**目录：**

**交互式通信**

**延时确认**

**Nagle算法**

**流量控制与窗口管理**

**紧急机制**

**与窗口管理相关的攻击**

**总结**

### 交互式通信：

TCP流量研究表明，通常90%或更多的TCP报文段都包含大批量数据，如Web、文件、邮件等。其数据段通常较大（1500字节或更大）。

交互式TCP是指，需要在客户端与服务器之间传输用户输入信息，如按键、鼠标、短消息等动作。其数据段会比较小（几十字节）。

对于交互式TCP，若采用较小的报文段来承载，其每个交换分组中有效数据字节较少，需要好分很高的代价；若报文段较大，则会产生更大的延时，对延时敏感的应用（如在线游戏、协同工具等）造成影响。

因此需要权衡相关因素，找到折中的办法。

每个交互按键通常都会生成一个单独的数据报。每个输入都会生成4个TCP报文段：客户端的输入、服务器对输入的确认、服务器生成的回显、客户端对回显的确认。通常第2和第3段可以合并。

### 延时确认：

很多情况下，TCP并不对每个到来的数据报都返回ACK，累积确认可以允许TCP延时一段时间发送ACK，以便将ACK与同方向上需要传的数据结合发送。这种捎带传输的方法经常用于批量数据传输。

显然，TCP不能任意时长地延迟ACK，否则会引起不必要的重传。该时延应该小于500ms，实践中时延最大取200ms。

采用延时ACK的方法会减少ACK的传输数目，可以一定程度的减轻网络负载。

### Nagle算法：

在小数据报传输中，如交互式应用，这些小包会造成相当高的网络传输代价（数据部分过小），会加重拥塞严重影响网络性能。

因此需要减少小包的发送。

Nagle算法规定，当一个TCP连接中由在传数据（存在未收到ACK的数据），小的报文段（长度小于SMSS）就不能被发送，直到所有ACK都接收到。并且，TCP需要将小报文段整合到一个报文段中发送（重新组包）。

Nagel算法中，传输的包数目更少长度更大，但传输时延也更长。

#### 延时ACK与Nagle算法结合：

在此情况下，在收到ACK前，新的小数据包无法被发送，导致形成一个短暂的死锁，持续到延时ACK计时器结束。在死锁期间整个连接处于空闲状态，使性能变差。

因此在某些情况下并不适合使用Nagle算法，典型的包括要求时延尽量小的应用（如远程控制，多人网络游戏等）。此时，需要禁用Nagle算法。

### 流量控制与窗口管理：

每个TCP报文段（除了建立之初的包交换）都包含一个有效的序列号、一个ACK号以及一个窗口大小字段。

窗口大小字段表明接收端可用缓存空间的大小，以字节为单位。该字段长度为16位，使用窗口缩放选项可以设置大于65535的值。

接收端会为即将到来的数据预留存储空间，当程序忙于其它操作时，预留存储空间会越来越小，导致窗口大小减小。

#### 滑动窗口：

每个TCP活动连接的两端都维护一个发送窗口结构和一个接收窗口结构。TCP以字节（而非包）来维护其窗口结构。

**发送窗口：**

1. 记录了已确认、在传（未确认）以及（能传但）还未传的数据的序列号。
2. 窗口大小由返回的ACK中的窗口大小字段控制。
3. 在窗口左边界与有边界之间的数据才可被发送。窗口左边界不能左移，因为它控制的是已确认的ACK号。
4. 当左右边界相等时称之为零窗口，此时发送端不能再发送新数据。此时发送端开始探测对方窗口，伺机增大窗口。

**接收窗口：**

1. 记录了已接受并确认的数据，以及它能接收的最大序列号。
2. 该窗口可以保证接收数据的正确性。避免存储重复的数据以及不应接收的数据。
3. 到达序列号小于左窗口边界的，则认为是重复数据而丢弃；到达序列号大于右边界的则超出处理范围，也被丢弃。
4. 由于TCP的累计确认机制，只有当到达数据序列号等于左边时，数据才不会被丢弃，窗口才能向前滑动。

但可以通过使用SACK选项来处理，此时处于窗口内的其它报文段也可以被接收确认。窗口滑动规则仍与前者保持一致。

#### 零窗口与TCP持续计时器：

TCP是通过接收端的通告窗口大小来实现流量控制的。当窗口值变为0时，不可继续发送。当接收端重新获得可用空间时，会给发送端传输一个窗口更新。此报文通常为纯ACK，不能保证其可靠传输。

若包含窗口更新的ACK丢失，通信双方就会一直处于等待状态。为了防止这种死锁的发生，发送端会在出现零窗口时，采用一个持续计时器，间歇性的查询接收端的窗口是否已增长，强制接收端返回ACK。

RFC1122建议在一个RTO后发送第一个窗口探测（即计时器初始值为RTO），随后以指数时间间隔发送（类似于Karn算法第二部分）。

窗口探测采用可靠传输，但也可能放弃执行重传操作，因为TCP不会停止发送窗口探测。

#### 糊涂窗口综合征（SWS）：

当通告窗口较小时，发送端会立即发送数据填满该窗口，导致连接中出现大量小数据包，这种现象被称为糊涂窗口综合征。

为了避免SWS的出现：

1. 接收端：不应通告小的窗口值

通告窗口大小为0，直到窗口增至一个全长报文段（MSS）或者接收端缓存空间的一半（取二者中较小的）。

1. 发送端：不应发送小的报文段，由Nagle算法控制

满足一下条件之一才能发送报文：

a.全长报文段

b.数据段长度>=通告过的最大窗口值的一般

c.没有在传数据；或禁用了Nagle算法

#### 大容量缓存与自动调优：

在多数情况下，上层应用指定的缓存大小会被忽略，而是由操作系统来指定一个较大的固定值或动态变化的固定值。

### 紧急机制：

当接收到URG置位的数据报时，会进入称为紧急模式的特殊状态。直到应用停止紧急数据写操作，并且所有序列号大于紧急指针的数据都已确认。

大量的RFC文档中对紧急指针的阐述都存在语义上的模糊和二义性。实际紧急指针也几乎不会使用。

### 与窗口管理相关的攻击：

TCP窗口管理可能收到多种攻击，主要形式为资源耗尽。

1. 通告窗口会使得TCP传输减慢，会更长时间占用资源。这一点被用于针对传输性能较差的网络攻击（蠕虫）。
2. 针对已知的持续计时器的缺陷，采用客户端多SYN cookies技术。导致资源耗尽。

### 总结：

1. 交互式数据传输的报文段通常小于SMSS。接收方接收到这些分组时可能会采取延时确认的方法，希望将这些ACK与需要发送给对方的数据一起捎带传输。可以减少传输报文段的数据，但也会引入额外的延时。
2. 对于RTT较大的连接，通常使用Nagle算法来减少较小报文段数目。
3. TCP通过在其发送的每个ACK中包含一个窗口通告来实现流量控制。
4. 通告窗口值可能为0，表明接收端缓存已满。这是发送端停止发送，并以一定间隔不断地发送窗口探测，发送间隔类似于超时重传。直到收到ACK表明窗口变大，或收到接收端主动发送地窗口通告表明有可用的缓存空间。这种以一定间隔连续发送的行为可能被用于资源耗尽攻击。
5. 当通告窗口较小时，发送端会立即发送数据填满该窗口，导致连接中出现大量小数据包，这种现象被称为糊涂窗口综合征。发送端：避免发送小数据包。接收端：避免通告小窗口。
6. 接收窗口大小受限于其缓存大小。较新的操作系统会忽略上层应用指定的缓存大小，采用自动调优的方法高效的自动分配缓存大小。