# 网络编程-第11天

## 1 内容回顾

数据的通信过程: 就是发送之前层层打包, 对方收到之后再层层解包.

7层模型和4层模型:

协议格式: TCP和UDP

UDP: 端口

TCP: 端口, 有序号和确认序号, 目的是使数据有序且不丢失.

IP: 32位IP, 网络字节序, TTL(下一跳), TTL的目的是防止网络阻塞.

socket概念: 一个文件描述符, 2个缓冲区, 2个队列

服务端的编码流程:

创建socket

将lfd与IP和端口进行绑定

侦听listen

将socket有主动态变为被动态

调用accept函数阻塞等待新的连接到来

while循环read 和 write

close文件描述符lfd和cfd

客户端的编码流程:

创建socket

调用connect连接远端的服务器

while循环 read 和 write

close文件描述符cfd

## 2 学习目标

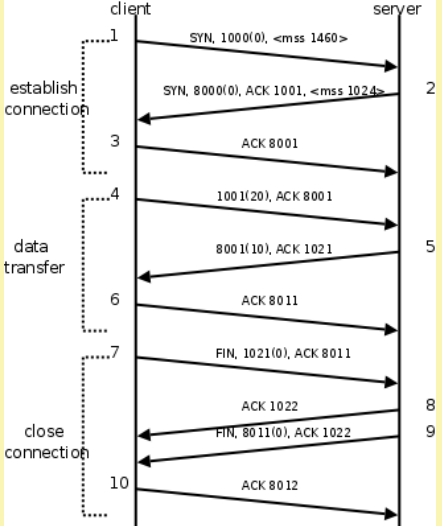
* 掌握三次握手建立连接过程
* 掌握四次握手关闭连接的过程
* 掌握滑动窗口的概念
* 掌握错误处理函数封装
* 实现多进程并发服务器
* 实现多线程并发服务器

## 3 三次握手和四次挥手

思考: 为什么TCP是面向连接的安全可靠的传输????

TCP是面向连接的安全的数据传输, 在客户端与服务端建立建立的时候要经过三次握手的过程, 在客户端与服务端断开连接的时候要经历四次挥手的过程, 下图是客户端与服务端三次握手建立连接, 数据传输和断开连接四次挥手的全过程.

TCP时序:



说明讲义中图的含义.

SYN: 表示请求, ACK:表示确认

服务端发送的SYN和客户端发送的SYN本身也会占1位.

单独讲解三次握手过程, 以图解形式说明.

上图中ACK表示确认序号, 确认序号的值是对方发送的序号值+数据的长度, 特别注意的是SYN和FIN本身也会占用一位.

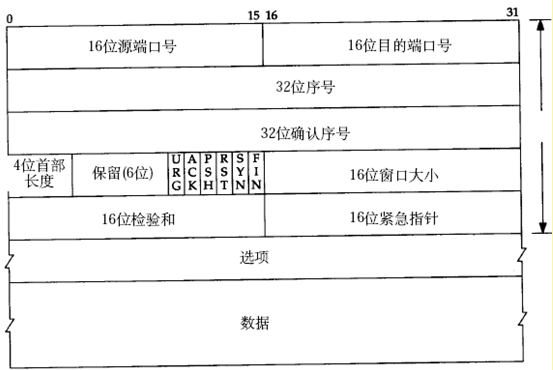
注: SYS----->synchronous

ACK----->acknowledgement

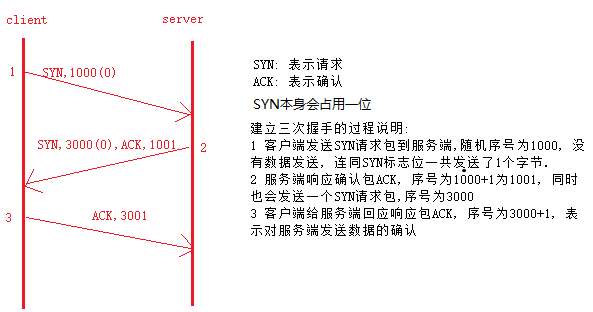
FIN------>finish

三次握手和四次挥手的过程都是在内核实现的.

下图是TCP数据报格式

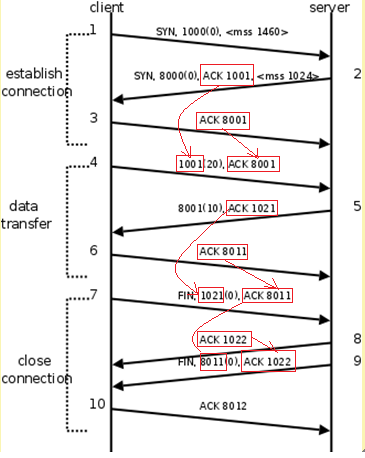


窗口大小: 指的是缓冲区大小



通信的时候不再需要SYN标识位了, 只有在请求连接的时候需要SYN标识位.

传输数据的时候的随机序号seq就是最近一次对方发送给自己的ACK的随机序号值, 而发给对方的ACK就是上次刚刚发给对方的ACK的值.



图中发送的ACK确认包表示给对方发送数据的一个确认, 表示你发送的数据我都收到了, 同时告诉对方下次发送该序号开始的数据.

由于每次发送数据都会收到对方发来的确认包, 所以可以确认对方是否收到了, 若没有收到对方发来的确认包, 则会进行重发.

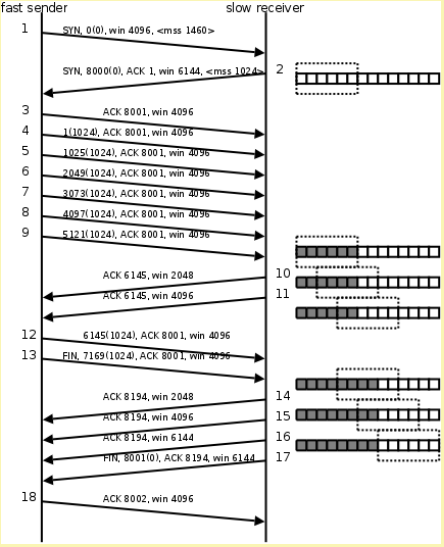
mss: 最大报文长度, 告诉对方我这边最多一次能收多少, 你不能超过这个长度.

win: 表示告诉对方我这边缓存大小最大是多少.

## 4 滑动窗口

主要作用: 滑动窗口主要是进行流量控制的.

见下图:如果发送端发送的速度较快，接收端接收到数据后处理的速度较慢，而接收缓冲区的大小是固定的，就会导致接收缓冲区满而丢失数据。TCP协议通过“滑动窗口（Sliding Window）”机制解决这一问题。



详细说明参考讲义

图中win表示告诉对方我这边缓冲区大小是多少, mss表示告诉对方我这边最多一次可以接收多少数据, 你最好不要超过这个长度.

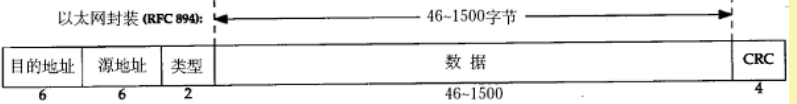
在客户端给服务端发包的时候, 不一定是非要等到服务端返回响应包, 由于客户端知道服务端的窗口大小, 所以可以持续多次发送, 当发送数据达到对方窗口大小了就不再发送, 需要等到对方进行处理, 对方处理之后可继续发送.

mss和MTU

MTU: 最大传输单元

MTU:通信术语最大传输单元（Maximum Transmission Unit，MTU）

是指一种通信协议的某一层上面所能通过的最大数据包大小（以字节为 单位). 最大传输单元这个参数通常与通信接口有关（网络接口卡、串 口等), 这个值如果设置为太大会导致丢包重传的时候重传的数据量较大, 图中的最大值是1500, 其实是一个经验值.



mss: 最大报文长度, 只是在建立连接的时候, 告诉对方我最大能够接收多少 数据, 在数据通信的过程中就没有mss了.

## 5 函数封装思想

函数封装的思想-处理异常情况

结合man-page和errno进行封装.

在封装的时候起名可以把第一个函数名的字母大写, 如socket可以封装成Socket, 这样可以按shift+k进行搜索, shift+k搜索函数说明的时候不区分大小写, 使用man page也可以查看, man page对大小写不区分.

像accept，read这样的能够引起阻塞的函数，若被信号打断，由于信号的优先级较高, 会优先处理信号, 信号处理完成后，会使accept或者read解除阻塞, 然后返回, 此时返回值为 -1，设置errno=EINTR;

errno=ECONNABORTED表示连接被打断,异常.

errno宏:

在/usr/include/asm-generic/errno.h文件中包含了errno所有的宏和对应的错误描述信息.

粘包的概念

粘包: 多次数据发送, 收尾相连, 接收端接收的时候不能正确区分第一次发 送多少, 第二次发送多少.

粘包问题分析和解决??

方案1: 包头+数据

如4位的数据长度+数据 -----------> 00101234567890

其中0010表示数据长度, 1234567890表示10个字节长度的数据.

另外, 发送端和接收端可以协商更为复杂的报文结构, 这个报文结 构就相当于双方约定的一个协议.

方案2: 添加结尾标记.

如结尾最后一个字符为\n \$等.

方案3: 数据包定长  
 如发送方和接收方约定, 每次只发送128个字节的内容, 接收方接收定 长128个字节就可以了.

wrap.c代码解读和分析.

要求能看懂代码, 会使用即可.

## 6 高并发服务器

如何支持多个客户端---支持多并发的服务器

由于accept和read函数都会阻塞, 如当read的时候, 不能调用accept接受新的连接, 当accept阻塞等待的时候不能read读数据.

第一种方案: 使用多进程, 可以让父进程接受新连接, 让子进程处理与客户端通信

思路: 让父进程accept接受新连接, 然后fork子进程, 让子进程处理通信, 子进程处理完成后退出, 父进程使用SIGCHLD信号回收子进程.

代码实现:

第二种方案: 使用多线程, 让主线程接受新连接, 让子线程处理与客户端通信; 使用多线程要将线程设置为分离属性, 让线程在退出之后自己回收资源.

思考: 如何不使用多进程或者多线程完成多个客户端的连接请求

可以将accept和read函数设置为非阻塞, 调用fcntl函数可以将文件描述符设置为非阻塞, 让后再while循环中忙轮询.

预习: TCP状态转换, 高并发服务器select poll epoll