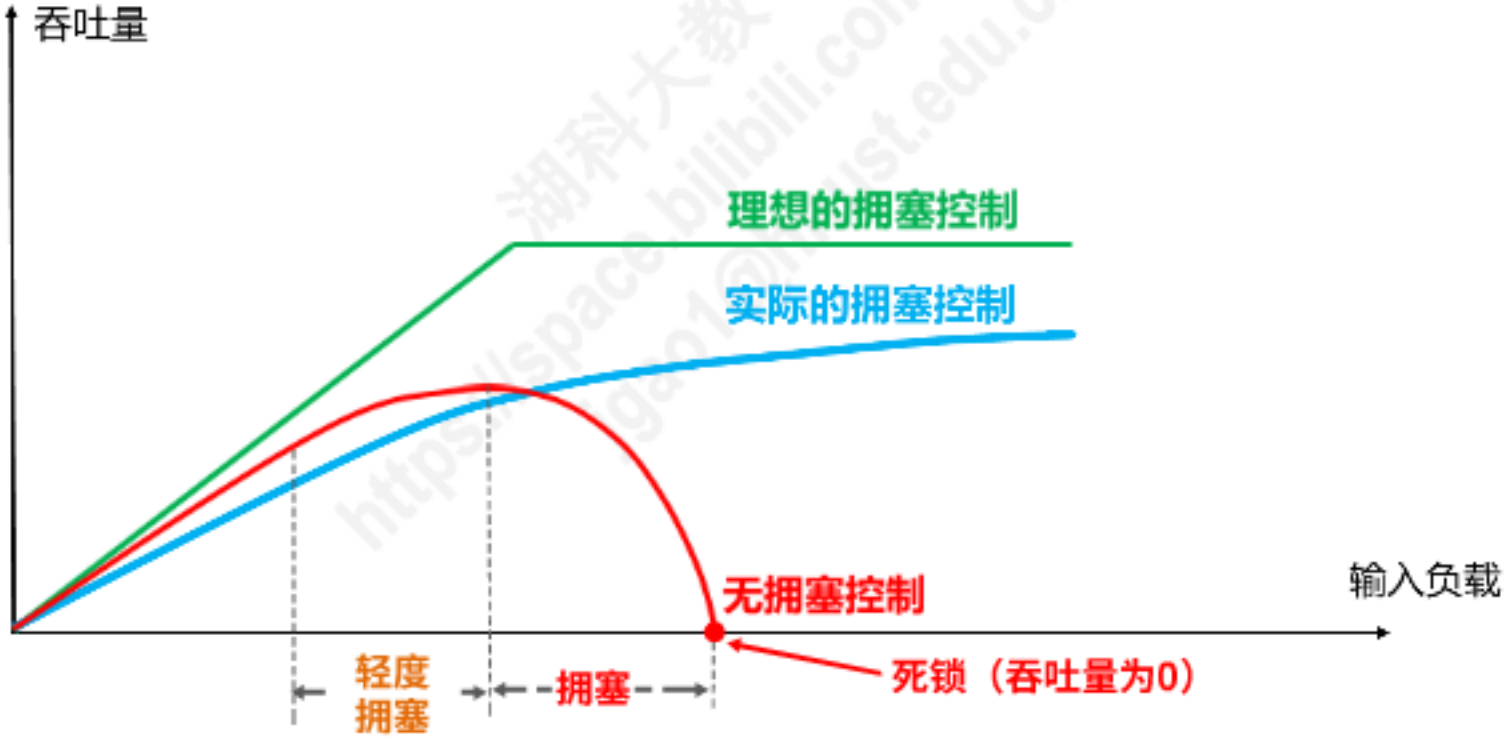
5.5 TCP的拥塞控制

5.5 TCP的拥塞控制

- 在某段时间，若**对网络中某一资源的需求超过了该资源所能提供的可用部分，网络性能就要变坏**。这种情况就叫做**拥塞**(congestion)。
  - 在计算机网络中的链路容量（即带宽）、交换结点中的缓存和处理机等，都是网络的资源。
- 若**出现拥塞而不进行控制**，整个网络的**吞吐量将随输入负荷的增大而下降**。



5.5 TCP的拥塞控制

慢开始 (slow-start)	拥塞避免 (congestion avoidance)	快重传 (fast retransmit)	快恢复 (fast recovery)

下面介绍这四种拥塞控制算法的基本原理，假定如下条件：

1. 数据是单方向传送，而另一个方向只传送确认。
2. 接收方总是有足够大的缓存空间，因而发送方发送窗口的大小由网络的拥塞程度来决定。
3. 以最大报文段MSS的个数为讨论问题的单位，而不是以字节为单位。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 发送方维护一个叫做**拥塞窗口cwnd**的状态变量，其值**取决于网络的拥塞程度**，并且**动态变化**。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 发送方维护一个叫做**拥塞窗口cwnd**的状态变量，其值**取决于网络的拥塞程度**，并且**动态变化**。
  - 拥塞窗口**cwnd的维护原则**：只要网络**没有出现拥塞**，**拥塞窗口**就再**增大**一些;但只要网络**出现拥塞**，**拥塞窗口就减少**一些。
  - 判断出现**网络拥塞的依据**：没有按时收到应当到达的确认报文（即**发生超时重传**）。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 发送方维护一个叫做**拥塞窗口cwnd**的状态变量，其值**取决于网络的拥塞程度**，并且**动态变化**。
  - 拥塞窗口**cwnd的维护原则**：只要网络**没有出现拥塞**，**拥塞窗口**就再**增大**一些;但只要网络**出现拥塞**，**拥塞窗口就减少**一些。
  - 判断出现**网络拥塞的依据**：没有按时收到应当到达的确认报文（即**发生超时重传**）。
- 发送方将拥塞窗口作为**发送窗口swnd**，即**swnd = cwnd**。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 发送方维护一个叫做**拥塞窗口cwnd**的状态变量，其值**取决于网络的拥塞程度**，并且**动态变化**。
  - 拥塞窗口**cwnd的维护原则**：只要网络**没有出现拥塞**，**拥塞窗口**就再**增大**一些;但只要网络**出现拥塞**，**拥塞窗口就减少**一些。
  - 判断出现**网络拥塞的依据**：没有按时收到应当到达的确认报文（即**发生超时重传**）。
- 发送方将拥塞窗口作为**发送窗口swnd**，即**swnd = cwnd**。
- 维护一个慢开始门限**ssthresh**状态变量：

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 发送方维护一个叫做**拥塞窗口cwnd**的状态变量，其值**取决于网络的拥塞程度**，并且**动态变化**。
  - 拥塞窗口**cwnd的维护原则**：只要网络**没有出现拥塞**，**拥塞窗口**就再**增大**一些;但只要网络**出现拥塞**，**拥塞窗口就减少**一些。
  - 判断出现**网络拥塞的依据**：没有按时收到应当到达的确认报文（即**发生超时重传**）。
- 发送方将拥塞窗口作为**发送窗口swnd**，即**swnd = cwnd**。
- 维护一个慢开始门限**ssthresh**状态变量：
  - 当cwnd < ssthresh时，使用慢开始算法;
  - 当cwnd > ssthresh时，停止使用慢开始算法而改用拥塞避免算法;
  - 当cwnd = ssthresh时，既可使用慢开始算法，也可使用拥塞避免算法。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd
swnd=cwnd
ssthresh

接收方

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=1
swnd=cwnd
ssthresh

接收方

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=1
swnd=cwnd
ssthresh

接收方

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

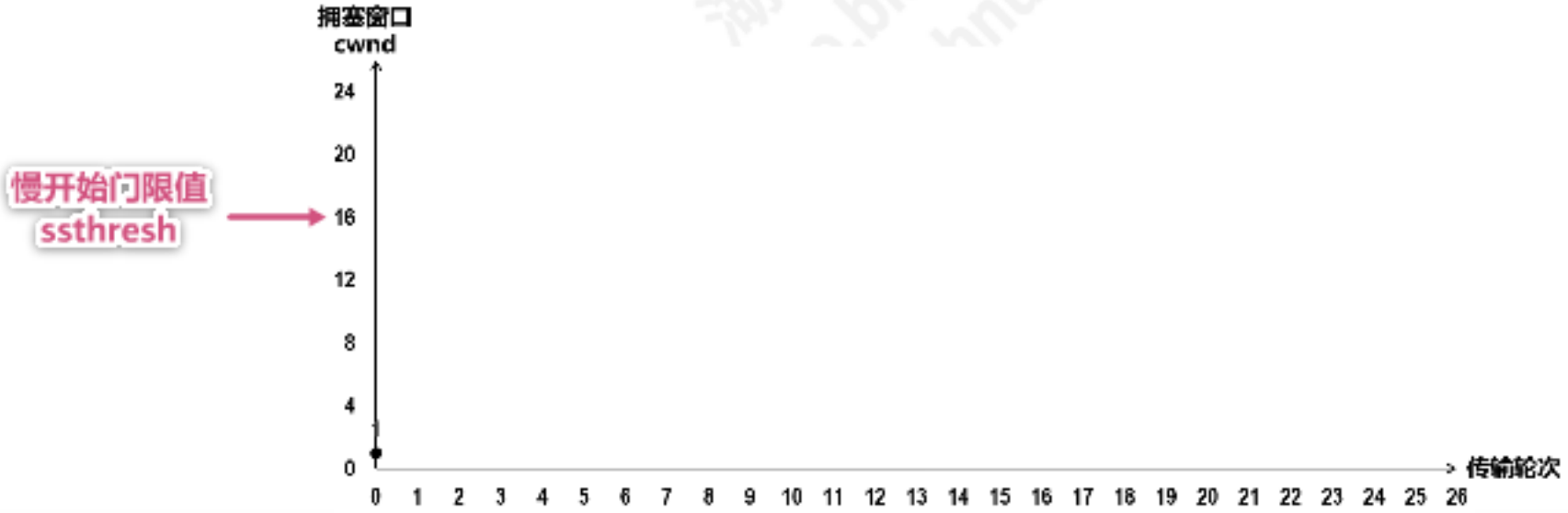
发送方

cwnd=1
swnd=cwnd
ssthresh=16

对报文段0的确认

TCP数据报文段0

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

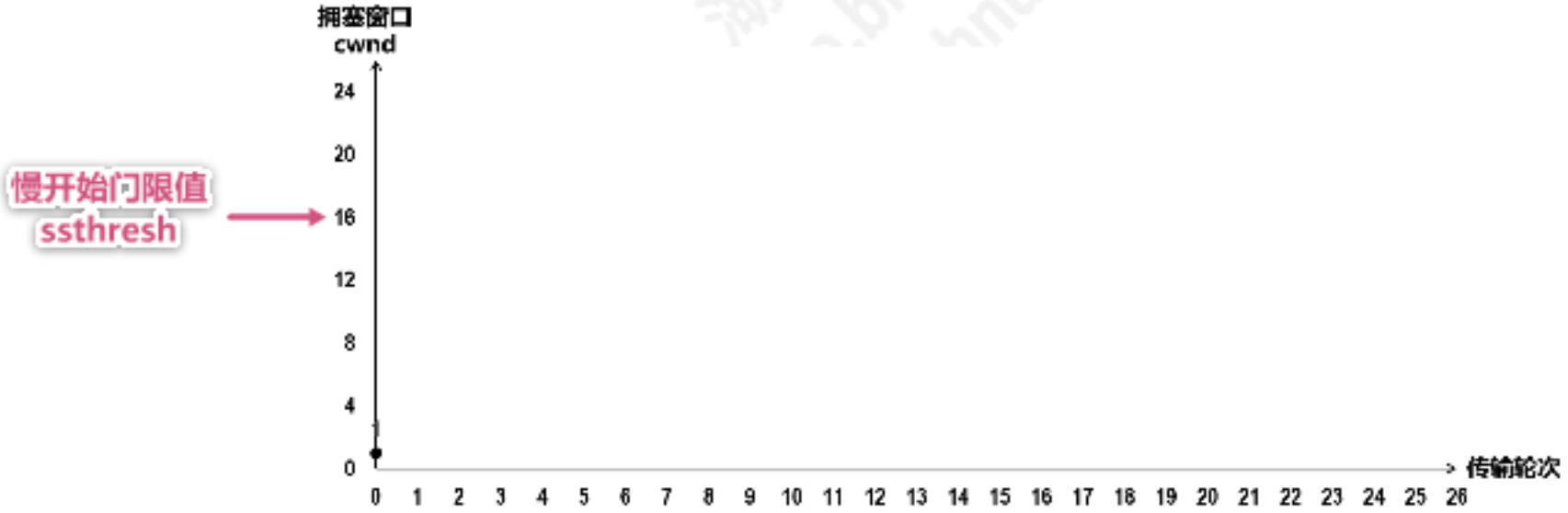
发送方

cwnd=1+1=2
swnd=cwnd
ssthresh=16

对报文段0的确认

TCP数据报文段0

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

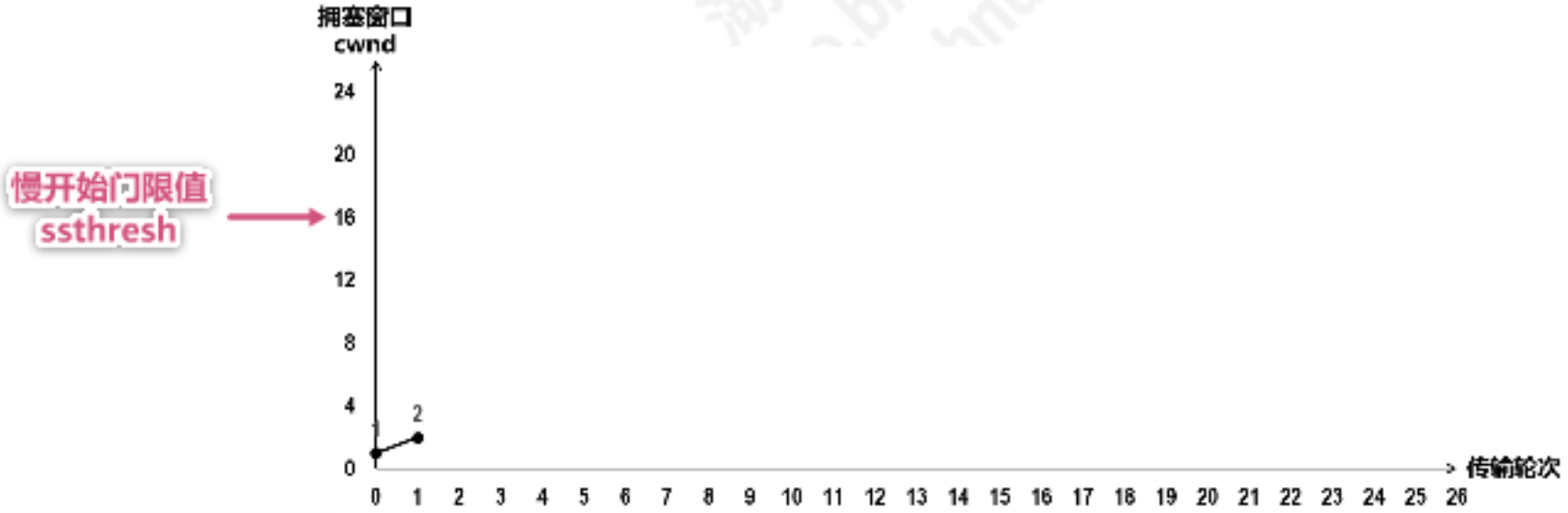
发送方

cwnd=1+1=2
swnd=cwnd
ssthresh=16

对报文段1～2的确认

TCP数据报文段1～2

接收方



慢开始门限值
ssthresh

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

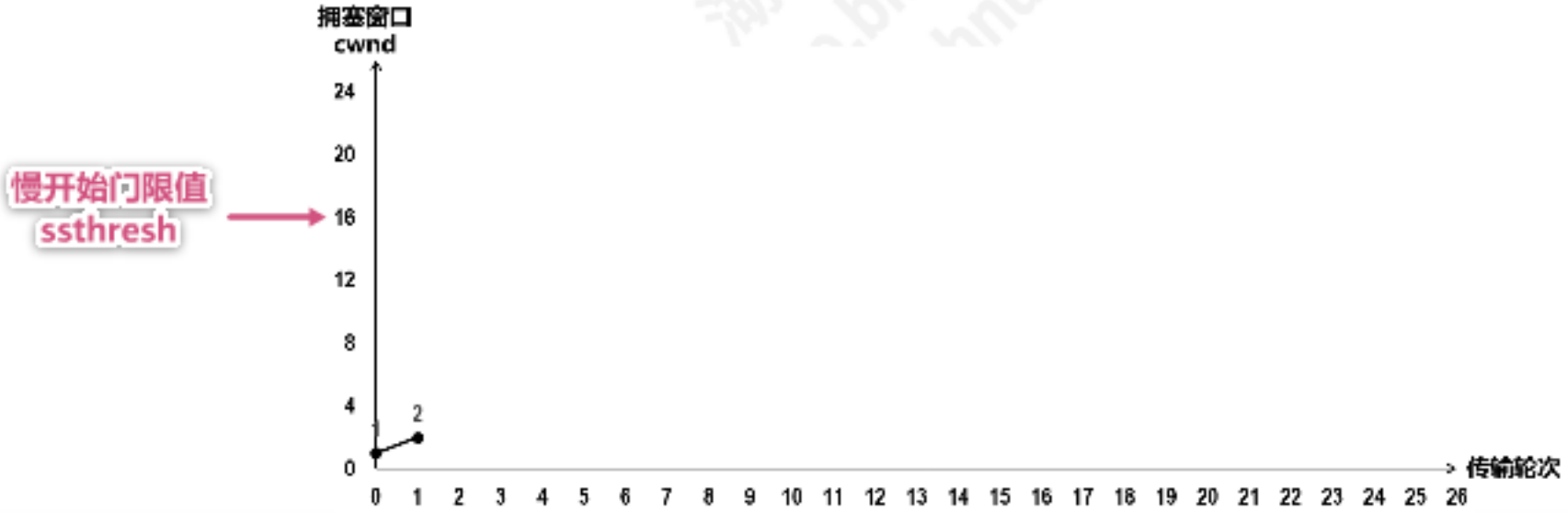
发送方

cwnd=2+2=4
swnd=cwnd
ssthresh=16

对报文段1～2的确认

TCP数据报文段1～2

接收方



慢开始门限值
ssthresh

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

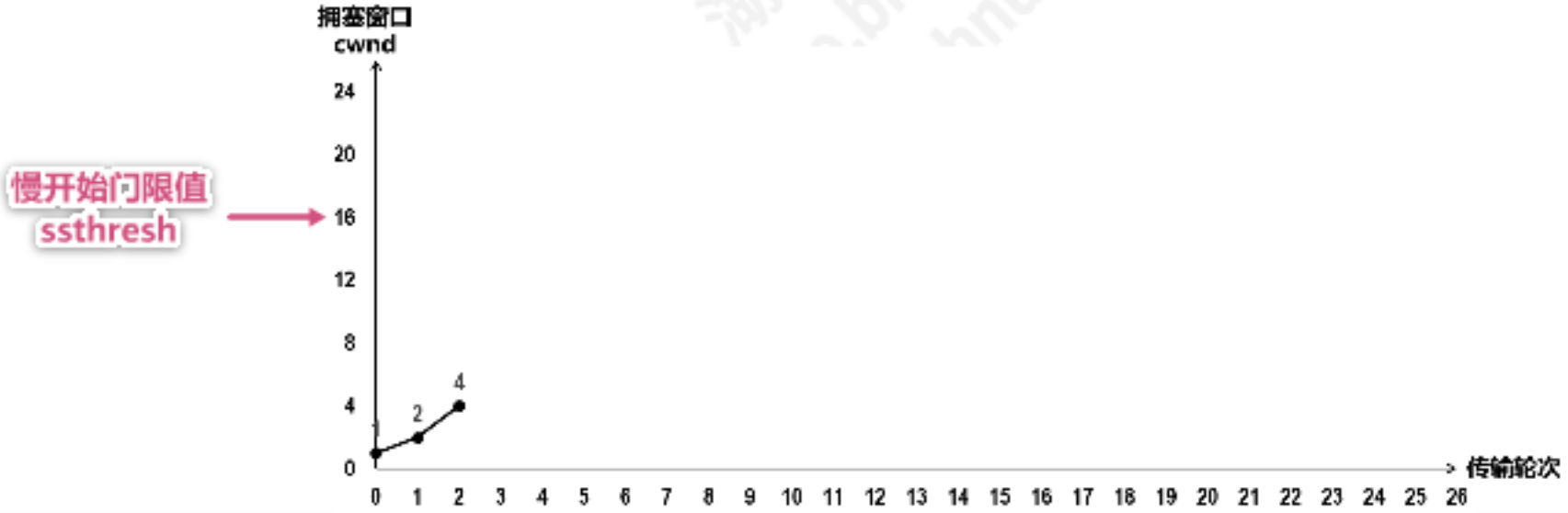
发送方

cwnd=2+2=4
swnd=cwnd
ssthresh=16

对报文段3~6的确认

TCP数据报文段3~6

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

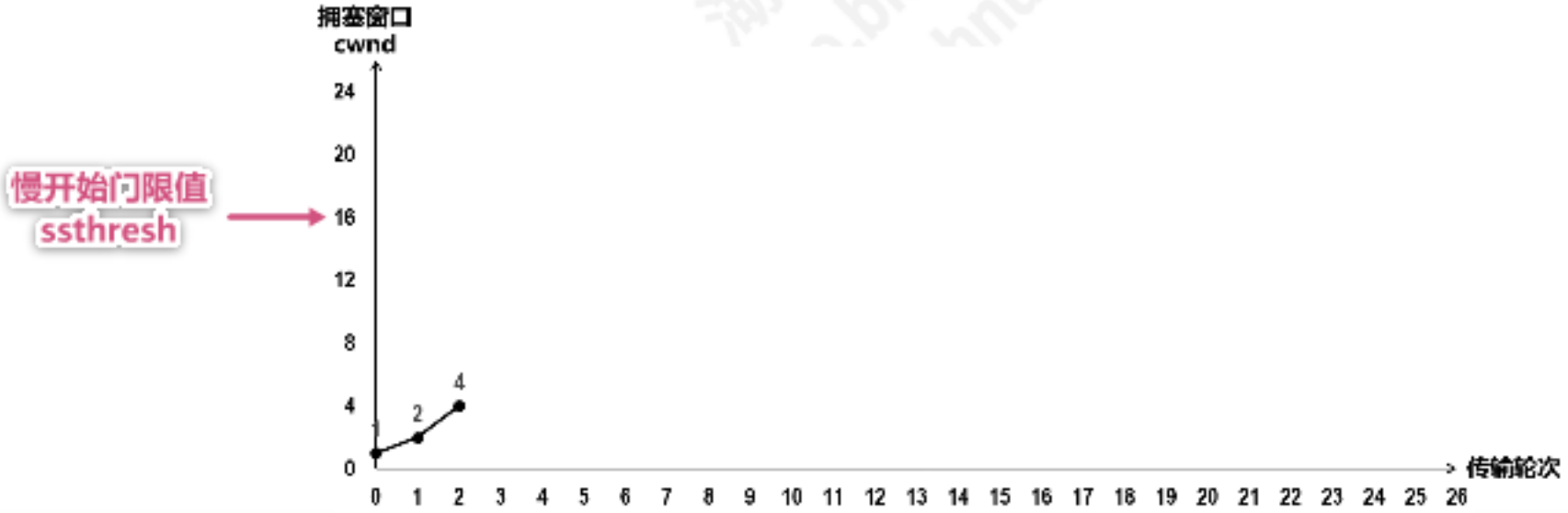
发送方

cwnd=4+4=8
swnd=cwnd
ssthresh=16

对报文段3~6的确认

TCP数据报文段3~6

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

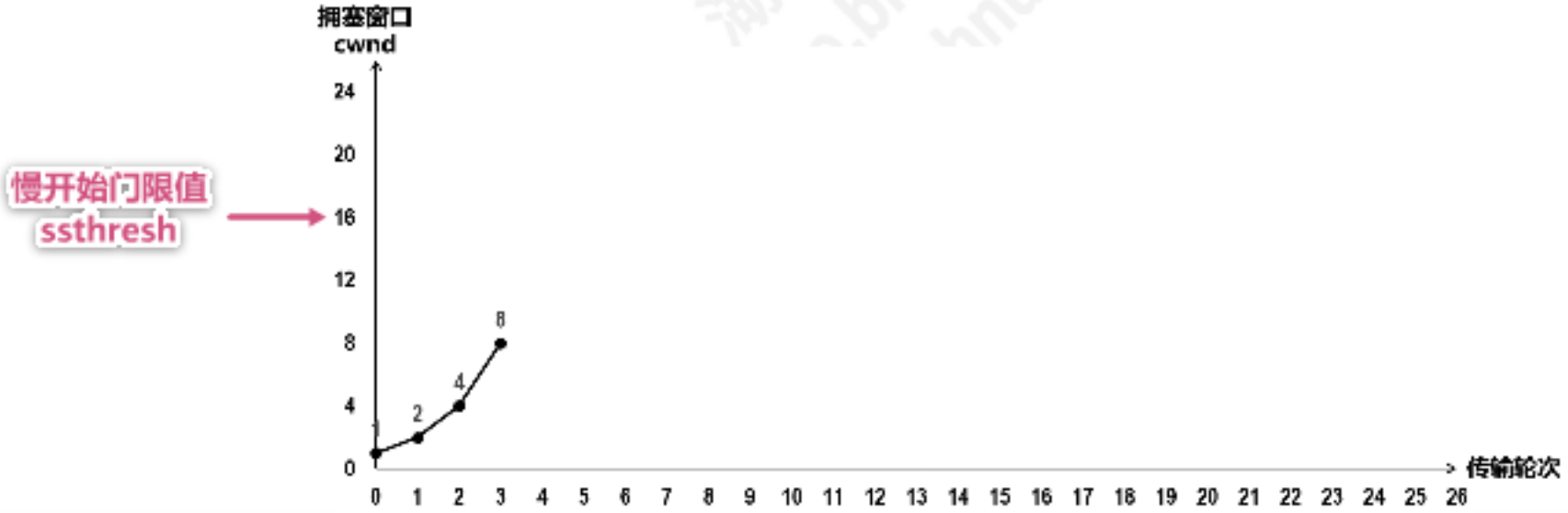
发送方

cwnd=4+4=8
swnd=cwnd
ssthresh=16

对报文段7~14的确认

TCP数据报文段7~14

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

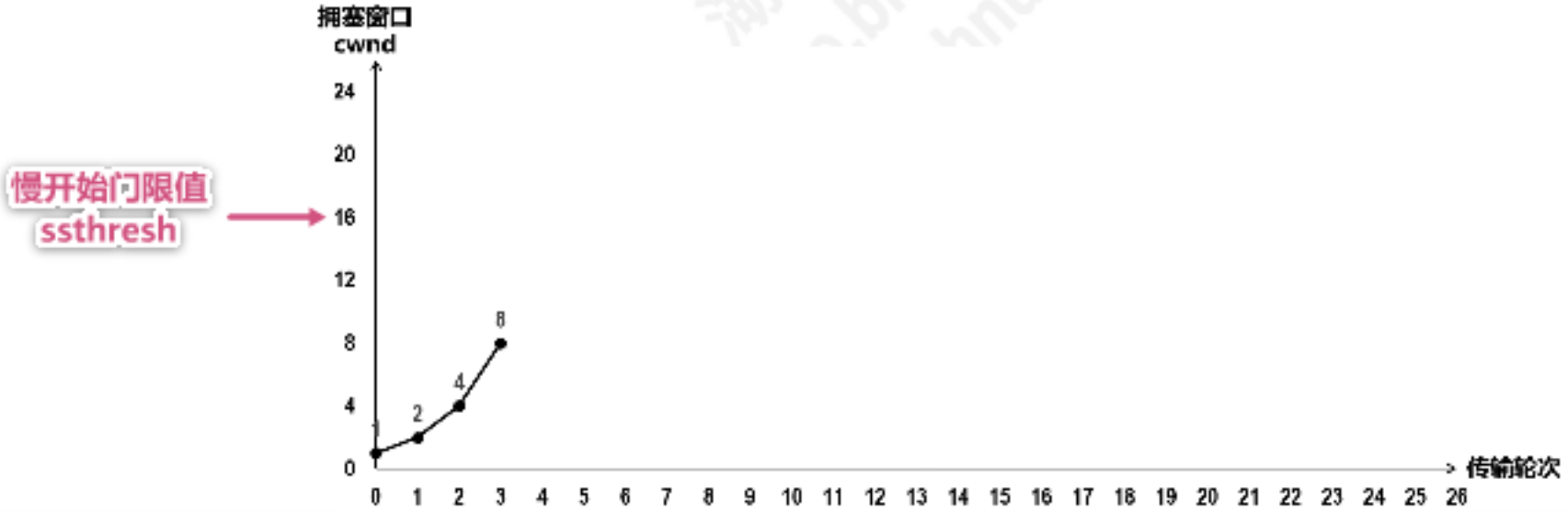
发送方

cwnd=8+8=16
swnd=cwnd
ssthresh=16

对报文段7~14的确认

TCP数据报文段7~14

接收方



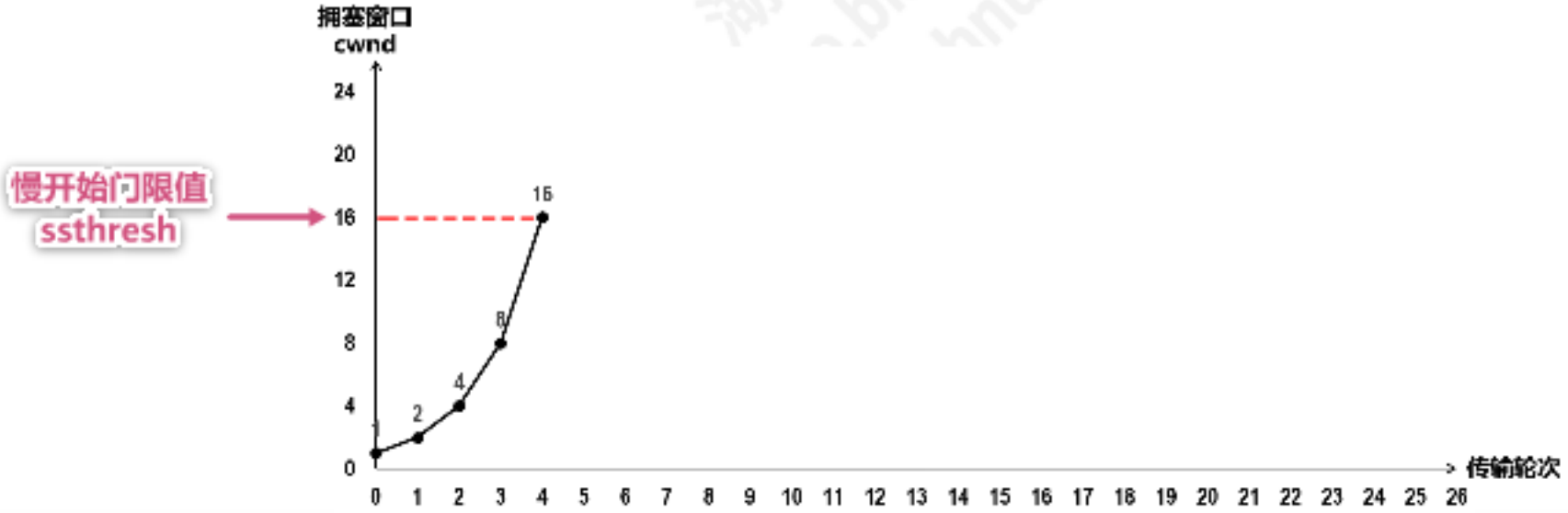
5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=8+8=16
swnd=cwnd
ssthresh=16

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

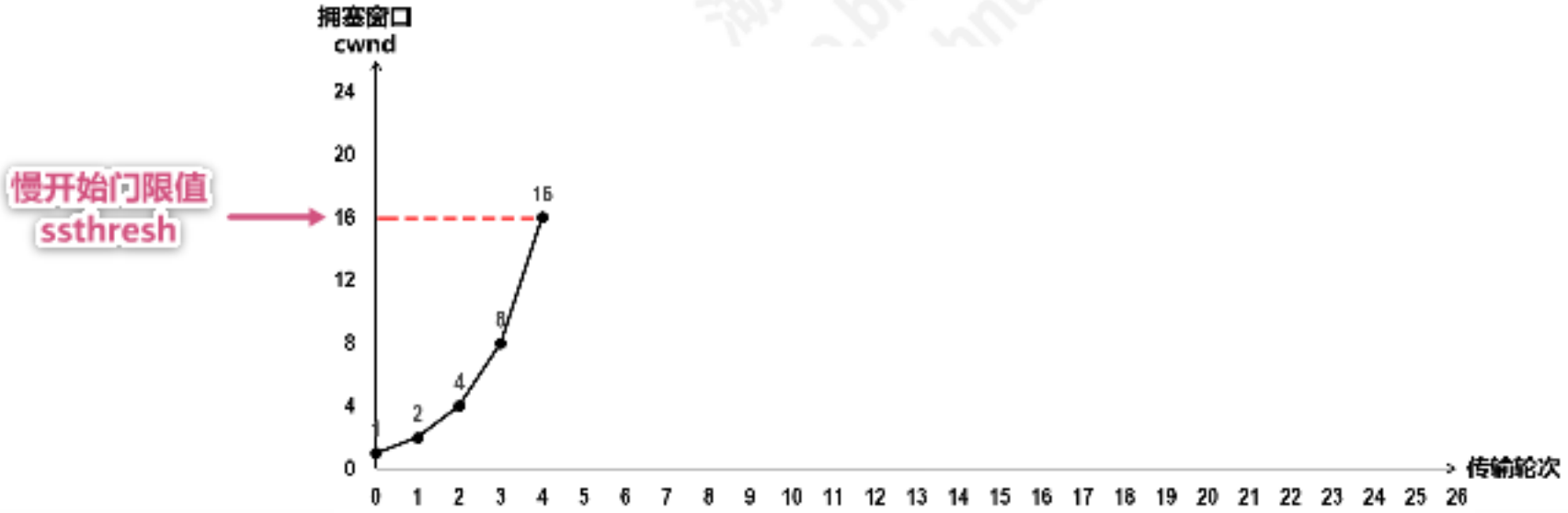
发送方

cwnd=8+8=16
swnd=cwnd
ssthresh=16

对报文段15~30的确认

TCP数据报文段15~30

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=16+1=17
swnd=cwnd
ssthresh=16

对报文段15~30的确认

TCP数据报文段15~30

接收方

慢开始门限值 ssthresh

5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

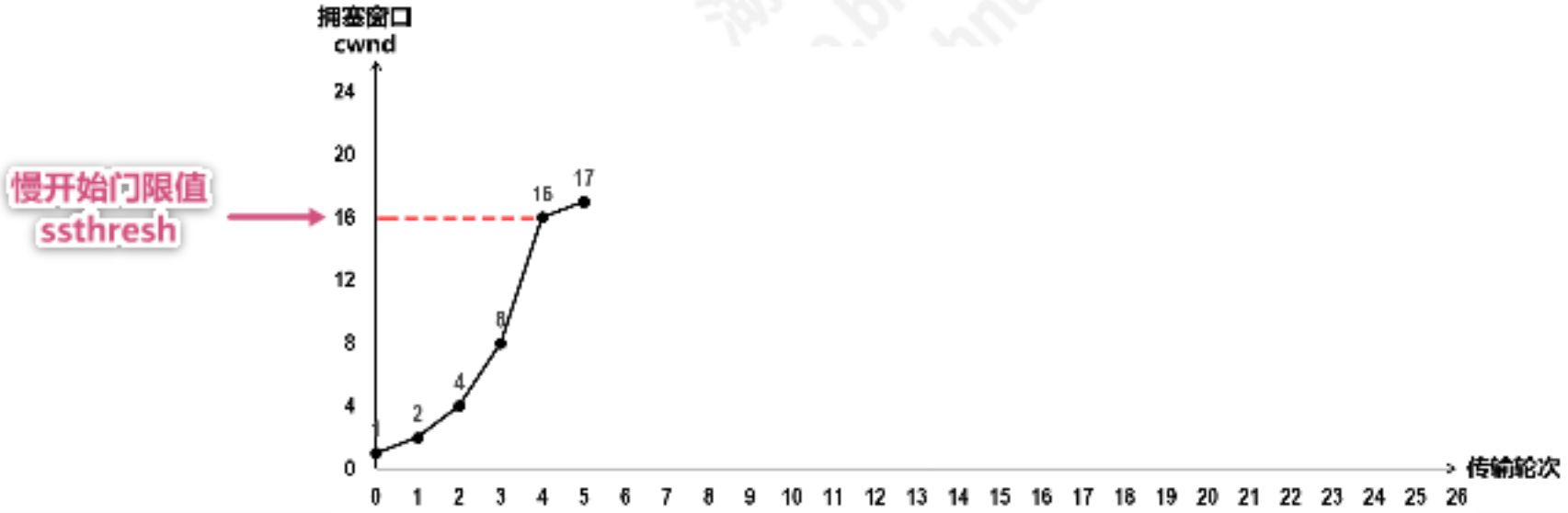
发送方

cwnd=16+1=17
swnd=cwnd
ssthresh=16

对报文段31~47的确认

TCP数据报文段31~47

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=17+1=18
swnd=cwnd
ssthresh=16

对报文段31~47的确认

TCP数据报文段31~47

接收方

慢开始门限值 ssthresh

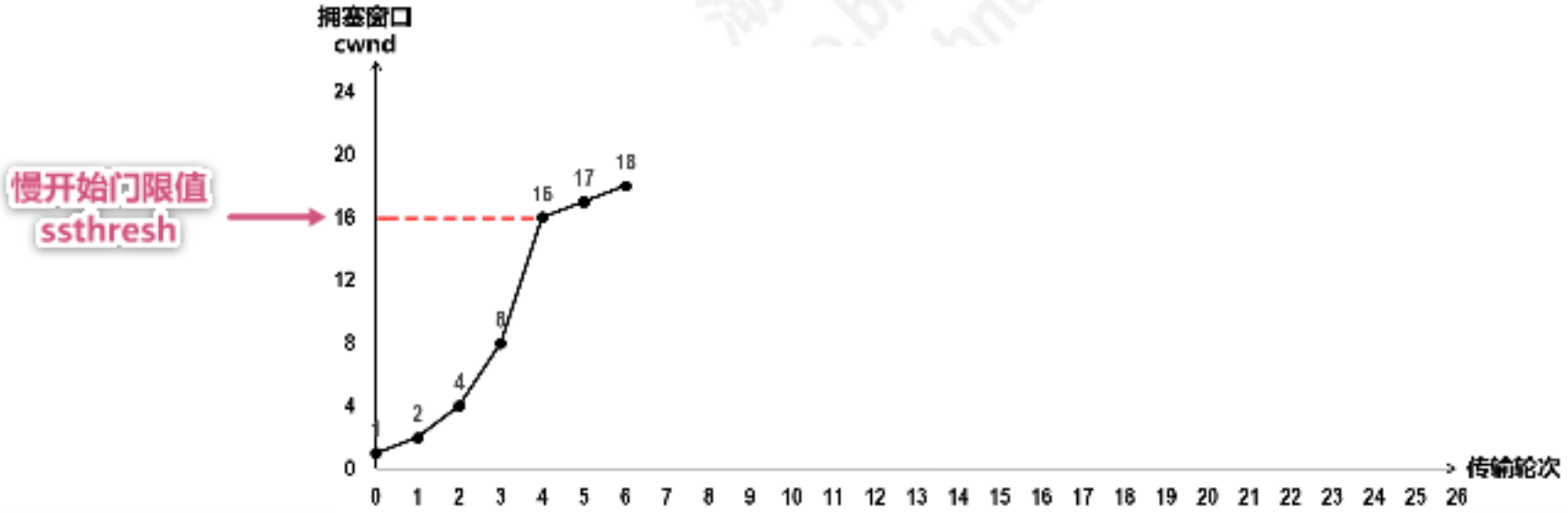
5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=17+1=18
swnd=cwnd
ssthresh=16

接收方



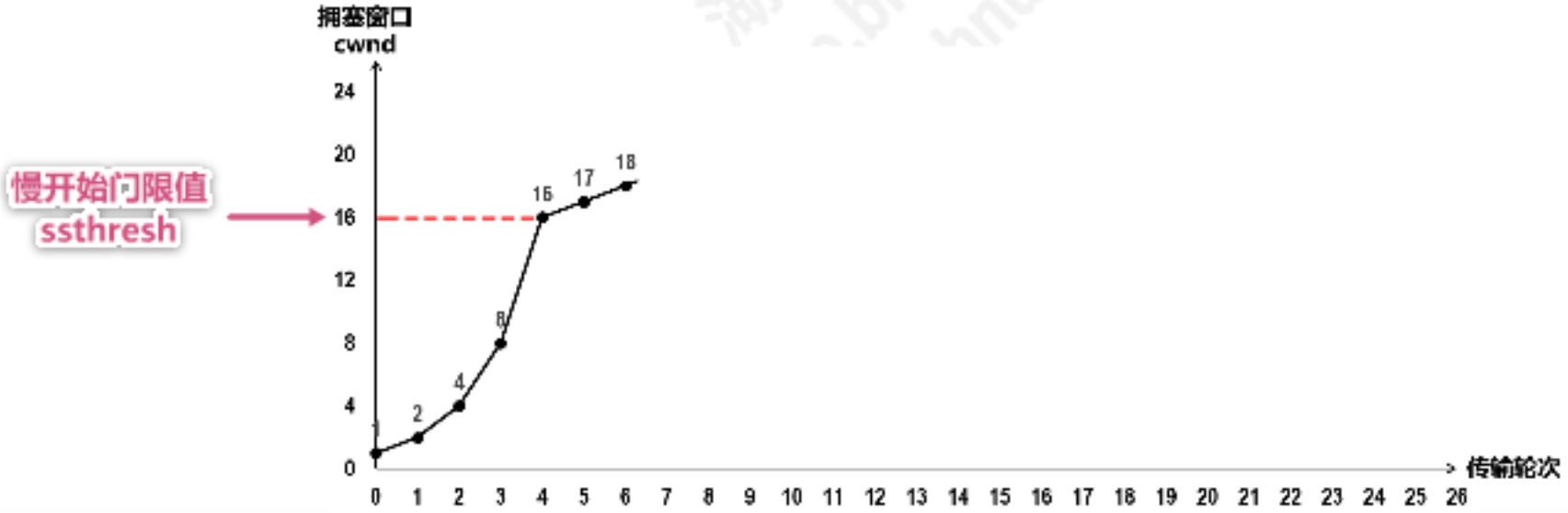
5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=23+1=24
swnd=cwnd
ssthresh=16

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

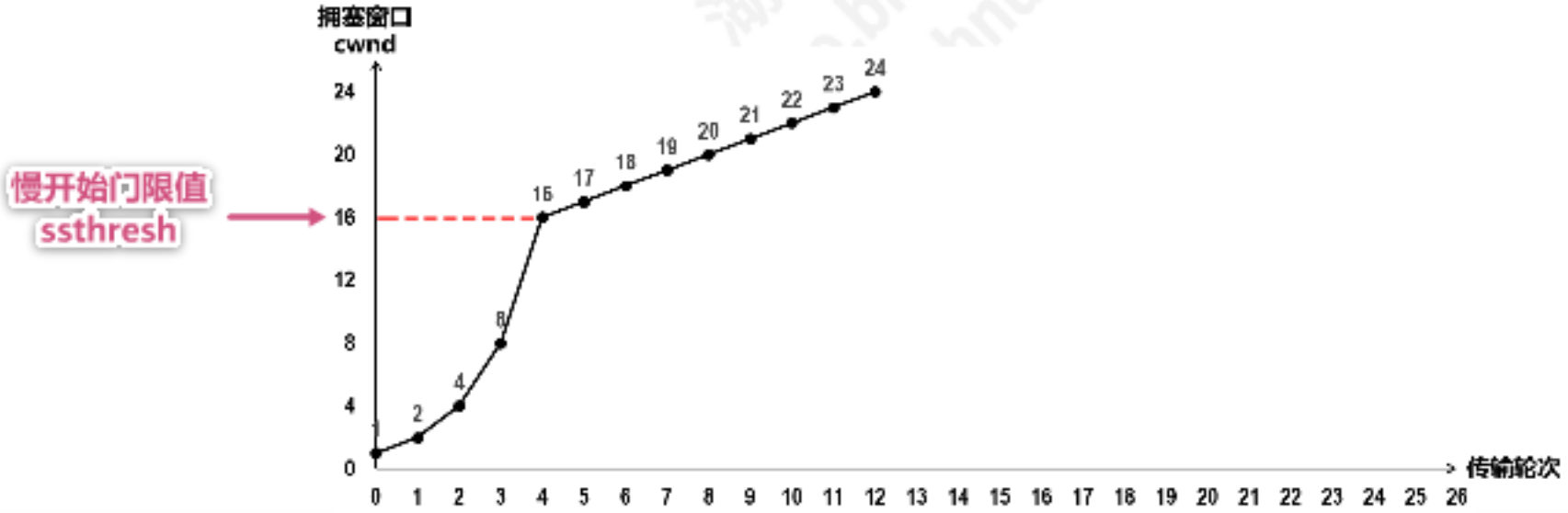
发送方

cwnd=23+1=24
swnd=cwnd
ssthresh=16

TCP数据报文段171~194

部分报文段丢失

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

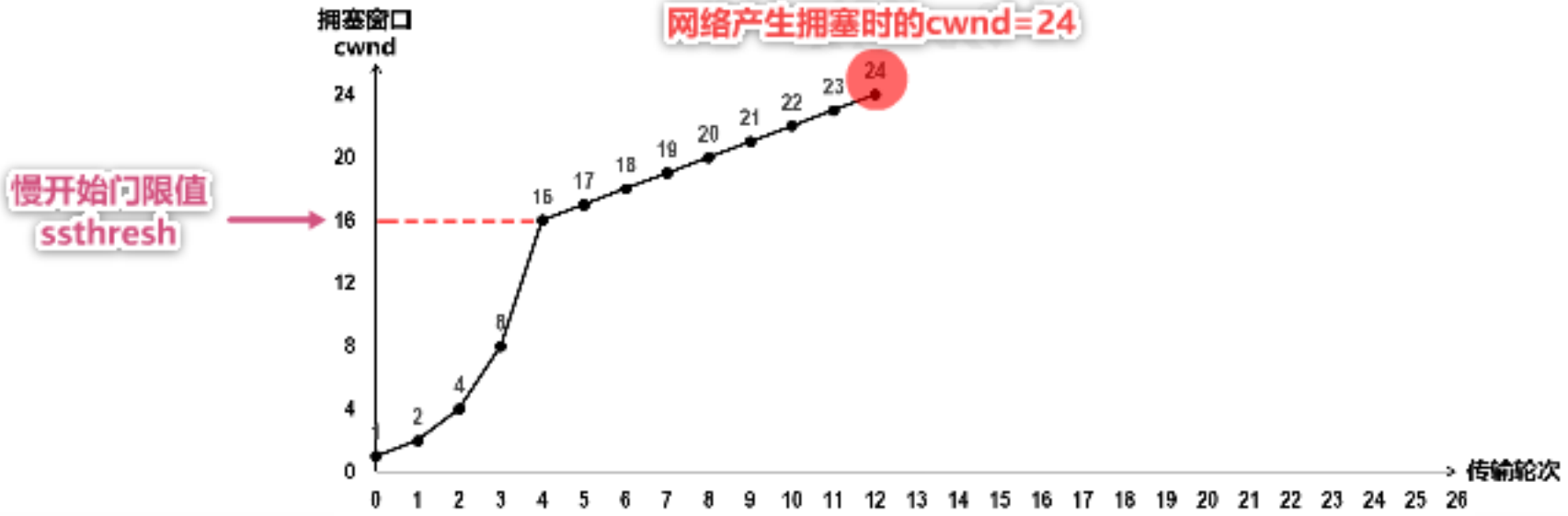
cwnd=24
swnd=cwnd
ssthresh=12

重传计时器超时

判断网络很可能出现了拥塞，进行以下工作：

1. 将ssthresh值更新为发生拥塞时cwnd值的一半；

接收方



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

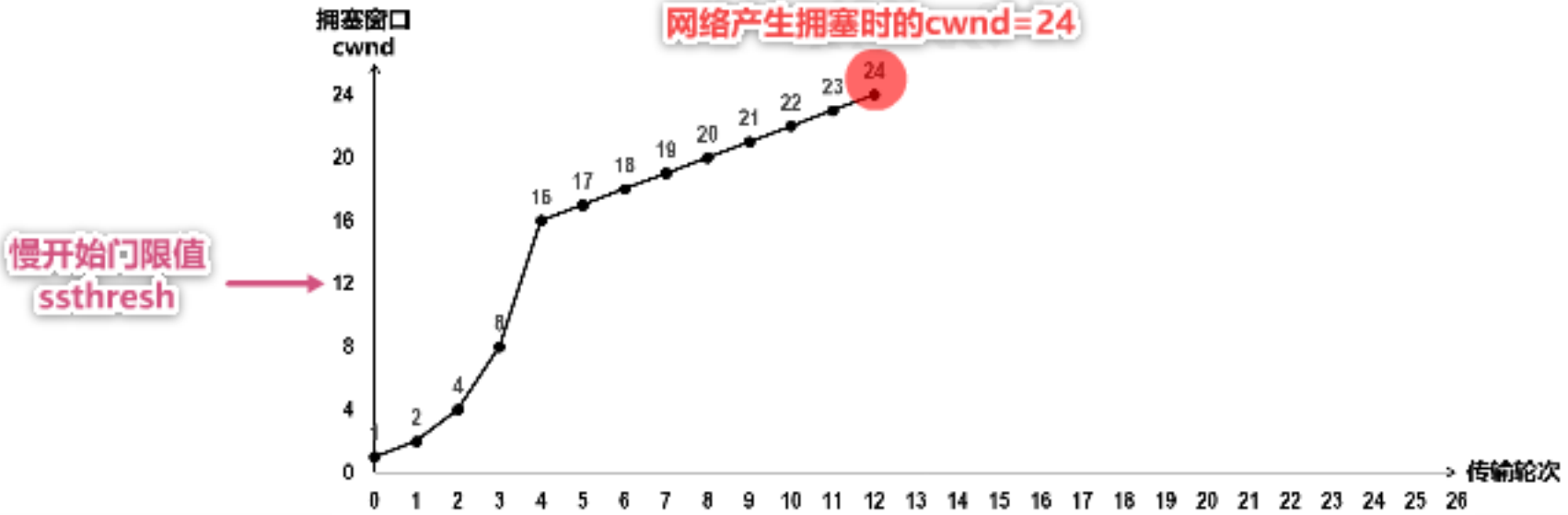
cwnd=1
swnd=cwnd
ssthresh=12

接收方

重传计时器超时

判断网络很可能出现了拥塞，进行以下工作：

1. 将ssthresh值更新为发生拥塞时cwnd值的一半；
2. 将cwnd值减少为1，并重新开始执行慢开始算法。



5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

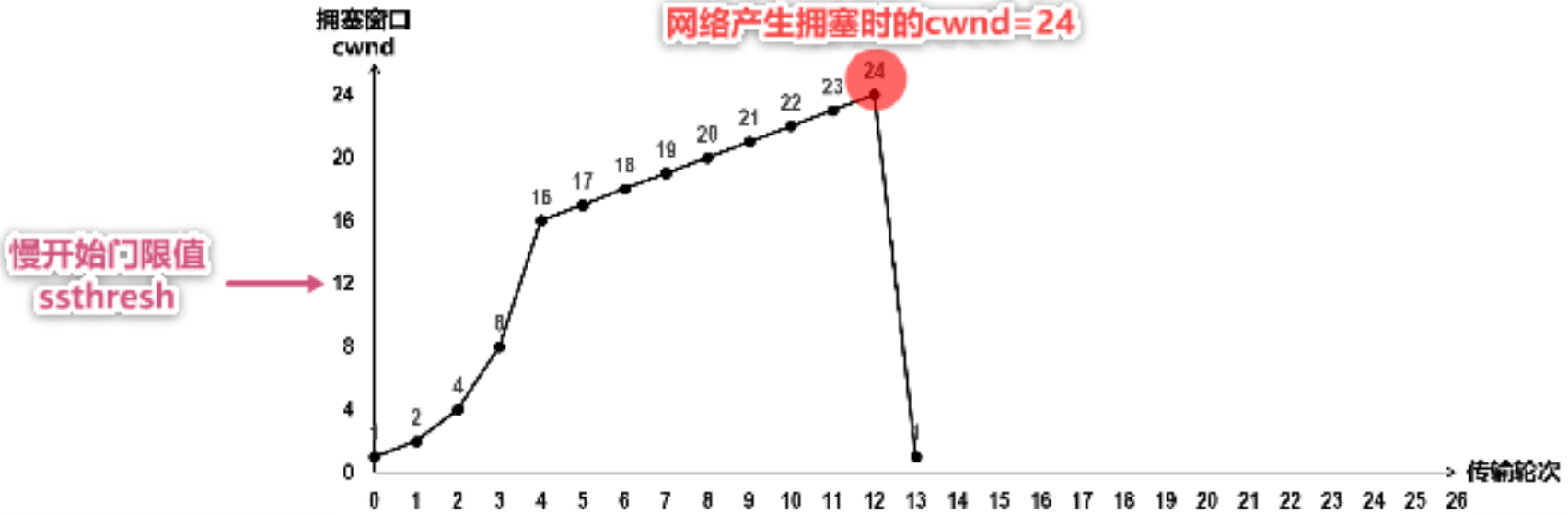
cwnd=1
swnd=cwnd
ssthresh=12

接收方

重传计时器超时

判断网络很可能出现了拥塞，进行以下工作：

1. 将ssthresh值更新为发生拥塞时cwnd值的一半；
2. 将cwnd值减少为1，并重新开始执行慢开始算法。



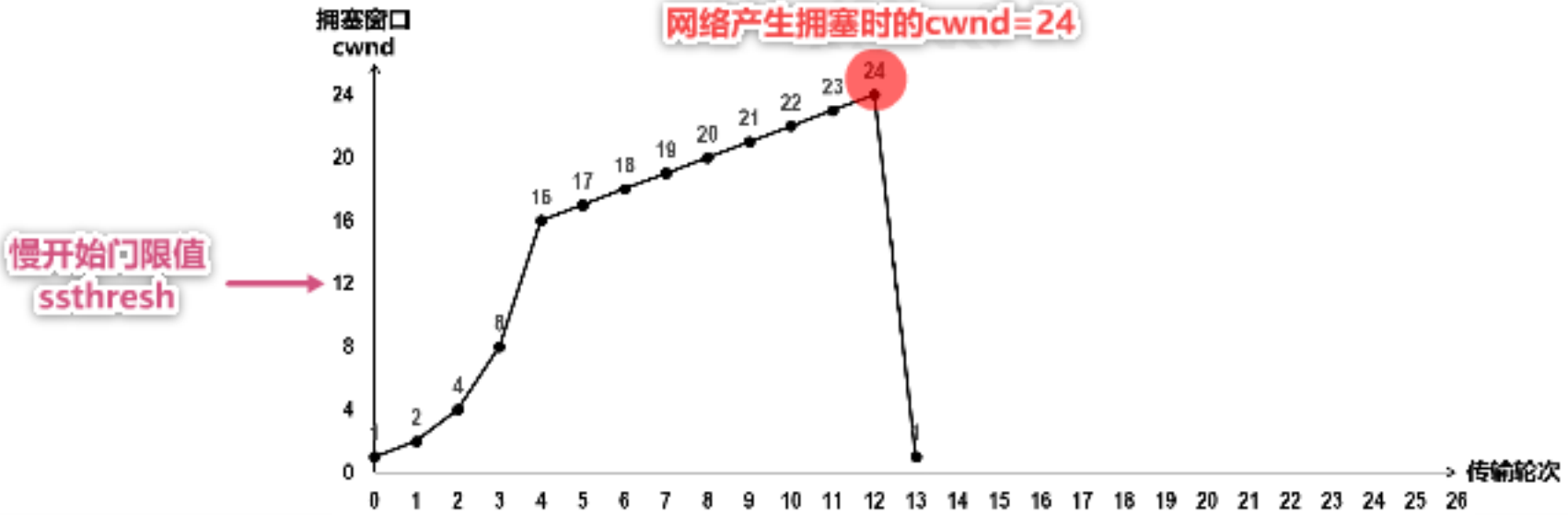
5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=1
swnd=cwnd
ssthresh=12

接收方



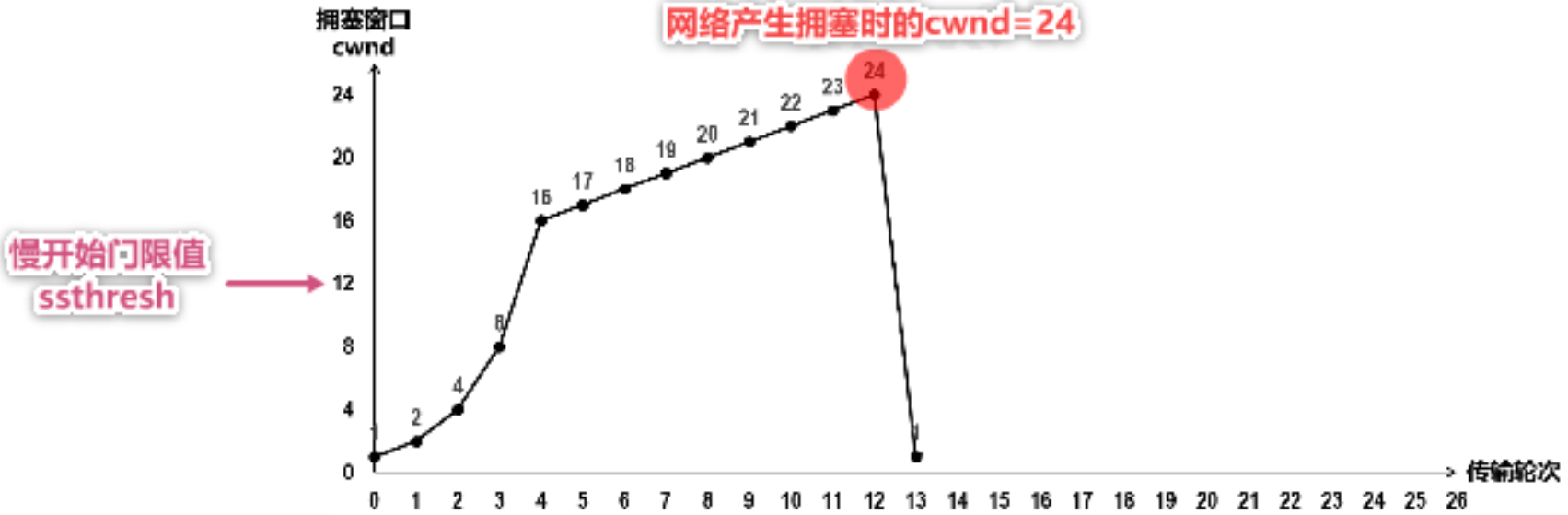
5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=12
swnd=cwnd
ssthresh=12

接收方



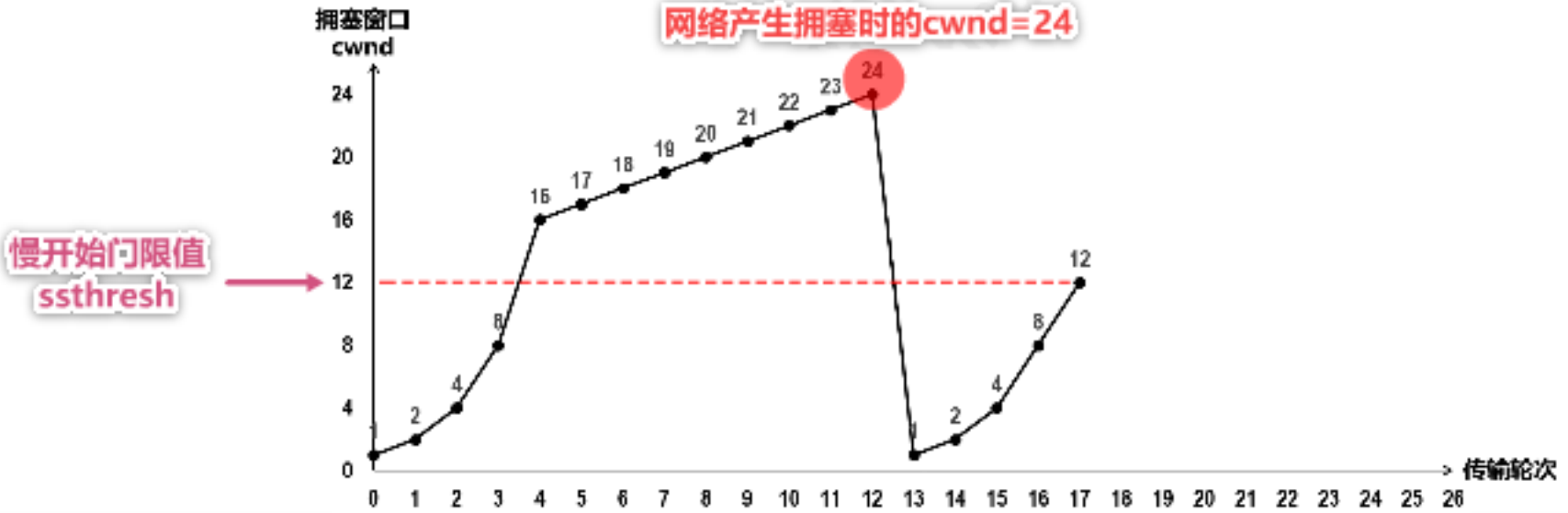
5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=12
swnd=cwnd
ssthresh=12

接收方



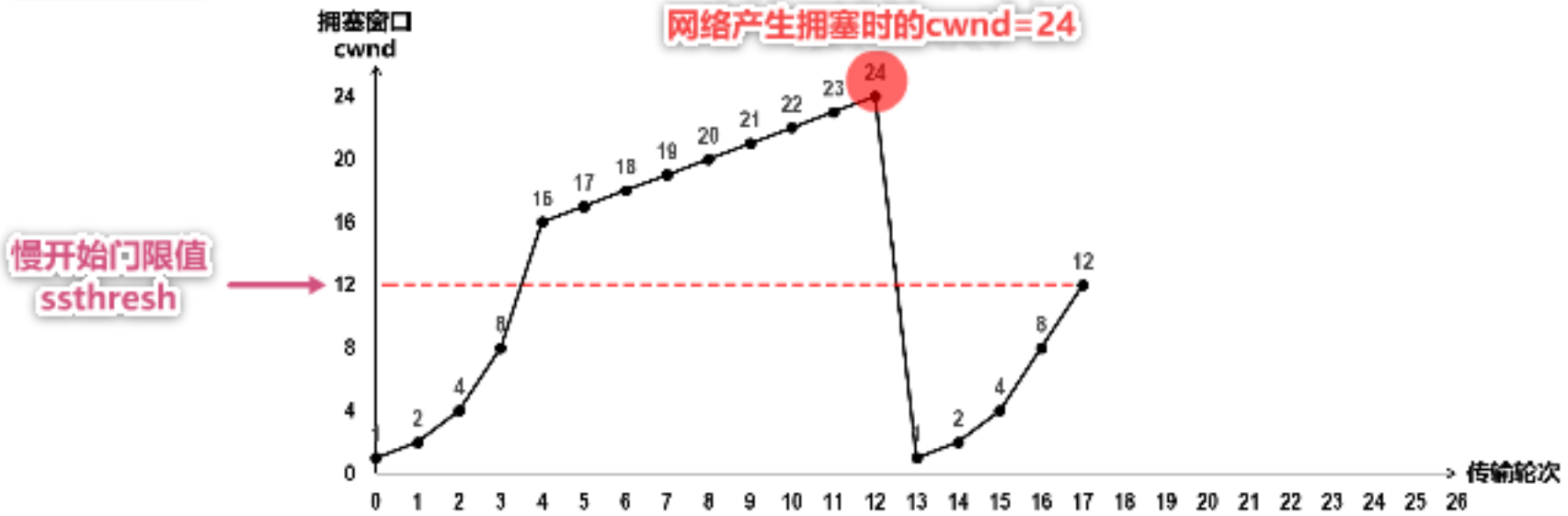
5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=12
swnd=cwnd
ssthresh=12

接收方



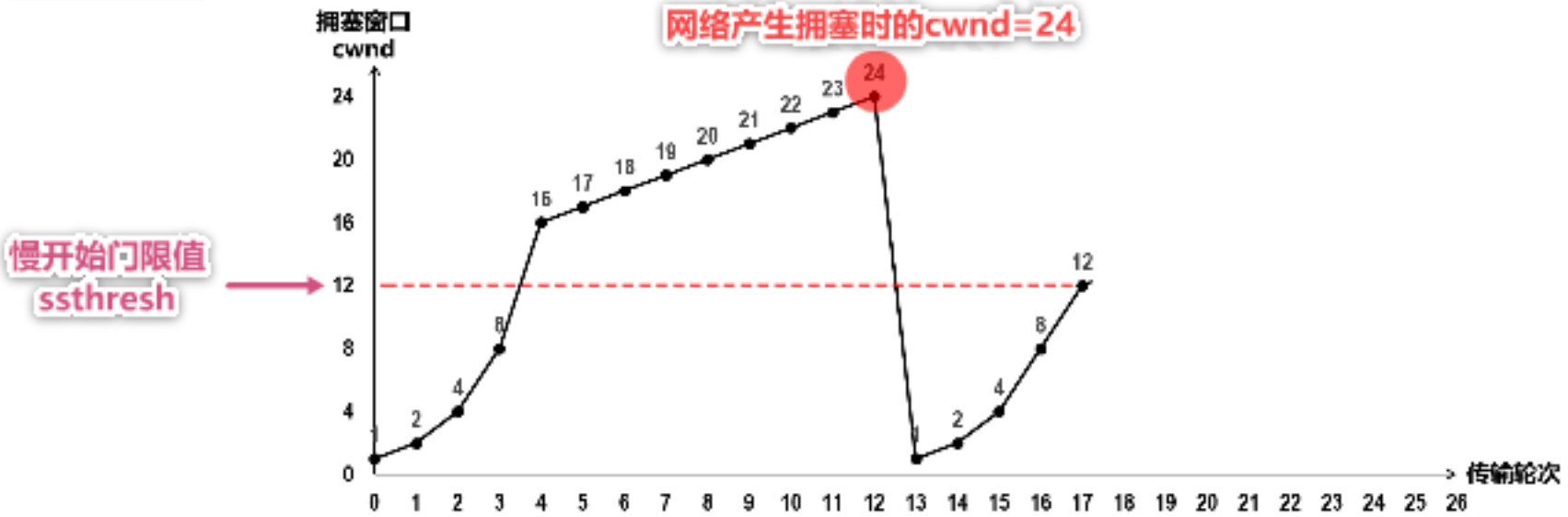
5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

cwnd=16
swnd=cwnd
ssthresh=12

接收方



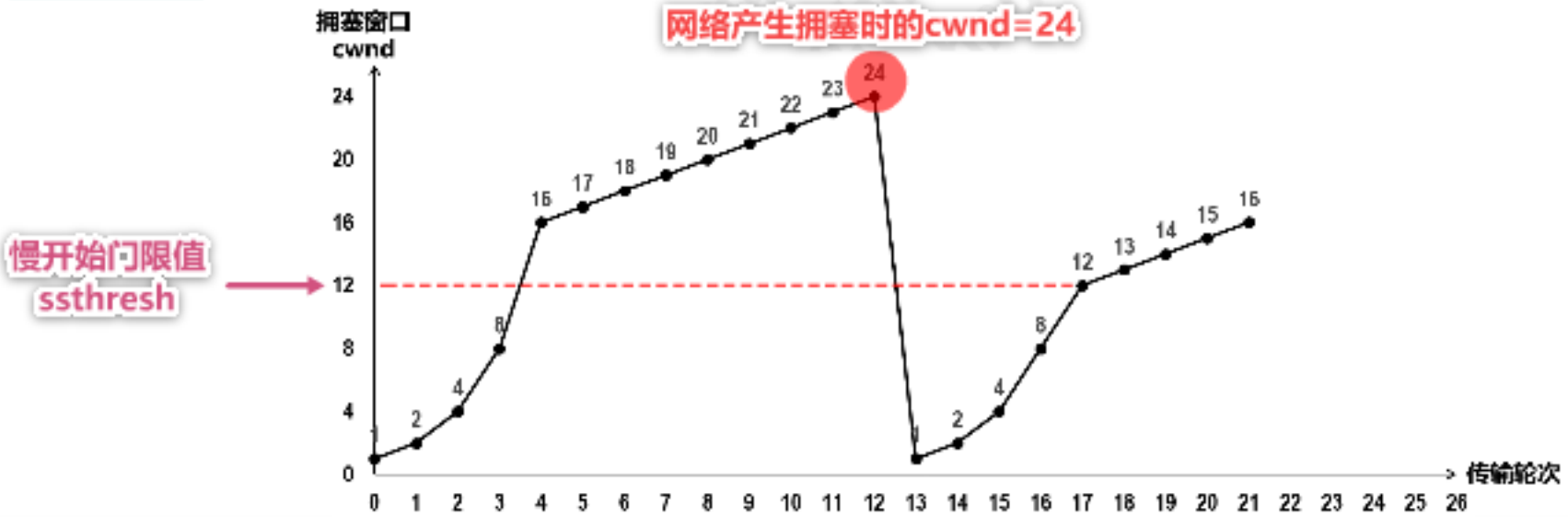
5.5 TCP的拥塞控制

慢开始 (slow-start) | **拥塞避免 (congestion avoidance)** | 快重传 (fast retransmit) | 快恢复 (fast recovery)

发送方

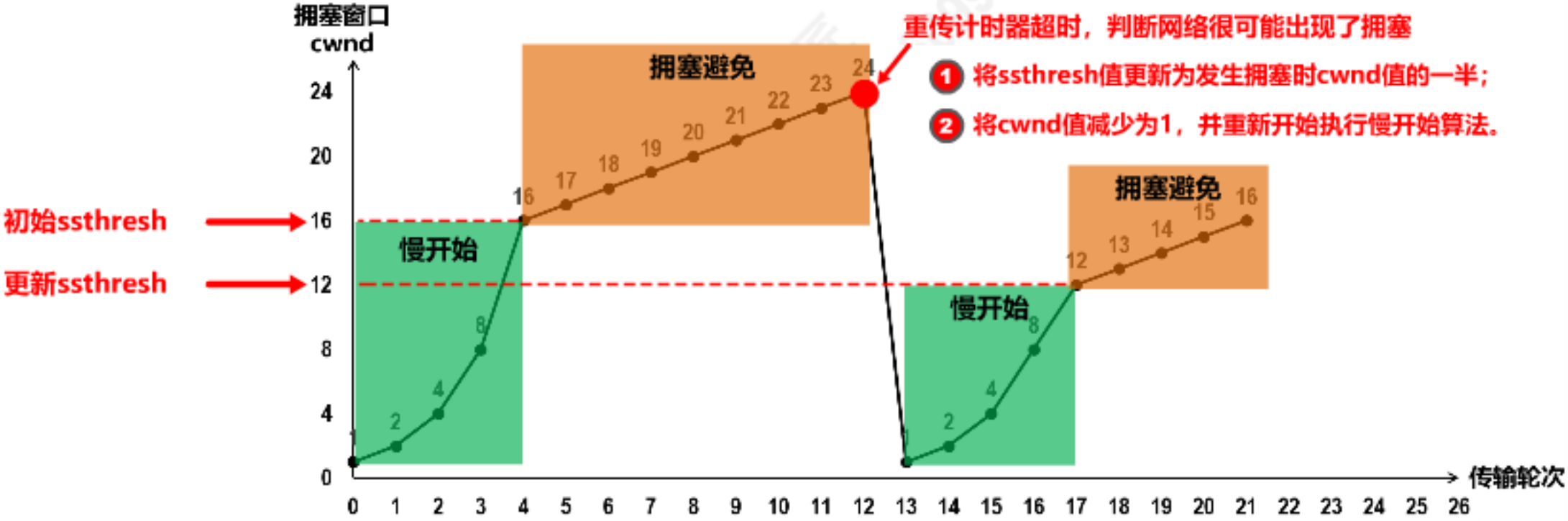
cwnd=16
swnd=cwnd
ssthresh=12

接收方



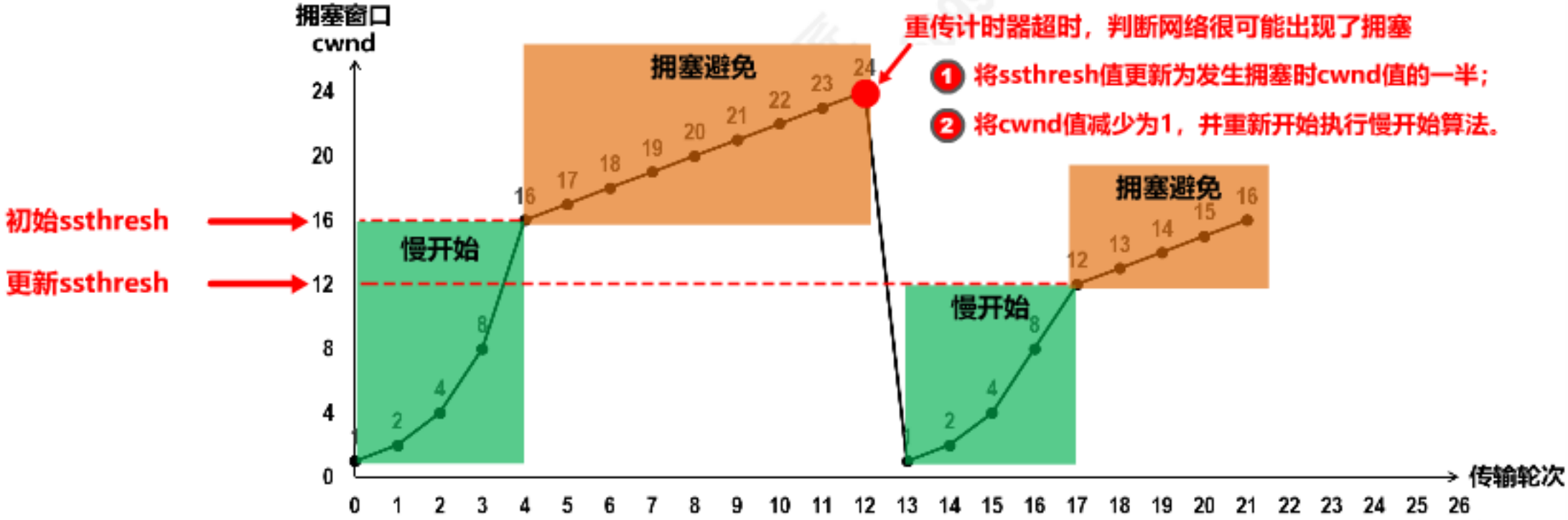
5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

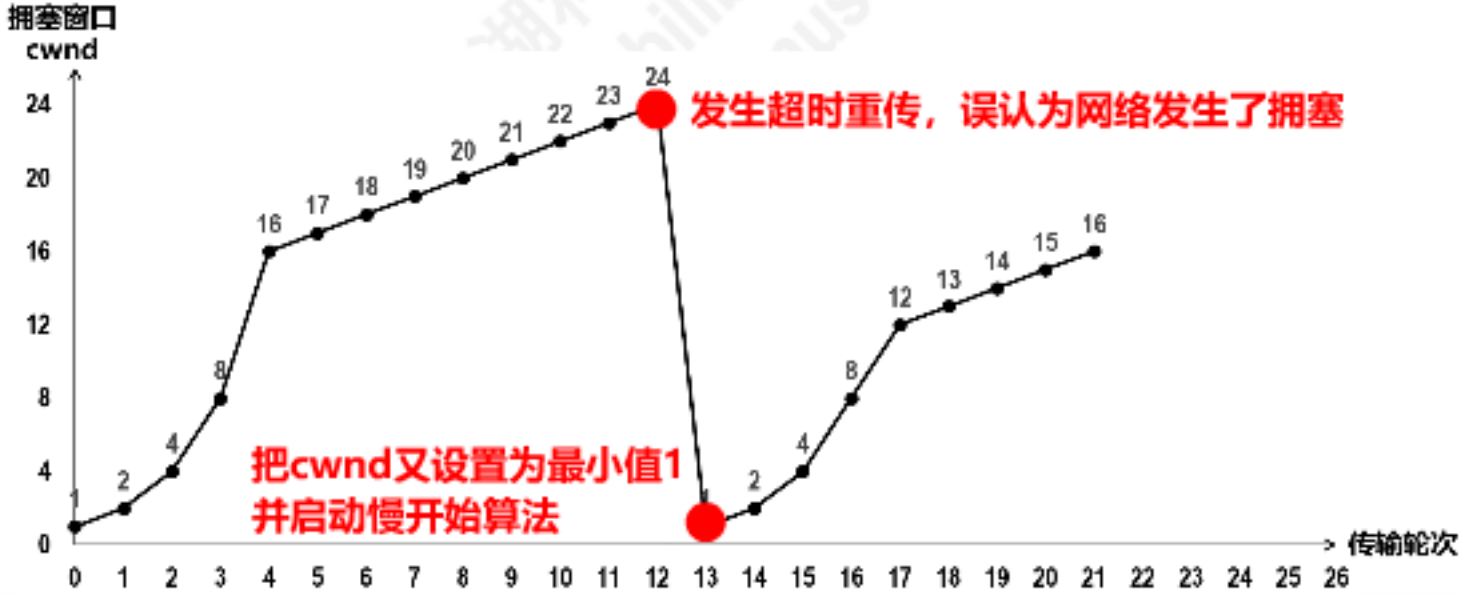


- “慢开始”是指一开始向网络注入的报文段少，并不是指拥塞窗口cwnd增长速度慢；
- “拥塞避免”并非指完全能够避免拥塞，而是指在拥塞避免阶段将拥塞窗口控制为按线性规律增长，使网络比较不容易出现拥塞；

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 慢开始和拥塞避免算法是1988年提出的TCP拥塞控制算法（TCP Tahoe版本）。
- 1990年又增加了两个新的拥塞控制算法(**改进TCP的性能**)，这就是快重传和快恢复（TCP Reno版本）。
  - 有时，**个别报文段**会在网络中**丢失**，但实际上网络**并未发生拥塞**。
    - 这将导致**发送方**超时重传，并**误认为**网络发生了**拥塞**；
    - **发送方把拥塞窗口cwnd又设置为最小值1**，并错误地启动慢开始算法，因而**降低了传输效率**。



5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 慢开始和拥塞避免算法是1988年提出的TCP拥塞控制算法（TCP Tahoe版本）。
- 1990年又增加了两个新的拥塞控制算法(**改进TCP的性能**)，这就是快重传和快恢复（TCP Reno版本）。
  - 有时，**个别报文段**会在网络中**丢失**，但实际上网络**并未发生拥塞**。
    - 这将导致**发送方**超时重传，并**误认为**网络发生了**拥塞**；
    - **发送方把拥塞窗口cwnd又设置为最小值1**，并错误地启动慢开始算法，因而**降低了传输效率**。
- 采用快重传算法可以**让发送方尽早知道发生了个别报文段的丢失**。
- 所谓快重传，就是使发送方**尽快进行重传**，而**不是等超时重传计时器超时**再重传。
  - 要求接收方不要等待自己发送数据时才进行捎带确认，而是要**立即发送确认**；
  - 即使收到了失序的报文段也要立即发出对已收到的报文段的**重复确认**。
  - 发送方一旦**收到3个连续的重复确认**，就将相应的报文段**立即重传**，而不是等该报文段的超时重传计时器超时再重传。

5.5 TCP的拥塞控制

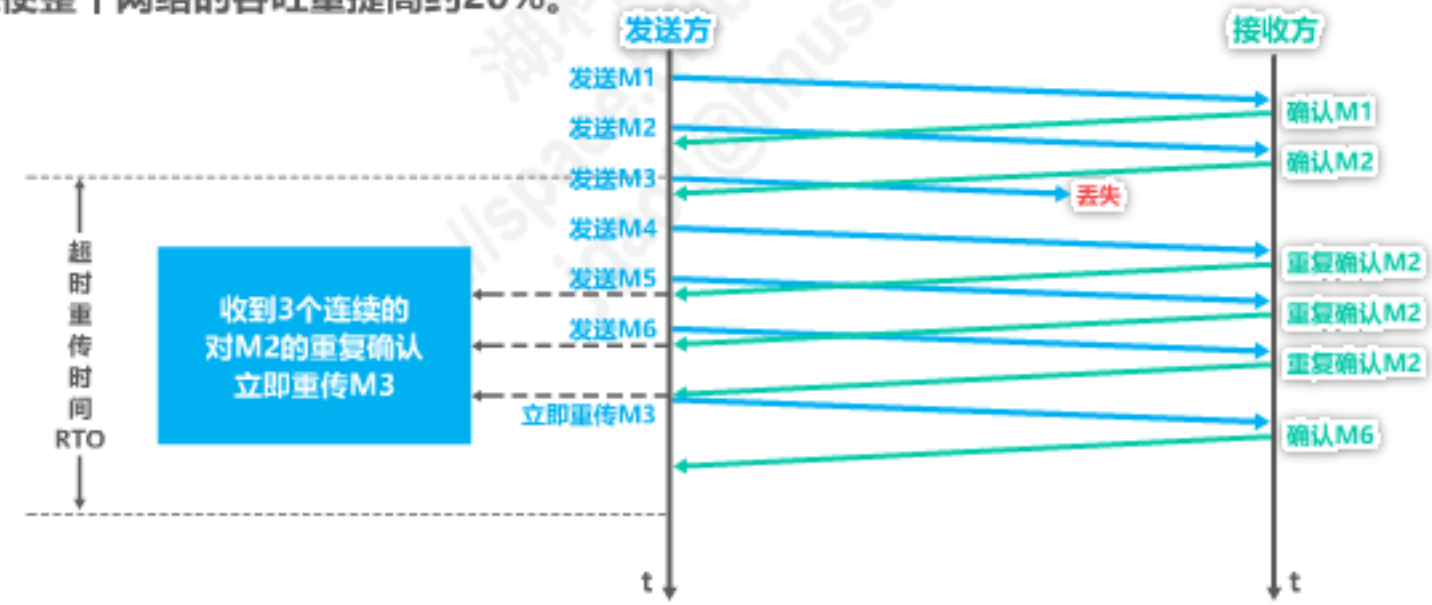
慢开始 (slow-start)	拥塞避免 (congestion avoidance)	快重传 (fast retransmit)	快恢复 (fast recovery)

- 所谓快重传，就是使发送方**尽快进行重传**，而**不是等超时重传计时器超时**再重传。
  - 要求接收方不要等待自己发送数据时才进行捎带确认，而是要**立即发送确认**；
  - 即使收到了失序的报文段也要立即发出对已收到的报文段的**重复确认**。
  - 发送方一旦**收到3个连续的重复确认**，就将相应的报文段**立即重传**，而不是等该报文段的超时重传计时器超时再重传。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | **快重传 (fast retransmit)** | 快恢复 (fast recovery)

- 所谓快重传，就是使发送方**尽快进行重传**，而**不是等超时重传计时器超时**再重传。
  - 要求接收方不要等待自己发送数据时才进行捎带确认，而是要**立即发送确认**；
  - 即使收到了失序的报文段也要立即发出对已收到的报文段的**重复确认**。
  - 发送方一旦**收到3个连续的重复确认**，就将相应的报文段**立即重传**，而不是等该报文段的超时重传计时器超时再重传。
  - 对于个别丢失的报文段，发送方不会出现超时重传，也就不会误认为出现了拥塞（进而降低拥塞窗口cwnd为1）。使用快重传可以使整个网络的吞吐量提高约20%。



5.5 TCP的拥塞控制

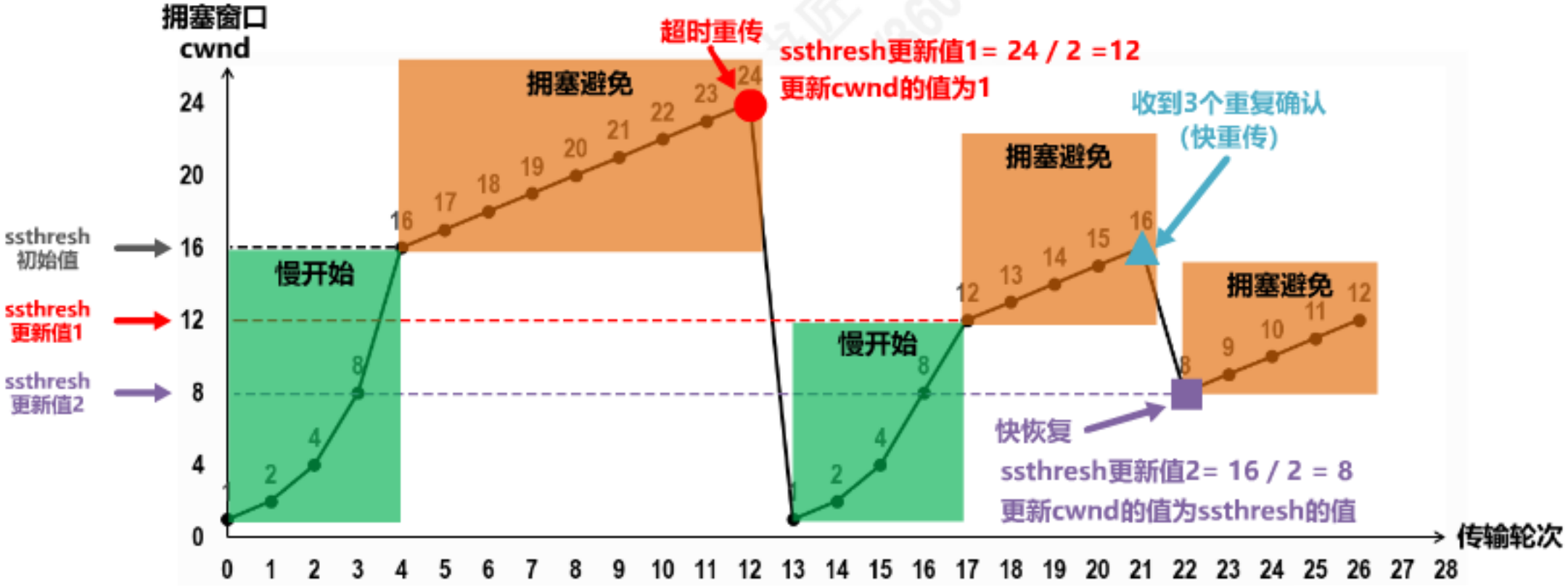
慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)

- 所谓快重传，就是使发送方**尽快进行重传**，而**不是等超时重传计时器超时**再重传。
  - 要求接收方不要等待自己发送数据时才进行捎带确认，而是要**立即发送确认**；
  - 即使收到了失序的报文段也要立即发出对已收到的报文段的**重复确认**。
  - 发送方一旦**收到3个连续的重复确认**，就将相应的报文段**立即重传**，而不是等该报文段的超时重传计时器超时再重传。
  - 对于个别丢失的报文段，发送方不会出现超时重传，也就不会误认为出现了拥塞（进而降低拥塞窗口cwnd为1）。使用快重传可以使整个网络的吞吐量提高约20%。

- 发送方一旦**收到3个重复确认**，就知道现在只是丢失了个别的报文段。于是不启动慢开始算法，而**执行快恢复算法**；
  - **发送方将慢开始门限ssthresh值和拥塞窗口cwnd值调整为当前窗口的一半；开始执行拥塞避免算法。**
  - 也有的快恢复实现是把快恢复开始时的拥塞窗口cwnd值再增大一些，即等于新的ssthresh + 3。
    - 既然发送方收到3个重复的确认，就表明有3个数据报文段已经离开了网络；
    - 这3个报文段不再消耗网络资源而是停留在接收方的接收缓存中；
    - 可见现在网络中不是堆积了报文段而是减少了3个报文段。因此可以适当把拥塞窗口扩大些。

5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)



5.5 TCP的拥塞控制

【2009年 题39】一个TCP连接总是以1KB的最大段长发送TCP段，发送方有足够多的数据要发送。当拥塞窗口为16KB时发生了超时，如果接下来的4个RTT（往返时间）内的TCP段的传输都是成功的，那么当第4个RTT时间内发送的所有TCP段都得到肯定应答时，拥塞窗口大小是 C

A. 7KB　　B. 8KB　　C. 9KB　　D. 16KB

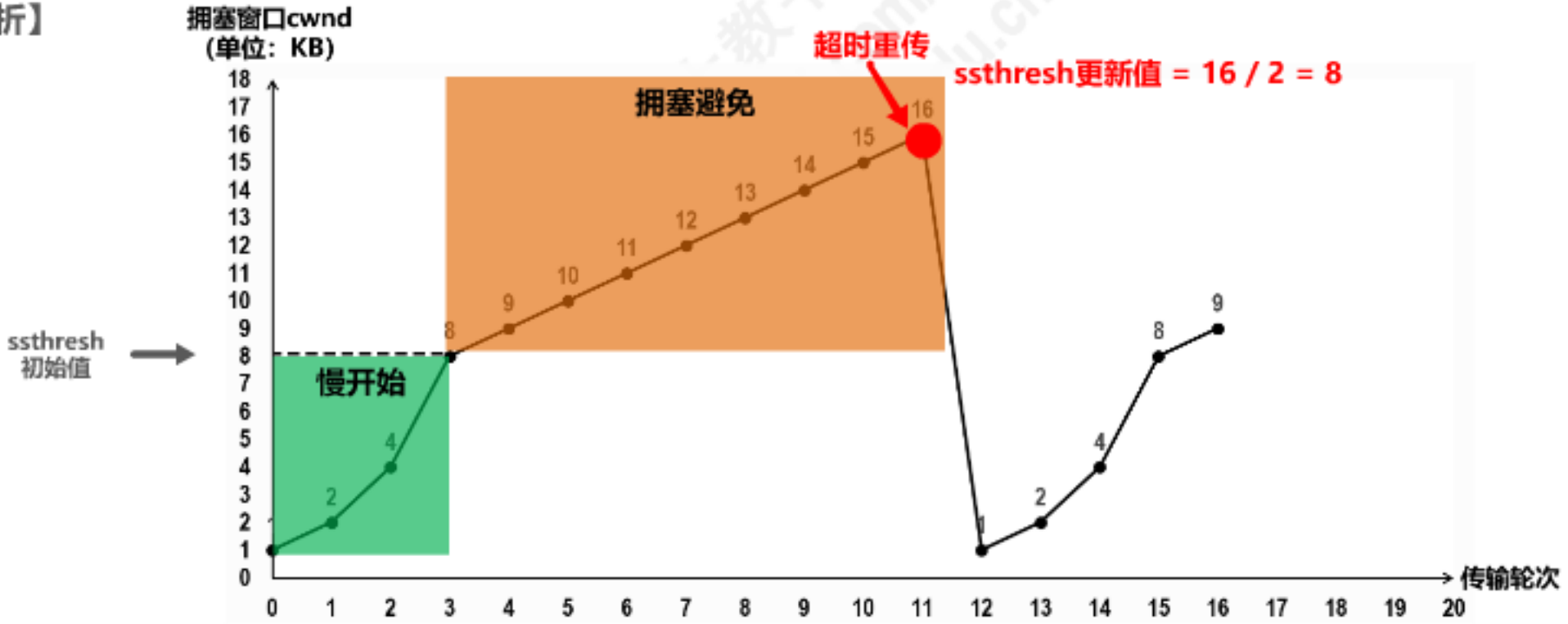
【解析】

5.5 TCP的拥塞控制

【2009年 题39】一个TCP连接总是以1KB的最大段长发送TCP段，发送方有足够多的数据要发送。当拥塞窗口为16KB时发生了超时，如果接下来的4个RTT（往返时间）内的TCP段的传输都是成功的，那么当第4个RTT时间内发送的所有TCP段都得到肯定应答时，拥塞窗口大小是 C

A. 7KB　　B. 8KB　　C. 9KB　　D. 16KB

【解析】

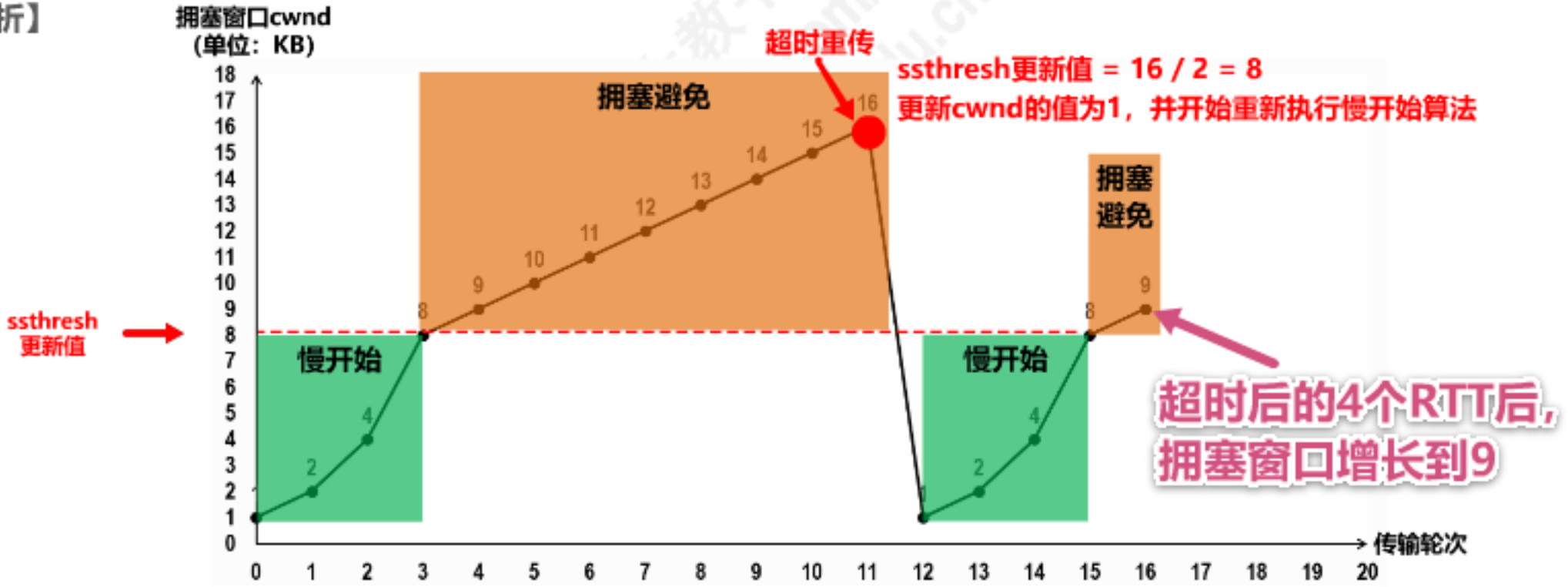


5.5 TCP的拥塞控制

【2009年 题39】一个TCP连接总是以1KB的最大段长发送TCP段，发送方有足够多的数据要发送。当拥塞窗口为16KB时发生了超时，如果接下来的4个RTT（往返时间）内的TCP段的传输都是成功的，那么当第4个RTT时间内发送的所有TCP段都得到肯定应答时，拥塞窗口大小是 C

A. 7KB　　B. 8KB　　C. 9KB　　D. 16KB

【解析】

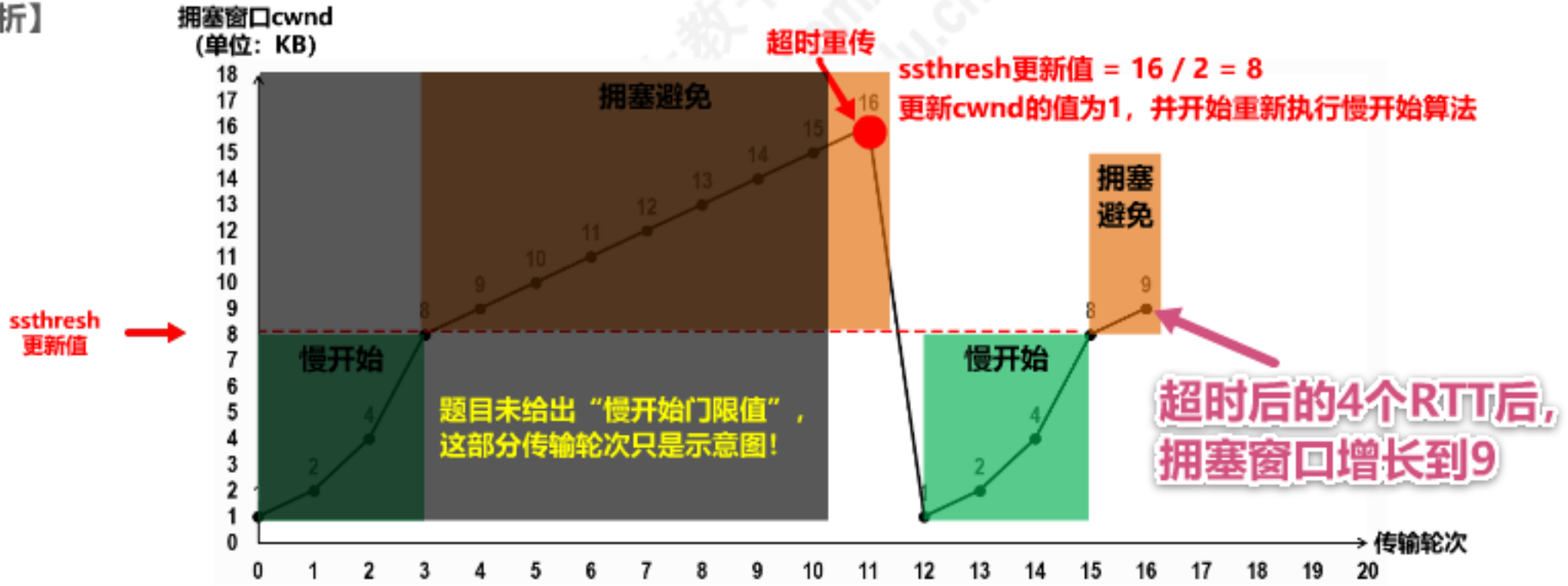


5.5 TCP的拥塞控制

【2009年 题39】一个TCP连接总是以1KB的最大段长发送TCP段，发送方有足够多的数据要发送。当拥塞窗口为16KB时发生了超时，如果接下来的4个RTT（往返时间）内的TCP段的传输都是成功的，那么当第4个RTT时间内发送的所有TCP段都得到肯定应答时，拥塞窗口大小是 C

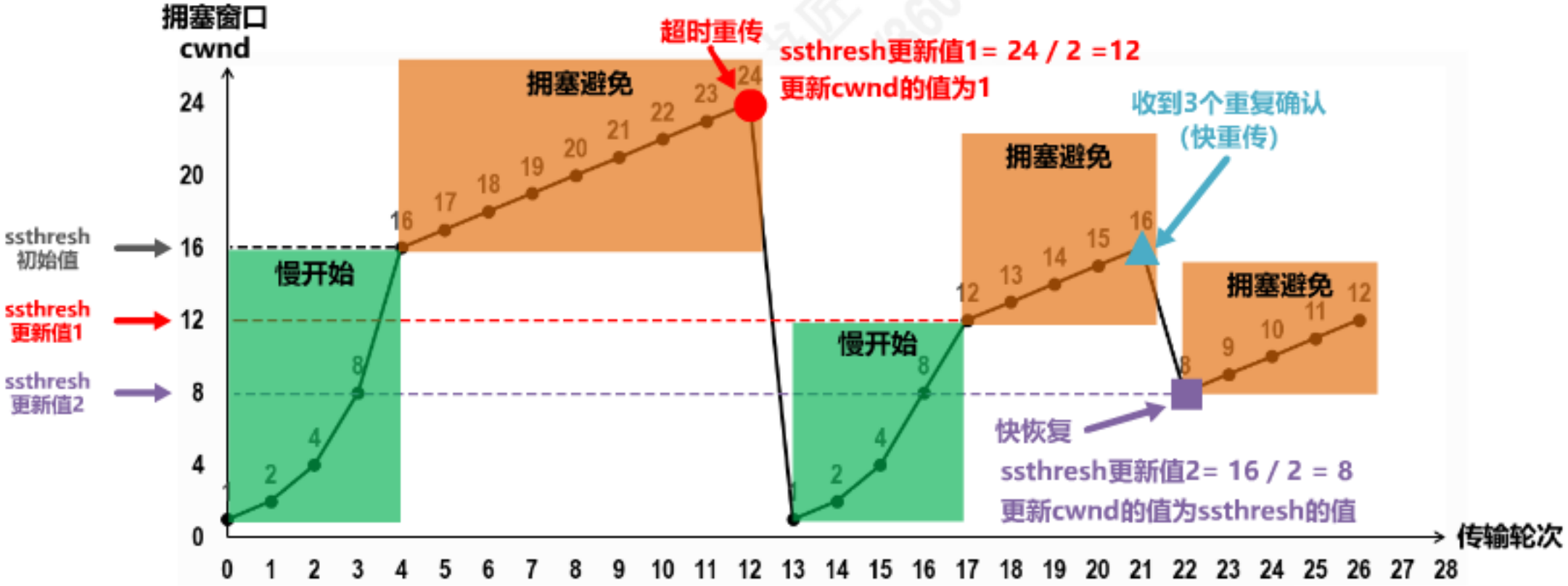
A. 7KB　　B. 8KB　　C. 9KB　　D. 16KB

【解析】

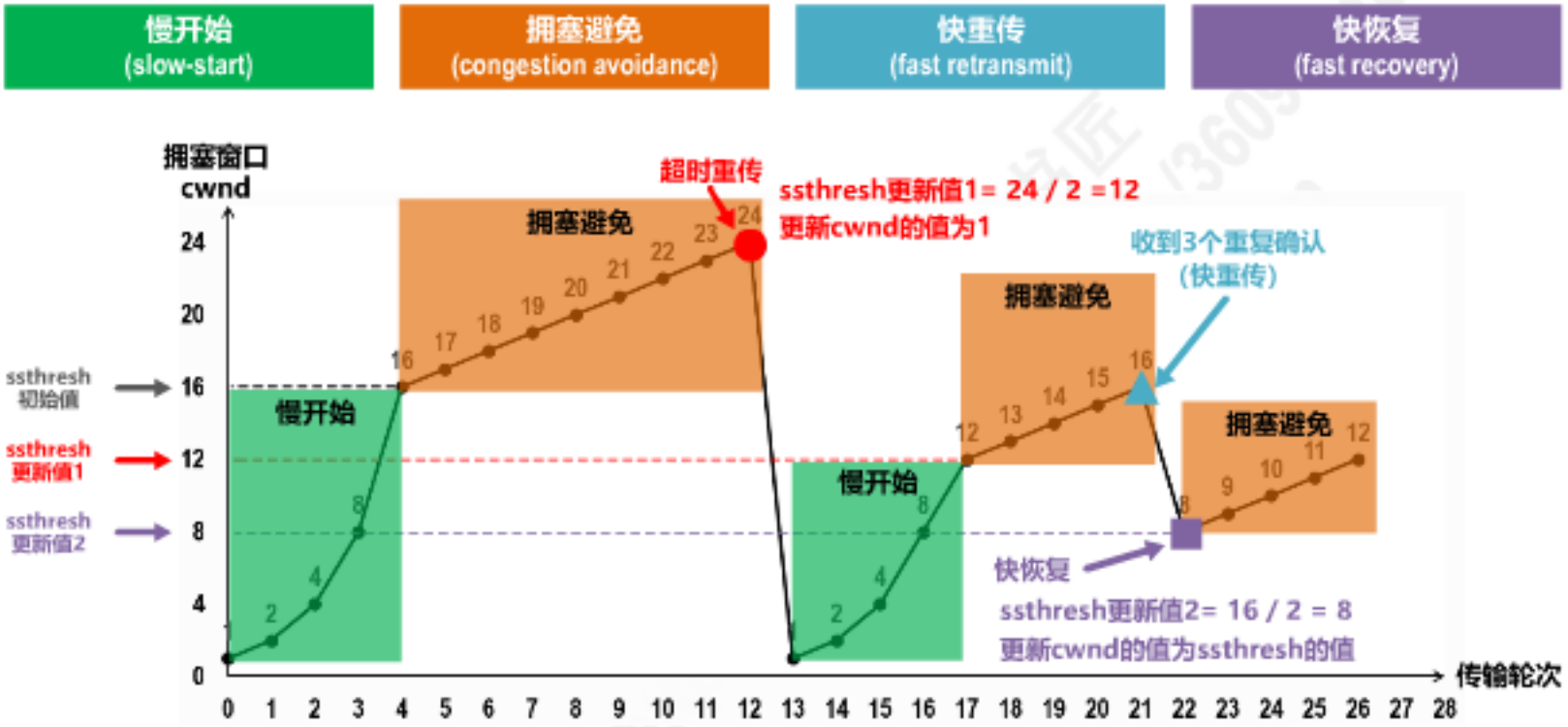


5.5 TCP的拥塞控制

慢开始 (slow-start) | 拥塞避免 (congestion avoidance) | 快重传 (fast retransmit) | 快恢复 (fast recovery)



5.5 TCP的拥塞控制


慢开始 (slow-start)
拥塞避免 (congestion avoidance)
快重传 (fast retransmit)
快恢复 (fast recovery)
拥塞窗口 cwnd
超时重传
ssthresh更新值1= 24 / 2 =12
更新cwnd的值为1
拥塞避免
慢开始
收到3个重复确认 (快重传)
拥塞避免
慢开始
拥塞避免
ssthresh 初始值
ssthresh 更新值1
ssthresh 更新值2
快恢复
ssthresh更新值2= 16 / 2 = 8
更新cwnd的值为ssthresh的值
传输轮次