TCP 协议“三次握手”的误解和正确认识

缪志敏 刘莹 岳淑贞

中国人民解放军陆军工程大学指挥控制工程学院，南京 210007

摘 要 TCP 协议需要在传输数据之前采用“三次握手”的方式建立双向的逻辑连接。但是对于采用“三次握手”来保证正确可靠的原因和其中的一些深层次内涵，则是存在较多误解。本文介绍了 TCP 协议连接建立时“握手”的必要性和起始序号选择方法演变，并深入解读了“三次握手”的相关内涵。

关键字 TCP，连接管理，三次握手

Misunderstanding and Correct Understanding of TCP Protocol's "Three Handshakes"

Miao zhimin Liu ying Yue shuzhen

College of Command and Control Engineering, Army Engineering University of PLA, Nanjing 210007, China; olivermiao@126.com

Abstract—A bidirectional logical connection is established applying a "three way handshake" method in TCP protocol before transmitting data. However, there are many misunderstandings about the reasons for using "three way handshake" to ensure accuracy and reliability. In this paper, the necessity of "handshake" during the establishment of TCP protocol connections and the evolution of the method for selecting starting sequence numbers are introduced, and the relevant connotations of "three way handshake"is deeply interpreted.

Keywords— TCP, connection management, three way handshake

1 引 言

学过计算机网络的人都知道， TCP/IP 是互联网的最核心协议之一。TCP 协议在 IP 协议之上实现端到端用户（应用进程）之间的可靠数据传输，但其依托的 IP 协议却是不太可靠的，TCP 协议交由 IP 协议传输的报文段可能会丢失或不按顺序到达。为了保证用户数据的可靠传输，TCP 引入了字节编号、滑动窗口、累积确认和超时重发等机制实现每个数据字节的按序可靠交付。

这些机制需要收发双方的TCP协议维护数据交换过程的状态信息，比如接收窗口。另外，为了保证数据传输可靠，TCP 协议的收发双方必须确定对方当前能够接收数据，并分配接收窗口的数据缓冲区，建立 TCB（TCP Control Block）。这些 TCB 和状态信息的初始化和维护就依靠收发进程之间的 TCP 连接来实现。由于连接需要占用计算机资源，所以在需要时打开连接，在所有数据交付后关闭连接，连接打开和关闭统称为连接管理。

一个TCP连接的唯一标识是该连接两端的套接字，即（IP1:Port1,IP2:Port2）。一对套接字之间的 TCP 连接会反复建立和拆除，TCP 协议必须让接收方有能力辨别收到的报文段是不是当前连接的。这个能力就依靠 TCP 连接建立时的“三次握手”来实现。如果 TCP 的连接管理机制设计不当，可能导致 TCP 连接死锁，甚至将旧连接中的数据错误地作为当前连接的数据交付给用户。

2 TCP 连接建立的“三次握手”需求

Charles M.Kozierok 的“The TCP/IP Guide”[1]一书第 47 章“TCP 基本操作：连接的建立、管理和终止”中讲到，如果每次建立连接都从默认序号（比如 1）开始发送的话，有可能会造成前后不同连接的 TCP 报文段相互混淆。为了实现 TCP 协议的可靠传输，必须保证接收端能够正确区分是此前已经关闭连接的报文段还是当前连接的报文段。有多种办法可以解决上述问题。方法之一是每次建立连接都使用不同的端口号；方法之二是增加一个连接号字段，用于区分不同连接；方法之三是采用比较大的序号空间，大到足以保证端到端之间传输的数据单元在最大可能延迟时间内不会有重复的序号出现，并且在建立新连接时，起始序号在上一次连接的末尾序号上递增；方法之四是旧连接关闭后等待一个报文段寿命期再建立新的连接，保证前一连接的报文段都已不在网络中传输了。但是，以上

这些方法都有一个共同的缺陷，即要求 TCP 记住前面都用过了哪些（端口号、连接号、末尾序号或连接关闭时间），最怕宕机，一旦 TCP 甚至计算机崩溃重启，以前的记忆就全没有了，TCP 的可靠数据传输就有可能无法保证。

TCP 的连接管理参考了上述这些方法，采用报文段中的字节序号来标识一个连接中传输的数据，再通过三次握手来保证接收方有足够的信息可以辨认出旧连接的报文段。Tomlinson 于 1978 年在其论文“Selecting Sequence Numbers”[3]中提出了用“three way handshake”来解决新旧连接报文段辨识问题。Tomlinson 解决方案的本质是在建立连接时做好两件事，一是恰当选择数据字节起始序号，每次连接不同；二是起始序号在连接建立时由收发双方正确同步，互相确认。

关于起始序号选择，Tomlinson[3]的思路是，在网络层的 IP 协议对传输报文有寿命 T 限制和 TCP 的序号空间足够大（32 比特）的基础上，每次新建立连接都选择一个“很不一样”的起始序号（Initial Sequence Number，ISN）开始发送。 既然每次新建立连接时都必须由每个发送端各自选择一个与以往不同的数据起始序号（接收端不知道），接收端与发送端之间就必须进行起始序号同步（让接收端知道）。同步过程通过连接请求报文段和连接确认报文段来实现。连接发起的客户端 A 要在其连接请求 SYN 报文段的序号(SequenceNumber)字段中填入其选择的 ISNa，服务器端 B 在其连接响应 ACK 报文段中进行确认（ACK 序号值为 ISNa+1）。同样的原理，服务器端 B 也要在其 SYN 报文段中给出 ISNb，客户端 A 必须对该序号进行确认（ACK=ISNb+1）。这个过程就是连接建立的“握手”，通过“握手”过程让每个连接方向的接收端知道（确认）了发送端选择的起始序号。在序号确认之前是不能发送数据的，因此 SYN 报文段不能携带数据。Tomlinson 认为，这个确认起始序号的“握手”过程必须且只能采用 “three way handshake” 才不至于发生错误（详见第三节）。

在国内最流行的《计算机网络（第 8 版）》（谢希仁编著）[4]中首次将 three way handshake 翻译为三报文握手，此前都是翻译成三次握手。本文中暂时还是统一用“三次握手”，以便跟大家此前的认识关联更紧密一些。

3 TCP 连接的起始序号选择

起始序号选择的目的在前一节已经讲明，每次新连接都要选择一个“很不一样”的 ISN，以保证在新连接建立后，即使前一个连接的报文段到达接收端，其报文段的数据序号也不会落在接收窗口内。

首先，报文段（在 IP 层也叫分组）在网络中是有寿命（生存期）的（记为 T），过了这个时间还没有到达接收端的报文段都会被 IP 层丢弃。其次，TCP 的可靠传输机制采用接收窗口来判断一个报文段是否要收下，序号落在窗口内的就收下。

根据 Tomlinson 的分析，如果报文段的寿命是 T，那么一个序号使用（标识特定数据）后，在 T 时间内就不能再用于标识后续的数据，这可以称为“序号禁区”。图中禁区的左边实线就是序号的实际使用曲线，序号禁区的水平宽度就是 T。报文段的寿命 T 和接收窗口（序号范围）大小将会影响（限制）一个连接的最大传输速率（序号的使用速率），因此要综合权衡 T、最大传输速率和序号位数之间的关系。众所周知，TCP 选择了 32 位序号，足够大。

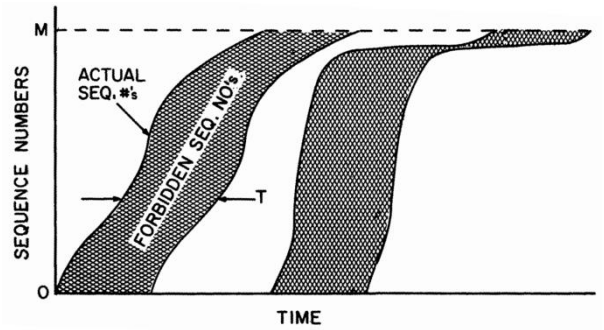


图 1 TCP 报文段序号的禁区问题

3．1 早期 TCP 协议的序号选择方法

前面提到过的方法之三，是采用比较大的序号空间，并且在建立新连接时，起始序号在上一次连接的末尾序号上递增。这个方法能够只利用序号就实现新旧连接报文段的辨识问题，但怕宕机重启。

在上述方法基础上，Tomlinson（1978）[3]提出了利用本地时钟来选择初始序号 ISN 的方法，并解释了序号选择和使用的一些限制。早期 TCP 协议（RFC 793[2]）就采用了本地时钟生成 ISN 方案：利用一个基于机器时钟的计数器来选择 ISN。根据早期 TCP 协议的约定，这个计数器占用 32 比特，每 4 微秒加 1，达到 32 比特数的最大值(4294967295)后回绕到 0 重新计数。

利用本地时钟来选择初始序号 ISN 的方法，也是有限制的。在 Tomlinson（1978）[3]的基础上，Tanenbaum 在其著名教材《Computer Networks(second Edition,1989)》[5]给出了更容易理解的描述，说明了基于时钟的 ISN 选取存在报文段序号禁区问题图中实线是基于时钟决定的 ISN 取值线，阴影区就是序号使用的禁区，在某个时刻，禁区内的序号是不能使用的，否则会有接收方无法辨识旧连接报文段的风险，TCP 可靠数据传输需要排除所有这些风险。在实际中，ISN 取值线应该是阶梯形的，因为时钟取值是离散的。图

2 中还画出了 3 种（虚线）实际序号使用情况：如果实际使用序号的速率快于 ISN 增长速率，序号使用就会遇到禁区（右侧虚线，从下面进入禁区）；如果连接持续的时间足够长，发送方持续较长时间发送，实际使用序号也迟早将遇到序号禁区（左侧虚线，从上面进入禁区）。用禁区中的序号发送新的分组，就有可能前后两个内容不同的分组使用了相同的序号，而间隔却未达到分组寿命 T。要是这时旧连接关闭，正好新连接打开，旧连接的报文就有可能先于新连接的报文到达接收方，造成数据错认。解决办法是，发送速率不能快于 ISN 增长速率，并且发送方时刻监视发送的序号是否遇到序号禁区，在进入禁区之前要重新同步序号，又从当时的 ISN 开始发送；或者暂停发送 T 时间也行。序号禁区的存在也说明了，一个 TCP 连接的存续时间不宜太长，如果建立新连接能够用新的端口会更加安全。

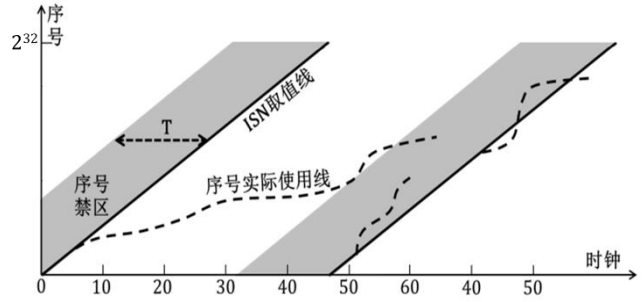


图 2 基于时钟选择 ISN 的禁区现象

3．2 当下 TCP 协议的序号选择方法

1985 年, Morris [6]描述了一种通过猜测 TCP 连接初始序号进行网络攻击的方法。著名的“The TCP/IP Guide”[1]一书也给出了解释：“基于时钟的 ISN 选择”这种方法存在着一个严重问题，因为这样就使得每次 TCP 连接的 ISN 可以预测。互联网上的黑客可以侦听到 TCP 连接建立报文段，根据 TCP 连接建立报文段的发送时间及其 ISN 对每个网络终端的 ISN 进行分析，然后就可以预测（猜测）该终端每一次 TCP 连接建立的 ISN。已经有黑客已经利用过这一 TCP 安全漏洞，比如著名的 Mitnick 欺骗攻击[7]就利用了 TCP 起始序号猜测技术。

为了封堵上述 TCP 安全漏洞，避免 ISN 可推测，当下的 TCP 协议通过随机数选 ISN。通过随机数来产生 ISN 当然可以杜绝 TCP 起始序号猜测，但随机数也不是随便选的。要保证前后两次随机数差额足够大，使得初始序号不会落在当时的“序号禁区”内，尽管 TCP 的序号空间很大（232），完全随机的选也是无法保证的，还是有一定的概率造成已失效或关闭连接的报文段到达接收端被误认为新连接的报文段。

为了克服完全的随机数可能造成的初始序号问题，又能确保外部即使侦听到以往连接的 ISN 值，也无法轻易推算得出下一次的 ISN 值。 RFC6528 [8]是有关 TCP 起始序号选取的最新一个文档，其中规定了一个与时钟序号相结合的起始序号 ISN 随机生成算法：

ISN = M + F(localip, localport, remoteip, remoteport, secretkey)

M 就是原有基于时钟的序号计数器，32 位，每隔 4 微妙加 1，用来保证随机数也不会太随机；

F 是一个伪随机函数，这个 F 函数必须是外部不可计算的。这个 F 函数可以是 TCP 连接套接字（源 IP 地址、目的 IP 地址、源端口号、目的端口号）和一个本机秘密数据的一个加密散列（hash）函数，RFC1321 [9]中的 MD5 就是比较好的散列函数。

其中 M 每 4μs 加 1，只适用于 2Mb/s 的传输速率（考虑到图 2 的序号禁区），现在的系统应采用更快的计数器，比如 64ns 加 1。

其中 F 函数用到的秘密数据 secretkey 的私密性是 F 函数值私密性的保证，很重要，要有足够的长度，但 128 位应该够了。

4 TCP 连接建立最好“三次握手”

Tomlinson 首先提出了“three way handshake”，而后 Sunshine 和 Dalal 在其论文“Connection Management in Transport Protocols”[11]中给出了更加透彻的解释，把他归入“连接管理”过程中，说明了“三次握手”可以正确识别过时的连接请求报文段，不会误建连接。

TCP 协议的连接建立过程需要完成两项工作，一是确认对方已经做好接收数据前的准备（比如分配接收缓存）；二是完成 TCP 报文段中数据字节序号（sequence number ）的同步，使得收发双方都确认从同一个序号开始发送和接收。TCP 每个连接方向上发送端选择的第一个数据字节的编号（32 比特）即起始序号 x 和 y 在握手的过程中得到各自的接收端确认。

TCP 的连接建立过程为什么只能“三次握手”，而不能“四次握手”。

如果采用“四次握手”，连接建立过程应该是这样的：A 向 B 发起连接（发出 SYN 报文段，起始序号 ISNa），B 收到后回复 ACK 报文段（ACK 序号=ISNa+1），并且 B 向 A 发出反方向的 SYN 报文段，起始序号 ISNb，A 收到 SYN 报文段后再回复 ACK 报文段（ACK 序号=ISNb+1）。

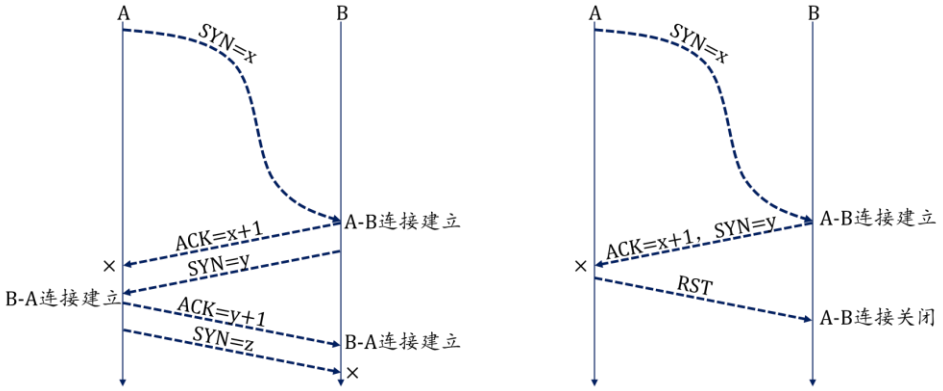
假如 B 收到的是一个过期的连接请求报文段（ISNa+1），假设这时尚未建立起连接，B 自然会同意建立连接，并发出 ACK 报文段。A 收到 B 的 ACK 报文段（ISNa+1）时认为没有发出过 SYN，自然直接丢

弃 ACK。而 A 收到 B 的反方向 SYN 报文段（ISNb），以为是一个新的连接请求（没有足够信息可以判定为无效并丢弃），遂回复 ACK 报文段（序号 ISNb+1），并再次发出（反方向）新的 SYN 报文段（ISNa2）。B 收到 ACK（序号 ISNb+1）就认为已经建立了双向连接，再收到 SYN 报文段（ISNa2）就认为是重复的过期连接请求，就予以丢弃。到这时，B 认为已经建立了连接（这是一个不该建立的连接），等待接收数据；而 A 则处于等待 B 的 ACK 状态，如图 3（a）所示。处于上述状态的 A、B 之间不能建立新的连接，也称为死锁。这个死锁是可以靠 A 的超时机制解锁的，但是浪费了时间。

如果采用严格的“三次握手”就可以避免上述问题：A 向 B 发起连接（发出 SYN 报文段，起始序号 ISNa），B 收到后回复 ACK+SYN 报文段（序号 ISNa+1，ISNb），A 再回复 ACK 报文段（序号 ISNb+1）。

假如 B 收到的是一个过期的连接请求，A 收到 B 的 ACK+SYN 报文段（序号 ISNa+1，ISNb）时认为没有发过 SYN，自然直接丢弃 ACK+SYN（实际上就把 ACK 和 SYN 一并丢弃了），也就不会向 B 发出 ACK 报文段（序号 ISNb+1），甚至这时可以发出一个 RST 报文段，A、B 同时中止连接建立过程，如图 3（b）所示。客户端 A 从报文段的 ACK 序号是否等于 ISNa+1 就可以判断这个连接响应是否是针对最新这个连接请求的。服务器 B 同样也要核对 ACK 序号，比如在 B 发出 ACK+SYN 报文段（ACK=x+1，SYN=y）后，如果有一个过期的 A 发给 B 的 ACK 报文段到达 B，其中的 ACK 序号不会是 y+1，因此可以判断这是一个过期的重复报文段，直接丢弃。

以上例子说明，TCP 连接建立的“握手”过程，可以且必须把被动连接方的 ACK 和 SYN 两个报文段合并在一起发送，“四次握手”变成“三次握手”才是可靠无误的。



（a）四次握手的情形　　（b）三次握手的情形

图 3　三次握手与四次握手的比较

5　对TCP“三次握手”的误解

“三次握手”作为 TCP 协议的核心概念，经常被用来考核学生，比如“为什么 TCP 需要‘三次握手’？”，绝大多数回答“为了防止已失效的连接请求报文段又传送到了服务端，因而产生错误”。 这个答案没有正确理解三次握手的全貌。TCP 协议连接建立采用“三次握手”，确实是为了解决“网络中存在延迟的重复报文段”的问题，但并非单单是“为了防止已失效的连接请求报文段又传送到了服务端，因而产生错误”。

首先是，TCP 为了实现可靠数据传输，每次建立连接时每个传输方向都要选择一个“不太一样”的起始序号（还涉及选择的方法）；其次是连接建立时通过 SYN 报文段和 ACK 报文段互相确认这个起始序号（“握手”）；第三是必须采用“三次握手”来识别是否为“已失效的连接请求报文段又传送到了服务端”。当下对 TCP 连接建立采用“三次握手”原因的误解或片面理解还是普遍存在。

误解 1：只讲握手过程，不讲序号同步

Tomlinson 提出的“three way handshake”解决方案的本质是要在建立 TCP 连接时做两件事，一是合理选择数据单元的起始序号，保证网络中任何时候都不会有一个序号标识着 2 个或更多套接字相同的数据单元；二是在收发双方之间正确同步起始序号。

在国内流行的计算机网络教材中关于 TCP 连接建立的“三次握手过程”大多都是这样描述的（大意）：“A 选择一个初始序号 seq=x，向 B 发出连接请求报文段（SYN）；B 收到后向 A 发送确认报文段（ACK+SYN），确认号是 ack=x+1，其中也包括自己选择的初始序号 seq=y；A 收到 B 的确认报文段后，还要向 B 给出确认报文段（ACK），确认号 ack=y+1。”在这些描述中，

大多并没解释为什么需要“选择一个初始序号 seq=x”以及 x 代表什么、如何选择，也没有解释收到确认报文段（ACK）后要不要验证序号 x+1。

实际上，“为了防止已失效的连接请求报文段又传送到了服务端，因而产生错误”，序号是判断连接请求 SYN 和确认 ACK 报文段是否为失效重复报文段的依据。每个收到 ACK 的 TCP 实体都要根据 ACK 中的数据序号与自己发出的 SYN 起始序号进行比对，如果不是当前连接的序号就直接丢弃 ACK 报文段。

误解 2：四次握手也行，三次更加简洁

在《计算机网络（第 8 版）》（谢希仁编著）[4]中还有这么一句话“请注意，在图 5-28 中 B 发送给 A 的报文段，也可拆成两个报文段。可以先发送一个确认报文段(ACK=1，ack=x+1)，然后再发送一个同步报文段(SYN=1，seq=y)。这样的过程就变成了四报文握手，但效果是一样的。”

CSDN 博主「Andrew.J」博文[12]认为“当然 TCP 连接还可以四次握手，但三次握手是最节省资源的连接方式，因此 TCP 连接应该为三次握手。”

但以上这些解释都是不恰当的，也就是误解。因为 TCP 连接建立过程中，“三次握手”比“四次握手”不仅仅是“简洁”这么简单，而是可以避免“已失效的旧连接请求报文段又传送到了服务端，因而错误地建立起连接”。“四次握手”不但要多一次“握手”，还不能避免“已失效的旧连接请求报文段又传送到了服务端，因而错误地建立起连接”。

误解 3：“四次握手”数据传输会出错

这里我们需要假定，无论是“三次握手”还是“四次握手”，都是包含了“序号选择和同步”的（如“误解 1”的解释），每次连接建立都选择不一样的起始序号，“握手”过程实现起始序号同步。

如果采用“四次握手”，一旦遇到已失效的旧连接请求报文段又传送到了服务端，就会错误地建立起连接.，图 3（a）所示。图中，主机 B 以为 TCP 连接已经建立，并一直等待主机 A 发来数据；而 A 则处于等待 B 的 ACK 状态。这个错误建立起的连接，在“Connection Management in Transport Protocols”[11]中被称为“死锁”。这个“死锁”并不会造成数据传输的错误。这个“死锁”使得套接字被占用，处于上述状态的 A、B（套接字不变）之间不能建立新的连接；其次，主机 A、B 为这个 TCP 连接分配的资源（比如缓冲区、时钟等等）就一直占用，白白浪费。超时机制虽然可以解锁，但总是浪费了时间。如果 B 是服务器，在解锁之前，互联网中大量存在的已失效的旧连接请求报文段就会占用大量服务器资源。

6 结束语

TCP 协议运行在一个不可靠的 IP 服务之上，因此 TCP 报文段可能会丢失、延迟、重复或错序，协议必须使用超时机制重新发送没有及时到达接收方的报文段。如果重发的数据报文段在旧连接已经关闭且新连接恰好又建立后到达接收方，TCP 协议必须有足够的机制保证接收方能够分辨是不是当前连接的报文段。这就是 TCP 连接建立“三次握手”的初衷，在每次建立连接时选择与已关闭连接不太一样的起始序号，并跳过“三次握手”来互相确认当前连接的起始序号，实际上也是收发窗口的同步。有了起始序号的选择和同步，接收方就有足够的信息区分新老连接的数据报文段。

很多人对“三次握手”的必要性存在着一些误解，以为三次握手可以，四次握手也可以，只是三次握手更加简洁。实际上是，四次握手不行，反而会出错，导致收到 SYN 握手报文的一方无法判断这个 SYN 报文是否为过期的重复报文，只能当其为当前的连接请求，从而错误地建立连接。TCP 协议的“三次握手”对于连接两端之间的正确同步是必要的，也是充分的。采用“三次握手”就可以避免延迟到达的失效连接请求报文段错误地建立起无效的连接。

参 考 文 献

[1] Charles M.Kozierok: The TCP/IP Guide: A Comprehensive, llustrated Internet Protocols Reference, No Starch Press., 2005. ISBN 1-59327-047-X
[2] Postel J., Transmission Control Protocol, RFC 793, September 1981.
[3] R.S. Tomlinson, Selecting Sequence Numbers, ACM Operating Systems Review, Vol.9, No.3, July 1975.
[4] 谢希仁，计算机网络（第 8 版），电子工业出版社，2021.05
[5] A. S. Tanenbaum，Computer Networks (second Edition)，Pearson Education Inc.,1989. ISBN:0131668366
[6] R.T. Morris, A Weakness in the 4.2BSD UNIX TCP/IP Software, CSTR 117, 1985.
[7] Kevin D.Mitnick，The Art of Deception，Wiley，2002. ISBN 9780471237129
[8] F. Gont， S. Bellovin，Defending against Sequence Number Attacks，RFC 6528，February 2012.
[9] Rivest, R., The MD5 Message-Digest Algorithm,RFC 1321, April 1992.
[10] Eastlake, D., Schiller, J., and S. Crocker, Randomness Requirements for Security, RFC 4086, June 2005.
[11] C. A. Sunshine and Y. K. Dalal, Connection Management in Transport Protocols, Computer Networks, vol.2, 1978, 454-473.
[12] Andrew.J，CSDN 博文，https://blog.csdn.net/u011168837/article/details/110848170， 2020-12-08.