


# TCP/IP之拥塞控制



六尺帐篷 (/u/f8e9b1c246f1)

2017.05.31 20:26\* 字数 1910 阅读 345 评论 0 喜欢 15

(/u/f8e9b1c246f1)

[编辑文章 \(/writer#/notebooks/12835308/notes/12982341\)](#)

拥塞(Congestion)

给一个非正式定义就是：“太多发送主机发送了太多数据或者发送速度太快，以至于网络无法处理”

如果网络中发生了拥塞，会出现如下表现：

- 分组丢失（路由器缓存溢出）
- 分组延迟过大（在路由器缓存中排队）

和可靠数据传输一样都是网络领域中的top-10的问题。

拥塞现象是指到达[通信子网]中某一部分的分组数量过多，使得该部分网络来不及处理，以致引起这部分乃至整个网络性能下降的现象，严重时甚至会导致网络通信业务陷入停顿，即出现[死锁]现象。这种现象跟公路网中经常所见的交通拥挤一样，当节假日公路网中车辆大量增加时，各种走向的车流相互干扰，使每辆车到达目的地的时间都相对增加（即延迟增加），甚至有时在某段公路上车辆因堵塞而无法开动(即发生局部[死锁])

我们先讨论一下拥塞的成因和代价

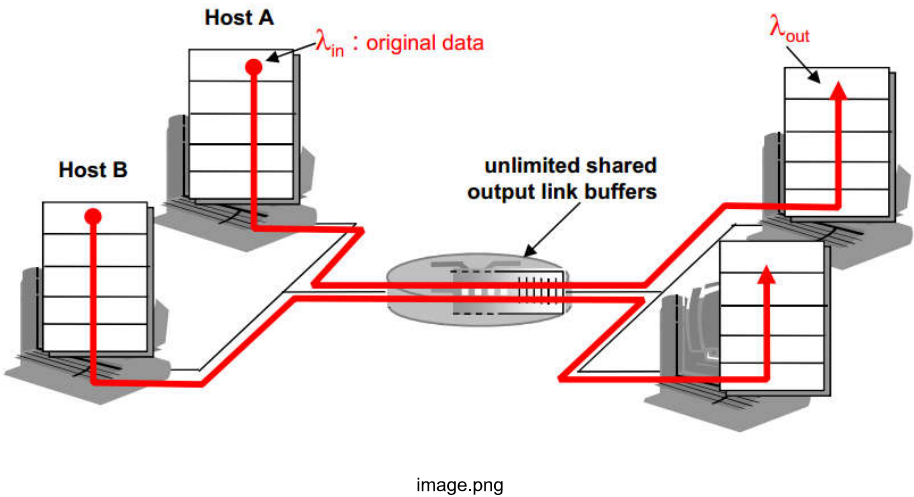
## 拥塞的成因和代价

我们通过假设不同的场景渐进式分析拥塞的成因和代价

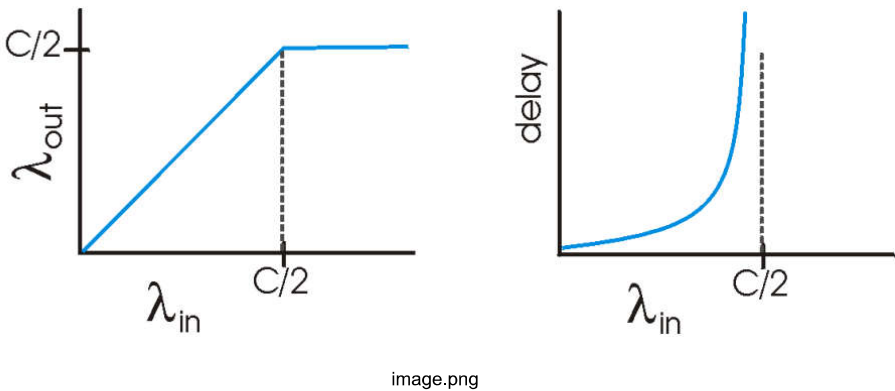
### 场景一

我们假设

- 两个senders,两个receivers
- 一个路由器, 无限缓存
- 没有重传



我们假设原始数据的发送速度为 $\lambda_{in}$ ,到达接受方的速度为 $\lambda_{out}$ ，由于路由器的速度限制，即使无限缓存，到达的最大速度也无法超过路由器的速度



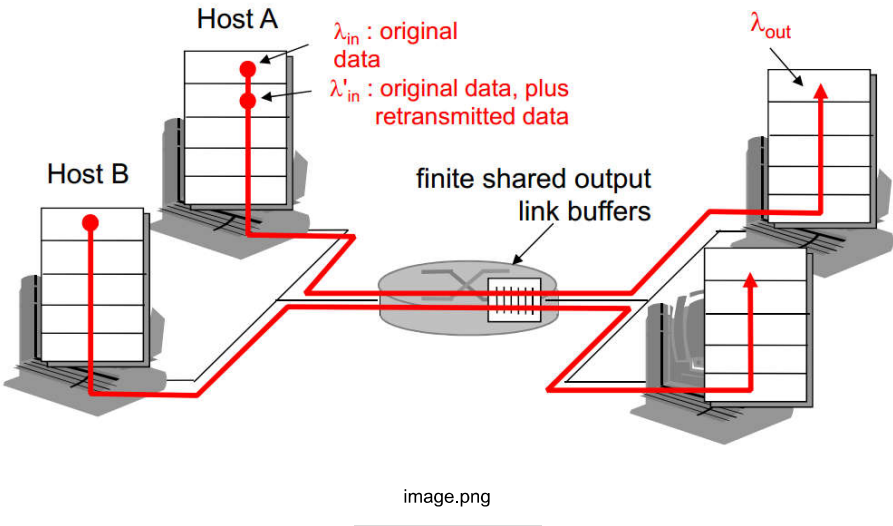
即使发送速度再快，到达速度也无法超过路由器的速度，所以时延在拥塞发生的时候会无限增大。

- 拥塞时分组延迟太大
- 达到最大throughput

场景2

我们假设

- 一个路由器, 有限buffers
- Sender重传分组



可以分为以下几种情况讨论

- 情况a: Sender能够通过某种机制获知路由器buffer信息，有空闲才发

📄

🔖

🔗

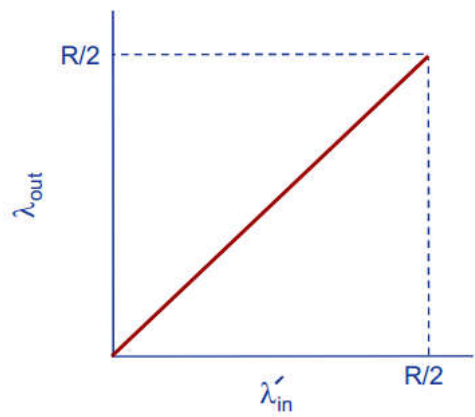


image.png

- 情况b: 丢失后才重发，显然这样到达速度会减小

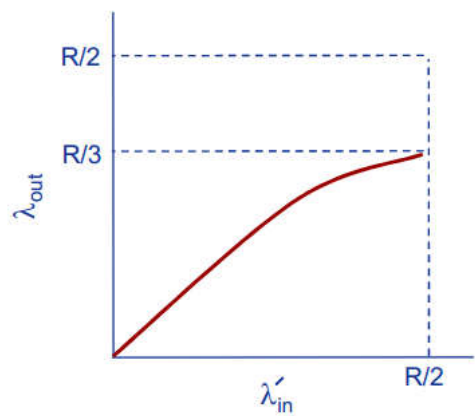


image.png

- 情况c: 分组丢失和定时器超时后都重发，显然到达速度进一步减小

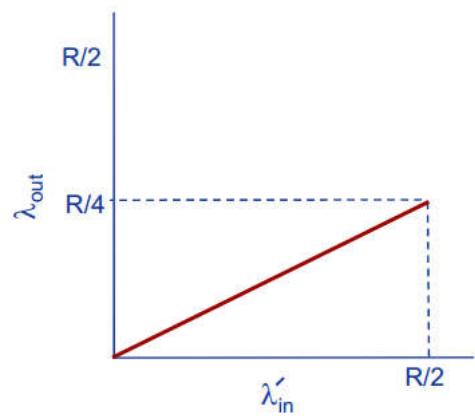


image.png

拥塞的代价:

- 对给定的“goodput”，要做更多的工作 (重传)
- 造成资源的浪费

场景三

- 四个发送方
- 多跳

📄

🔖

🔗

- 超时/重传

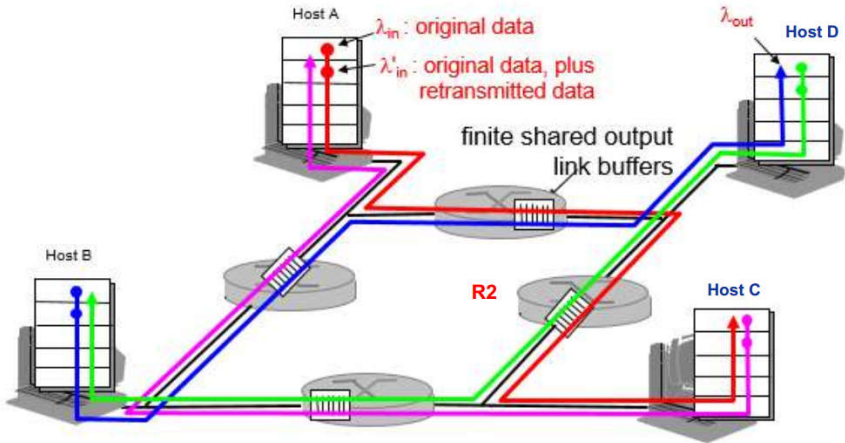


image.png

随着拥塞的严重，整个网络可能陷入瘫痪，到达速度趋近于0，类似于死锁的状态

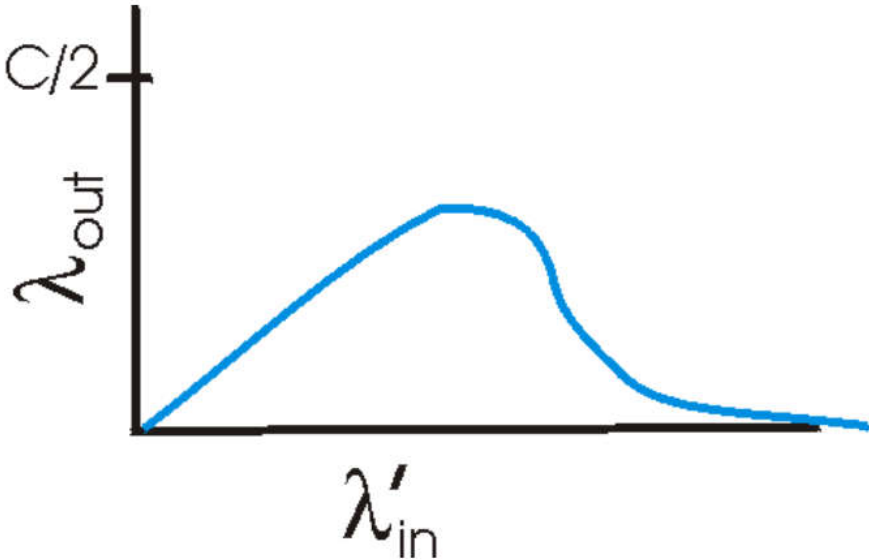


image.png

拥塞的另一个代价:

- 当分组被drop时，任何用于该分组的"上游"传输能力全都被浪费掉，相当于白传了，浪费了资源和传输能力

拥塞控制的方法

端到端拥塞控制:

- 网络层不需要显式的提供支持
- 端系统通过观察loss, delay等网络行为判断是否发生拥塞
- TCP采取这种方法
  - 网络辅助的拥塞控制: 路由器向发送方显式地反馈网络拥塞信息
  - 简单的拥塞指示(1bit): SNA, DECbit, TCP/IP ECN, ATM) 指示发送方应该采取何种速度

🏠

🔖

🔗

## 案例：ATM ABR拥塞控制

- ABR: available bit rate
  - “弹性服务”
  - 如果发送方路径“underloaded”
  - 使用可用带宽
  - 如果发送方路径拥塞
  - 将发送速率降到最低保障速率
- RM(resource management)cells
  - 发送方发送
  - 交换机设置RM cell位(网络辅助)
- NI bit: rate不许增长
- CI bit: 拥塞指示
  - RM cell由接收方返回给发送方

## TCP拥塞控制

### TCP拥塞控制的基本原理

Sender限制发送速率

$$\text{LastByteSent} - \text{LastByteAcked} \leq \text{CongWin}$$

$$\text{rate} \approx \frac{\text{CongWin}}{\text{RTT}} \text{ Bytes/sec}$$

image.png

CongWin可以动态调整以改变发送速率，并且反映所感知到的网络拥塞。

那么问题来了，如何感知网络拥塞：

- Loss事件=timeout或3个重复ACK
- 发生loss事件后，发送方降低速率

感知到网络拥塞后，需要动态调整发送速率，以减轻网络的拥塞状况，如何调整发送速率，一般有两个方法：

- 加性增—乘性减: AIMD
- 慢启动: SS

### 加性增—乘性减: AIMD

顾名思义，这种方法就是先简单的增加，遇到拥塞的情况，就乘性减少。

原理：逐渐增加发送速率，谨慎探测可用带宽，直到发生loss。

Additive Increase: 每个RTT将CongWin增大一个MSS——拥塞避免

Multiplicative Decrease: 发生loss后将CongWin减半。

AIMD方法会使congwin呈锯齿状的波动



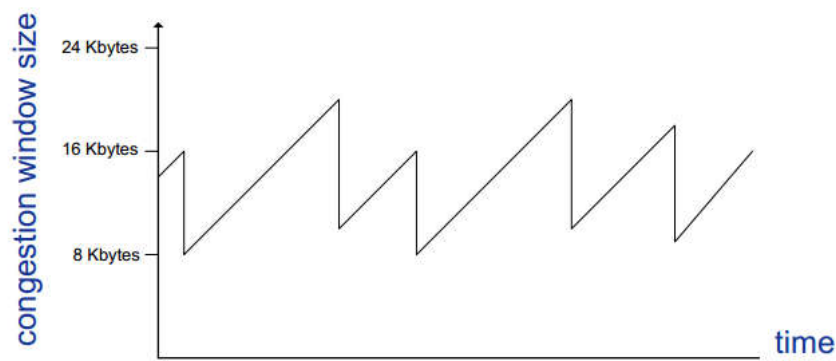


image.png

首先慢慢增加，当遇到拥塞时，减为一半，然后又继续慢慢增加，直到遇到拥塞后又减为一半，这样往复就会出现锯齿状的波动。

TCP慢启动: SS

我们考虑下面这种情况：

TCP连接建立时，

CongWin=1

□ 例：MSS=500 byte,

RTT=200msec

□ 初始速率=20k bps

我们发现在这种情况下，可用带宽可能远远高于初始速率，如果我们采用加性增的方法就太慢了，我们希望快速增长到可用带宽。

这就慢启动算法的思想：

当连接开始时，指数性增长。指数性增长。每个RTT将CongWin翻倍。收到每个ACK进行操作。初始速率很慢，但是快速攀升。

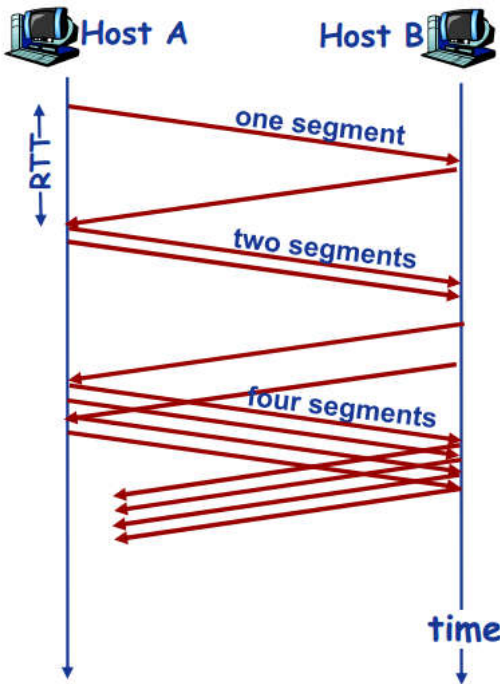


image.png

那么问题来了，我们什么时候才进行线性增长来避免拥塞控制？  
我们通过设置一个变量 Threshold，Loss事件发生时，Threshold被设为Loss事件前 CongWin值的1/2。然后如果cogwin到了Threshold，就开始线性增长。

📄

🔖

🔗

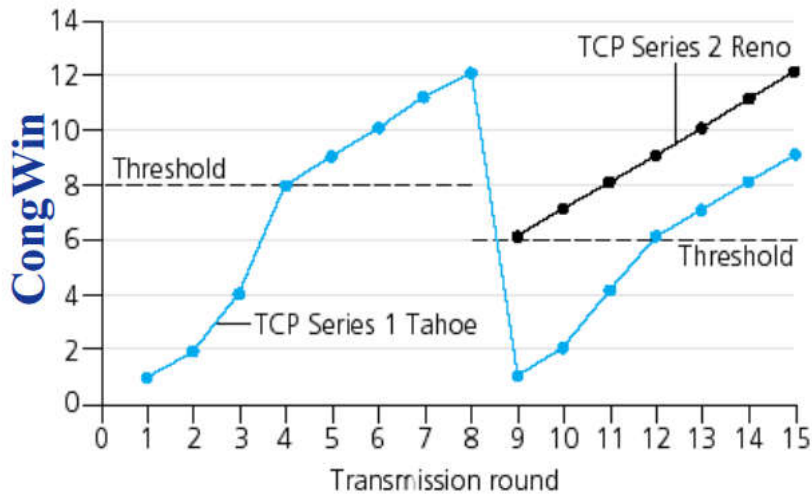


image.png

如图所示，初始的Threshold变量为8，所以当指数增长大于等于8的时候，就开始线性增长，然后直到发生loss事件，Threshold变为发生loss事件时的一半，也就是6，然后继续指数增长到Threshold，又开始线性增长。

那么我们如何判断loss事件的发生呢？

我们分为两种情况来处理：

- 3个重复ACKs:  
CongWin切到一半，然后线性增长
- Timeout事件:  
CongWin直接设为1个MSS，然后指数增长，达到threshold后,再线性增长

我们想想这样做的原因，因为3个重复ACKs表示网络还能够传输一些 segments，timeout事件表明拥塞更为严重。

慢启动算法：

```

Th = ?
CongWin = 1 MSS
/* slow start or exponential increase */
While (No Packet Loss and CongWin < Th) {
  send CongWin TCP segments
  for each ACK increase CongWin by 1
}
/* congestion avoidance or linear increase */
While (No Packet Loss) {
  send CongWin TCP segments
  for CongWin ACKs, increase CongWin by 1
}
Th = CongWin/2
If (3 Dup ACKs) CongWin = Th;
If (timeout) CongWin=1;

```

□ 一个TCP连接总是以1 KB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16 KB时发生了超时，如果接下来的4个RTT（往返时间）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，拥塞窗口大小是多少？

□ 解：threshold=16/2=8 KB, CongWin=1 KB, 1个RTT后， CongWin=2 KB，2个RTT后， CongWin=4 KB，3个RTT后， CongWin=8 KB，Slowstart is over; 4个RTT后， CongWin=9 KB



六尺帐篷 (/u/f8e9b1c246f1)

写了 245418 字，被 15240 人关注，获得了 1429 个喜欢

(/u/f8e9b1c246f1)

喜欢15







更多分享

(http://cwb.assets.jianshu.io/notes/images/1298234

被以下专题收入，发现更多相似内容

投稿管理

- + 收入我的专题
- 

Android知识 (/c/3fde3b545a35?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Android开发 (/c/d1591c322c89?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

Android... (/c/58b4c20abf2f?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

计算机网络 (/c/89e56c7637e1?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

iOS Dev... (/c/3233d1a249ca?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

首页投稿 (/c/bDHhpK?utm\_source=desktop&utm\_medium=notes-included-collection)
- 

程序员 (/c/NEt52a?utm\_source=desktop&utm\_medium=notes-included-collection)

展开更多





