# 【计算机网络 (谢希仁) 习题题解】第5章 运输层 (5)——TCP的运输连接管理



TCP 是面向连接的协议。运输连接是用来传送 TCP 报文的。运输连接有三个阶段，即：**连接建立**、**数据传送**和**连接释放**。运输连接的管理是使运输连接的建立和释放都能正常地进行。

在 TCP 连接建立过程中要解决三个问题：

- 要使每一方能够确知对方的存在。
- 要允许双方协商一些参数 (如最大窗口值、是否使用窗口扩大选项和时间戳选项以及服务质量等)。
- 能够对运输实体资源 (如缓存大小、连接表中的项目等) 进行分配。

TCP 连接的建立采用 C/S 方式。主动发起连接建立的应用进程叫做**客户** (client)，被动等待连接建立的应用进程叫做**服务器** (server)。

### TCP 的连接建立

TCP 连接建立的过程叫做握手，握手需要在客户和服务器之间交换三个 TCP 报文段。下图画出了三报文握手 (three way (three message) handshake) [RFC 973] 建立 TCP 连接的过程。

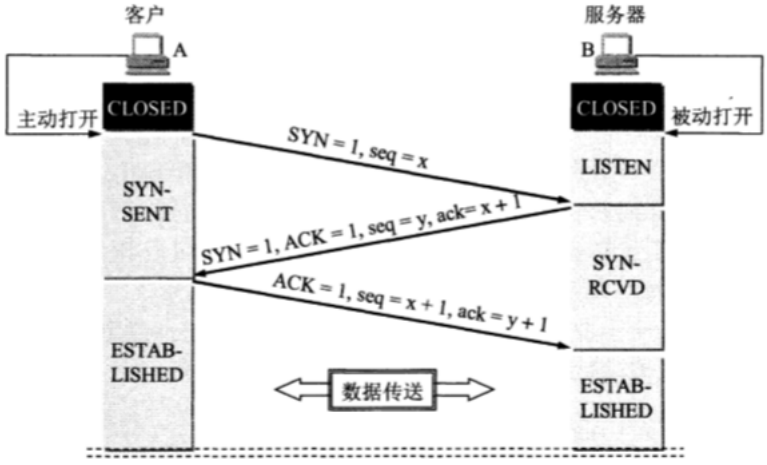



图 5-28 用三报文握手建立 TCP 连接



- 最初两端的 TCP 进程都处于 CLOSED (关闭) 状态。
- 一开始，B 的 TCP 服务器进程先创建**传输控制块** TCB (Transmission Control Block)，准备接受客户进程的连接请求。然后服务器进程就处于 LISTEN (收听) 状态，等待客户的连接请求。如有，即作出响应。
- A 的 TCP 客户进程也是首先创建 TCB。然后，在打算建立 TCP 连接时，向 B 发出连接请求报文段，这时首部中的同步位 SYN = 1，同时选择一个初始序号 seq = x。
  TCP 规定，SYN 报文段 (即 SYN = 1 的报文段) 不能携带数据，但要**消耗掉一个**

- B 收到连接请求报文段后，如同意建立连接，则向 A 发送确认。在确认报文段中应把 SYN 位和 ACK 位都置 1，确认号是 ack = x + 1，同时也为自己选择一个初始序号 seq = y。
  注意，这个报文段也不能携带数据，同样**要消耗掉一个序号。**
  这时 TCP 服务器进程进入 SYN-RCVD (同步收到) 状态。
- TCP 客户进程收到 B 的确认后，还要向 B 给出确认。确认报文段的 ACK 置 1，确认号 ack = y + 1，而自己的序号 seq = x + 1。
  TCP 的标准规定，ACK 报文段可以携带数据。但**如果不携带数据则不消耗序号**，在这种情况下，下一个数据报文段的序号仍是 seq = x + 1。
  这时，TCP 连接已经建立，A 进入 ESTABLISHED (已建立连接) 状态。
- 当 B 收到 A 的确认后，也进入 ESTABLISHED 状态。

上面给出的连接建立过程叫做**三报文握手。**
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。

---

**TCP 的连接释放**

数据传输结束后，通信的双方都可释放连接。

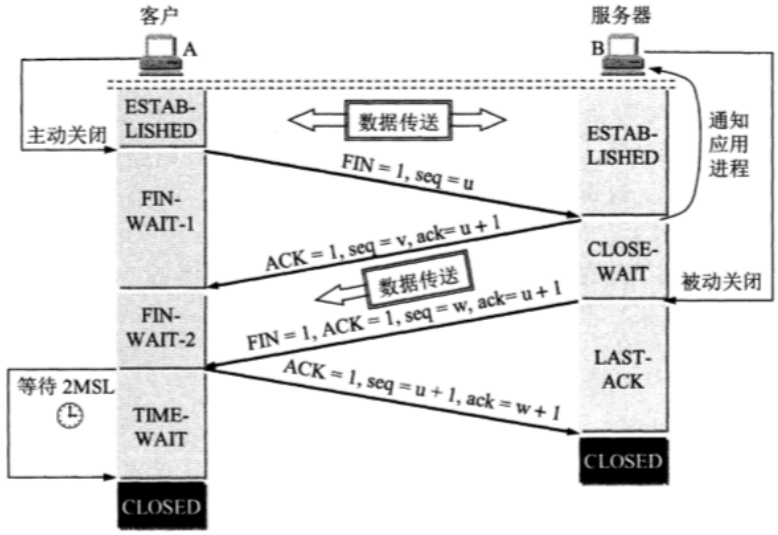



图 5-29　TCP 连接释放的过程



- 现在 A 和 B 都处于 ESTABLISHED 状态。
- A 的应用进程先向其 TCP 发出连接释放报文段，并停止再发送数据，主动关闭 TCP 连接。
  A 把连接释放报文段首部的终止控制位 FIN 置 1，其序号 seq = u，它等于前面已传送过的数据的最后一个字节的序号加 1。
  这时 A 进入 FIN-WAIT-1 (终止等待 1) 状态，等待 B 的确认。
  注意，FIN 报文段即使不携带数据，它也消耗掉一个序号。
- B 收到连接释放报文段后即发出确认，确认号是 ack = u + 1，而这个报文段自己的序号是 v，等于 B 前面已传送过的数据的最后一个字节的序号加 1。然后 B 就进入 CLASE-WAIT (关闭等待) 状态。
  TCP 服务器进程这时应通知高层应用进程，因而从 A 到 B 这个方向的连接就释放了，这时的 TCP 连接处于**半关闭** (half-close) 状态，即 A 已经没有数据要发送了，但 B 若发送数据，A 仍要接收。也就是说，从 B 到 A 这个方向的连接并未关闭，这个状态可能会持续一段时间。
- A 收到来自 B 的确认后，就进入 FIN-WAIT-2 (终止等待 2) 状态，等待 B 发出的连接释放报文段。
- 若 B 已经没有要向 A 发送的数据，其应用进程就通知 TCP 释放连接。
  这时 B 发出的连接释放报文段必须使 FIN = 1。现假定 B 的序号为 w (在半关闭状

- A 在收到 B 的连接释放报文段后，必须对此发出确认。在确认报文段中把 ACK 置 1，确认号 ack = w + 1，自己的序号是 seq = u + 1。然后进入到 TIME-WAIT (时间等待) 状态。
  注意，现在 TCP 连接还没有释放掉。必须经过**时间等待计时器** (TIME-WAIT timer) 设置的时间 2MSL 后，A 才进入到 CLOSED 状态。
  时间 MSL 叫做**最长报文段寿命** (Maximum Segment Lifetime)，RFC 793 建议设为 2 分钟。
  当 A 撤销相应的 TCB 后，就结束了这次的 TCP 连接。
- B 只要收到了 A 发出的确认，就进入 CLOSED 状态。同样，B 在撤销相应的 TCB 后，就结束了这次的 TCP 连接。

上述的 TCP 连接释放过程是四报文握手。

为什么 A 在 TIME-WAIT 状态必须等待 2MSL 的时间呢?

- 为了保证 A 发送的最后一个 ACK 报文段能够到达 B。这个 ACK 报文段有可能丢失，处在 LAST-ACK 状态的 B 会超时重传 FIN + ACK 报文段，而 A 就能在 2MSL 时间内收到这个重传的报文段。接着 A 重传一次确认，重新启动 2MSL 计时器。最后 A 和 B 都正常进入到 CLOSED 状态。
  如果 A 在 TIME-WAIT 状态不等待一段时间，而是在发送完 ACK 报文段后立即释放连接，那么就无法收到 B 重传的 FIN + ACK 报文段，因而也不会再发送一次确认报文段。这样，B 就无法进入 CLOSED 状态。
- 防止“已失效的连接请求报文段”出现在本连接中。A 在发送完最后一个 ACK 报文段后，再经过时间 2MSL，就可以使本连接持续的时间内所产生的所有报文段都从网络中消失。使下一个新的连接中不会出现这种旧的连接请求报文段。

TCP 还设有一个**保活计时器** (keepalive timer)。假设客户已与服务器建立了 TCP 连接，但客户端主机突然出故障。服务器不能再收到此客户发来的数据。
可以使用保活计时器使服务器不要再等待。服务器每收到一次客户的数据，就重新设置保活计时器，时间的设置通常是两小时。若两小时没有收到客户数据，服务器就发送一个探测报文段，以后则每隔 75 秒发送一次，若一连发送 10 个探测报文段后仍无客户的响应，服务器就认为客户端出现了故障，接着就关闭这个连接。

---

**TCP 的有限状态机**

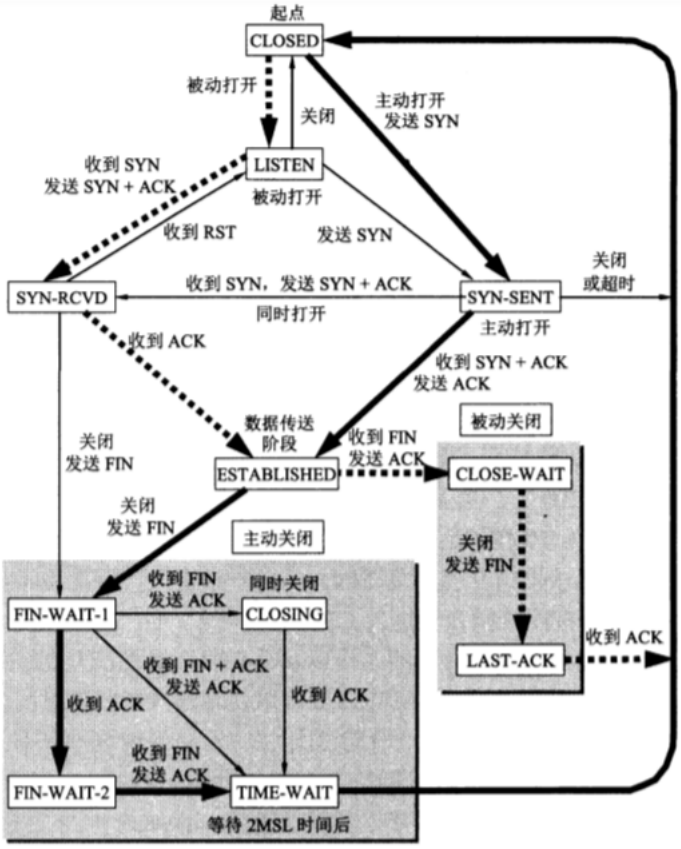



图 5-30　TCP 的有限状态机



上图中，粗实线箭头表示对客户进程的正常变迁，粗虚线箭头表示对服务器进程的正常变迁，另一种细线箭头表示异常变迁。

---

**5-41** 用 TCP 传送 512 字节的数据。设窗口为 100 字节，而 TCP 报文段每次也是传送 100 字节的数据。再设发送方和接收方的起始序号分别选为 100 和 200，试画出类似于图 5-28 的工作示意图。从连接建立阶段到连接释放都要画上。

**解**：发送方 A，接收方 B。

建立连接：
A->B：SYN = 1，seq = 100
B->A：SYN = 1，ACK = 1，seq = 200，ack = 101
A->B：ACK = 1，seq = 101，ack = 201

传送数据：
A->B：ACK = 1，seq = 101，ack = 201，DATA(100B)
B->A：ACK = 1，seq = 201，ack = 201，rwnd = 100
A->B：ACK = 1，seq = 201，ack = 201，DATA(100B)
B->A：ACK = 1，seq = 201，ack = 301，rwnd = 100
A->B：ACK = 1，seq = 301，ack = 201，DATA(100B)
B->A：ACK = 1，seq = 201，ack = 401，rwnd = 100
A->B：ACK = 1，seq = 401，ack = 201，DATA(100B)
B->A：ACK = 1，seq = 201，ack = 501，rwnd = 100
A->B：ACK = 1，seq = 501，ack = 201，DATA(100B)
B->A：ACK = 1，seq = 201，ack = 601，rwnd = 100
A->B：ACK = 1，seq = 601，ack = 201，DATA(12B)
B->A：ACK = 1，seq = 201，ack = 613，rwnd = 100

释放连接：
A->B：FIN = 1，ACK = 1，seq = 613，ack = 201
B->A：ACK = 1，seq = 201，ack = 614
B->A：FIN = 1，ACK = 1，seq = 201，ack = 614

注意，SYN 报文段和 FIN 报文段即使不携带数据，也要消耗掉一个序号。ACK 报文段如果不携带数据，则不消耗序号。

---

**5-42** 在图 5-29 中所示的连接释放过程中，在 ESTABLISHED 状态下，服务器进程能否先不发送 ack = u + 1 的确认？(因为后面要发送的连接释放报文段中仍有 ack = u + 1 这一信息。)

**解**：如果 B 不再发送数据了，是可以把两个报文段合并成为一个，即只发送 FIN + ACK 报文段。
但如果 B 还有数据要发送，而且要发送一段时间，那就不行，因为 A 迟迟收不到确认，就会以为刚才发送的 FIN 报文段丢失了，就超时重传这个 FIN 报文段，浪费网络资源。

---

**5-43** 在图 5-30 中，在什么情况下会发生从状态 SYN_SENT 到状态 SYN_RCVD 的变迁?

**解**：当 A 和 B 都作为客户，即同时主动打开 TCP 连接。这时每一方的状态变迁都是：CLOSED->SYN-SENT->SYN-RCVD->ESTABLISHED

---

**5-44** 试以具体例子说明为什么一个运输连接可以有多种方式释放。可以设两个互相通信的用户分别连接在网络的两结点上。

**解**：设 A，B 建立了运输连接。
协议应考虑一下实际可能性：
A 或 B 故障，应设计超时机制，使对方退出，不至于死锁。
A 主动退出，B 被动退出。
B 主动退出，A 被动退出。

---

**5-45** 解释为什么突然释放运输连接就可能会丢失用户数据，而使用 TCP 的连接释放方法就可保证不丢失数据。

**解**：当主机 1 和主机 2 之间连接建立后，主机 1 发送了一个 TCP 报文段并正确抵达主机 2，接着主机 1 发送另一个 TCP 报文段，这次很不幸，主机 2 在收到第二个 TCP 数据段之前发出了释放连接请求，如果就这样突然释放连接，显然主机 1 发送的第二个 TCP 报文段会丢失。
而使用 TCP 的连接释放方法，主机 2 发出了释放连接的请求，那么即使收到主机 1 的确认后，只会释放主机 2 到主机 1 方向的连接，即主机 2 不再向主机 1 发送数据，而仍然可接收主机 1 发来的数据，所以可保证不丢失数据。

---

**5-46** 试用具体例子说明为什么在运输连接建立时要使用三次握手。说明如不这样做可能会出现什么情况。

★ 为什么 A 最后还要发送一次确认呢？这主要是为了防止已失效的连接请求报文段突然又传送到了 B，因而产生错误。

- “已失效的连接请求报文段”是这样产生的。假定出现一种异常情况，即 A 发出的第一个连接请求报文段在某些网络结点长时间滞留，以致延误到连接释放以后的某个时间才到达 B。
  本来这是一个早已失效的报文段。但 B 收到此失效的连接请求报文段后，误认为是 A 又发出一次新的连接请求。于是就向 A 发出确认报文段，同意建立连接。
- 假定不采用三报文握手，那么只要 B 发出确认，新的连接就建立了。
  由于现在 A 并没有发出建立连接的请求，因此不会理睬 B 的确认，也不会向 B 发送数据。但 B 却以为新的运输连接已建立，一直等待 A 发来数据。B 的许多资源就这样浪费了。
- 采用三报文握手的办法，可以防止上述现象的发生。例如在上述的异常情况下，A

**5-47** 一个客户向服务器请求建立 TCP 连接。客户在 TCP 连接建立的三报文握手中的最后一个报文段中捎带上一些数据，请求服务器发送一个长度为 L 字节的文件。假定：

(1) 客户和服务器之间的数据传输速率是 R 字节/秒，客户与服务器之间的往返时间是 RTT (固定值)。

(2) 服务器发送的 TCP 报文段的长度都是 M 字节，而发送窗口大小是 nM 字节。

(3) 所有传送的报文段都不会出现差错 (无重传)，客户收到服务器发来的报文段后就及时发送确认。

(4) 所有的协议首部开销都可忽略，所有确认报文段和连接建立阶段的报文段的长度都可忽略 (即忽略这些报文段的发送时间)。

试证明，从客户开始发起连接建立到接收服务器发送的整个文件多需的时间 T 是：

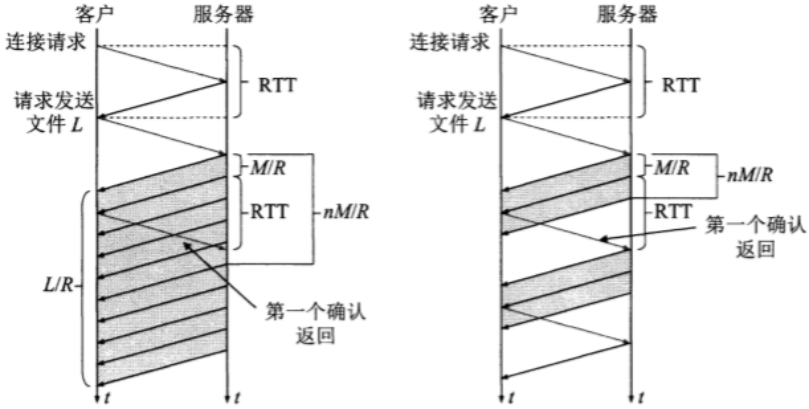
T = 2 RTT + L/R，当 nM > R (RTT) + M

或 T= 2 RTT + L/R + (K - 1)[M/R + RTT - nM/R]，当 nM < R (RTT) + M

其中，K = ⌈ L/nM ⌉，符号 ⌈x⌉ 表示若 x 不是整数，则把 x 的整数部分加 1。

(提示：求证的第一个等式发生在发送窗口较大的情况，可以连续把文件发送完。求证的第二个等式发生在发送窗口较小的情况，发送几个报文段后就必须停顿下来，等收到确认后再继续发送。建议先画出双方交互的时间图，然后再进行推导。)

**解**：发送窗口的两种不同情况如下图所示。




(a) 当 $nM > R\,(\mathrm{RTT}) + M$　　(b) 当 $nM < R\,(\mathrm{RTT}) + M$

图 (a)：

M/R 是一个报文段的发送时间。

nM 是窗口大小，nM/R 是把窗口内的数据都发送完所需要的时间。

如果 nM/R > M/R + RTT ，那么服务器在发送窗口内的数据还没有发送完，就收到客户的确认，因此，服务器可以连续发送，直到全部数据发送完毕。

这个不等式两边都乘以 R，就得出等效的条件：

当 nM > R (RTT) + M 发送窗口内的数据还没有发送完就收到确认，因此，服务器可以连续发送，直到全部数据发送完毕。

因此，客户接收全部数据所需的时间是：

T = 2 RTT + L/R，当 nM > R(RTT) + M

图 (b)：

当 nM > R (RTT) + M 时，服务器把发送窗口内的数据发送完毕时还收不到确认，因此必须停止发送。从图中可知，停止的时间间隔是 M/R + RTT - nM/R。

整个文件 L 要划分为 K = ⌈L/nM⌉ 次传送，停止的时间间隔有 (K-1) 个。这样就证明了求证的公式：

T = 2 RTT + L/R + (K-1)[M/R + RTT - nM/R]，当 nM < R (RTT) + M

---

**5-62** TCP 连接处于 ESTABLISHED 状态。以下的事件相继发生：

(1) 收到一个 FIN 报文段。

(2) 应用程序发送“关闭”报文。

在每一个事件之后，连接的状态是什么？在每一个事件之后发生的动作是什么？

**解**：见图 5-30。

(1) 这是对服务器进程的正常变迁。当收到 FIN 时，服务器端向客户端发送 ACK 报文

没有数据要发送了。这时服务器就应当发送 FIN 报文段给客户，然后进入到 LAST-ACK 状态，并等待来自客户端的最后的确认。

---

**5-63** TCP 连接处于 SYN-RCVD 状态。以下的事件相继发生：
(1) 应用程序发送“关闭”报文。
(2) 收到 FIN 报文段。
在每一个事件之后，连接的状态是什么？在每一个事件之后发生的动作是什么？

**解**：(1) 这是异常变迁。应用进程向对方发送 FIN 报文段，进入 FIN-WAIT-1 状态。
(2) 这是异常变迁。应用进程收到 FIN 报文段，向对方发送 ACK 报文段，进入 CLOSING 状态。

---

**5-64** TCP 连接处于 FIN-WAIT-1 状态。以下的事件相继发生：
收到 ACK 报文段。
收到 FIN 报文段。
发生了超时。
在以上的每一个事件之后，连接的状态是什么？在每一个事件之后发生的动作是什么？

**解**：客户收到 ACK 报文段后，进入 FIN-WAIT-2 状态。
客户收到 FIN 报文段后，向对方发送 ACK 报文段，进入 TIME-WAIT 状态。
发生了超时，也就是等待 2MSL 时间后，客户进入 CLOSED 状态。

---

**5-68** 在 TCP 的连接建立的三报文握手过程中，为什么第三个报文段不需要对方的确认？这会不会出现问题？

- 假定 A 是客户，是发起 TCP 连接建立一方。B 是服务器。现在假定三报文握手过程中的第三个报文段 (也就是 A 发送的第二个报文段——ACK 报文段) 丢失了，而 A 并不知道。这时，A 以为对方收到了这个报文段，以为 TCP 连接已经建立，于是就开始发送数据报文段给 B。
- B 由于没有收到三报文握手中的最后一个报文段 (A 发送的 ACK 报文段)，因此 B 就不能进入 TCP 的 ESTABLISHED 状态。B 的这种状态可以叫做 “半开连接”，即仅仅把 TCP 连接打开了一半。在这种状态下，B 虽然已经初始化了连接变量和缓存，但是不能接收数据。
  通常，B 在经过一段时间后，如果还没有收到来自 A 的确认报文段，就终止这个半开连接状态，那么 A 就必须重新建立 TCP 连接。
  因此，在这种情况下，第三个报文段 (A 发送的第二个报文段) 的丢失，就导致了 TCP 连接无法建立。
- 但是，假定 A 在这段时间内，紧接着就发送了数据。
  TCP 具有累计确认的功能。在 A 发送的数据报文段中，自己的序号也没有改变，仍然是和丢失的 ACK 报文段的序号一样 (丢失的那个 ACK 报文段不消耗序号)，并且确认位 ACK = 1，确认号也是 B 的初始序号加 1。当 B 收到这个报文段后，从 TCP 的首部就可以知道，A 已确认了 B 刚才发送的 SYN + ACK 报文段，于是就进入了 ESTABLISHED 状态。接着，就接收 A 发送的数据。
  在这种情况下，A 丢失的第二个报文段对 TCP 的连接建立就没有影响。
- A 在发送第二个报文段时，可以有两种选择：
  (1) 仅仅是确认而不携带数据，数据接着在后面发送。
  (2) 不仅是确认，而且携带上自己的数据。
- 在第一种选择时，A 在下一个报文段发送自己的数据。但下一个报文的首部中仍然包括了对 B 的 SYN + ACK 报文段的确认，即和第二种选择发送的报文段一样。
- 在第二种选择时，A 省略了单独发送一个确认报文段。

A 发送的第二个报文段仅仅是 ACK 报文段，是个可以省略的报文段，即使丢失了也无妨，只要下面紧接着就可以发送数据报文段即可。

【计算机网络 (谢希仁) 习题题解】目录

**文章知识点与官方知识档案匹配，可进一步学习相关知识**

网络技能树　首页　概览　41817 人正在系统学习中

《计算机网络》谢希仁版--第二章物理层.pdf　12-24
《计算机网络》谢希仁版--第二章思维导图（适合期末复习或系统理解知识点

《计算机网络》谢希仁版--第六章应用层.pdf　12-24
《计算机网络》谢希仁版--第六章思维导图（适合期末复习或系统理解知识点

7 条评论　半山先生　热评　为啥谢书上 传输数据那块 除了第一次A...　写评论

...谢希仁版(五)运输层(重点)_4、tcp 采用自适应算法来计算加权平均往返...　12-22
TCP的流是指流入到进程或从进程流出的字节序列。虽然应用程序和TCP的交互式一次一个数据…

...1.设tcp的ssthresh初始值为8(单位为报文段)。当拥塞窗口上升到12时...　12-23
答:以太网帧的数据字段的最大长度是1500字节。UDP数据报的首部是8个字节,所以整个UDP数…

【计算机网络(微课版)】第5章 传输层 课后习题及答案　为了远大的理想，冲锋！　2万+
5.1 试说明运输层在协议栈中的地位和作用。运输层的通信和网络层的通信有什么重要区别？ 解…

第七版《计算机网络》运输层.xmind　03-14
结合课程PPT与第七版计网书籍

【广工考试笔记】计算机网络考试速成笔记　12-22
应用层:报文 为用户的应用进程直接提供服务 传输层:报文段 负责向两个主机中进程之间的通信…

计算机网络期末复习重要知识点_在链路上产生的传播时延与链路的带宽无...　12-7
(包含协议有TCP、UDP) 网络层:网络层负责为分组交换网上的不同主机提供通信服务。在发送…

《计算机网络》谢希仁版--第四章网络层.pdf　12-24
《计算机网络》谢希仁版--第四章思维导图（适合期末复习或系统理解知识点

TCP三次握手，4次挥手流程　06-29
用TCP传送512字节的数据，设窗口为100字节，而TCP报文段每次也是传送100字节的数据。再…

...a用tcp传送512字节的数据给b,b用tcp传送640字节的数据给a。设a、b...　12-22
A .执行计算机数据处理的软件模块 B .由自主计算机互联起来的集合体 C .多个处理器通过共享内…

【计算机网络】课后习题重点_一个udp用户数据报的数据字段为8192...　12-21
1-19 长度为100字节的应用层数据交给传输层传送,需加上20字节的TCP首部。再交给网络层传…

计网-习题　小飞猪的博客　4263
习题一：假设一个IP分组头部长度为20B，数据字段长度为2000B。分组从源主机到目的主机要…

计网复习——传输层习题　NickHan_cs的博客　8091
计网复习——传输层习题 1. 主机A向主机B连续发送了两个TCP报文段，其序号分别为70和100…

计算机网络知识点总结-第五章:运输层　12-3
(1)用户数据报协议(UDP) (2)传输控制协议(TCP) 两种协议在协议栈中的位置: UDP: 在传送数据…

...如果收到重复的报文段时不予理睬(即悄悄地丢弃它而其他什么-CSDN博 ...　12-15
只是在计算检验和时,临时添加在UDP用户数据报或TCP报文段的前面,得到一个临时的UDP用户…

计算机网络（第八版）第五章章末习题　m0_67837149的博客　2569
计算机网络（第八版）第五章章末习题

计算机网络教程第5版-第5章运输层（传输层）　ZMiao的博客　1956
解答：从通信和信息处理的角度看，运输层向它上面的应用层提供端到端通信服务，它属于面向…

对tcp协议的可靠数据传输的理解（水）　2301_76684181的博客　45
但是tcp协议重传最多一个丢失的分组甚至不会重传分组，因为n+1（n就是丢失的分组）的分组…

《计算机网络》谢希仁版--第五章运输层.pdf　12-24
《计算机网络》谢希仁版--第五章思维导图（适合期末复习或系统理解知识点

计算机网络（谢希仁-第八版）第五章习题全解 qq_56919740的博客 3万+
还没更完，近期可能会更新

计算机网络习题解答(教材 计算机网络 谢希仁编著)3 热门推荐 大师的资源 4万+
第六章网络互连6-03作为中间系统。转发器、网桥、路由器和网关有何区别？答：转发器：是物…

超时重传时间例题计算机网络,计算机网络（5）TCP之… weixin_29011667的博客 378
计算机网络(5)TCP之重传机制计算机网络(5)TCP之重传机制重传机制超时重传数据包丢失确认…

计算机网络：第五章习题 匿名用户 3560
简答/计算题 5-1 试说明运输层在协议栈中的地位和作用，运输层的通信和网络层 的通信有什么…

TCP状态中 time_wait 的作用 线上幽灵 1597
TCP状态中 time_wait 的作用： 客户端接收到服务器端的 FIN 报文后进入此状态，此时并不是直…

（~最新合集~）计算机网络谢希仁第七版 第五章课后… weixin_43899069的博客 5839
5—01 试说明运输层在协议栈中的地位和作用，运输层的通信和网络层的通信有什么重要区别？…

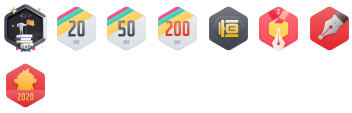
计算机网络谢希仁第五章 最新发布 11-25
计算机网络谢希仁第五章主要讲述了运输层的相关内容，包括传输控制协议TCP和用户数据报协…



今岁成蹊
码龄7年 暂无认证

| 102 | 4万+ | 200万+ | 19万+ | |
|---|---|---|---|---|
| 原创 | 周排名 | 总排名 | 访问 | 等级 |

| 1994 | 301 | 283 | 50 | 1430 |
|---|---|---|---|---|
| 积分 | 粉丝 | 获赞 | 评论 | 收藏 |



私信 关注


搜博主文章

**热门文章**

【计算机网络 (谢希仁) 习题题解】目录 37920

【计算机网络 (谢希仁) 习题题解】第4章 网络层 (2)——划分子网；CIDR 13658

【计算机网络 (谢希仁) 习题题解】第2章 物理层 13399

【计算机网络 (谢希仁) 习题题解】第5章 运输层 (1)——UDP 12022



Beta

Beta

举报

【计算机网络 (谢希仁) 习题题解】第3章 数据链路层 10944

**分类专栏**


**最新评论**

【计算机网络 (谢希仁) 习题题解】第5章 …
A_xxxxxx: 我觉得博主这个答案是正确的，除了建立连接那次消耗掉了一个序号，7 …

【计算机网络 (谢希仁) 习题题解】第4章 …
今岁成蹊: 是的，已修改

【计算机网络 (谢希仁) 习题题解】第4章 …
嗯嗯....: 4-55 第二题是不是应该160 AND 192才可以啊

【计算机网络 (谢希仁) 习题题解】第5章 …
m0_74021670: 我理解是A->B的ack=202，因为A希望B发送的序号是202，但是B— …

【python 让繁琐工作自动化】第16章 处…
今岁成蹊: 代码的第23的作用其实就是将 response.text 中的内容赋值给 weatherDa …

**您愿意向朋友推荐“博客详情页”吗?**


**最新文章**

【C# 7.0 in a Nutshell】第4章 C#的高级特性——委托

【C# 7.0 in a Nutshell】第3章 在C#中创建类型——类

【C# 7.0 in a Nutshell】第2章 C#语言基础——数组

2022年 2篇　　2021年 34篇
2020年 51篇　　2019年 15篇

**目录**



Beta

Beta

举报