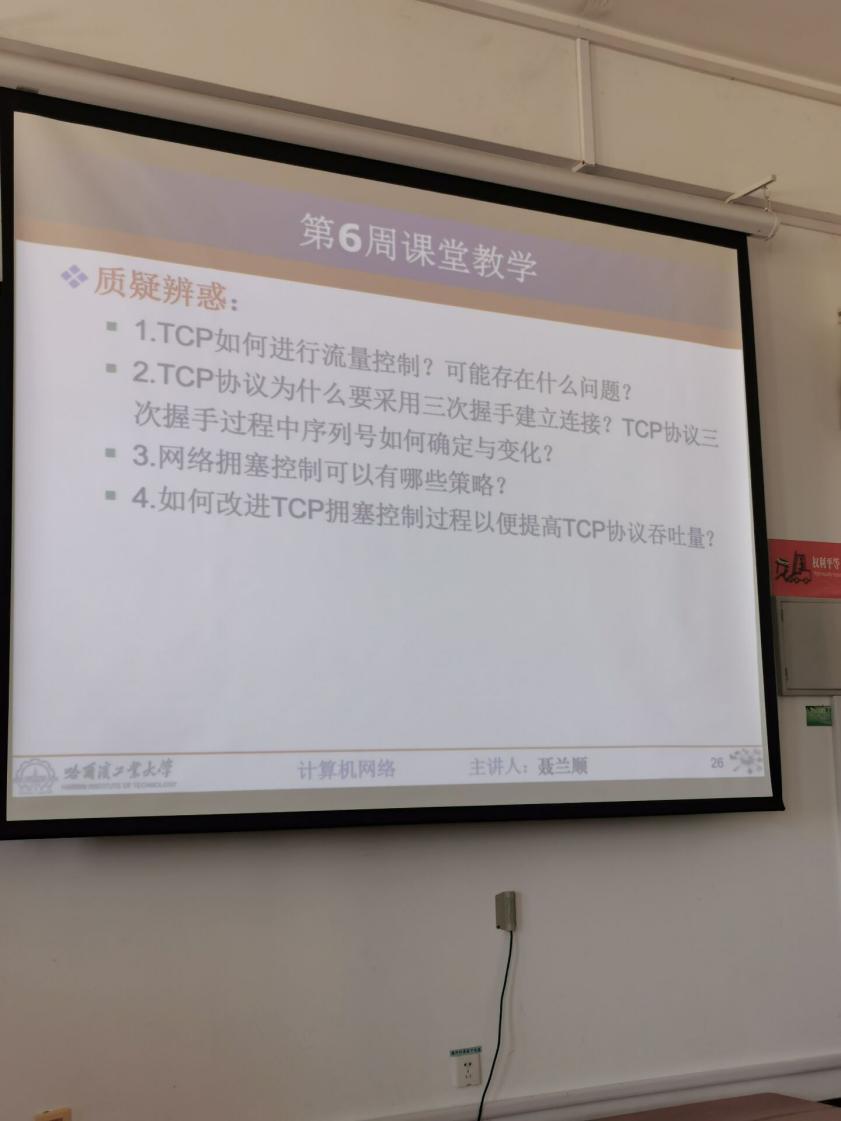
**计算机网络课堂讨论**

6班B组 ：范天泷 陈鋆 罗鹏豪 高天赋

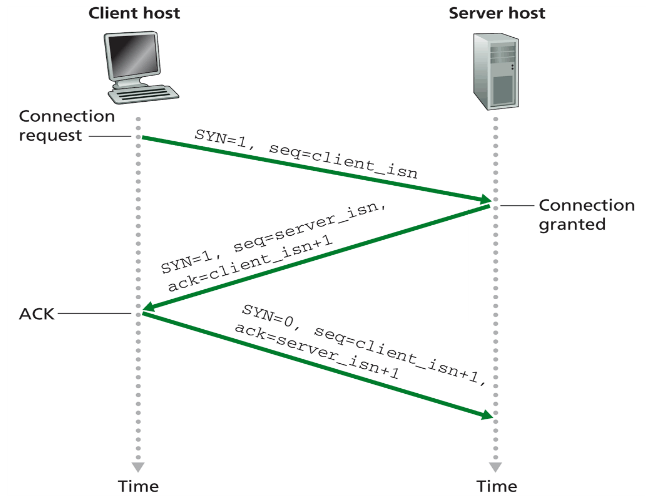


1. TCP报文段中设置了接收窗口字段，接收方可以在其中指示其愿意接收的字节数量。接收方在计算自己的缓存剩余量后通过接收窗口字段传达给发送方，发送方采用与拥塞控制相类似的手段（控制发送窗口大小）来进行流量控制。同时，为了避免rwnd = 0时发送方不再向接收方发送数据而导致连接阻塞中断，流量控制中补充了rwnd = 0发送方向接收方发送一字节报文段的控制方法。

对网络时延要求很高的流媒体很不友好；同时，这个补充的控制方法有一个问题就是每一次发送一字节报文段会导致网络效率很低（RTT较长的情况下等待时间长，中间传输时间可能接收方已经处理完了，但却白白等待很久）我想到的一个可能的解决办法是：接收到rwnd = 0 的消息后，等待一小段时间（参考之前计时器时间计算的方法）再发送一定大小的报文。

1. 我们在MOOC的TCP概述中学到过：TCP连接是全双工的，即同一连接中能够传输双向的数据流。通过三次握手建立连接的主要目的就是为了保证连接是全双工的，通过这样的三次握手，客户端与服务端建立起可靠的全双工连接，开始传输数据。三次握手中，1 、 2两次握手是客户端先向服务端发送一个连接请求，而后服务端发还一个连接请求，由此确保了服务端能够接收到客户端的信息并且做出正确应答。而2 、 3次握手则是同理确立了客户端能够接收到服务端的信息并且做出正确应答，通过两端的互相确立保证，确保了全双工连接的可靠性。

第一次握手时的初始序列号（ISN）是客户端随机产生的一个值（同时选择时为了避免某些安全性攻击，要有适当的随机策略选择）；第二次握手中服务端同理随机生成一个序列号初始值，同时ack\_seq=client\_isn+1；第三次握手时客户端发送的序列号为client\_isn+1,同时ack\_seq=server\_isn+1.



1. 从网络体系结构来看可以分成两块，一块是在网络边缘，即从端主机方面来对网络拥塞进行控制（从源头上进行治理），如我们已经学过的TCP的拥塞控制就是通过控制发送窗口大小（发送量，类比交通拥塞的管理）、接收方乱序缓存策略也可以说是避免网络拥塞的一种方法（但不是主动控制）、之前学过的代理服务器也能够做到。而另一块则是从在网络核心中，从路由器等网络核心结点来对网络拥塞进行控制：例如通过（由于还没学到这一部分，所以本处是我自己所构想的，可能与现实机制不太相符）首先没法实时实现的（基础设施类，道路加宽加大）可以改变网络拥塞情况，如增大路由器内存、提高路由器嵌入式芯片的计算能力，其次是可以实时实现的（交通导流），如路由器的排队策略（信号灯）、实时路由算法（交警导流）等等。
2. 举几个现有的改进的TCP拥塞控制过程：

对快速恢复阶段的策略进行改进：TCP New Reno，在Reno的快速恢复中，一旦出现3次重复确认，TCP发送方会重发重复确认对应序列号的分段并设置定时器等待该重发分段包的分段确认包，当该分段确认包收到后，就立即退出快速恢复阶段，进入拥塞控制阶段，但如果某个导致重复确认的分段包到遇到重复确认期间所发送的分段包存在多个丢失的话，则这些丢失只能等待超时重发，并且导致拥塞窗口多次进入拥塞控制阶段而多次下降。而New Reno的快速恢复中，一旦出现3次重复确认，TCP发送方先记下3次重复确认时已发送但未确认的分段的最大序列号，然后重发重复确认对应序列号的分段包。如果只有该重复确认的分段丢失，则接收方接收该重发分段包后，会立即返回最大序列号的分段确认包，从而完成重发；但如果重复确认期间的发送包有多个丢失，接收方在接收该重发分段后，会返回非最大序列号的分段确认包，从而发送方继续保持重发这些丢失的分段，直到最大序列号的分段确认包的返回，才退出快速恢复阶段。避免了进入超时重发而导致的吞吐率下降。

改进拥塞的测量确认方法：TCP Westwood，不同于以往其他拥塞控制算法使用丢失来测量，其通过对确认包测量来确定一个“合适的发送速度”，并以此调整拥塞窗口和慢启动阈值，使链接尽可能地使用更多的带宽，提高吞吐率。同类的还有Google的TCP BBR。

对拥塞窗口进行改进：微软的Compound TCP，通过同时维护两个拥塞窗口，一个常规的AIMD窗口，以及一个基于延迟的窗口。最终实际使用的滑动窗口大小是这两个窗口的和。AIMD窗口与TCP Reno的增加方式相同。如果延迟小，基于延迟的窗口将迅速增加以提高网络的利用率。一旦经历了排队，延迟窗口将逐渐减小以补偿增加的AIMD窗口。这样的目的是保持两者的总和大致恒定，使算法估计带宽时延积的路径。具体来说，当检测到排队时，基于时延的窗口因估计的队列大小而减少，在对高带宽时延积网络中性能优异。