石家庄铁道大学
SHIJIAZHUANG TIEDAO UNIVERSITY

计算机网络

第 17 讲　运输层（III）

上讲内容回顾

- 可靠传输的工作原理
  停止等待协议
  连续ARQ协议
- TCP报文段的首部格式

本讲内容

- TCP可靠传输的实现

  滑动窗口，超时重传时间的选择，选择确认

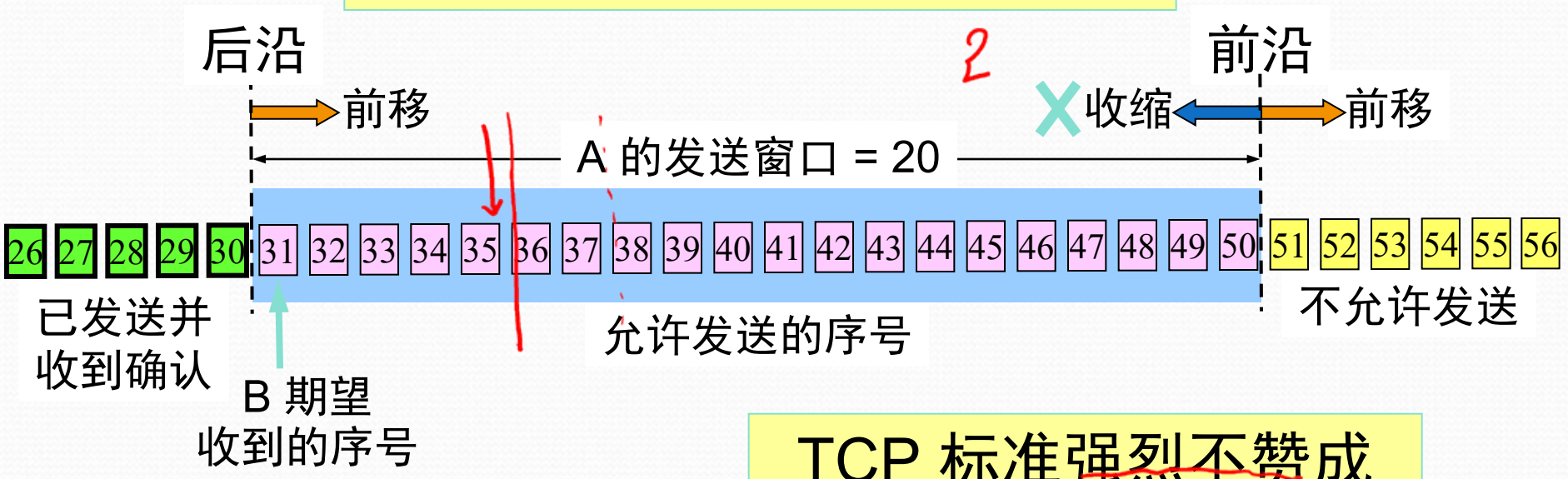
- TCP的流量控制

  利用滑动窗口实现流量控制，传输效率

TCP 可靠传输的实现

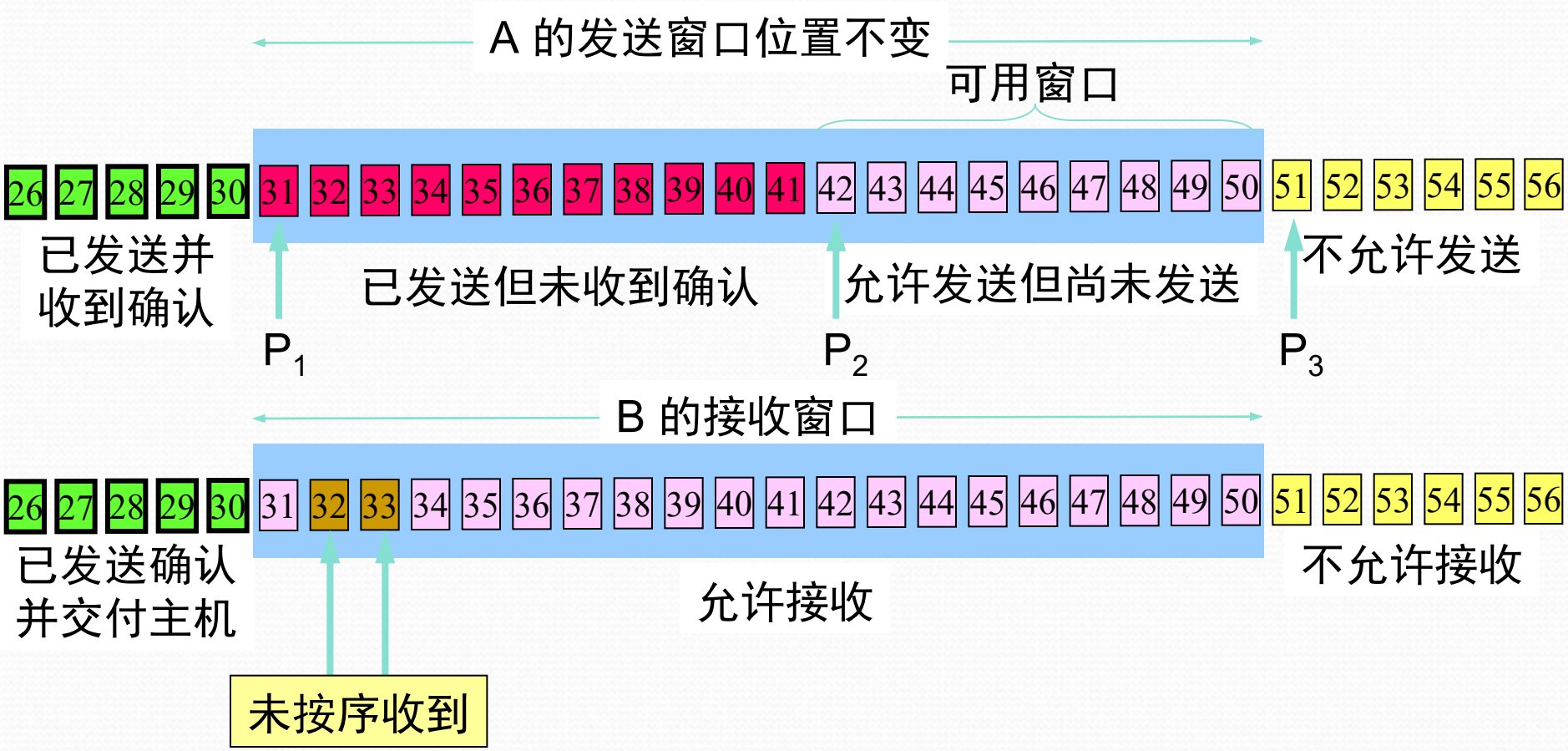
以字节为单位的滑动窗口

根据 B 给出的窗口值
A 构造出自己的发送窗口



TCP 标准强烈不赞成
发送窗口前沿向后收缩

A 发送了 11 个字节的数据

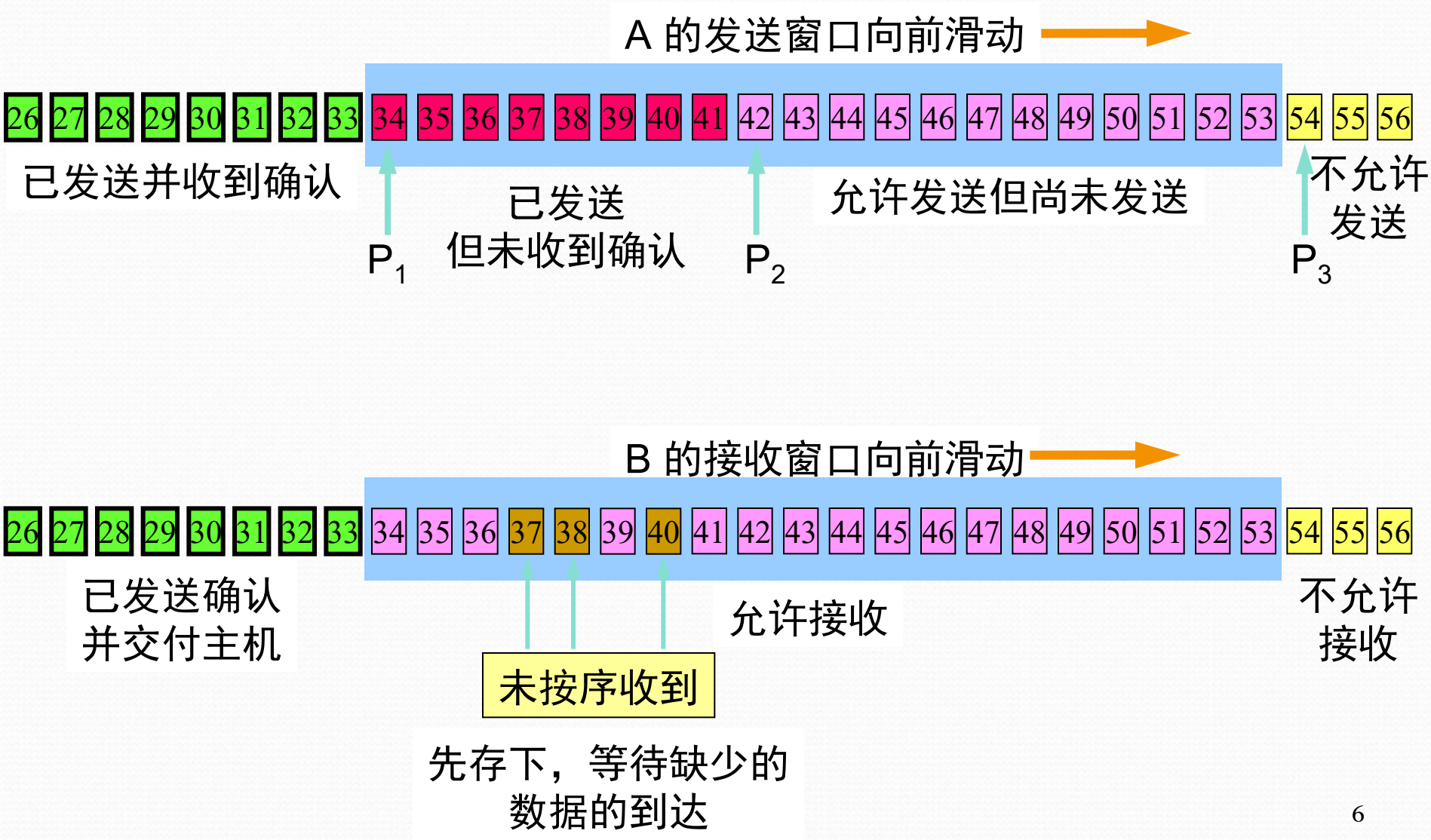


$P_3 - P_1$ = A 的发送窗口（又称为通知窗口）

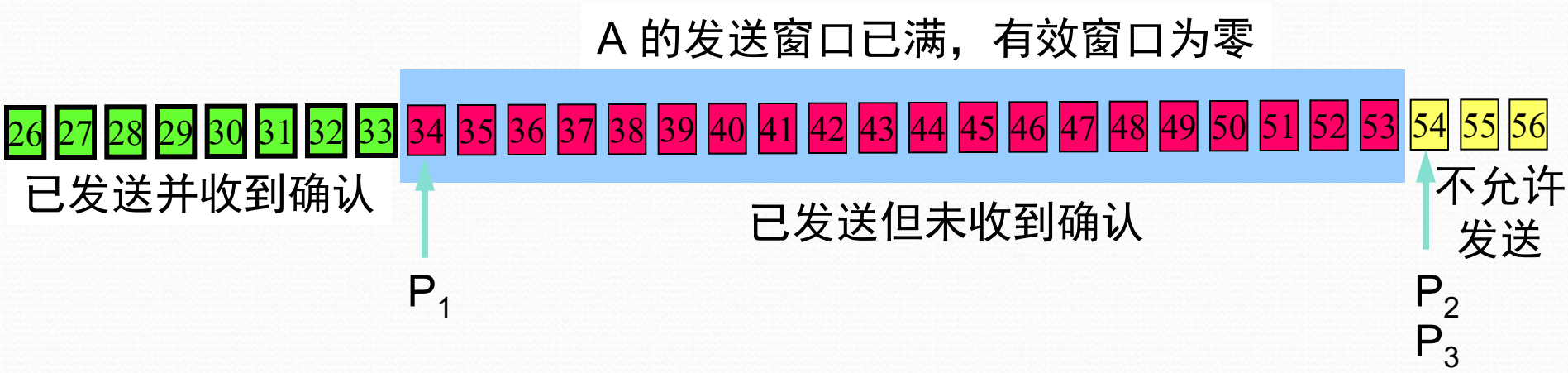
$P_2 - P_1$ = 已发送但尚未收到确认的字节数

$P_3 - P_2$ = 允许发送但尚未发送的字节数（又称为可用窗口）

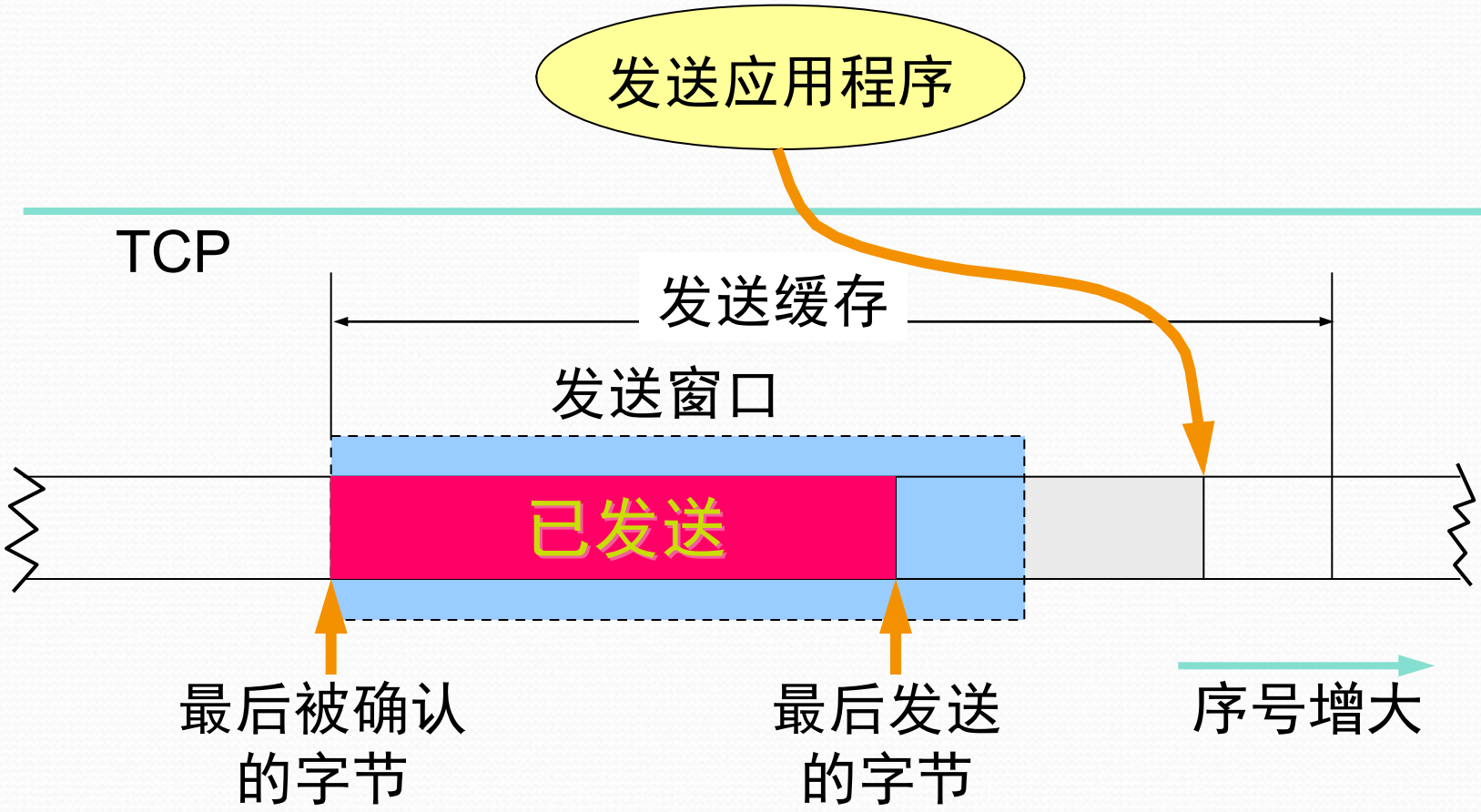
A 收到新的确认号，发送窗口向前滑动



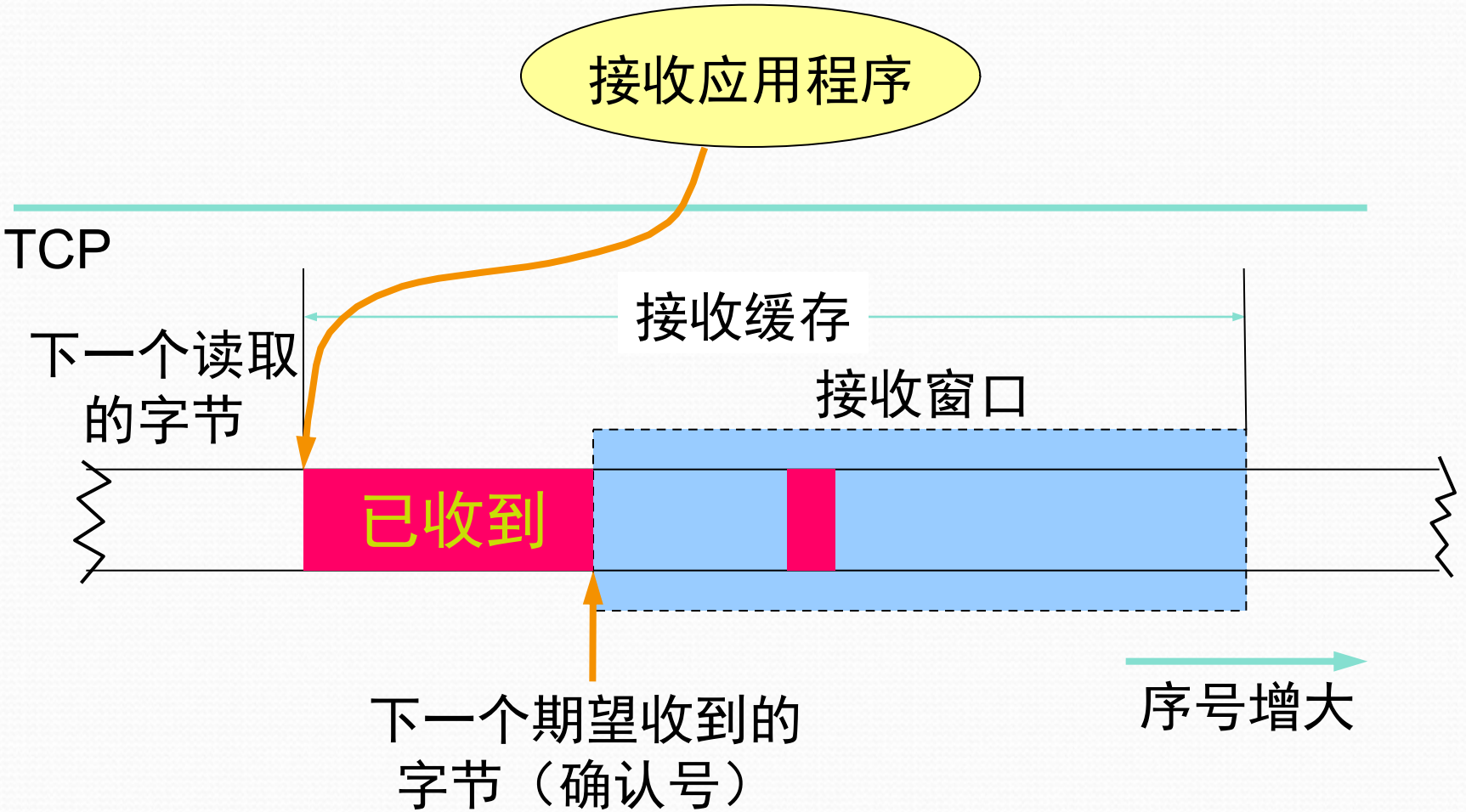
A 的发送窗口内的序号都已用完，但还没有再收到确认，必须停止发送。



发送缓存



接收缓存



发送缓存与接收缓存的作用

- 发送缓存用来暂时存放：
  - 发送应用程序传送给发送方 TCP 准备发送的数据；
  - TCP 已发送出但尚未收到确认的数据。
- 接收缓存用来暂时存放：
  - 按序到达的、但尚未被接收应用程序读取的数据；
  - 不按序到达的数据。

需要强调三点

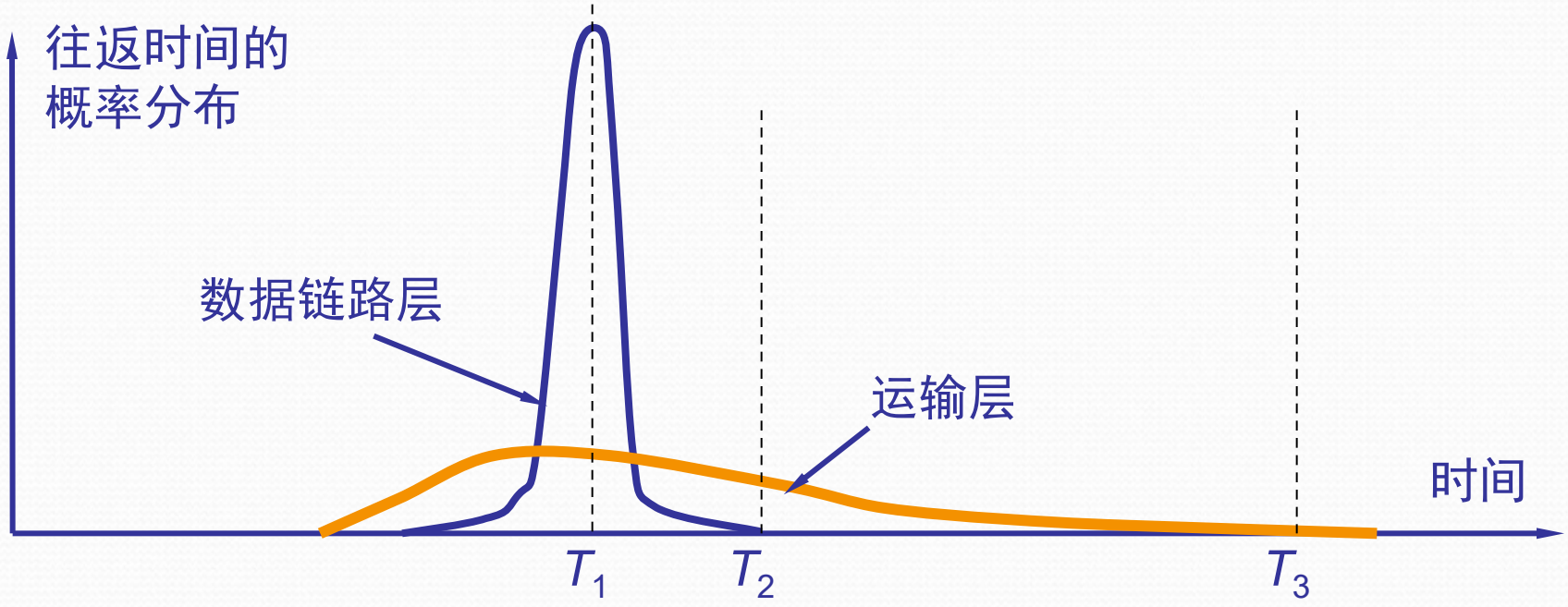
- A 的发送窗口并不总是和 B 的接收窗口一样大（因为有一定的时间滞后）。
- TCP 标准没有规定对不按序到达的数据应如何处理。通常是先临时存放在接收窗口中，等到字节流中所缺少的字节收到后，再按序交付上层的应用进程。
- TCP 要求接收方必须有累积确认的功能，这样可以减小传输开销。

超时重传时间的选择

- 重传机制是 TCP 中最重要和最复杂的问题之一。
- TCP 每发送一个报文段，就对这个报文段设置一次计时器。只要计时器设置的重传时间到但还没有收到确认，就要重传这一报文段。

往返时延的方差很大

- 由于 TCP 的下层是一个互联网环境，IP 数据报所选择的路由变化很大。因而运输层的往返时间的方差也很大。



加权平均往返时间

- TCP 保留了 RTT 的一个加权平均往返时间 RTT_S（这又称为平滑的往返时间）。
- 第一次测量到 RTT 样本时，RTT_S 值就取为所测量到的 RTT 样本值。以后每测量到一个新的 RTT 样本，就按下式重新计算一次 RTT_S：

$$\text{新的 } RTT_S = (1-\alpha) \times (\text{旧的 } RTT_S) + \alpha \times (\text{新的 RTT 样本}) \qquad (5\text{-}4)$$

- 式中，$0 \le \alpha < 1$。若 α 很接近于零，表示 RTT 值更新较慢。若选择 α 接近于 1，则表示 RTT 值更新较快。
- RFC 2988 推荐的 α 值为 1/8，即 0.125。

超时重传时间 RTO (RetransmissionTime-Out)

- RTO 应略大于上面得出的加权平均往返时间 RTT_S。
- RFC 2988 建议使用下式计算 RTO：
- $$RTO = RTT_S + 4 \times RTT_D \qquad (5\text{-}5)$$
- RTT_D 是 RTT 的偏差的加权平均值。
- RFC 2988 建议这样计算 RTT_D。第一次测量时，RTT_D 值取为测量到的 RTT 样本值的一半。在以后的测量中，则使用下式计算加权平均的 RTT_D：

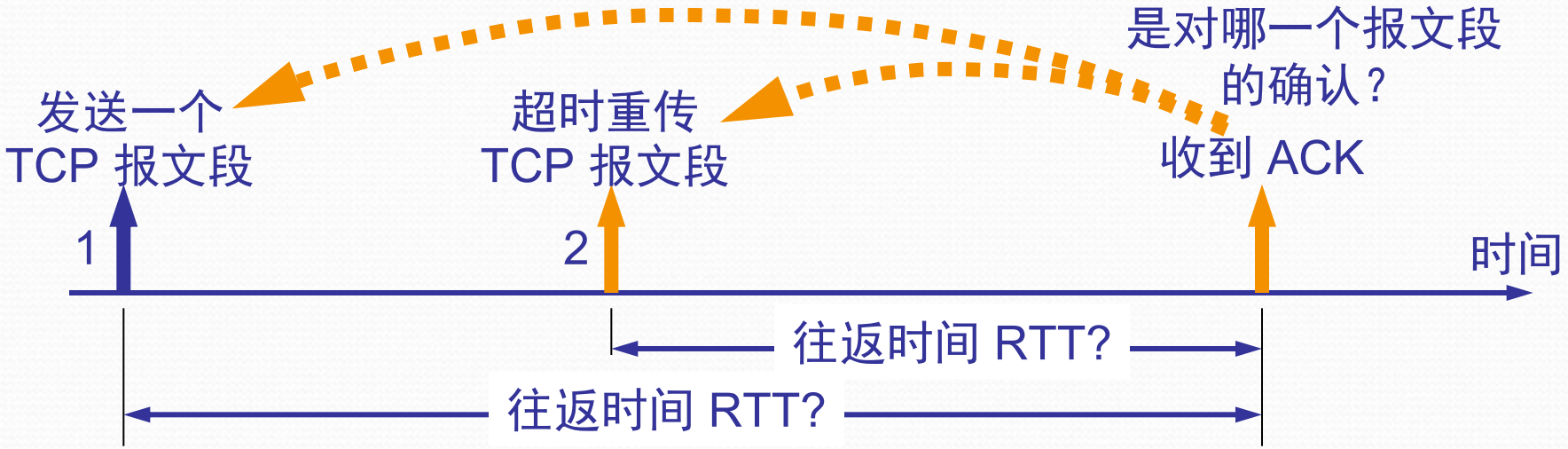
新的 $RTT_D = (1 - \beta) \times$ (旧的RTT_D)

$$+ \beta \times |RTT_S - \text{新的 RTT 样本}| \qquad (5\text{-}6)$$

- β 是个小于 1 的系数，其推荐值是 1/4，即 0.25。

往返时间的测量相当复杂

- TCP 报文段 1 没有收到确认。重传（即报文段 2）后，收到了确认报文段 ACK。
- 如何判定此确认报文段是对原来的报文段 1 的确认，还是对重传的报文段 2 的确认？



Karn 算法

- 在计算加权平均时延RTT_S时，只要报文段重传了，就不采用其往返时间样本。
- 这样得出的加权平均往返时间 RTT_S 和超时重传时间 RTO 就较准确。

修正的 Karn 算法

- 报文段每重传一次，就把 RTO 增大一些：

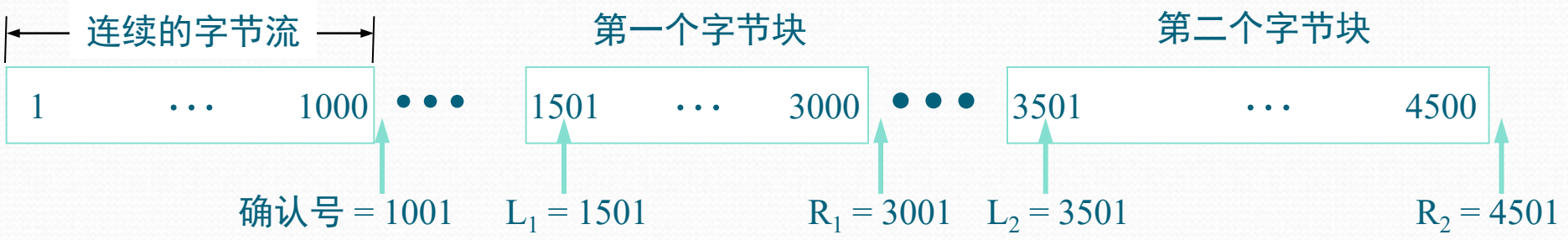
$$\text{新的 RTO} = \gamma \times (\text{旧的 RTO})$$

- 系数 γ 的典型值是 2 。
- 当不再发生报文段的重传时，才根据报文段的往返时延更新平均往返时延 RTT 和超时重传时间 RTO 的数值。
- 实践证明，这种策略较为合理。

选择确认 SACK (Selective ACK)

- 接收方收到了和前面的字节流不连续的两个字节块。
- 如果这些字节的序号都在接收窗口之内，那么接收方就先收下这些数据，但要把这些信息准确地告诉发送方，使发送方不要再重复发送这些已收到的数据。

接收到的字节流序号不连续



- 和前后字节不连续的每一个字节块都有两个边界：左边界和右边界。图中用四个指针标记这些边界。
- 第一个字节块的左边界 L_1 = 1501，但右边界 R_1 = 3001。
- 左边界指出字节块的第一个字节的序号，但右边界减 1 才是字节块中的最后一个序号。
- 第二个字节块的左边界 L_2 = 3501，而右边界 R_2 = 4501。

RFC 2018 的规定

- 如果要使用选择确认，那么在建立 TCP 连接时，就要在 TCP 首部的选项中加上“允许 SACK”的选项，而双方必须都事先商定好。
- 如果使用选择确认，那么原来首部中的“确认号字段”的用法仍然不变。只是以后在 TCP 报文段的首部中都增加了 SACK 选项，以便报告收到的不连续的字节块的边界。
- 由于首部选项的长度最多只有 40 字节，而指明一个边界就要用掉 4 字节，因此在选项中最多只能指明 4 个字节块的边界信息。

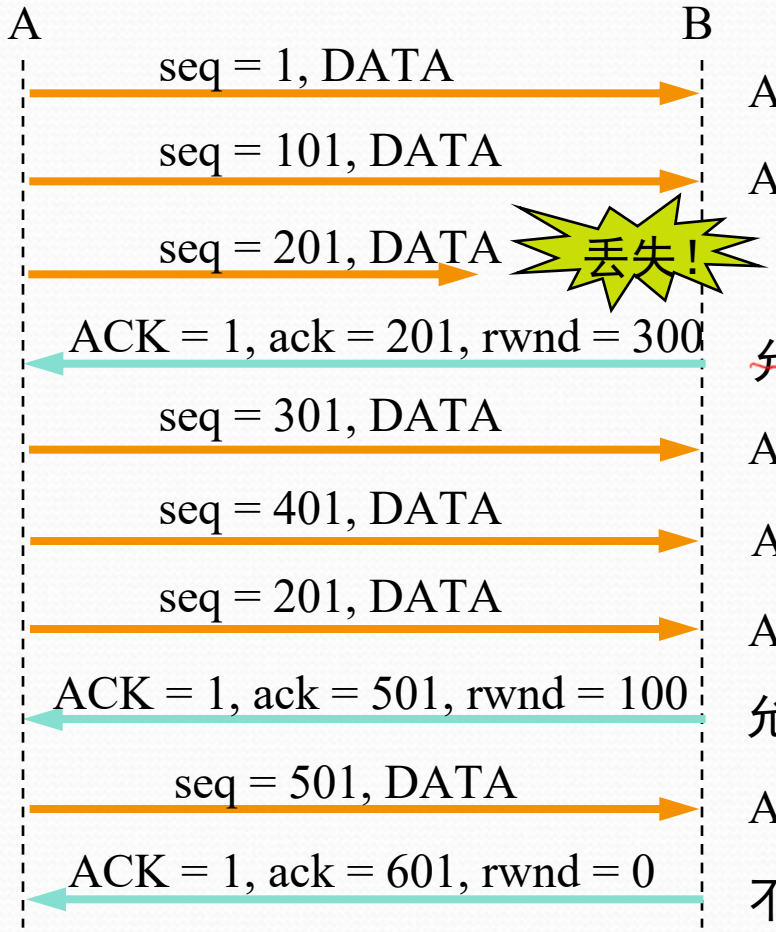
TCP 的流量控制

利用滑动窗口实现流量控制

- 一般说来，我们总是希望数据传输得更快一些。但如果发送方把数据发送得过快，接收方就可能来不及接收，这就会造成数据的丢失。
- 流量控制(flow control)就是让发送方的发送速率不要太快，要让接收方来得及接收。
- 利用滑动窗口机制可以很方便地在 TCP 连接上实现流量控制。

流量控制举例

A 向 B 发送数据。在连接建立时，
B 告诉 A：“我的接收窗口 rwnd = 400（字节）”。



A 发送了序号 1 至 100，还能发送 300 字节

A 发送了序号 101 至 200，还能发送 200 字节

允许 A 发送序号 201 至 500，共 300 字节

A 发送了序号 301 至 400，还能再发送 100 字节新数据

A 发送了序号 401 至 500，不能再发送新数据了

A 超时重传旧的数据，但不能发送新的数据

允许 A 发送序号 501 至 600 共 100 字节

A 发送了序号 501 至 600，不能再发送了

不允许 A 再发送（到序号 600 为止的数据都收到了）

持续计时器 (persistence timer)。

- TCP 为每一个连接设有一个持续计时器。
- 只要 TCP 连接的一方收到对方的零窗口通知，就启动持续计时器。
- 若持续计时器设置的时间到期，就发送一个零窗口探测报文段（仅携带 1 字节的数据），而对方就在确认这个探测报文段时给出了现在的窗口值。
- 若窗口仍然是零，则收到这个报文段的一方就重新设置持续计时器。
- 若窗口不是零，则死锁的僵局就可以打破了。

必须考虑传输效率

- 可以用不同的机制来控制 TCP 报文段的发送时机:
- 第一种机制是 TCP 维持一个变量，它等于最大报文段长度 MSS。只要缓存中存放的数据达到 MSS 字节时，就组装成一个 TCP 报文段发送出去。
- 第二种机制是由发送方的应用进程指明要求发送报文段，即 TCP 支持的推送(push)操作。
- 第三种机制是发送方的一个计时器期限到了，这时就把当前已有的缓存数据装入报文段（但长度不能超过 MSS）发送出去。

本讲总结

TCP可靠传输的实现

滑动窗口，超时重传时间的选择，选择确认

TCP的流量控制

利用滑动窗口实现流量控制，传输效率

作业

- 5-29，5-32