

**1.**消息头固定字节数 (即**消息头长度**不可变)

**2.**消息头至少拥有两个字段(**消息类型**和**消息体长度**(或整个消息长度,即**消息头长度固定**值))

**3.多种不同类型**的消息**共用同一种格式的消息头** (即消息体类型不同,即**不同类型的消息头序列化之后字节是同一个固定常数值**)

sturct MessaHeader{

MessaHeader(int a,int b):type(a),body\_lenght(b){}

int type;//消息类型

int body\_lenght; //消息体长度

}

MessaHeader loginheader(login,100);

//序列化消息头 无需memcpy

char\* p=(char\*)&loginheader, 长度sizeof(MessaHeader )

//反序列化

MessaHeader\* header=(MessaHeader\*) p

服务端要使用一个结构维护众多客户端连接的socket句柄,并为每一个客户端socket分配一个自定义的用户数据(自定义用户数据结构),这是因为当socket->readAll()读取到的长度不足以构成一条消息时，要往这个用户数据上保存，等下一次socket->readAll()时把用户数据和socket->readAll()合并上去,并清空用户数据

例如std::map<socket,userdata> maps;

使用iocp或epoll模型(第三方跨平台网络库 libuv libhv asio)会有一种更高效的结构可以将每一个客户端socket和对应的用户数据关联起来，这个结构是操作系统内核管理的一种结构，至少比用map关联更高效

**序列化库**(protobuf) 跨语言跨平台

至少比json xml这种方式跨语言跨平台 要好