计算机网络讲义要点

赵增华

天津大学智能与计算学部

2023 年秋

参考教材：

“Computer networking: a top-down approach”, by Jim Kurose, Keith Ross, Pearson, 8th Edition, 2021 年。

辅助教材：“Computer Networks”, 第 5 版英文影印版 ,A. Tanenbaum, 清华大学出版社, 2011 年。

教学目标：

通过对 Internet 经典协议的学习和剖析，让学生理解网络系统设计、网络协议设计所面临的问题和常用的解决方法。课程结束后学生能够根据工作/科研的需要设计、实现相应的网络系统和协议，并进行性能评价。授课过程中强调对问题的描述（why），然后是解决方案（how）。

学习方法：

学而不思则罔，思而不学则殆。

多读，多思，多实践。

多读：通过大量阅读理解网络、协议的基本概念，原理。阅读推荐的教材、参考书、自行在网上查找的相关内容。

多思：思考为什么。网络系统/每层（个）协议设计面临什么问题？如何解决？为什么？

多实践：1）通过网络协议分析软件（如 Wireshark）深入理解各层协议的执行过程。（WiresharkLab）

2）通过设计、实现网络协议，掌握协议的设计方法和实现技术。（Socket programming, RDT, routing protocol）

3）通过网络构建、配置，理解网络的实际部署和管理。（Lab Practice）

第三章 传输层

教学要求：

1. 理解传输层在网络协议栈中的位置及主要功能。(3.1)
2. 了解 UDP 协议（3.3）.
3. 熟练掌握经典的可靠数据传输协议 RDT 的工作原理，包括停等协议（stop and wait）,GBN (Go Back N),和 SR (Selective Repeat)。要求能够使用 FSM 设计可靠传输协议和其它网络协议。(3.4)
4. 掌握 TCP 协议可靠传输机制、三次握手建立连接的方法、流量控制机制。（3.5）
5. 理解拥塞控制的基本原理；掌握 TCP 协议拥塞控制协议的工作机制。（3.6，3.7）

一、深入理解 RDT

1.1 可靠数据传输的应用场景

在有丢包、出错的不可靠的情况，需要数据传输是可靠的（没有丢包，没有出错，没有乱序）应用场景都可以考虑使用 RDT。

（1）RDT 不是传输层所特有的，在协议各层只要有需要就可以使用 RDT。

我们课程采用 top-down 方式，先讲传输层，后讲数据链路层。由于 RDT 是传输层的重要功能，因此把 RDT 放到了传输层来讲。但是在其他协议层也可以使用 RDT。

举两个例子。1）在应用层，如果传输层使用 UDP（User Datagram Protocol）Socket，而网络应用对可靠传输有一定的需求，协议设计也可以考虑使用 RDT。QUIC [3-1]就是这类协议。2）某些数据链路层协议的设计也采用了 RDT，如 WiFi 的数据链路层协议 CSMA/CA 就使用了 stop-and-wait 类型的 RDT。这是因为无线信号在传输过程中容易受到环境干扰、多径效应等影响导致丢包，因而物理层不可靠。所以 WiFi 需要提供一定程度的可靠性服务，以减少丢包对上层协议的不良影响。

（2） RDT 不是 Internet 所特有的，只要在不可靠的环境下需要可靠传输数据，就可以考虑使用 RDT。

比如工业生产环境中常用的串口通信 RS-485。

[3-1] RFC 9000 – QUIC: A UDP-Based Multiplexed and Secure Transport. IETF. doi:10.17487/RFC9000. RFC 9000.

1.2 可靠传输需要解决的问题

包（packet）出错，包丢失，包乱序。

数据包（data）和确认包（ACK）都可能出现上述问题。

1.3 可靠传输的解决方案

差错检测；ACK；出错重传；超时重传（设置计时器 Timer）；序列号。

其中，ACK、出错重传和超时重传都是以“延迟”为代价换取“可靠”。这里不考虑 FEC（Forward Error Correction）的方法。FEC 是用“带宽”为代价换取“可靠”的。天下没有免费的午餐。

1.4 可靠传输的性能评价

协议的设计会严重影响到网络的端到端性能。因此，协议实现后要做性能评价，确保协议的性能。性能评价参数常用的有两个：最大链路利用率和端端有效吞吐率。

最大链路利用率（maximum link utilization）：理论分析时使用的性能评价参数，即理想情况（无丢包，无差错，无乱序）下协议所能达到的链路利用率。这里的“链路”是逻辑上的，指需要可靠数据传输的双方之间的逻辑链路。

端端吞吐率（goodput）：即接收端有效吞吐率，指接收端单位时间内正确收到的按序到达的数据量。是实验验证时常用的性能参数。

二、设计 RDT 协议：stop-and-wait, pipelined RDT (GBN, SR)

协议的设计只要能够解决所面临的问题，能运行，就是满足了基本要求。协议的设计没有对错之分，只有性能的好坏。协议设计追求性能的优化，同时兼顾实现的代价（cost）。

RDT 协议是以延迟（重传）为代价，换取可靠性（gain）。因此，如何降低延迟（cost）提高链路利用率是 RDT 协议设计要解决的关键问题。

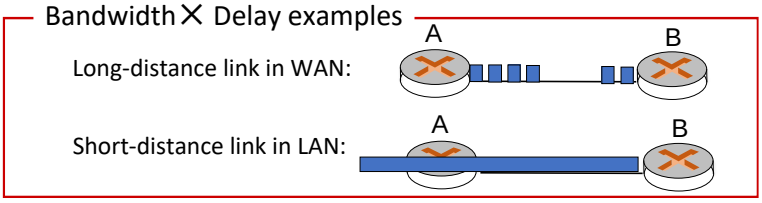
两类经典的 RDT 协议（按照一次发送数据包的个数分类）：

停等 RDT 协议（stop-and-wait）：发送端一次只发送一个数据包，直到该数据包被正确接收才发送下一个数据包。

流水线 RDT 协议（pipelined RDT）：发送端可以连续发送多个数据包。该类协议按实现方式又有两个经典案例：GBN（Go-Back-N）回退 N 帧； SR（Selective Repeat）选择重传。

1) 根据网络场景选择 RDT 的类型

理解延迟带宽积（bandwidth× delay）的含义。能够根据网络的 bandwidth × delay 合理确定所需设计的 RDT 的类别：所需要设计的 RDT 是 stop-and-wait 还是 pipelined RDT。

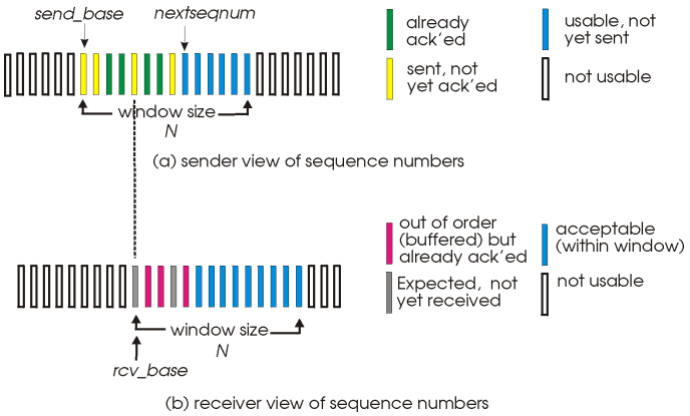


如果 bandwidth×delay 远大于数据包长度，则采用 pipelined RDT，以达到较高的链路利用率。

如果 bandwidth×delay 和数据包长度相当或者小于数据包长度，则使用 stop-and-wait 就能达到较高的链路利用率。比如 WLAN（WiFi）采用 stop-and-wait 机制提高链路的可靠性。

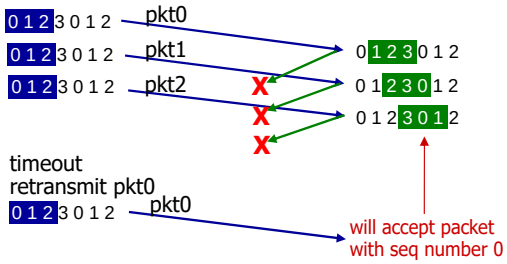
2) Pipelined RDT 协议实现所采用的滑动窗口（sliding window）技术

滑动窗口技术本质上是对接收方和发送方缓冲区的管理。通过该技术实现对数据包的高效管理，确保发送出去的每个数据包都能被正确接收到，按序无差错交付数据包。



设发送端窗口大小为 N_s，接收端窗口大小为 N_r。对于 GBN 协议，N_r=1；对于 SR 协议，N_s=N_r>1。

为确保滑动窗口机制的正确运行，发送端和接收端窗口大小和数据包序列号长度要满足一定的关系。设序列号长度为 n 比特，则 $N_s+N_r <= 2^n$。这样才能避免出现下图所示的出错情况：

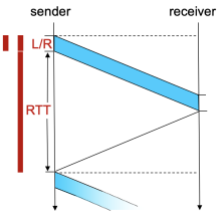


3) 能够使用 FSM（Finite State Machine）辅助设计 RDT 协议。

4) 性能评价

停等 RDT 协议的最大链路利用率：

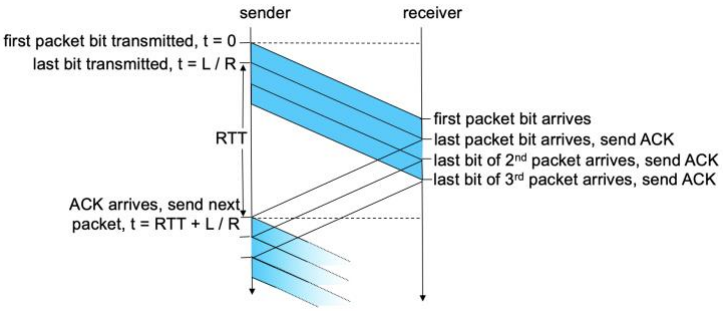
$$U_{sender} = \frac{L/R}{RTT + L/R}$$



流水线 RDT 协议的最大链路利用率：

$$U_{sender} = \frac{N_s \times L/R}{RTT + L/R}$$

N_s: the size of sending window



三、理解 TCP 可靠传输机制中提升性能的主要技术

TCP 可靠传输中采取了很多技术提升性能（链路利用率、吞吐率），主要包括以下几点：

1） 采用流水线可靠传输机制。

TCP 的 RDT 主要解决 Internet 中端到端的数据可靠传输问题。大多数情况下两个端节点间物理距离很远，而骨干网的带宽又很高，因此端端逻辑链路的 bandwidth×delay 远大于一个数据包的长度。基于此，TCP 采用流水线的可靠传输机制以提高链路利用率。

发送端和接收端都设有数据缓冲区。使用滑动窗口机制管理发送端和接收端的缓冲区。另外，由于 TCP 支持数据的双向传输，即全双工方式（duplex），因此 TCP 的发送端和接收端都设有两个数据缓冲区，一个用于发送，一个用于接收。

2） 动态设置超时间隔（Timeout Interval）。

采用指数滑动平均算法动态估计 RTT（Round Trip Time）的均值，并据此设置超时间隔。合理设置超时间隔，既能减少长期等待造成的时间浪费，又能降低由于超时间隔过短而造成的重传浪费网络资源。

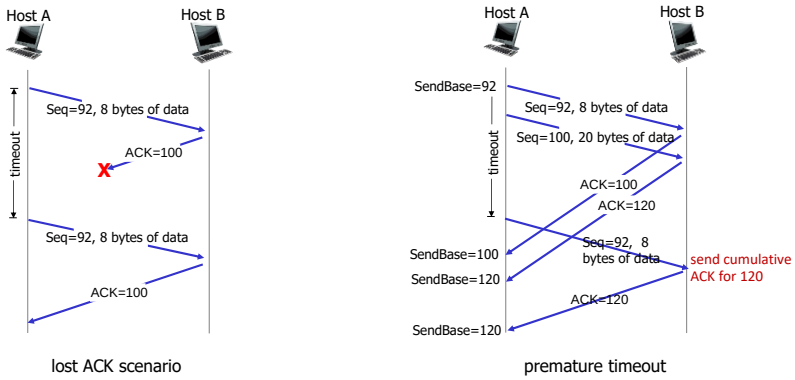


图3-1 超时间隔对数据传输的影响

具体计算方法：

$DevRTT_i=(1-\beta)\times DevRTT_{(i-1)}+\beta\times|SampleRTT_i-EstimatedRTT_{(i-1)}|$

$EstimatedRTT_i=(1-\alpha)\times EstimatedRTT_{(i-1)}+\alpha\times SampleRTT_i$

$TimeoutInterval= EstimatedRTT+4\times DevRTT$

其中：

SampleRTT：RTT采样值。

EstimatedRTT|：RTT的估计值。

DevRTT：RTT估计偏差。

$\alpha=0.125$； $\beta=0.25$

注意：迭代计算时要先计算DevRTT，再计算EstimatedRTT

3） 捎带应答（piggybacked ACK）。

利用 TCP 数据双向传输的特点，在数据包中携带 ACK 信息。不需要单独发送 ACK，避免了短小的 ACK 包的传输。可以有效减少网络资源的占用。

4） 累积应答（accumulative ACK）。

对当前序号数据包的 ACK 代表该序号之前的所有数据包都已经正确接收了。如图 3-2 所示，接收端没有收到 ACK100，在收到 ACK120 后认为 120 之前的所有数据包都已经正确接收了，因此直发送 seq120，可以减少由于之前数据包的 ACK 丢失导致的不必要的重传。

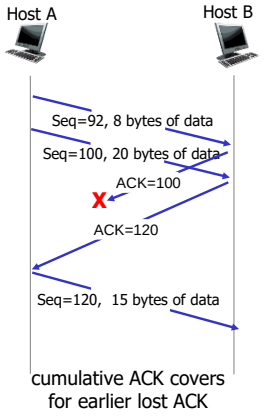


图 3-2 累积应答可以有效减少数据包重传

5) 快速重传（fast retransmission）。

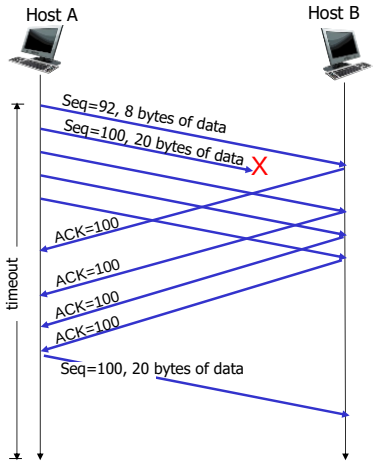


图 3-1 快速重传示例

接收端收到数据包发送 ACK 时，ACK 的数据包序号为最近接收到的按序到达的数据包的序号，不一定是当前收到的数据包序号。因此如果接收端收到多个乱序到达的数据包，则接收端所相应发送的 ACK 包的序号有可能相同。这种情况下发送端会连续收到多个重复序号的 ACK 包。如图 3-1 所示，当发送端收到 3 个重复序号的 ACK 包时，意味着这个数据包大概率已经丢失了，就立即重传该序号的数据包，而不是继续等待它超时再重传。这就是快速重传机制。该机制可以有效减少重传的等待时间，提高链路利用率。

6) 使用单一计时器（single timer）。

计时器用于对最早发出去还没有 ACK 的数据包的超时重传。不需要对每个数据包都启动一个计时器，能节约计时器资源。

注意：TCP 做为 Internet 的经典协议，其设计和实现都致力于以较小的代价

获取较好的性能。采用了很多提升性能的技术，并不限于上面所列的几项。更没有单纯使用 GBN 或者 SR 的技术。

四、理解 TCP 连接建立和拆除的工作流程及原因

TCP 建立连接本质上是要为数据包的双向可靠传输预约资源，包括 TCP client 和 server 缓冲区的分配、各种变量的初始化。TCP 建立连接后才开始数据传输、流量控制和拥塞控制。

理解三次握手建立连接的原因和方法。

三次握手：主要为解决连接过程中出现的丢包、乱序问题。

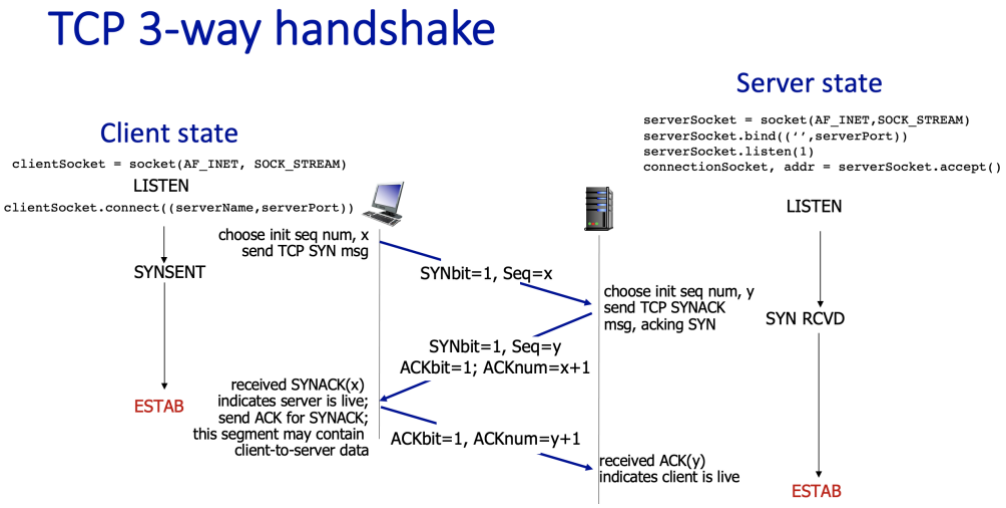


图 4-1 TCP 三次握手建立连接过程

理解 client 和 server 分别拆除连接的原因和方法。

TCP 双向数据传输，两边需要分别终止该方向上的数据传输。

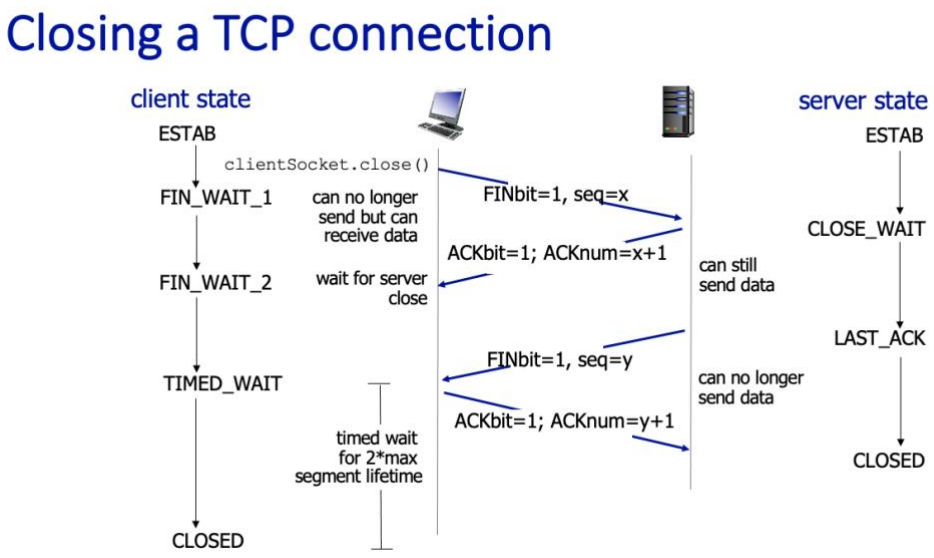


图 4-2 TCP 双向拆除连接的过程

五、理解 TCP 的流量控制（flow control）

解决的问题：由于互联网接入设备的多样性，导致接入设备的处理能力不同。TCP 发送端的数据包经过网络传输到达接收端后，由于发送速度快而接收端处理速度慢而导致的丢包。

解决的方法：在数据包头增加接收端可用缓冲区大小，用以控制发送端的发送速率。如图 5-1 所示。

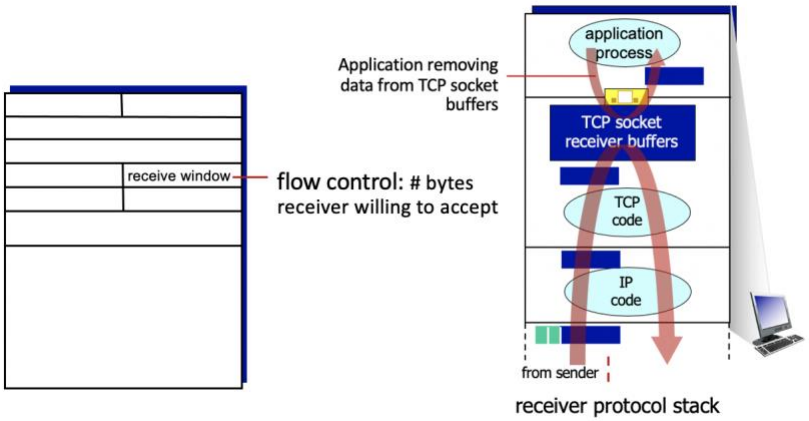


图 5-1 TCP 流量控制示意图

六、熟练掌握 TCP 拥塞控制（congestion control）；理解拥塞控制的原因和解决方法

解决的问题：由于互联网端系统业务的突发性，TCP 发送端的发送速率较大，短时间内向网络中注入了大量数据包，核心网络中的路由器不能及时转发出去而导致丢包。

解决的方法：调整发送端的发送速率，AIMD（Additive Increase Multiplicative Decrease）。

面临的挑战：TCP 在端系统上的传输层，网络丢包发生在核心网络路由器的网络层，TCP 如何检测到拥塞，如何根据网络拥塞程度调整发送端的速率？

AIMD:

基本思想是摸着石头过河。如图 6-1 所示，先从小速率开始不断增大发送速率，直到发生拥塞再降速。

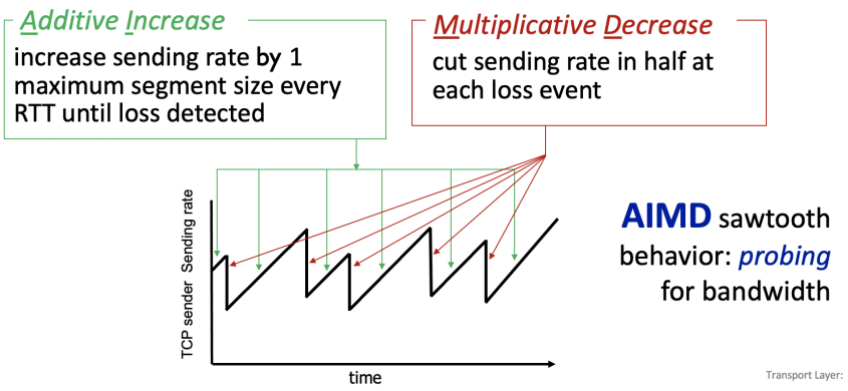


图 6-1 TCP AIMD 拥塞控制示意图

拥塞的判定方法：

（1）三次重复 ACK （Duplicate ACK）

（2）超时

TCP 拥塞控制的 3 个状态（阶段）：

慢启动（slow start）；

拥塞避免（congestion avoidance）；

快速恢复（fast recovery）。

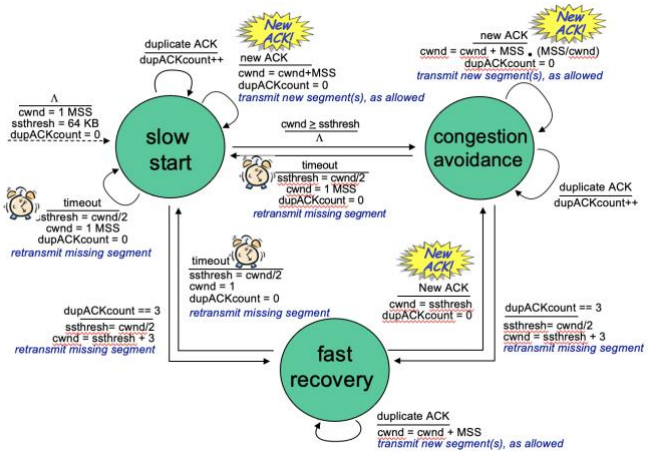


图 6-2 TCP Reno 拥塞控制 FSM 图

要求能够判断在不同事件（超时、收到三次重复 ACK 包）发生时 TCP 如何做拥塞窗口调整，如教材图 3.52 Evaluation of TCP's congestion window。

注：第 8 版教材图 3.52 有误，更改为下图 6-3。原因：TCP Reno 的 fast recovery 阶段是暂态的，通常在 1 个 RTT 之内就结束了。结束后如果转到 congestion avoidance 状态，则拥塞窗口 cwnd 会调整回到刚进入 fast recovery 状态时 cwnd 的 1/2，然后线性增长。因此在发生“收到三次重复 ACK 包”事件后的 1 个 RTT 内如果转入到 congestion avoidance 状态在图中是很难把 fast recovery 状态表现出来的，直接忽略即可。同理，课后习题 P40 的图 3.61 也做了相应修改，更正

过的图已经放在智慧树的习题里了。做题时要使用智慧树习题里的图。

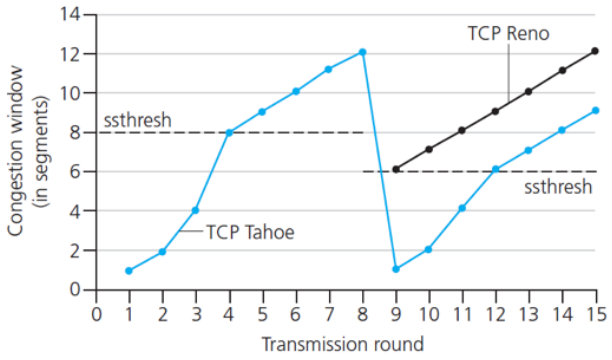


图 6-3 TCP 拥塞控制运行过程示例