[面试·网络] TCP/IP（五）：TCP 协议详解

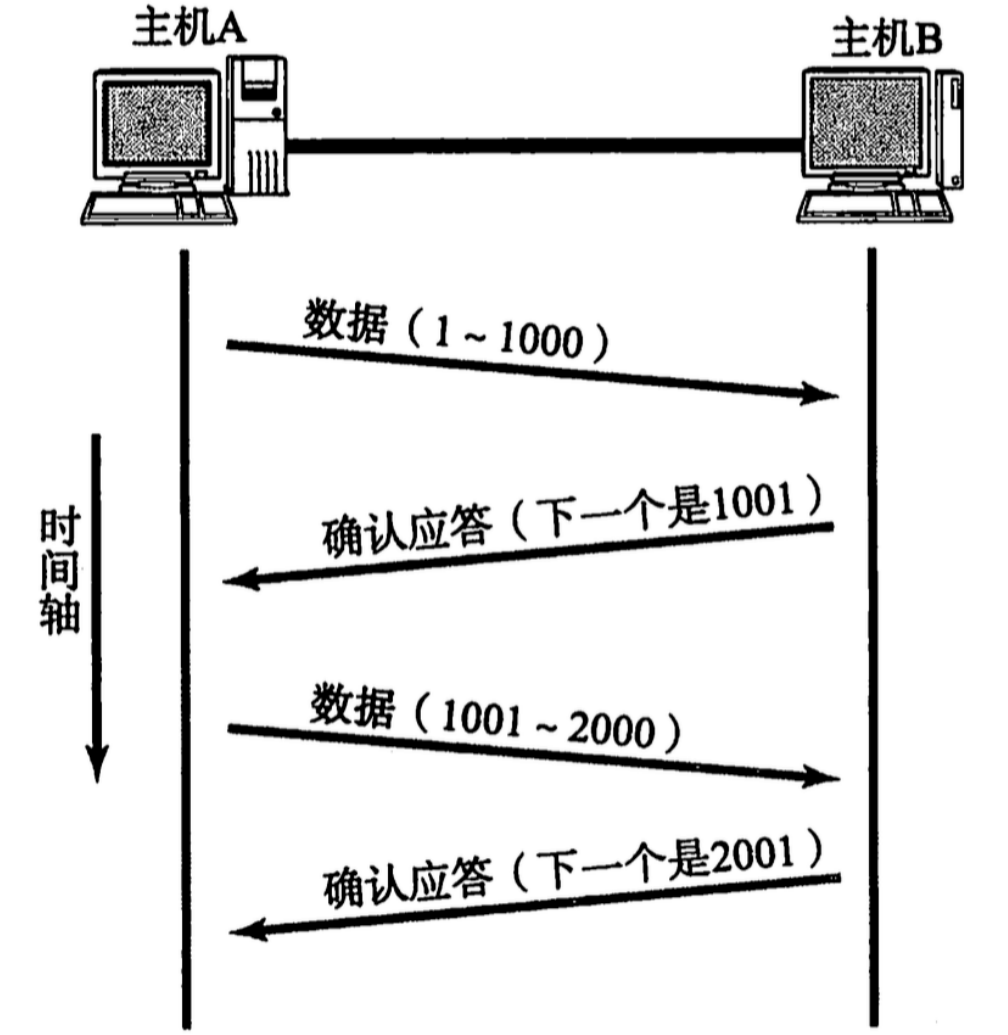
上一节 中讲过，TCP 协议是面向有连接的协议，它具有丢包重发和流量控制的功能，这是它区别于 UDP 协议最大的特点。本文就主要讨论这两个功能。

数据包重发

数据发送

丢包重发的前提是发送方能够知道接收方是否成功的接收了消息。所以，在 TCP 协议中，接收端会给发送端返回一个通知，也叫作确认应答（ACK），这表示接收方已经收到了数据包。

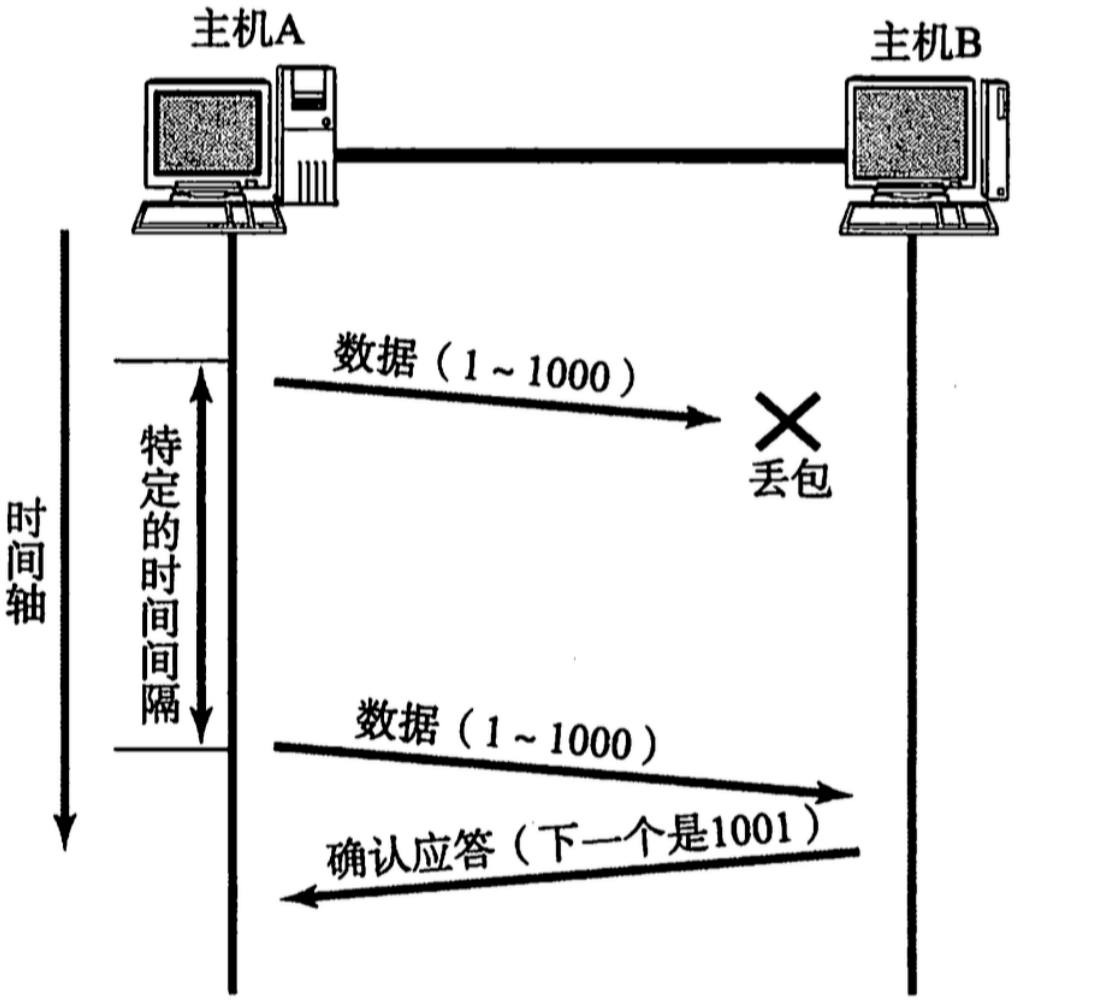
根据上一节对 TCP 首部的分析得知，ACK 的值和下次发送数据包的序列号相等。因此 ACK 也可以理解为：“发送方，下次你从这个位置开始发送！”。下图表示了数据发送与确认应答的过程：



当数据从主机A发送到主机B时，主机B会返回给主机A一个确认应答。

ACK 确认

数据包和 ACK 应答都有可能丢失，在这种情况下，发送方如果在一段时间内没有收到 ACK，就会重发数据：



当数据由主机A发出后如果因网络拥堵等原因丢失的话，该数据将无法到达主机B。此时，如果主机A在一个特定时间间隔内都未收到主机B发来的确认应答，将会对此数据进行重发。

未收到 ACK 时重发数据

即使网络连接正常，由于延迟的存在，接收方也有可能收到重复的数据包，因此接收方通过 TCP 首部中的 SYN 判断这个数据包是否曾经接收过。如果已经接收过，就会丢弃这个包。

重传超时时间(RTO)

如果发送方等待一段时间后，还是没有收到 ACK 确认，就会启动超时重传。这个等待的

时间被称为重传超时时间(RTO，Retransmission TimeOut)。RTO 的值具体是多久呢?

首先，RTO 的值不是固定的，它是一个动态变化的时间。这个时间总是略大于连接往返时间（RTT，Round Trip Time）。这个设定可以这样理解：“数据发送给对方，再返回到我这里，假设需要 10 秒，那我就等待 12秒，如果超过 12 秒，那估计就是回不来了。”

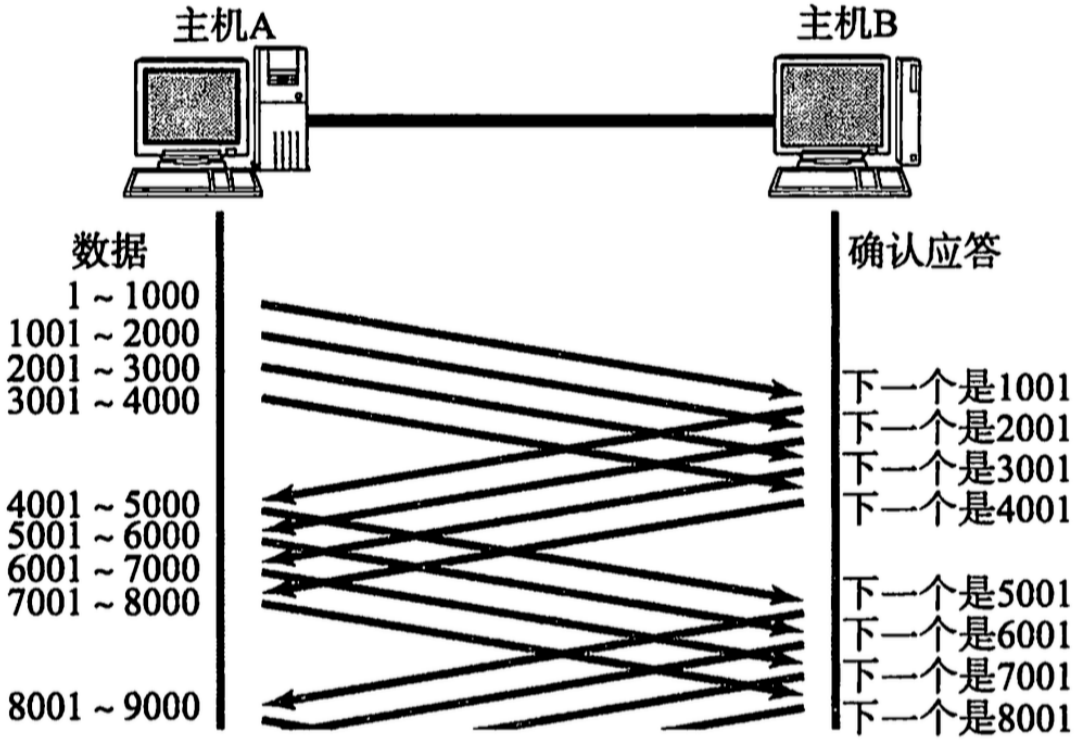
RTT 是动态变化的，因为谁也不知道网络下一时刻是否拥堵。而当前的 RTO 需要根据未来的 RTT 估算得出。RTO 不能估算太大，否则会多等待太多时间；也不能太小，否则会因为网络突然变慢而将不该重传的数据进行重传。

RTO 有自己的估算公式，可以保证即使 RTT 波动较大，它的变化也不会太剧烈。感兴趣的读者可以自行查阅相关资料。

TCP 窗口

按照之前的理论，在数据包发出后，直至 ACK 确认返回以前，发送端都无法发送数据，而且包的往返时间越长，网络利用效率和通信性能就越低。前两张图片形象的解释了这一点。

为了解决这个问题，TCP 使用了“窗口”这个概念。窗口具有大小，它表示无需等待确认应答就可以继续发送数据包的最大数量。比如窗口大小为 4 时，数据发送的示意图如下:



- 根据窗口为4000字节时返回的确认应答，下一步就发送比这个值还要大4000个序列号为止的数据。这跟前面每个段接收确认应答以后再发送另一个新段的情况相比，即使往返时间变长也不会影响网络的吞吐量。

窗口大小为 4

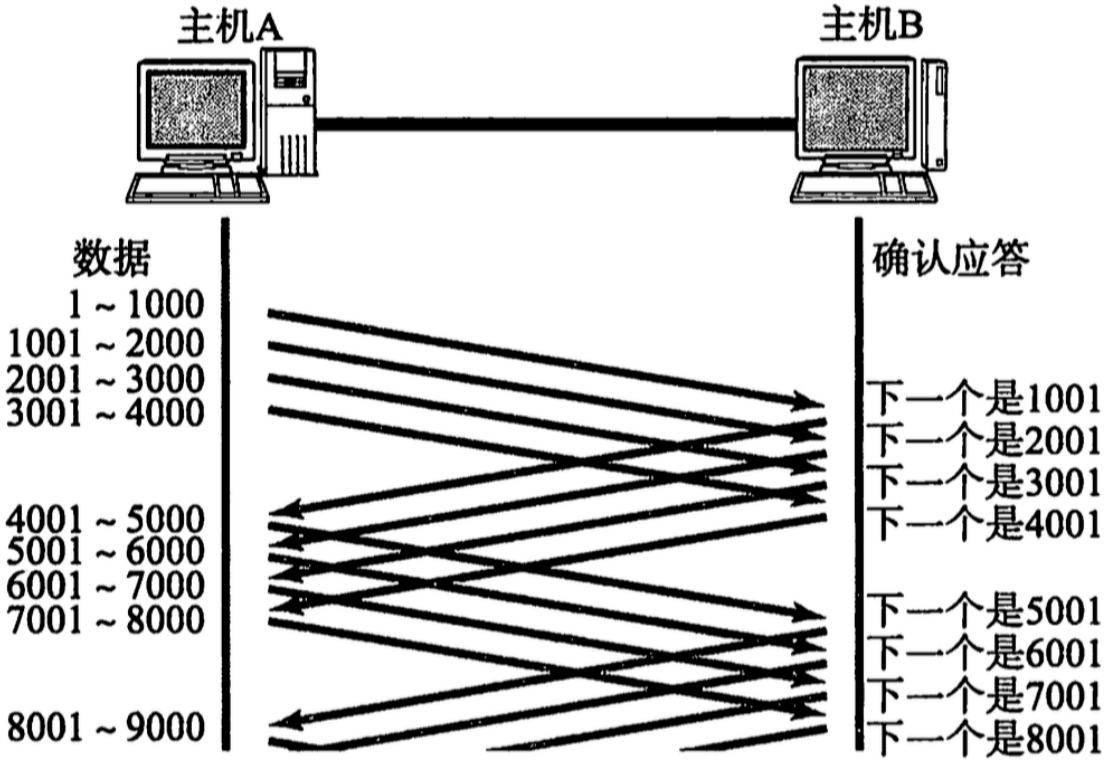
不等确认就连续发送若干个数据包会不会有问题呢？我们首先来看数据包丢失问题。

我们知道 TCP 首部中的 ACK 字段表示接收方已经收到数据的最后位置。因此，接收方成功接收到了 1–1000 字节的数据后，它会发送一个 ACK = 1001 的确认包。假设 1001–2000 字节的数据包丢失了，由于窗口长度比较大，发送方会继续发送 2001–3000 字节的数据包。接收端并不会返回这个数据包的确认，因为它最后收到的数据还是 1–1000 字节的数据包。

因此，接收端返回的数据包的 ACK 依然是 1001。这表示：“喂，发数据的，别往后发了，你第 1001 字节开始的数据还没来呢”。可以想见，发送端以后每次发送数据包得到的确认中，ACK 的值都是 1001。当连续收到三次确认之后，发送方会意识到：“对方还没有接收到数据，这个包需要重传”。

因此，引入窗口的概念后，被发送的数据不能立刻丢弃，需要缓存起来以备将来需要重发。

利用窗口发送数据的过程可以用下图表示：

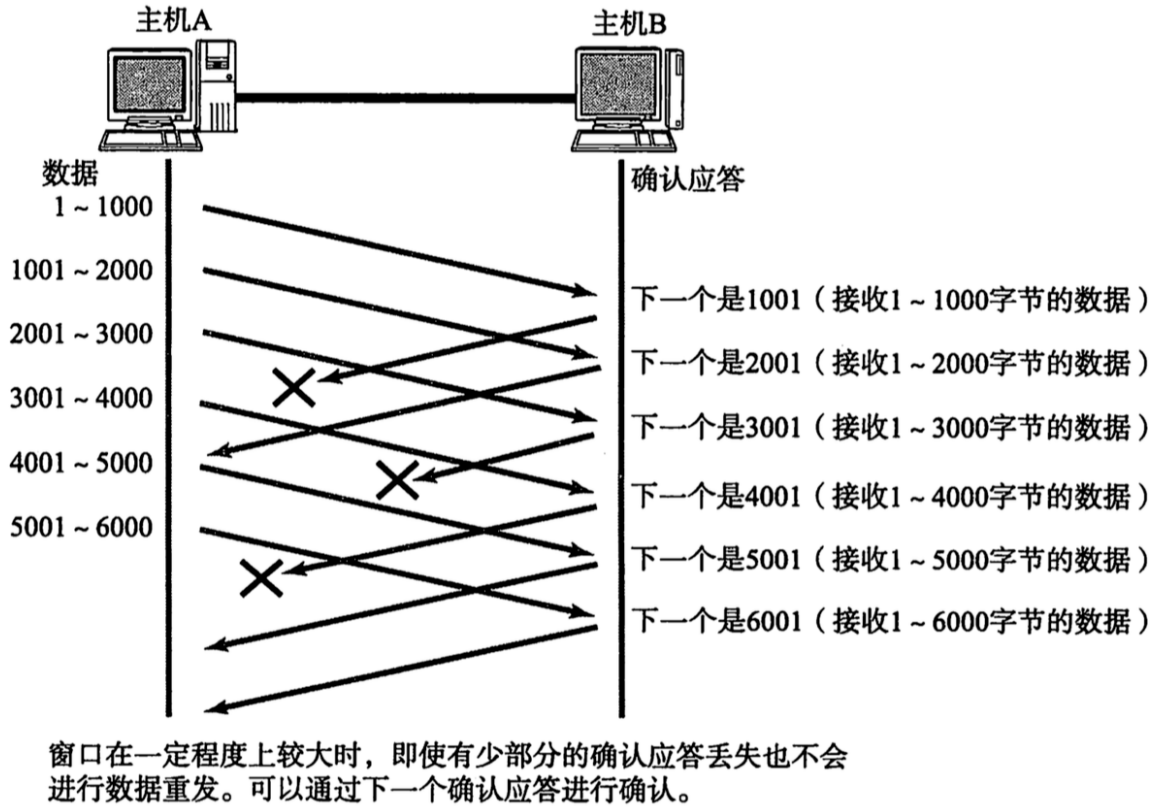


- 根据窗口为4000字节时返回的确认应答，下一步就发送比这个值还要大4000个序列号为止的数据。这跟前面每个段接收确认应答以后再发送另一个新段的情况相比，即使往返时间变长也不会影响网络的吞吐量。

快速重传

如果是数据包没有丢失，但是确认包丢失了呢？这就是窗口最擅长处理的问题了。假设发送发收到的确认包中的 ACK 第一次是 1001，第二次是 4001。那么我们完全可以相信中间的两个包是成功被接收的。因为如果有没接收到的包，接收方是不会增加 ACK 的。

在这种情况下，如果不使用窗口，发送方就需要重传第二、三个数据包，但是有了窗口的概念后，发送方就省略了两次重传。因此使用窗口实际上可以理解为“空间换时间”。



某些确认包丢失时不用重发

流量控制

窗口大小

如果窗口过大，会导致接收方的缓存区数据溢出。这时候本该被接收的数据反而丢弃了，就会导致无意义的重传。因此，窗口大小是一个可以改变的值，它由接收端主机控制，附加在 TCP 首部的“窗口大小”字段中。

慢启动

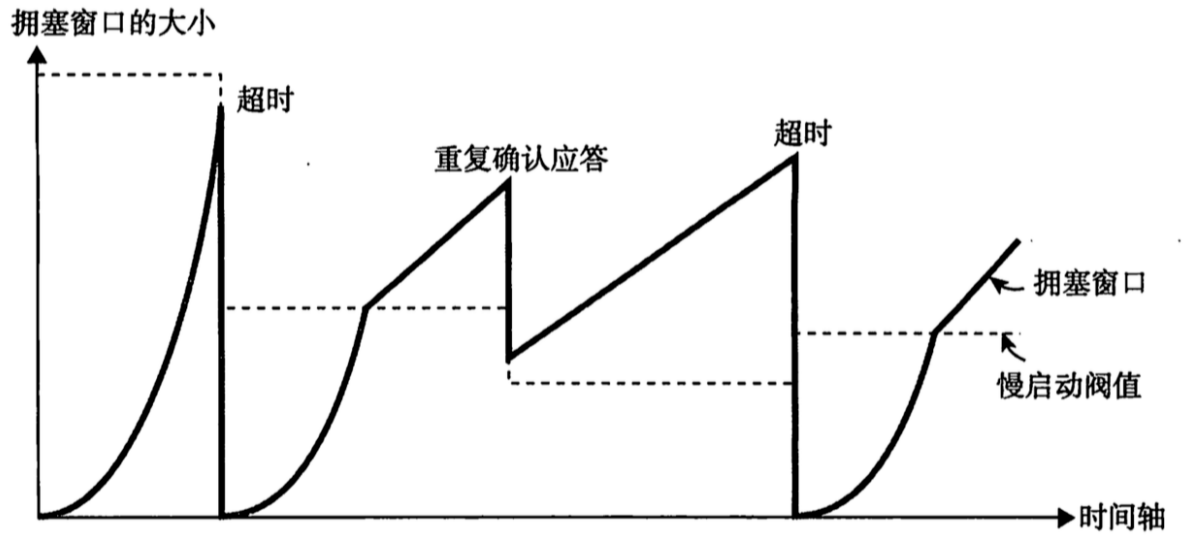
在连接建立的初期，如果窗口比较大，发送方可能会突然发送大量数据，导致网络瘫痪。因此，在通信一开始时，TCP 会通过慢启动算法得出窗口的大小，对发送数据量进行控制。

流量控制是由发送方和接收方共同控制的。刚刚我们介绍了接收方会把自己能够承受的最大窗口长度写在 TCP 首部中，实际上在发送方这里，也存在流量控制，它叫拥塞窗口。TCP 协议中的窗口是指发送方窗口和接收方窗口的较小值。

慢启动过程如下：

1. 通信开始时，发送方的拥塞窗口大小为 1。每收到一个 ACK 确认后，拥塞窗口翻倍。
2. 由于指数级增长非常快，很快地，就会出现确认包超时。
3. 此时设置一个“慢启动阈值”，它的值是当前拥塞窗口大小的一半。
4. 同时将拥塞窗口大小设置为 1，重新进入慢启动过程。
5. 由于现在“慢启动阈值”已经存在，当拥塞窗口大小达到阈值时，不再翻倍，而是线性增加。
6. 随着窗口大小不断增加，可能收到三次重复确认应答，进入“快速重发”阶段。
7. 这时候，TCP 将“慢启动阈值”设置为当前拥塞窗口大小的一半，再将拥塞窗口大小设置成阈值大小（也有说加 3）。
8. 拥塞窗口又会线性增加，直至下一次出现三次重复确认应答或超时。

以上过程可以用下图概括：



窗口大小变化示意图

强烈建议读者对照上述八个步骤理解这幅图！