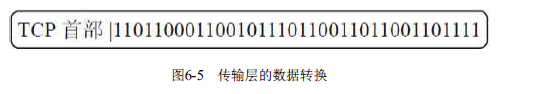
ISO七层网络模型（协议）

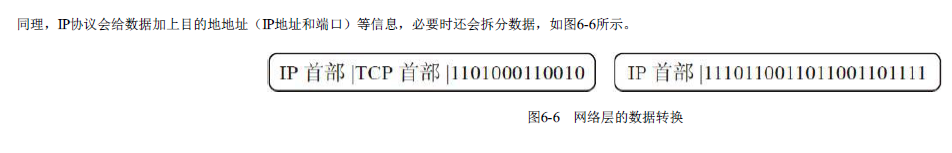
1. 应用层（应用程序提供服务）
2. 表示层（格式化数据，为应用程序提供通用接口）
3. 会话层（在两个节点间建立端连接）
4. 传输层（面向连接或无连接的数据传递， TCP和UDP属于传输层协议）
5. 网络层（通过寻址来建立两个节点间的连接，IP协议属于网络层协议）
6. 数据链路层（将数据分帧，添加校验机制，处理流控制）
7. 物理层（原始比特流的传输）

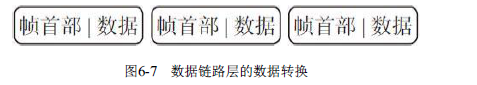
应用层，表示层，会话层统称为应用层

传输层，网络层，数据链路层，物理层为底层，由操作系统提供

应用层将要发送的数据转成二进制传递给传输层；传输层在数据前面添加传输首部（每条消息有唯一的编号）；网络层在传输数据前面加上IP首部(IP地址和端口)；数据链路层将比特流组合成帧，以帧为单位进行传送，对端程序只要检测最后一位，能初步判定数据是否正确发送，如果不正确则要求重发；物理层是指数据通过物理介质（电缆，光纤）进行传输。







TCP协议：

三次握手目的：同步连接双方的序列号和确认号，并交换TCP窗口大小的信息。

数据传输过程：发送方发送一个数据，并不能确保接收方一定能收到数据，于是发送方等待接收方的回应，如果太长时间没有收到回应，发送方会重新发送数据。

四次挥手（目的是断开连接）

第一次挥手：主机1（可以是客户端也可以是服务器）向主机2发送一个终止信号（FIN），此时，主机1进入FIN\_WAIT\_1状态，它没有需要发送的数据，只需要等待主机2的回应。

第二次挥手：主机2收到了主机1发送的终止信号（FIN），向主机1回应一个ACK。收到ACK的主机1进入FIN\_WAIT\_2状态。

第三次挥手：在主机2把所有数据发送完毕后，主机2向主机1发送终止信号（FIN），请求关闭连接。

第四次挥手：主机1收到主机2发送的终止信号（FIN），向主机2回应ACK。然后主机1进入TIME\_WAIT状态（等待一段时间，以便处理主机2的重发数据）。主机2收到主机1的回应后，关闭连接。至

此，TCP的四次挥手便完成了，主机1和主机2都关闭了连接。

常用协议：



Socket 看作不同主机间的进程双向通信的端点

服务端新建一个套接字，绑定指定IP和端口，监听客户端连接

客户端调用Connect连接服务器，服务器通过Accept接收客户端连接，此过程，操作系统会进行三次握手。

客户端和服务器通过write和read发送和接收数据，操作系统会完成TCP数据的确认，重发等步骤。

通过Close关闭连接，操作系统会进行四次挥手

每台计算机可以分配0到65535共65536个端口。其中0到1023号端口称为众所周知的端口号，它们被分配给一些固定的服务，比如80端口分配给

WWW服务，21端口分配给FTP服务。我们在浏览器中输入一个网址（网址通过域名系统转换为IP地址）时不必指定端口号，因为在默认情况下WWW服务的端口是80。

服务端要处理多个客户端消息，它需要用一个数组来维护所有客户端的连接。每个客户端都有自己的Socket和缓冲区。

防止sql注入

所谓sql注入是通过输入请求，把sql命令插入到sql语句中，以达到欺骗服务器执行恶意sql命令。

解决方案：在拼接sql语句前，对用户输入的字符串进行安全性检测，使用正则表达式判断，把-;,\/()[]}{@\*!’特殊符号判定为不安全字符串。

粘包和分包：

TCP协议本身的机制，客户端和服务器会维持一个连接发送数据；

粘包： 如果发送的网络数据包太小，TCP会合并较小的数据包再发送，接收端便无法区分哪些数据是发送端自己分开的。

分包：如果发送的数据包太大，TCP可能会将它拆分成多个包发送，接收端一次接收可能只收到一部分数据

解决方案：在每个数据包前面加上长度字节，每次收到数据后，先读取长度字节，如果缓冲区的数据长度大于要提取的字节数，则取出相应的字节，否则等待下一次数据接收。

正常情况，断开TCP连接会经历四次挥手。如果客户端电脑死机，或者网线被拔出，四次挥手则不能完成。服务器在同一时间能接入的客户端数量是有限的，如果出现太多的死连接，新连接便无法连入。

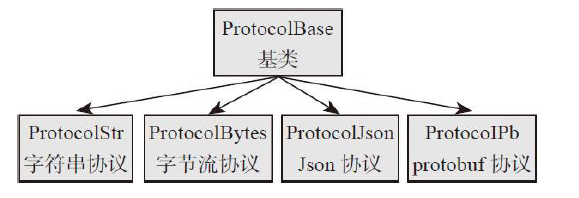
解决方案：引入心跳机制，心跳机制规定客户端每隔一段时间要给服务端发送一个特定信号，服务器会记录客户端最后一次发送心跳信号的时间，如果相隔太久，便认为客户端连接已经断开，于是主动断开连接。在服务端开启一个每秒执行的定时器，遍历所有连接，并判断连接的心跳时间

协议：

一套通用的服务端框架，要支持不同游戏使用的各种协议格式。所有协议都从字节流中读取消息，又把某种形式的消息转换为字节流发送出去。只要定义好解码编码的接口，便能够支持多种协议。

把一次发送的完整数据称为消息，一条消息的前4个字节用于粘包分包处理消息长度，随后才是消息内容（协议内容）。协议类型是基于

ProtocolBase的类，每个类都有自己的编码和解码方法，编码解码的内容是消息中除去消息长度的字节流。



通用的接口包括解码（Decode）、编码（Encode）、获取协议名称（GetName）和用于打印输出的描述（GetDesc）。

字符串协议：优点简单直观。字符串协议形式是”参数1（协议名），参数2，参数3”；缺点是客户端只要发送一段含有逗号的字符串便会引起混淆

字节流协议是一种最基本的协议。它把所有参数放入byte[]结构中，客户端和服务端按照约定的数据类型和顺序解析各个参数。

消息分发：应用可能有成百上千条协议，如果每次新增协议都添加if..else ..语句并处理会比较麻烦，而且容易造成代码混乱。

消息分发包含事件分发和协议分发，事件分发指在某种情况下会发生的事情，比如玩家上线和玩家下线时处理的事件。协议分发指服务端收到协议时，用什么方法去处理它。

规定在handleConnMsg和handlePlayerMsg中使用“Msg+协议名”来处理对应的协议。服务端使用反射机制调用实现这一功能

