New Reno TCP 拥塞控制算法

1120202944 董若扬

北京理工大学计算机学院 07112005 班，北京 100081
(dry@bit.edu.cn)

New Reno TCP Congestion Control Algorithm

Dong Ruoyang
(Class 07112005,School of Computer Science, Beijing Institute of Technology, Beijing 100081)

Abstract This article introduces three TCP congestion control algorithms: Tahoe TCP, Reno TCP, and New Reno TCP, and analyzes their advantages and limitations. Tahoe TCP uses three mechanisms, slow start, congestion avoidance, and fast retransmit, to control the size of the congestion window and avoid network congestion. Reno TCP adds fast recovery mechanism on the basis of Tahoe TCP, which can recover multiple lost packets within one timeout, improving the transmission efficiency. New Reno TCP improves Reno TCP and can recover all lost packets within one timeout, further improving the transmission efficiency. This article also compares the performance of New Reno TCP with other congestion control algorithms and finds that New Reno TCP performs well in throughput, delay, and fairness. The advantages and limitations of the New Reno TCP congestion control algorithm are detailed, and it is pointed out where the New Reno TCP algorithm may affect performance in specific environments and corresponding improvement suggestions are given. This article summarizes the advantages and disadvantages of New Reno TCP and points out future research directions.

Key words New Reno TCP , Congestion Control

摘要 本文介绍了 TCP 拥塞控制的三种算法：Tahoe TCP，Reno TCP 和 New Reno TCP，并分析了它们的优势与局限。Tahoe TCP 使用慢开始，拥塞避免和快重传三个机制来控制拥塞窗口的大小，避免网络拥塞。Reno TCP 在 Tahoe TCP 的基础上增加了快恢复机制，可以在一次超时内恢复多个丢失的分组，提高了传输效率。New Reno TCP 对 Reno TCP 进行了改进，可以在一次超时内恢复所有丢失的分组，进一步提高了传输效率。本文还比较了 New Reno TCP 与其他拥塞控制算法的性能，发现 New Reno TCP 在吞吐量，延迟和公平性方面都有较好的表现。详细 New Reno TCP 拥塞控制算法的优势与局限性，指出了 New Reno TCP 算法在哪些特定的环境下可能会影响运行性能，并给出了相应的改进意见。本文总结了 New Reno TCP 的优点和缺点，并指出了未来的研究方向。

关键词 New Reno TCP，拥塞控制

随着互联网的快速发展，网络拥塞问题日益严重，对网络性能产生极大影响。为了解决这一问题，研究人员提出了多种拥塞控制算法。本文主要研究 New Reno TCP 拥塞控制算法，分析其在网络性能优化方面的应用和优势。New Reno 算法是 TCP 拥塞控制算法中的一种改进型算法，是 Reno 算法的改进版本[1]。与 Reno 算法相比，New Reno 在处理多个拥塞数据包时具有更好的性能表现。本文首先介绍 New Reno TCP 拥塞控制算法的原理和实现，然后分析算法的优势和局限，最后比较 New Reno TCP 与其他拥塞控制算法的性能。

1. Tahoe TCP 拥塞控制

Tahoe TCP 是 TCP 协议的最早版本，其拥塞控制包括三个算法：慢启动、拥塞避免和快重传。这三个算法都基于拥塞窗口 cwnd 这一变量，以控制发送方的数据传输速率。拥塞窗口的大小会根据网络的拥塞程度进行动态调整，以适应网络的变化。

Tahoe TCP 拥塞控制是 TCP 协议发展历史上的一个里程碑，它有效地提高了 TCP 的性能和稳定性。但是它也有一些缺点，例如它不能区分数据丢失是由于网络拥塞还是其他原因造成的[2]，它对每次发生拥塞都采用相同的处理方式，它不能利用接收方反馈的更多信息来优化发送方的行为等。后续的 TCP 版本对这些缺点进行了改进和优化。

1.1 慢开始

TCP 在建立连接并开始发送 TCP 报文段时，采用慢开始算法。首先，将拥塞窗口 cwnd 设置为 1 个最大报文段长度 MSS。每当接收到一个新报文段的确认时，就将 cwnd 加 1（增加一个 MSS），逐步增加发送方的 cwnd，从而使注入网络的分组速率更加合理。例如，当 A 向 B 发送数据时，A 先将 cwnd 设置为 1。发送第一个报文段后，收到 B 对第一个报文段的确认后，将 cwnd 从 1 增加到 2。然后 A 接着发送两个报文段，收到 B 对这两个报文段的确认后，将 cwnd 从 2 增加到 4。下次就可以一次发送 4 个报文段。

慢启动算法的目的是在连接建立后，不要一开始就发送大量的数据，而是先探测一下网络的状况，逐渐增加拥塞窗口的大小。慢启动算法的过程如下：

- 当连接建立时，将 cwnd 初始化为 1 个最大报文段（MSS）大小。
- 每当收到一个确认报文（ACK），将 cwnd 增加 1 个 MSS 大小。这样，cwnd 会随着往返时间（RTT）呈指数增长。[3]
- 当 cwnd 达到一个慢启动门限（ssthresh）时，停止慢启动过程，进入拥塞避免过程。[4]

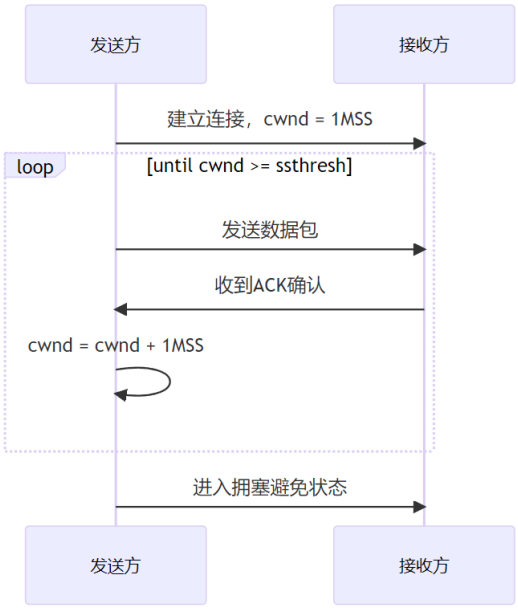

图 1 慢开始算法

Fig. 1 Slow start algorithm

慢开始算法中的“慢”并不是指拥塞窗口 cwnd 增长速率慢，而是指 TCP 开始发送报文段时先设置 cwnd 为 1，只发送一个报文段以试探网络的拥塞情况，然后逐渐增加 cwnd。这是一种有效的措施，可以防止网络出现拥塞。在应用慢开始算法时，每个传输轮次（即往返时延 RTT）结束后，cwnd 的值会加倍，即 cwnd 的增长遵循指数规律。因此，慢开始会不断增加 cwnd 的值，直到达到一个设定的慢开始门限 ssthresh，然后切换到拥塞避免算法。

1.2 拥塞避免

基本思路是通过缓慢增大拥塞窗口 cwnd 来控制网络拥塞，拥塞避免算法的具体实现是在每经过一个往返时延 RTT 后将发送方拥塞窗口 cwnd 加 1，而非加倍。这种增长速率是线性规律的，即加法增大，相比慢开始算法，更为缓慢。

在慢开始阶段和拥塞避免阶段，发送方一旦检测到网络出现拥塞（即未及时收到确认），就会将慢开始门限 ssthresh 设置为当前拥塞窗口 cwnd 值的一半（但不得小于 2）。然后将 cwnd 重新设置为 1，执行慢开始算法。这样做的目的是尽快减少注入网络中的分组数量，并给发生拥塞的路由器足够的时间来处理积压的分组。

慢开始算法和拥塞避免算法都采用了“乘法减小”和“加法增大”的机制。在检测到超时（即可能出现网络拥塞）时，无论是在慢开始阶段还是拥塞避免阶段，都会将慢开始门限值 ssthresh 设置为当前拥塞窗口的一半，并执行慢开始算法。当网络拥塞频繁发

生时，ssthresh 值会快速下降，从而大幅减少向网络注入的分组数量。而“加法增大”则是指在拥塞避免阶段，每当收到所有报文段的确认后（即经过一个 RTT），就会将拥塞窗口 cwnd 增加 MSS 大小，以缓慢增大拥塞窗口，避免网络过早发生拥塞。

拥塞避免算法的目的是在网络接近饱和时，避免引起过多的拥塞。拥塞避免算法的过程如下：

- 当 cwnd 大于 ssthresh 时，进入拥塞避免阶段。
- 在拥塞避免阶段，每当收到一个 RTT 内的所有确认报文时，将 cwnd 增加 1 个 MSS 大小。这样，cwnd 会随着 RTT 呈线性增长。
- 当发生超时或收到三个重复的确认报文时，认为网络发生了拥塞，将 ssthresh 设置为当前 cwnd 的一半，然后重新进入慢启动过程。

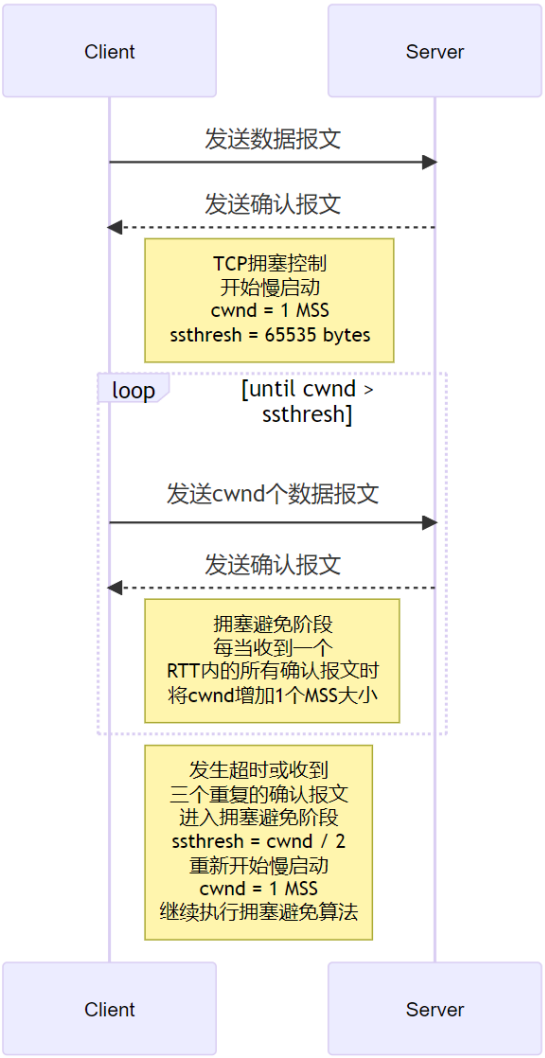

图 2 拥塞避免算法

Fig. 2 Congestion avoidance algorithm

1.3 快重传

TCP 的可靠传输机制中，快速重传技术利用冗余 ACK 来检测数据包的丢失情况，同时也可用于检测网络拥塞，因为数据包的丢失可能是拥塞的信号。快速重传不会取消重传计时器，但在某些情况下可以更早地重传丢失的报文段。

当发送方收到连续三个重复的 ACK 报文时，它会立即重传尚未被接收方收到的报文段，而无需等待该报文段设置的重传计时器超时。

快重传算法的目的是在检测到数据丢失时，尽快地重传丢失的数据。快重传算法的过程如下：

- 当发送方收到三个重复的确认报文时，认为最早发送的报文段丢失了。
- 不等待超时定时器到期，立即重传丢失的报文段。
- 重传后继续执行拥塞避免算法。

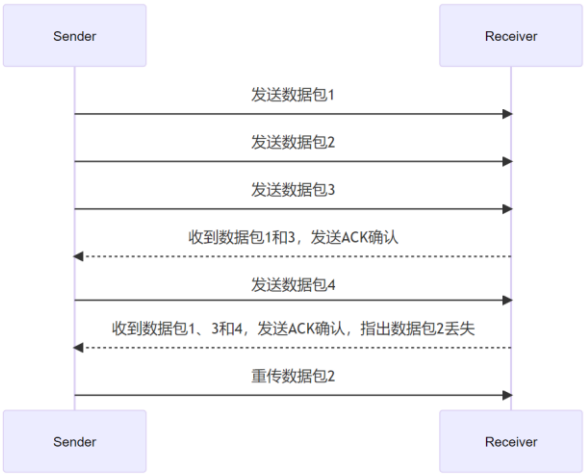

图 3 快重传算法

Fig. 3 Fast retransmit algorithm

例如发送方发送了 1，2，3 三个数据包给接收方，但是第二个数据包 2 丢失了。接收方只收到了 1 和 3 两个数据包，于是发送一个 ACK 确认收到了 1 和 3。但是发送方收到了这个 ACK 后并没有马上重传数据包 2，而是继续发送数据包 4。此时接收方收到了 4 和之前重复的 ACK，于是它发送一个 ACK 确认收到了 1，3 和 4，并在 ACK 中指出了 2 号数据包丢失的情况。当发送方收到这个 ACK 时，它就知道数据包 2 已经丢失，同时也知道接收方已经收到了 1，3 和 4。于是发送方可以立即重传数据包 2，而不必等待超时时间到达。这样可以避免不必要的等待，提高数据传输的效率。如果发送方在收到第一个 ACK 时就立即重传 2 号数据包，就可以避免等待超时时间，从而加快数据传输速度。

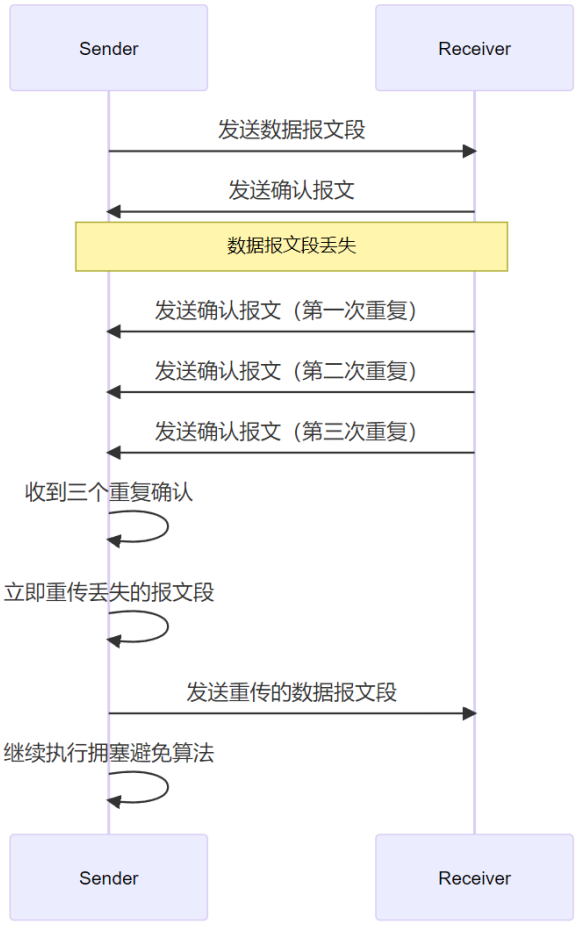

图 4 快重传算法示例

Fig. 4 Example of fast retransmit algorithm

2. Reno TCP 拥塞控制

Reno TCP 拥塞控制是一种 TCP 的拥塞控制算法，它在 Tahoe 的基础上增加了快速恢复（Fast Recovery）的机制。Reno TCP 拥塞控制算法相比 Tahoe 算法可以提高网络的吞吐量和响应速度，但是它也有一些缺点。首先，它只能处理一次拥塞中最多丢失一个分段的情况，如果在同一个 RTT 内丢失了多个分段，它会误认为发生了多次拥塞，导致 cwnd 和 ssthresh 多次减半，降低网络性能。其次，它在快速恢复阶段过于激进地增加 cwnd，可能会导致网络中的数据量超过网络的承载能力，引起更多的拥塞和分段丢失。因此，后续的 TCP 版本对 Reno 算法进行了改进和优化。

2.1 快恢复

当发送方连续接收到三个冗余 ACK（即重复确认）时，会触发“乘法减小”算法，并将慢启动阈值（ssthresh）设置为当前拥塞窗口（cwnd）的一半，以避免网络拥塞。然而，发送方并不认为网络发生了（严重）拥塞，否则就不会出现几个报文段连续到达接收方，也不会连续收到重复确认。因此，快速恢复算法与慢启动不同，它将 cwnd 值设为更改后的 ssthresh 值，并开始执行拥塞避免算法（“加性增加”），逐渐线性增加拥塞窗口。

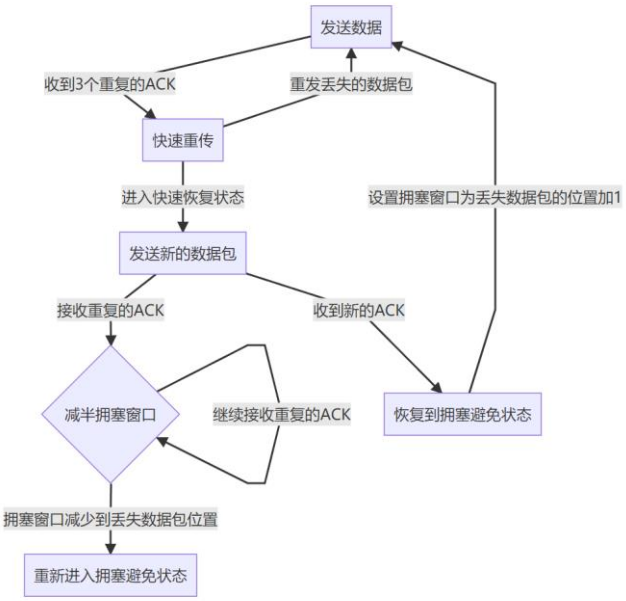

图 5 快恢复算法

Fig. 5 Fast recovery algorithm

例如发送方发送了 1，2，3，4 四个数据包给接收方，但是第二个数据包 2 丢失了。接收方只收到了 1，3 和 4 三个数据包，于是发送一个 ACK 确认收到了 1，3 和 4。但是发送方收到了这个 ACK 后并没有马上重传数据包 2，而是继续发送数据包 5。此时接收方收到了 5 和之前重复的 ACK，于是它发送一个 ACK 确认收到了 1，3，4 和 5，并在 ACK 中指出了 2 号数据包丢失的情况。

根据 Reno TCP 的快恢复算法，当发送方收到第三个重复的 ACK 时[5]，它会认为 2 号数据包已经丢失，但是它并不会立刻进入超时重传状态，而是将拥塞窗口减半，即从之前的 4 个数据包减为 2 个数据包，然后继续发送数据包 6 和 7。这样做的目的是为了避免进入超时重传状态，从而加快数据传输速度。

在快恢复状态下，发送方会等待一个新的 ACK 确认，如果收到了这个 ACK，就说明之前的重复 ACK 是由于网络延迟或其他原因引起的，而不是数据包丢失导致的。此时，发送方会将拥塞窗口恢复到之前的大小，即从 2 个数据包恢复到 4 个数据包，然后继续发送数据包。如果在快恢复状态下，发送方收到了一个新的 ACK 确认，它就会认为之前的重复 ACK 是由于数

据包丢失导致的，于是立即进入超时重传状态，重新发送丢失的数据包。

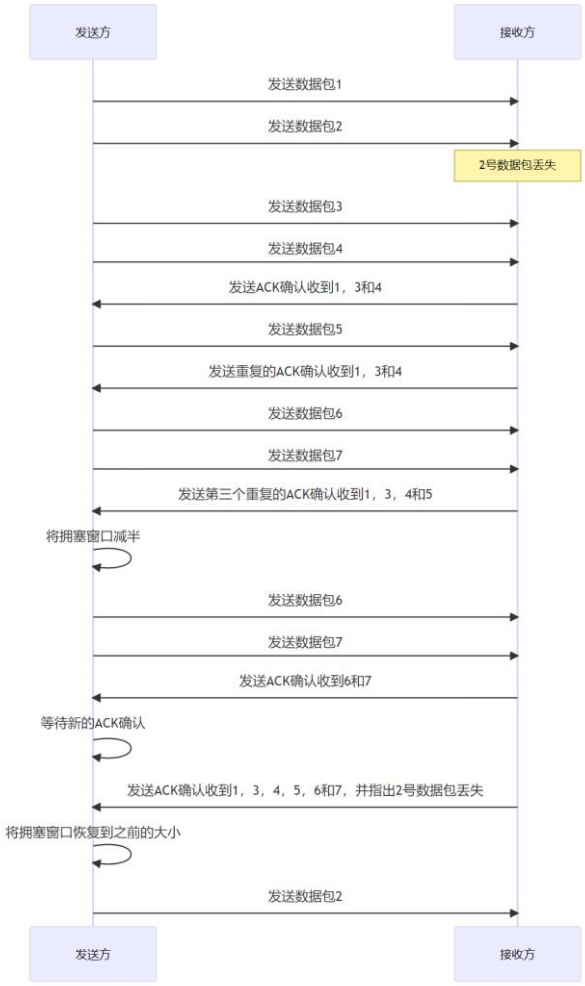

图 6 快恢复算法

Fig. 6 Example of fast recovery algorithm

3. New Reno TCP 拥塞控制

New Reno TCP 的核心思想是在保证网络稳定的前提下，尽可能地提高网络吞吐量和降低延迟。其主要改进了 Reno 算法中处理多个拥塞数据包的方式，通过精确地检测拥塞窗口内的数据包丢失，降低拥塞窗口的减小幅度，从而提高网络性能。

New Reno TCP 拥塞控制算法主要包括四个阶段：慢启动、拥塞避免、快速重传和快速恢复。[6]在慢启动阶段，拥塞窗口按指数增长，直至达到阈值。拥塞避免阶段中，拥塞窗口按线性增长。快速重传阶段在接收到三个相同的确认报文时触发，快速恢复阶段则在收到新数据包的确认报文后开始。

New Reno 主要改进了快速恢复阶段的处理方式。在 Reno 算法中，发送方在快速恢复阶段结束时，将拥塞窗口减半。然而，这会导致多个数据包丢失时，拥塞窗口过小，降低网络性能。New Reno 通过在快速恢复阶段持续更新拥塞窗口大小，使其更加精确地反映实际网络拥塞情况。

New Reno TCP 拥塞控制算法可以提高 TCP 的吞吐量和鲁棒性，适用于高速网络和高延迟网络的环境。

在实现 New Reno 算法时，TCP 协议栈需要维护以下几个重要的变量：

1) 拥塞窗口大小（cwnd）：用来控制发送方每一个 RTT 内可以发送的数据量，通常是动态调整的。
2) 拥塞阈值（ssthresh）：用来控制拥塞窗口大小的增长速率，通常是根据网络状态动态调整的。
3) 已发送但未被确认的数据段数量（unacked）：用来记录发送方已经发送但还未收到确认的数据段数量。
4) 最近一次收到的 ACK 序号（last_ack）：用来记录最近一次收到的 ACK 序号，以便判断是否接收到了重复的 ACK。

在 New Reno 算法中，当发生拥塞事件时，TCP 协议栈会根据以下步骤进行拥塞控制：

1) 将拥塞窗口大小减半，并将拥塞阈值设置为当前的拥塞窗口大小。
2) 将 unacked 变量设置为当前发送方已经发送但未被确认的数据段数量。
3) 当接收到一个新的 ACK 时，如果它确认了一个未确认的数据段，则将 unacked 减少该数据段的长度，如果它确认了多个未确认的数据段，则将 unacked 减少所有被确认的数据段的长度。
4) 如果接收到重复的 ACK，则将拥塞窗口大小加 1，直到接收到一个新的 ACK。
5) 如果接收到的 ACK 确认了新的数据段，则将拥塞窗口大小设置为 unacked 加 1，然后继续执行拥塞避免算法。

通过上述步骤，New Reno 算法能够更加准确地判断出哪些数据段已经被接收方接收到，从而避免了不必要的拥塞窗口减半，提高了 TCP 的性能和吞吐量。

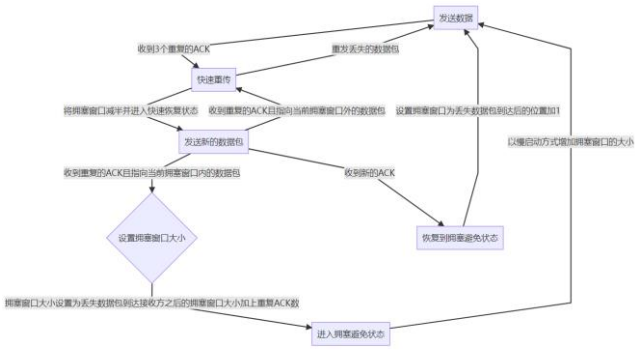

图 7 New Reno TCP **拥塞控制**

Fig. 7 New Reno TCP congestion control

例如发送方发送了一系列数据包，编号为 1 到 10，接收方只收到了 1，2，3，6 和 7 这五个数据包，而 4，5，8，9，10 这五个数据包全部丢失了。此时接收方会发送一个 ACK 确认收到了 1，2，3，6 和 7 这五个数据包，但是由于丢失了 4，5，8，9，10 这五个数据包，因此发送方会认为网络出现了拥塞。

此时，New Reno TCP 会根据拥塞窗口的大小决定需要重传多少个数据包。假设拥塞窗口大小为 4 个数据包，那么发送方会立即重传数据包 4，5，6 和 7 这四个数据包。由于接收方已经收到了数据包 1，2，3，6 和 7，因此它会继续发送 ACK 确认收到了这五个数据包。但是由于发送方在重传数据包 4，5，6 和 7 时，接收方可能会收到多个重复的 ACK，因为它在等待重传的同时也收到了已经确认过的数据包 6 和 7。此时，New Reno TCP 会记录之前未确认的数据包数量，即 4，5 这两个数据包未确认，然后将拥塞窗口的大小减半，即从 4 个数据包减为 2 个数据包。

接着，发送方会继续发送数据包 8 和 9，并等待接收方的确认。如果接收方在一定时间内没有确认收到这两个数据包，发送方会认为这两个数据包也丢失了，于是立即进入超时重传状态，重新发送数据包 8，9 和 10。如果接收方在这个过程中收到了 ACK 确认，那么发送方会根据拥塞窗口的大小决定需要发送多少个数据包，从而逐步提高数据的发送速度，以充分利用网络带宽。

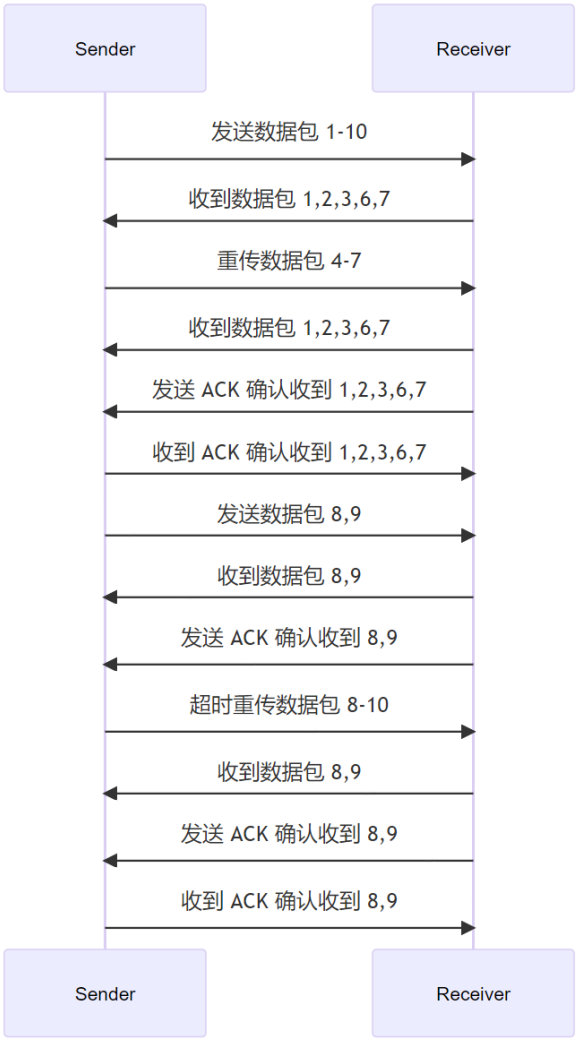

图 8 New Reno TCP **拥塞控制**

Fig. 8 Example of New Reno TCP congestion control

通过这种方式，New Reno TCP 可以更好地处理多个数据包的丢失情况。当出现多个数据包的丢失时，New Reno TCP 会根据拥塞窗口的大小决定需要重传的数据包数量，并根据接收方的确认情况逐步恢复数据的发送速度。这样可以更好地处理网络拥塞情况，提高数据传输的效率。

4. New Reno TCP 拥塞控制算法优势与局限

4.1 优势

New Reno TCP 拥塞控制算法的优势不仅在于其能够更有效地处理多个分段丢失的情况，还在于其在快速恢复阶段中能够更加充分地利用可用的网络带宽，避免了不必要的超时重传和慢启动。这是因为 New Reno 算法在进入快速恢复阶段后，会维持一个拥塞窗口大小为 ssthresh+3MSS，这样可以最大化利用网络带宽，以便更快地恢复丢失的分段。

此外，New Reno 算法还能够准确地判断一次拥

塞中丢失了多少个分段，这样可以更加精细地控制拥塞窗口的大小，从而提高网络吞吐量和健壮性。相比于 TCP Reno 算法，在网络拥塞恢复时，New Reno 算法的延迟较低，有助于提高网络效率。这是因为 New Reno 算法使用了更加有效的重传机制，避免了不必要的超时重传，从而减少了网络延迟和抖动。

另外，New Reno 算法还具有良好的适应性，能够自适应地调整拥塞窗口大小，以适应网络状况的变化。当网络状况较好时，New Reno 算法可以逐渐增加拥塞窗口的大小，以提高网络吞吐量；而当网络状况恶化时，New Reno 算法可以及时减小拥塞窗口的大小，以避免造成网络拥塞。

总之，New Reno TCP 拥塞控制算法的优势在于其能够更加精细地控制拥塞窗口大小，准确地判断丢失的分段数量，以及在快速恢复阶段中更加充分地利用可用的网络带宽，从而提高网络吞吐量和健壮性，降低网络延迟和抖动，具有较好的适应性和性能表现。相比于传统的 TCP Reno 算法，New Reno 算法在处理多个拥塞数据包时表现更加出色，能够更加有效地利用网络资源，提高网络的传输效率和可靠性。因此，New Reno TCP 拥塞控制算法已经成为当今网络传输领域中广泛使用的一种协议，被广泛应用于各类网络应用中，如视频流媒体、云计算、物联网等，为网络传输的高效和稳定提供了重要的支持。

4.2 局限

TCP 拥塞控制算法是一种用于避免网络拥塞的机制，它通过调整发送方的窗口大小来控制数据流量。New Reno 是 TCP 拥塞控制算法的一种改进版本，它在 Reno 的基础上增加了快速恢复的功能，即在发生一次超时后，只需要重传一个丢失的分组，而不是整个窗口的分组。这样可以减少重传的开销，提高网络吞吐量。

然而，New Reno TCP 拥塞控制算法也存在一些局限：

1) New Reno 只能处理一个分组丢失的情况，如果在一个窗口内有多个分组丢失，它仍然需要等待超时事件发生，才能重传所有丢失的分组。这会导致网络利用率降低，延迟增加。
2) New Reno 在快速恢复阶段，会将拥塞窗口减半，这会使得发送方的发送速率降低。如果网络条件恢复良好，发送方需要花费较长的时间来重新增加窗口大小，从而适应网络状况。这会影响网络性能。
3) New Reno 在快速恢复阶段，会维持一个较大的拥塞窗口，这会使得发送方对网络拥塞的反应不够灵敏。如果网络出现持续的拥塞，发送方可能会过度发送数据，造成更多的分组丢失和重传。这会加剧网络拥塞。

虽然 New Reno 在多个数据包丢失时的性能有所改进，但其在单个数据包丢失时的性能表现与 Reno 算法类似，仍存在提升空间。同时，New Reno 算法在高速网络和高延迟环境下的性能表现仍有待优化。

5. New Reno TCP 与其他拥塞控制算法的性能比较

New Reno TCP 是一种改进的 TCP 拥塞控制算法，它主要解决了 Reno TCP 在多个分段丢失时的性能问题。New Reno TCP 在快速恢复阶段，不会在收到一个新的 ACK 后就退出[7]，而是等待所有丢失的分段都被重传并确认后才退出。这样可以避免多次降低拥塞窗口和慢启动阈值，提高网络吞吐量和效率。

Table 1 Performance comparison of New Reno TCP with other congestion control algorithms

表 1 New Reno TCP 与其他拥塞控制算法的性能比较

指标	New Reno TCP	Reno TCP	Tahoe TCP	Vegas TCP
网络吞吐量	可以在多个分段丢失时保持较高的网络吞吐量，减少超时重传的次数和拥塞窗口的剧烈波动	较低的网络吞吐量，容易出现拥塞窗口的剧烈波动	较低的网络吞吐量，容易出现拥塞窗口的剧烈波动	过于保守地调整拥塞窗口，无法充分利用带宽
网络延迟	可以缩短快速恢复阶段的时间，减少分段在网络中的排队等待，降低网络延迟	快速恢复阶段的时间较长，分段在网络中的排队等待时间较长，增加网络延迟	快速恢复阶段的时间较长，分段在网络中的排队等待时间较长，增加网络延迟	过于敏感地检测网络拥塞，过早地进入拥塞避免阶段，增加分段的往返时间，增加网络延迟
网络公平性	可以提高网络公平性，减少拥塞窗口的震荡，使得多个 TCP 流之间能够更均匀地分享带宽资源	拥塞窗口的剧烈波动会影响网络公平性。	拥塞窗口的剧烈波动会影响网络公平性，使得多个 TCP 流之间无法均匀地分享带宽资源	过于谨慎地增加拥塞窗口，导致在与其他 TCP 流竞争时处于劣势，影响网络公平性

另外，New Reno TCP 拥塞控制算法并没有考虑网络拥塞的原因，它仅仅根据丢包情况来进行拥塞控制。因此，当网络出现拥塞时，可能并非由于网络带宽不足导致的，而是由于其他原因，例如路由器故障、网络拓扑变化等引起的。这种情况下，New Reno 算法可能会错误地减小拥塞窗口，从而影响网络性能。

此外，New Reno 算法也不能很好地处理拥塞窗口抖动的情况。当网络状况发生波动时，拥塞窗口可能会出现抖动，从而导致发送速率不稳定。这种情况下，New Reno 算法可能会出现过度反应或不足反应的情况，从而影响网络性能。

最后，New Reno 算法也存在一些安全问题。由于它仅根据丢包情况来进行拥塞控制，攻击者可以通过有意制造丢包来干扰网络性能，例如通过发送大量伪造的重复 ACK 来欺骗发送方进行快速恢复操作，从而导致网络拥塞。为了解决这个问题，一些改进的 TCP 拥塞控制算法引入了基于 ECN（Explicit Congestion Notification）的拥塞控制机制，通过在 IP 包头中标记拥塞状态，从而更精确地进行拥塞控制和避免攻击。

综上所述，New Reno TCP 是一种较为成熟和平衡的 TCP 拥塞控制算法，它在多个分段丢失时仍能保持较好的性能，并且与其他 TCP 流具有较好的协调性。

6. 总结

本文分析了 New Reno TCP 拥塞控制算法的具体实现以及其在改进 Reno 算法基础上提高网络性能方面的优势。New Reno 算法通过精确地判断网络拥塞状态并相应调整拥塞窗口大小，可以有效地控制网络流量，提高网络吞吐量和稳定性、降低延迟。与其他主流的拥塞控制算法相比，New Reno 算法在应对多个拥塞数据包的情况下表现出较好的性能，并且可以在一定程度上提高网络效率和公平性。

然而，New Reno 算法在处理单个数据包丢失和高速网络环境时的性能也存在提高的空间。总体来说，New Reno TCP 是一种相对成熟和平衡的拥塞控制算法，能够在保证网络稳定的前提下提高效率。本文的研究表明，利用 New Reno TCP 可以有效地优化网络性能，提高网络的吞吐量和响应速度。

参 考 文 献

[1] 贾永库. 基于非线性自适应 RED 算法的网络拥塞控制研究[D].西北大学,2010.

[2] 牛磊,王峰,刘冬冬,郭博.TCP Reno 拥塞控制的改进算法[J].福建电脑,2014,30(02):7-9+94.

[3] 连云凯. 高速网络的 TCP 拥塞控制策略及改进[D].广西师范大学,2007.

[4] 肖文曙,尹月静,许志勇. 一种数据传输的拥塞控制方法及装置[P].广东省：CN102204182B,2013-09-11.

[5] 陈金超,谢东亮.无线网络 TCP 拥塞控制算法研究综述[J].软件,2015,36(01):82-87.

[6] 胡飞飞,李云,刘期烈.TCP-BM:一种适用于异构网络的 TCP 协议改进策略[J]. 电子技术应用,2010,36(04):115-118.DOI:10.16157/j.issn.0258-7998.2010.04.018.

[7] 王伟伟. ad hoc 网络环境下的 TCP 性能分析及改进[D].山东大学,2005.