结构体：

typedef struct **icmp\_packet** {} icmp\_packet, \* picmp\_packet;

typedef struct **big\_packet** {} big\_packet, \* pbig\_packet;

typedef struct **icmp\_frag** {} icmp\_frag;

函数：

icmp\_frag\* **create\_icmp\_frag**()

bool **insert\_icmp\_frag**(icmp\_frag\* head, icmp\_packet packet)

：把信息填入分片，然后按序列号从小到大插入链表

bool **ressembly\_icmp\_frag**(icmp\_frag\* head, big\_packet\* ppacket)

：重新组装分片，只需将payload部分重组

unsigned short **calc\_checksum**(unsigned short\* data, int len)

int **my\_hash**(unsigned short ip\_id, unsigned short icmp\_identifier)

void **prepare\_headers**(ip\_hdr\* ipHdr, unsigned short\* pIpID)

bool **recv\_icmp\_packet**(int sock\_fd, icmp\_packet\* icmpPacket)

：主要通过recvfrom函数获取数据

bool **send\_icmp\_packet**(int sock\_fd, icmp\_packet\* icmpPacket)

int main(){

互斥锁；

创建线程；

重复或恰巧ip\_id和icmp\_id都冲突的握手包，直接丢弃，客户端检测超时重新生成ip\_id和icmp\_id；

不然就扩展链表解决hash碰撞；

发送第二次握手包；

收到第三次握手包；

发送对第三次握手包的确认；

收到普通数据包或文件数据包

}

void\* data\_receive\_func(void\* ptr)

：

客户端构造数据包（准备数据包ip头、icmp头以及有效负载部分）--发送第一次握手包--->接收第二次握手包--->第三次握手-->收到确认开始数据传输

服务器接收第一次握手包创建子线程-->响应握手包发送（2）-->响应握手包（4）--->接收数据包

**Client** **Server**

If not

**超时重传**

**相应子线程**

**处理分片**

**保存相关字段**

**子线程解析数据包**

**发送数据**

**创建子线程**

④

③

②

①

recv\_icmp\_packet

send\_icmp\_packet

RecvIcmpPacket

RecvIcmpPacket()

SendIcmpPacket

RecvIcmpPacket()

**构造数据包**

CreateICMPSocket()

PrepareHeaders()

obtainWorkingIP()

**四次握手**

是

发送结束，断开隐蔽信道

否

数据发送是否完成

接收服务端返回的ICMP数据包

发送ICMP数据包给服务端

将数据写入ICMP头部选项字段

构造ICMP包头，填充ICMP结构体

设置ICMP包目的地址

加载winsock库

创建ICMP原始套接字