2051138134	http-alt(8080)→35208 [FIN, ACK] Seq=2051138134 Ack=1204583179 Win=59 Len=0 TSval=3203004015 TSecr=3673419830
TCP	66	0	1204583179	35208→http-alt(8080) [ACK] Seq=1204583179 Ack=2051138135 Win=31 Len=0 TSval=3673419845 TSecr=3203004015

image.png

图中第二个红框就是回复的fin+ack，这样四次挥手变成三次了（如果一个包就是一次的话）。

我的理解：之所以绝大数时候我们看到的都是四次挥手，是因为收到fin后，知道对方要关闭了，然后OS通知应用层要关闭啥的，这里应用层可能需要做些准备工作，有一些延时，所以先回ack，准备好了再发fin 。握手过程没有这个准备过程所以可以立即发送syn+ack。

ack=seq+len

ack总是seq+len（包的大小），这样发送方明确知道server收到那些东西了。

但是特例是三次握手和四次挥手，虽然len都是0，但是syn和fin都要占用一个seq号，所以这里的ack都是seq+1。

No.	Source	Destination	Proto	Length	TCP Segment Len	Sequence number	Info
1	client	server	TCP	74	0	1608229138	48287→mysql(3306) [SYN] Seq=1608229138 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSva
2	server	client	TCP	60	0	4266193409	mysql(3306)→48287 [SYN, ACK] Seq=4266193409 Ack=1608229139 Win=65535 Len=0 MSS=1
3	client	server	TCP	54	0	1608229139	48287→mysql(3306) [ACK] Seq=1608229139 Ack=4266193410 Win=29200 Len=0
4	server	client	MyS…	132	78	4266193410	Server Greeting proto=10 version=5.6.28
5	client	server	TCP	54	0	1608229139	48287→mysql(3306) [ACK] Seq=1608229139 Ack=4266193488 Win=29200 Len=0
6	client	server	MyS…	251	197	1608229139	Login Request user=ren
7	server	client	TCP	60	0	4266193488	mysql(3306)→48287 [ACK] Seq=4266193488 Ack=1608229336 Win=65535 Len=0
8	server	client	MyS…	65	11	4266193488	Response OK
9	client	server	MyS…	91	37	1608229336	Request Query { select @@version_comment limit 1 }
10	server	client	TCP	60	0	4266193499	mysql(3306)→48287 [ACK] Seq=4266193499 Ack=1608229373 Win=65535 Len=0
11	server	client	MyS…	153	99	4266193499	Response
12	client	server	MyS…	74	20	1608229373	Request Query { select sleep(2) }
13	server	client	TCP	60	0	4266193598	mysql(3306)→48287 [ACK] Seq=4266193598 Ack=1608229393 Win=65535 Len=0
18	server	client	MyS…	117	63	4266193598	Response
19	client	server	MyS…	59	5	1608229393	Request Quit
20	client	server	TCP	54	0	1608229398	48287→mysql(3306) [FIN, ACK] Seq=1608229398 Ack=4266193661 Win=29200 Len=0
21	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [ACK] Seq=4266193661 Ack=1608229398 Win=65535 Len=0
22	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [ACK] Seq=4266193661 Ack=1608229399 Win=65535 Len=0
23	server	client	TCP	60	0	4266193661	mysql(3306)→48287 [FIN, ACK] Seq=4266193661 Ack=1608229399 Win=65535 Len=0
24	client	server	TCP	54	0	1608229399	48287→mysql(3306) [ACK] Seq=1608229399 Ack=4266193662 Win=29200 Len=0

image.png

看图中左边红框里的len+seq就是接收方回复的ack的数字，表示这个包接收方收到了。然后下一个包的seq就是前一个包的len+seq，依次增加，一旦中间发出去的东西没有收到ack就是丢包了，过一段时间（或者其他方式）触发重传，保障了tcp传输的可靠性。

三次握手中协商的其它信息

MSS 最大一个包中能传输的信息（不含tcp、ip包头），MSS+包头就是MTU（最大传输单元），如果MTU过大可能在传输的过程中被卡住过不去造成卡死（这个大小的包一直传输不过去），跟丢包还不一样。

SACK_PERM 用于丢包的话提升重传效率，比如client一次发了1、2、3、4、5 这5个包给server，实际server收到了 1、3、4、5这四个包，中间2丢掉了。这个时候server回复ack的时候，都只能回复2，表示2前面所有的包都收到了，给我发第二个包吧，如果server 收到3、4、5还是没有收到2的话，也是回复ack 2而不是回复ack 3、4、5、6的，表示快点发2过来。

但是这个时候client虽然知道2丢了，然后会重发2，但是不知道3、4、5有没有丢啊，实际3、4、5 server都收到了，如果支持sack，那么可以ack 2的时候同时告诉client 3、4、5都收到了，这样client重传的时候只重传2就可以，如果没有sack的话那么可能会重传2、3、4、5，这样效率就低了。

来看一个例子：

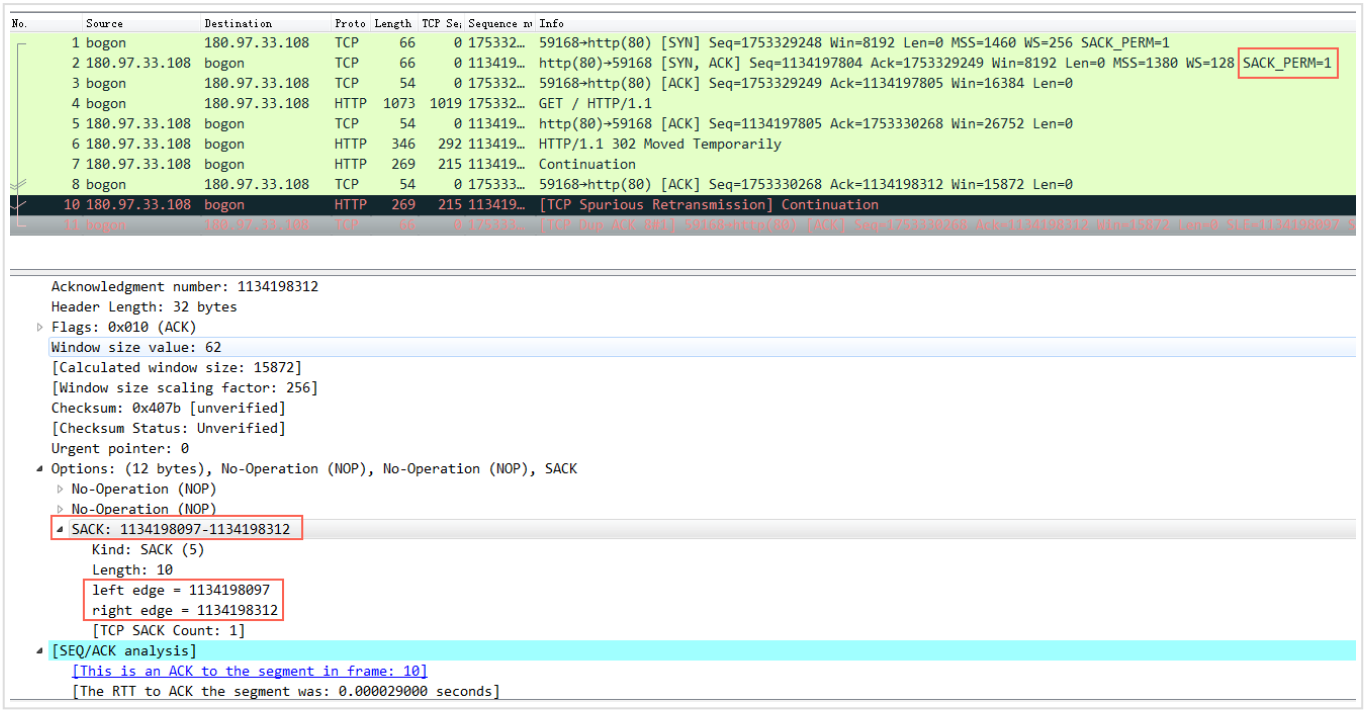

image.png

图中的红框就是SACK。

知识点：ack数字表示这个数字前面的数据**都**收到了。

总结下

tcp所有特性基本上核心都是为了**可靠传输**这个目标来服务的，然后有一些是出于优化性能的目的。

三次握手建连接的详细过程可以参考我这篇： 关于TCP 半连接队列和全连接队列

后续希望再通过几个案例来深化一下上面的知识。

说点关于学习的题外话

什么是工程效率，什么是知识效率

有些人纯看理论就能掌握好一门技能，还能举一反三，这是知识效率，这种人非常少；

大多数普通人都是看点知识然后结合实践来强化理论，要经过反反复复才能比较好地掌握一个知识，这就是工程效率，讲究技巧、工具来达到目的。

肯定知识效率最牛逼，但是拥有这种技能的人毕竟非常少。从小我们周边那种不怎么学的学霸型基本都是这类，这种学霸都还能触类旁通非常快的掌握一个新知识，非常气人。剩下的绝大部分只能拼时间+方法+总结等也能掌握一些知识。

非常遗憾我就是工程效率型，只能羡慕那些知识效率型的学霸。但是这事又不能独立看待有些人在某些方向上是工程效率型，有些方向就又是知识效率型（有一种知识效率型是你掌握的实在太多也就比较容易触类旁通了，这算灰色知识效率型）。

使劲挖掘自己在知识效率型方面的能力吧，即使灰色地带也行啊:)

> 企业级互联网架构Aliware，让您的业务能力云化：https://www.aliyun.com/aliware

#通信

‹ 就是要你懂 TCP | 最经典的TCP性能问题

面向万物互联的时序数据库HiTSDB ›