为什么 TCP 建立连接需要三次握手

2019-10-28 为什么这么设计 系统设计 TCP 三次握手

> 为什么这么设计（Why's THE Design）是一系列关于计算机领域中程序设计决策的文章，我们在这个系列的每一篇文章中都会提出一个具体的问题并从不同的角度讨论这种设计的优缺点、对具体实现造成的影响。如果你有想要了解的问题，可以在文章下面留言。

TCP 协议是我们几乎每天都会接触到的网络协议，绝大多数网络连接的建立都是基于 TCP 协议的，学过计算机网络或者对 TCP 协议稍有了解的人都知道 —— 使用 TCP 协议建立连接需要经过三次握手（three-way handshake）。

如果让我们简单说说 TCP 建立连接的过程，相信很多准备过面试的人都会非常了解，但是一旦想要深究『为什么 TCP 建立连接需要三次握手？』，作者相信大多数人都没有办法回答这个问题或者会给出错误的答案，这边文章就会讨论究竟为什么我们需要三次握手才能建立 TCP 连接？

> 需要注意的是我们会将重点放到为什么需要 TCP 建立连接需要**『三次握手』**，而*不仅仅*是为什么需要**『三次』**握手。

概述

在具体分析今天的问题之前，我们首先可以了解一下最常见的错误类比，这个对 TCP 连接过程的错误比喻误导了很多人，作者在比较长的一段时间内也认为它能够很好地描述 TCP 建立连接为什么需要三次握手：

1. 你听得到吗？
2. 我能听到，你听得到？
3. 我也能听到；

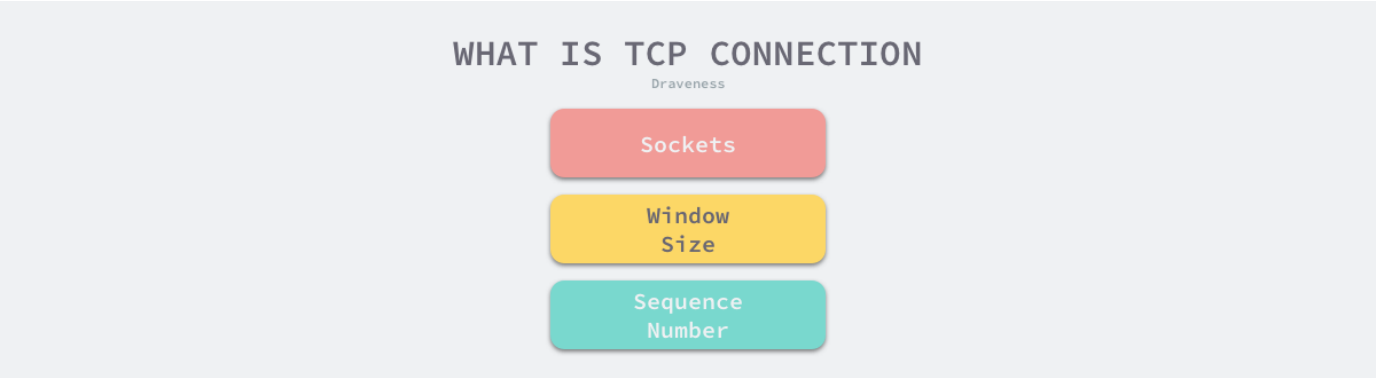
这种用类比来解释问题往往就会面临『十个类比九个错』的尴尬局面，如果别人用类比回答你的为什么，你需要仔细想一想它的类比里究竟哪里有漏洞；类比带来的解释往往只能有片面的相似性，我们永远也无法找到绝对正确的类比，它只在我们想要通俗易懂地展示事物的特性时才能发挥较大的作用，我们在文章的后面会介绍为什么这里的类比有问题，各位读者也可以带着疑问来阅读剩下的内容。

很多人尝试回答或者思考这个问题的时候其实关注点都放在了三次握手中的三次上面，这确实很重要，但是如果重新审视这个问题，我们对于『什么是连接』真的清楚？只有知道连接的定义，

我们才能去尝试回答为什么 TCP 建立连接需要三次握手。

> The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection.

RFC 793 - Transmission Control Protocol 文档中非常清楚地定义了 TCP 中的连接是什么，我们简单总结一下：用于保证可靠性和流控制机制的信息，包括 Socket、序列号以及窗口大小叫做连接。



所以，建立 TCP 连接就是通信的双方需要对上述的三种信息达成共识，连接中的一对 Socket 是由互联网地址标志符和端口组成的，窗口大小主要用来做流控制，最后的序列号是用来追踪通信发起方发送的数据包序号，接收方可以通过序列号向发送方确认某个数据包的成功接收。

到这里，我们将原有的问题转换成了『为什么需要通过三次握手才可以初始化 Sockets、窗口大小和初始序列号？』，那么接下来我们就开始对这个细化的问题进行分析并寻找解释。

设计

这篇文章主要会从以下几个方面介绍为什么我们需要通过三次握手才可以初始化 Sockets、窗口大小、初始序列号并建立 TCP 连接：

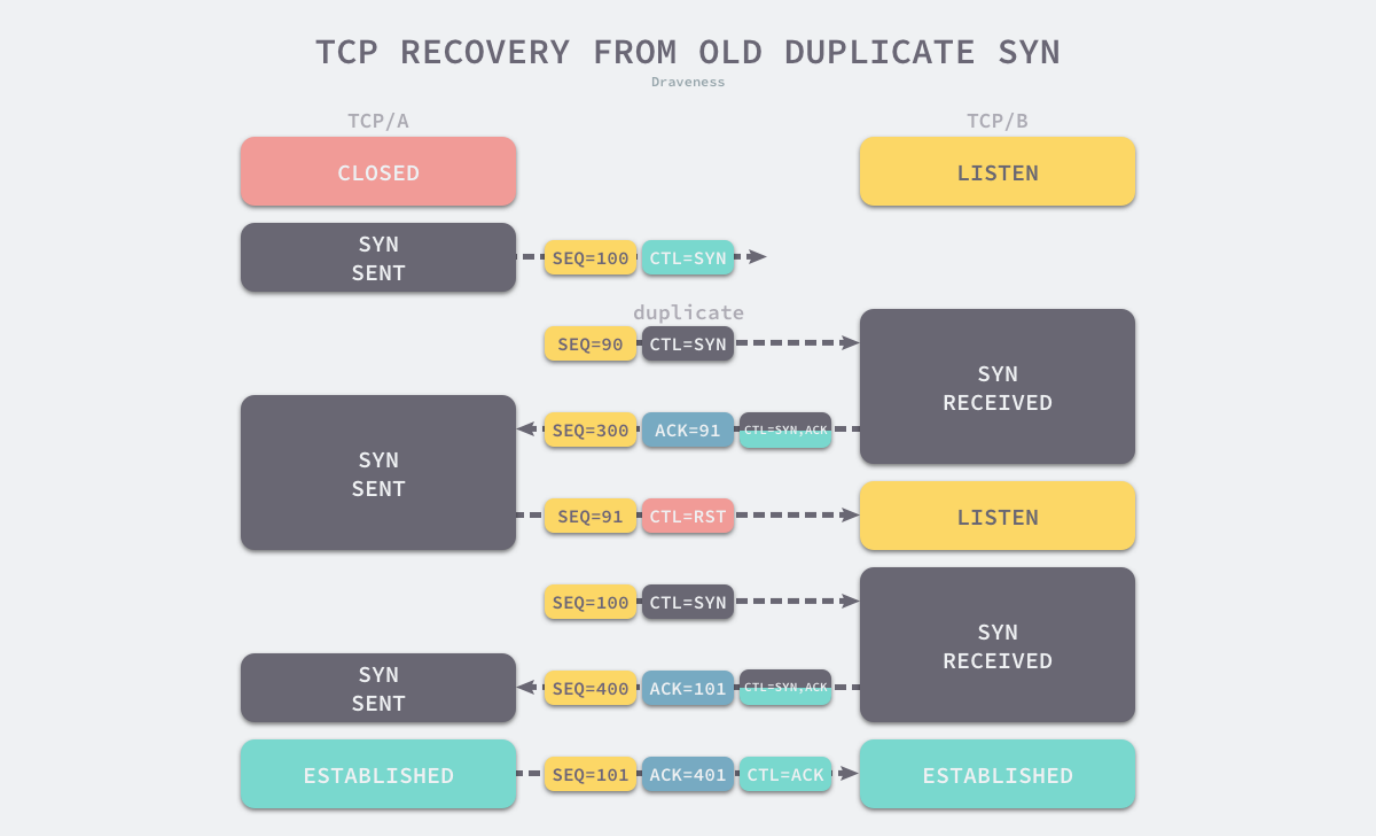
- 通过三次握手才能阻止重复历史连接的初始化；
- 通过三次握手才能对通信双方的初始序列号进行初始化；
- 讨论其他次数握手建立连接的可能性；

这几个论点中的第一个是 TCP 选择使用三次握手的最主要原因，其他的几个原因相比之下都是次要的原因，我们在这里对它们的讨论只是为了让整个视角更加丰富，通过多方面理解这一有趣的设计决策。

历史连接

RFC 793 - Transmission Control Protocol 其实就指出了 TCP 连接使用三次握手的首要原因 —— 为了阻止历史的重复连接初始化造成的混乱问题，防止使用 TCP 协议通信的双方建立了错误的连接。

> The principle reason for the three-way handshake is to prevent old duplicate connection initiations from causing confusion.



想象一下这个场景，如果通信双方的通信次数只有两次，那么发送方一旦发出建立连接的请求之后它就没有办法撤回这一次请求，如果在网络状况复杂或者较差的网络中，发送方连续发送多次建立连接的请求，如果 TCP 建立连接只能通信两次，那么接收方只能选择接受或者拒绝发送方发起的请求，它并不清楚这一次请求是不是由于网络拥堵而早早过期的连接。

所以，TCP 选择使用三次握手来建立连接并在连接引入了 RST 这一控制消息，接收方当收到请求时会将发送方发来的 SEQ+1 发送给对方，这时由发送方来判断当前连接是否是历史连接：

- 如果当前连接是历史连接，即 SEQ 过期或者超时，那么发送方就会直接发送 RST 控制消息中止这一次连接；
- 如果当前连接不是历史连接，那么发送方就会发送 ACK 控制消息，通信双方就会成功建立连接；

使用三次握手和 RST 控制消息将是否建立连接的最终控制权交给了发送方，因为只有发送方有足够的上下文来判断当前连接是否是错误的或者过期的，这也是 TCP 使用三次握手建立连接的最主要原因。

初始序列号

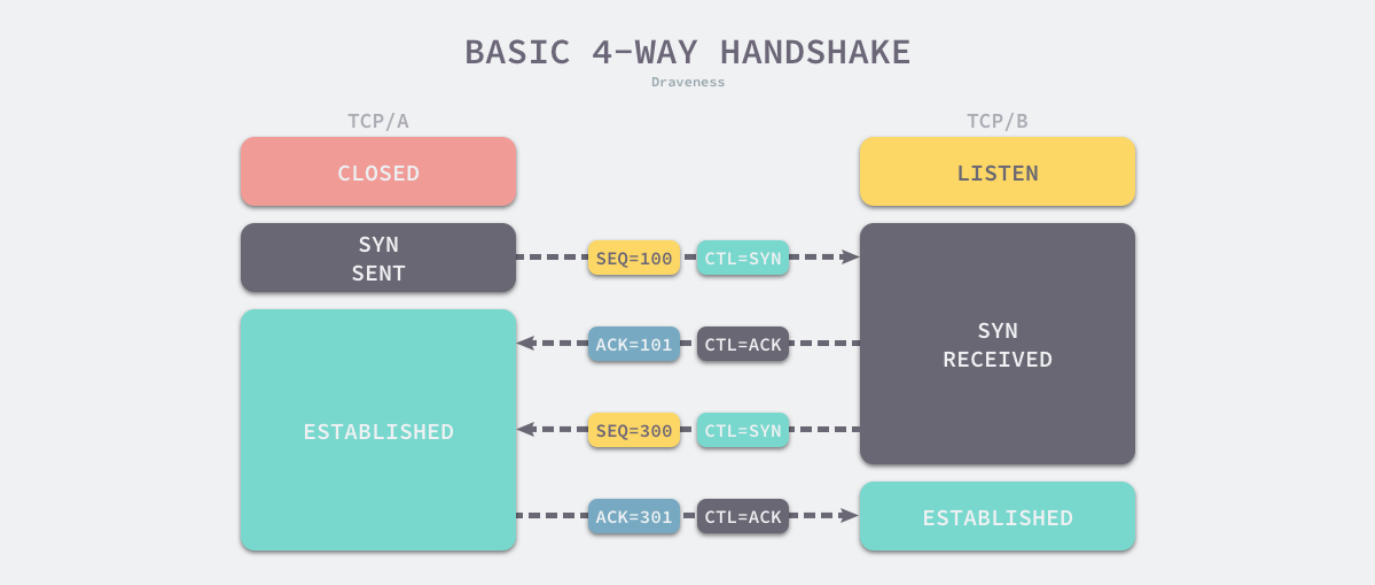
另一个使用三次握手的重要的原因就是通信双方都需要获得一个用于发送信息的初始化序列号，作为一个可靠的传输层协议，TCP 需要在不稳定的网络环境中构建一个可靠的传输层，网络的不确定性可能会导致数据包的缺失和顺序颠倒等问题，常见的问题可能包括：

- 数据包被发送方多次发送造成数据的重复；
- 数据包在传输的过程中被路由或者其他节点丢失；
- 数据包到达接收方可能无法按照发送顺序；

为了解决上述这些可能存在的问题，TCP 协议要求发送方在数据包中加入『序列号』字段，有了数据包对应的序列号，我们就可以：

- 接收方可以通过序列号对重复的数据包进行去重；
- 发送方会在对应数据包未被 ACK 时进行重复发送；
- 接收方可以根据数据包的序列号对它们进行重新排序；

序列号在 TCP 连接中有着非常重要的作用，初始序列号作为 TCP 连接的一部分也需要在三次握手期间进行初始化，由于 TCP 连接通信的双方都需要获得初始序列号，所以它们其实需要向对方发送 SYN 控制消息并携带自己期望的初始化序列号 SEQ，对方在收到 SYN 消息之后会通过 ACK 控制消息以及 SEQ+1 来进行确认。



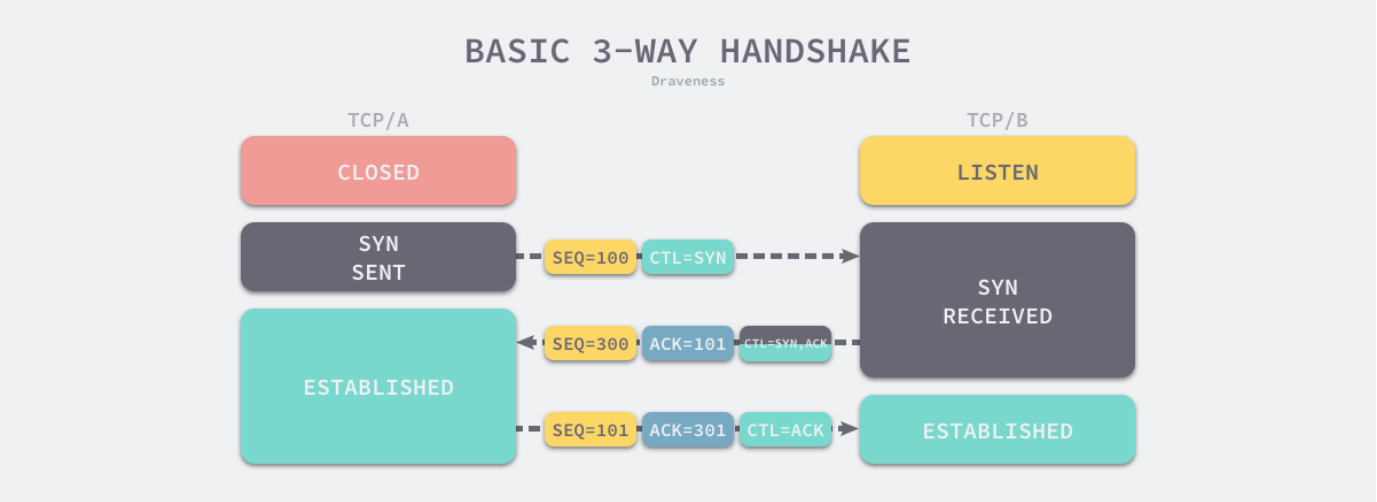
如上图所示，通信双方的两个 TCP A/B 分别向对方发送 SYN 和 ACK 控制消息，等待通信双方都获取到了自己期望的初始化序列号之后就可以开始通信了，由于 TCP 消息头的设计，我们可以将中间的两次通信合成一个，TCP B 可以向 TCP A 同时发送 ACK 和 SYN 控制消息，这也就帮助我们将四次通信减少至三次。

> A three way handshake is necessary because sequence numbers are not tied to a global clock in the network, and TCPs may have different mechanisms for picking the ISN's. The receiver of the first SYN has no way of knowing whether the segment was an old delayed one or not, unless it remembers the last sequence number used on the connection (which is not always possible), and so it must ask the sender to verify this SYN. The three way handshake and the advantages of a clock-driven scheme are discussed in [3].

除此之外，网络作为一个分布式的系统，其中并不存在一个用于计数的全局时钟，而 TCP 可以通过不同的机制来初始化序列号，作为 TCP 连接的接收方我们无法判断对方传来的初始化序列号是否过期，所以我们需要交由对方来判断，TCP 连接的发起方可以通过保存发出的序列号判断连接是否过期，如果让接收方来保存并判断序列号却是不现实的，这也再一次强化了我们在上一节中提出的观点 —— 避免历史错连接的初始化。

通信次数

当我们讨论 TCP 建立连接需要的通信次数时，我们经常会执着于为什么通信三次才可以建立连接，而不是两次或者四次；讨论使用更多的通信次数来建立连接往往是没有意义的，因为我们总可以使用更多的通信次数交换相同的信息，所以使用四次、五次或者更多次数建立连接在技术上都是完全可以实现的。



这种增加 TCP 连接通信次数的问题往往没有讨论的必要性，我们追求的其实是用更少的通信次数（理论上的边界）完成信息的交换，也就是为什么我们在上两节中也一再强调使用『两次握手』没有办法建立 TCP 连接，使用三次握手是建立连接所需要的最小次数。

总结

我们在这篇文章中讨论了为什么 TCP 建立连接需要经过三次握手，在具体分析这个问题之前，我们首先重新思考了 TCP 连接究竟是什么，RFC 793 - Transmission Control Protocol - IETF Tools 对 TCP 连接有着非常清楚的定义 —— 用于保证可靠性和流控制机制的数据，包括 Socket、序列号以及窗口大小。

TCP 建立连接时通过三次握手可以有效地避免历史错误连接的建立，减少通信双方不必要的资源消耗，三次握手能够帮助通信双方获取初始化序列号，它们能够保证数据包传输的不重不丢，还能保证它们的传输顺序，不会因为网络传输的问题发生混乱，到这里不使用『两次握手』和『四次握手』的原因已经非常清楚了：

- 『两次握手』：无法避免历史错误连接的初始化，浪费接收方的资源；
- 『四次握手』：TCP 协议的设计可以让我们同时传递 ACK 和 SYN 两个控制信息，减少了通信次数，所以不需要使用更多的通信次数传输相同的信息；

我们重新回到在文章开头提的问题，为什么使用类比解释 TCP 使用三次握手是错误的？这主要还是因为，这个类比没有解释清楚核心问题 —— 避免历史上的重复连接。到最后，我们还是来看一些比较开放的相关问题，有兴趣的读者可以仔细想一下下面的问题：

- 除了使用序列号是否还有其他方式保证消息的不重不丢？
- UDP 协议有连接的概念么，它能保证数据传输的可靠么？

如果对文章中的内容有疑问或者想要了解更多软件工程上一些设计决策背后的原因，可以在博客下面留言，作者会及时回复本文相关的疑问并选择其中合适的主题作为后续的内容。

Reference

- RFC 793 - Transmission Control Protocol - IETF Tools
- Why do we need a 3-way handshake? Why not just 2-way?

{% include related/whys-the-design.md %}

转载申请

Go 语言设计与实现

各位读者朋友，很高兴大家通过本博客学习 Go 语言，感谢一路相伴！ 《Go语言设计与实现》 的纸质版图书已经上架京东，本书目前已经四印，印数超过 10,000 册，有需要的朋友请点击 链接 或者下面的图片购买。

文章图片

你可以在 技术文章配图指南 中找到画图的方法和素材。

0 个反应

70 条评论 · 1+ 条回复 – *由 giscus 驱动*

最旧 最新

输入 预览 Aa

登录以评论

支持 Markdown GitHub 登录

ttttmr 2019 年 11 月 1 日

为什么DNS使用UDP?
也是面试问到了，我觉得是节省服务端资源，响应快，面试官好像觉得回答的不好，一直问还有呢，来请教一下博主

↑ 1 ❤ 4 0 条回复

draveness 2019 年 11 月 1 日 所有者

> 为什么DNS使用UDP?
> 也是面试问到了，我觉得是节省服务端资源，响应快，面试官好像觉得回答的不好，一直问还有呢，来请教一下博主

这是一个非常好的问题，我会分析一下的，多谢

↑ 1 0 条回复

beihai0xff 2019 年 11 月 1 日

为什么博客突然间无法访问了，能上谷歌，看着看着就无连接了

↑ 1 0 条回复

draveness 2019 年 11 月 1 日 所有者

> 为什么博客突然间无法访问了，能上谷歌，看着看着就无连接了

现在好了么，可能是服务器网络抖动

↑ 1 ☺ 0 条回复

 **beihai0xff** 2019 年 11 月 1 日

> 为什么博客突然间无法访问了，能上谷歌，看着看着就无连接了
>
> 现在好了么，可能是服务器网络抖动

现在可以了，这几天一直是这样，还以为是我梯子问题

↑ 1 ☺ 0 条回复

 **XiaTianliang** 2019 年 11 月 3 日

> **@ttttmr**
> 为什么DNS使用UDP?
> 也是面试问到了，我觉得是节省服务端资源，响应快，面试官好像觉得回答的不好，一直问还有呢，来请教一下博主

dns交换的信息很短，为了这么简短的请求建立tcp觉得太浪费了。

↑ 1 ☺ 0 条回复

 **leafduo** 2019 年 11 月 6 日

> **@ttttmr**
> 为什么DNS使用UDP?
> 也是面试问到了，我觉得是节省服务端资源，响应快，面试官好像觉得回答的不好，一直问还有呢，来请教一下博主

当时可能是节省时延和服务器资源之类的吧，现在来看 DNS 使用 UDP 的弊端暴露地越来越多了，比如安全、隐私和流量放大之类的问题

↑ 1 ☺ 0 条回复

 **leafduo** 2019 年 11 月 6 日

看完就想到了 TCP Fast Open，好奇它是怎么不用三次握手然后做到拒绝重复连接的，结果它直接接受了重复连接，要求应用层自己判断只有幂等操作才用 Fast Open。这个复杂性可能也是推不下去的原因之一吧。

↑ 1 ☺ ❤ 1 0 条回复

geckofu 2019 年 11 月 6 日

DNS 用的是 Anycast IP 地址, 每次查询都可能和不同的机器通信, 所以没有办法维持状态

↑ 1 ☺ 0 条回复

ttttmr 2019 年 11 月 6 日

> DNS 用的是 Anycast IP 地址, 每次查询都可能和不同的机器通信, 所以没有办法维持状态

好像有点道理，那么问题又来了😂，DoH肯定是基于TCP，比如1.1.1.1的那个，这是怎么做到的呢

↑ 1 ☺ 0 条回复

houbaron 2019 年 11 月 6 日

请问这些图都是用什么软件绘制的？似乎看过类似的用 Sketch 画的，如果是 Sketch，能分享下素材吗

↑ 1 ☺ 0 条回复

geckofu 2019 年 11 月 6 日

> **@ttttmr**
>
> > DNS 用的是 Anycast IP 地址, 每次查询都可能和不同的机器通信, 所以没有办法维持状态
> >
> > 好像有点道理，那么问题又来了😂，DoH肯定是基于TCP，比如1.1.1.1的那个，这是怎么做到的呢

我也不了解了 😂 还想到一个问题, 为什么 IP 可以上 HTTPS？不是域名才可以的么?

↑ 1 ☺ 0 条回复

leafduo 2019 年 11 月 6 日

> **@geckofu**
> DNS 用的是 Anycast IP 地址, 每次查询都可能和不同的机器通信, 所以没有办法维持状态

只有少数公共 DNS 是 anycast，并且实践中大部分连接是能保持连接状态的。

https://serverfault.com/questions/495719/anycast-dns-how-do-you-deal-with-tcp-dns-requests

↑ 1 ☺ ❤ 1 0 条回复

Maecenas 2019 年 11 月 11 日

> **@ttttmr**
> 为什么DNS使用UDP?
> 也是面试问到了，我觉得是节省服务端资源，响应快，面试官好像觉得回答的不好，一直问还有呢，来请教一下博主

DNS是一个比较简单的机制，这可能不是选择UDP的主要原因。TCP提供的、两个端点之间的可靠连接不是DNS需要的，DNS协议的可靠连接机制是靠多组冗余的DNS服务器实现的。DNS只有在需要可靠传输数据（zone transfer）和数据量较大（> 512 bytes）才会使用TCP。

↑ 1 ☺ 0 条回复

rsj217 2019 年 11 月 13 日

> 这种用类比来解释问题往往就会面临『十个类比九个错』的尴尬局面

是的，最开始学习写笔记的时候也喜欢这样类比，看起来貌似很 巧妙 ，实则很多非但不能解释清楚问题，还带来新的困惑。

↑ 1 ☺ 0 条回复

40 个隐藏项
加载更多...

oldthreefeng 2020 年 9 月 9 日 已编辑

@draveness 文章转载了~ 写的特别好。

↑ 1 ☺ 0 条回复

draveness 2020 年 9 月 9 日 所有者

> @draveness 文章转载了~ 写的特别好。

文章下面有如何转载的提示

↑ 1 ☺ 0 条回复

879913935 2020 年 9 月 16 日

除了使用序列号是否还有其他方式保证消息的不重不丢?

这个没理解，tcp如果没有序号，还能用什么方式实现

↑ 1 ☺ 0 条回复

zinwalin 2020 年 10 月 20 日

{% include related/whys-the-design.md %}

↑ 1 ☺ 0 条回复

Gpia 2020 年 11 月 11 日 已编辑

从标准定义看，连接包含：Socket、序列号、窗口大小和一些控制信息。
发送方和接收方协商的只有序列号和控制信息。这里的socket和窗口大小的初始化是发送方自己定的吧。
接收方一直是listen状态。

↑ 1 ☺ 0 条回复

hzh-test 2020 年 12 月 15 日

发送端发送连接请求如果没有收到确认，重发序列号是相同的（使用curl请求，wireshark抓包已证明），但是重试次数是有限的，之后用户可能会手动再次触发新的连接请求，此时的序列号是不同的，这时接受端就有可能收到多个不同的连接请求，序列号相同接受端可以辨认，但是这里是存在两个不同序列号的请求。如果只有两次握手，接收端就需要维护两个连接，但其中有一个是无用的，这就浪费资源了。这和 DDOS 攻击最大的区别是这个是非人为的无意的，但却会造成更大的危害。另外，发送端是可以伪造源IP地址的，这种接收端连的其实都不是发送端，这还能叫连接吗。
究其根源，是网络可不可达导致的问题。从发送端发出连接请求到发送端收到消息确认，在客户端看来确实是证明了网络可达并且连接已经建立，但是在接收端看来，单单收到连接请求并不能证明什么，因为这个连接请求有可能是旧的延迟的连接请求，此时客户端可能早就下线了。所以从接收端的角度来看，其也需要确认网络可达（或者说接收端在线），所以客户端收到消息确认后还得再回复一下接收端，这样一来一回就实现了双向证明。
综上，就算发送端伪造源IP地址，连接也无法建立，因为接收端发送的消息客户端收不到，自然也就不会再收到来自客户端的确认消息。另外，非人为的无意的类似 DDOS 攻击的行为也不会发生了，两全其美，一箭双雕。

这个问题面试遇到过，当时也只是在回答三次握手的最后一步，之后越想越不对劲，搜了很多答案都不满意，今天看到博主的文章，终于是彻底地把自己给说服了。

↑ 1 ☺ 0 条回复

pkwenda 2021 年 1 月 15 日

大佬用的这是什么画图工具?

↑ 1 ☺ 0 条回复

Dcell 2021 年 2 月 7 日

希望大佬能做一篇 http 协议的进化史，非常期待

↑ 1 ☺ 0 条回复

xinnjie 2021 年 3 月 3 日

> **@hzh-test**
> 发送端发送连接请求如果没有收到确认，重发序列号是相同的（使用curl请求，wireshark抓包已证明），但是重试次数是有限的，之后用户可能会手动再次触发新的连接请求，此时的序列号是不同的，这时接受端就有可能收到多个不同的连接请求，序列号相同接受端可以辨认，但是这里是存在两个不同序列号的请求。如果只有两次握手，接收端就需要维护两个连接，但其中有一个是无用的，这就浪费资源了。这和 DDOS 攻击最大的区别是这个是非人为的无意的，但却会造成更大的危害。另外，发送端是可以伪造源IP地址的，这种接收端连的其实都不是发送端，这还能叫连接吗。
> 究其根源，是网络可不可达导致的问题。从发送端发出连接请求到发送端收到消息确认，在客户端看来确实是证明了网络可达并且连接已经建立，但是在接收端看来，单单收到连接请求并不能证明什么，因为这个连接请求有可能是旧的延迟的连接请求，此时客户端可能早就下线了。所以从接收端的角度来看，其也需要确认网络可达（或者说接收端在线），所以客户端收到消息确认后还得再回复一下接收端，这样一来一回就实现了双向证明。
> 综上，就算发送端伪造源IP地址，连接也无法建立，因为接收端发送的消息客户端收不到，自然也就不会再收到来自客户端的确认消息。另外，非人为的无意的类似 DDOS 攻击的行为也不会发生了，两全其美，一箭双雕。
>
> 这个问题面试遇到过，当时也只是在回答三次握手的最后一步，之后越想越不对劲，搜了很多答案都不满意，今天看到博主的文章，终于是彻底地把自己给说服了。

手动点赞

↑ 1 ☺ 0 条回复

Maecenas 2021 年 3 月 16 日 已编辑

> **@birdkyle7918**
> 作者你好，请问你的图是用什么软件画的呀？很漂亮

最后一行：文章图片
你可以在 技术文章配图指南 中找到画图的方法和素材。

↑ 1 ☺ 0 条回复

LemonFish873310466 2021 年 9 月 10 日

> 想象一下这个场景，如果通信双方的通信次数只有两次，那么发送方一旦发出建立连接的请求之后它就没有办法撤回这一次请求，如果在网络状况复杂或者较差的网络中，发送方连续发送多次建立连接的请求，如果 TCP 建立连接只能通信两次，那么接收方只能选择接受或者拒绝发送方发起的请求，它并不清楚这一次请求是不是由于网络拥堵而早早过期的连接。

想问一下作者这边为什么server侧不能判断是旧的连接呢，根据序列号之类的不可以吗

↑ 1 ☺ 1 条回复

yzbmz5913 2022 年 7 月 1 日 ☺

因为序列号是随机初始化的，接收方收到一个更小的序列号可能是历史连接中延迟到达的，也可能是新一轮初始化的序列号

uknfire 2021 年 11 月 4 日

总结来讲，在TCP握手中，因为发起方知道更多的背景信息，所以应该由发起方主导控制这个连接。
因此协议给了发起方第二次发送消息的机会，根据已知的信息发送ACK接受连接，或者发送RST中止连接

↑ 1 ☺ 0 条回复

xincheng1997 2021 年 11 月 26 日

除了使用序列号是否还有其他方式保证消息的不重不丢？
有哪些实现呢？

↑ 1 ☺ 0 条回复

BBplux 2022 年 1 月 11 日

三次握手有了怎么能少了四次挥手

↑ 1 ☺ ❤ 1 0 条回复

vc60er 2022 年 2 月 21 日

期待博主的四次挥手

↑ 1 ☺ 0 条回复