测量船中心机与 USB 遥控终端 TCP 连接故障分析及改进方法

柏文良　施宇星　朱加勇
（中国卫星海上测控部）

摘　要　根据 TCP/IP 协议的技术特点，对测量船中心机与 USB 遥控终端 TCP 连接故障的产生原理进行分析，并提出相应的改进方法和采用 UDP 协议构建该网络数据链路的改进构想。

关键词　TCP/IP 协议　TCP　UDP　套接字（Socket）　遥控中心机　USB 遥控终端
中图分类号　V556　**文献标识码**　A　　**文章编号**　1674-5825（2009）02-0049-07

1 引言

微软的 TCP/IP 协议是在物理网上的一整套数据通信协议，它包括传输层的两个协议：TCP 协议和 UDP 协议。由于因特网的迅速流行，越来越多的应用程序具备了在网上与其它程序通信的能力。从 WIN95 开始微软把网络功能融进了它的操作系统，使得应用程序网络通信能力更为普及。因此，TCP/IP 协议也就成为网络应用程序基于的首选协议，测量船中心机系统实时应用软件（以下简称 TCAS）与 USB 遥控终端软件的通信，就采用了面向连接的 TCP 协议。

在历次的飞船海上测控任务中，TCAS 与 USB 遥控终端程序采用 TCP 协议构成可靠的网络数据链路，确保了远望号船正常向飞船发送各类遥控指令、数据注入等数据。但是 TCAS 与 USB 遥控终端的 TCP 连接也先后出现过 USB 遥控终端慢秒、错误数据导致中心机与 USB 遥控终端断开连接、中心机非正常重启后出现假连接等故障，本文就其中涉及软件的后两项问题进行分析，并提出相应的改进方法以及采用 UDP 协议的改进构想。

2 概　述

2.1 中心机与 USB 遥控终端的网络连接

测量船遥控中心机与 USB 遥控终端同处于测量船试验指挥局域网中，连接在遥测网交换机上。遥控中心机与 USB 遥控终端通过 TCP 协议建立网络连接，逻辑连接图如图 1 所示：

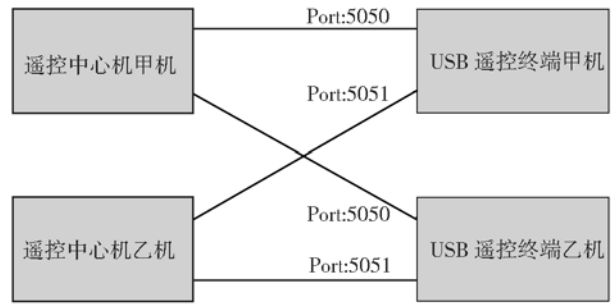

图 1　中心机与终端逻辑连接图

其中 USB 遥控终端的端口 5050 对应遥控中心机甲机的连接，端口 5051 对应遥控中心机乙机的连接。

2.2 中心机与 USB 遥控终端的 TCP 连接

中心机与 USB 遥控终端 TCP 网络通信的模式是典型的客户端/服务器模式（Client/Server model），该模式的建立基于以下两点：（1）非对等作用；（2）通信完全是异步的。客户端/服务器模式在操作过程中采取主动请求方式，其中 USB 遥控终端作为服务器端（Server），中心机处于客户端（Client）。

USB 遥控终端的服务器端（USB 遥控终端 RM-

来稿日期：2007-09-10
作者简介：柏文良（1979.04-），男，工程师，本科，主要从事测量船中心计算机工作。

CSP 进程)和中心机客户端(遥控中心机 TS 进程)建立和关闭过程:

2.2.1 服务器端

(1)USB 遥控终端启动进程 RMCSP，该进程建立两个 TCP 连接线程:RmcsRecThread5050()线程和 RmcsRecThread5051()线程,它们的作用是创建套接字并分别绑定 5050、5051 端口，接收并处理从端口 5050 和 5051 上传来的数据。

(2)创建 Socket:调用函数 socket(),建立流式套接字 Sock;

(3)绑定本机端口:调用函数 bind(),RmcsRecThread5050()线程和 RmcsRecThread5051()线程各调用此函数一次,绑定两个本机端口:5050 和 5051,分别对应接受遥控中心机甲机和乙机的连接;

(4)监听端口:listen();

(5)接受连接:accept(),此时服务器阻塞,等待客户端的连接请求;

(6)数据传输:连接成功后调用自定义发送数据函数 TCToRmcs ()和接收数据函数 RmcsRecData()进行数据传输;

(7)关闭 Socket:进程关闭。

2.2.2 客户端

(1)TCAS 支持软件创建 Socket:调用函数 socket(),建立流式套接字;

(2)建立连接:调用函数 connect(),向服务端发送连接请求;

(3)数据传输:连接成功后调用函数 send()和 recv()进行数据传输;

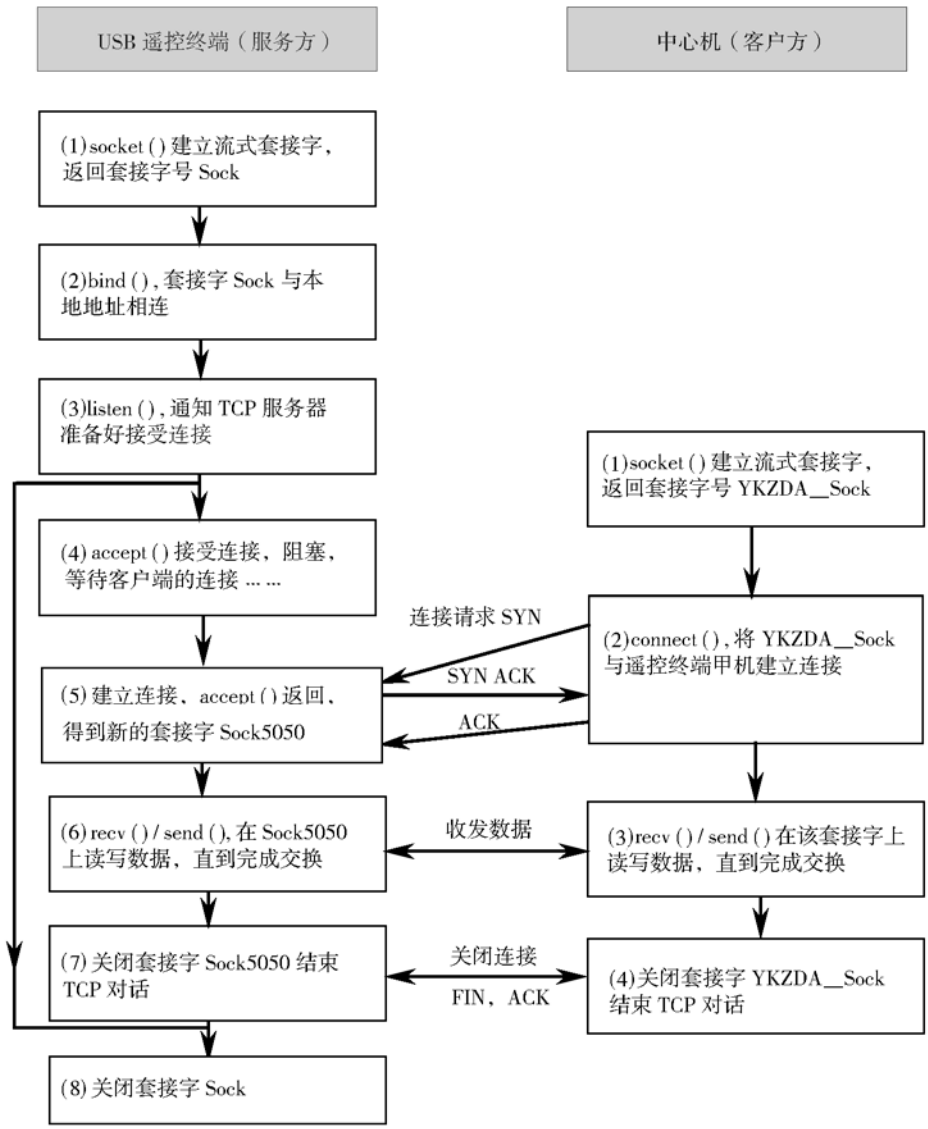

图 2　TCP 连接套接字调用时序关系图

(4)关闭 Socket:调用函数 closesocket()或 shutdown()。

USB 遥控终端的服务器端和中心机客户端 TCP 连接套接字调用时序关系如图 2 所示（以甲机互连为例)。

3 TCP 连接故障分析

3.1 错误数据导致遥控中心机与 USB 遥控终端断开连接故障分析

3.1.1 故障现象

该故障发生在测量船组织的飞船任务立体大回路演练过程中，演练采用测量船自行开发的中心接口模拟软件进行透明发令，发送轨控结束指令链时最后一条加密指令发出后,USB 遥控终端观察发现与中心机的 TCP 连接断开，但 TCAS 未判断出该连接已断开,并且 USB 遥控终端无法收到中心机发出的后续指令。

3.1.2 故障原因

通过检查发现，中心接口模拟软件发送的加密指令数据帧组装错误,导致中心机与 USB 遥控终端的 TCP 连接断开。

中心接口模拟软件中加密指令和数据注入共用的同一数据结构 S_SJZR，该数据结构中包括长度 L 和遥控帧序列字节数 N。其中数据注入的长度 L 是 528,因为 L 为一个字节表示,最大值为 255,因此结构中的 L 是模 256 后的值为 16,遥控帧序列字节数 N 为 516。加密指令 K133 正确的长度 L 为 44,遥控帧序列字节数 N 为 32。软件开发人员误将 K133 的长度 L 填成 16,但遥控帧序列字节数 N 为 32。这样的指令帧被发送到遥控终端后却导致了中心机与 USB 遥控终端的 TCP 连接断开。

3.1.3 故障分析

TCP 连接异常产生过程(以甲机互连为例)：

(1)中心机遥控甲机与 USB 遥控终端甲机建立 TCP 数据链路，时序调用关系如图 2 所示，用 netstat 命令在 USB 遥控终端甲机上可以查看到 TCP 状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.5:5050	129.23.22.33:1030	ESTABLISHED

其中 129.23.22.5:5050 代表 USB 遥控终端甲机本地的地址和端口号,129.23.22.33:1030 代表中心机遥控甲机的地址和端口号，状态 ESTABLISHED 表示连接已经建立好,是连接的正常状态。

(2)利用中心接口模拟器,发送一帧数据长度 L 为 16，但遥控帧序列字节数 N 为 32 的错误密指令 K133 数据给 USB 遥控终端甲机；

(3)USB 遥控终端甲机 RmcsRecThread5050()线程调用数据接收和处理 RmcsRecData ()(图 2 中第 6 步）函数从端口 5050 上接收到该错误数据,并在函数 RmcsRecData()中进行正确性判断(比对数据长度 L 和遥控帧序列字节数 N),RmcsRecData()判定数据长度 L 和遥控帧序列字节数 N 不匹配,返回数据错误标志 bRecStatus=DATA_ERROR；

(4)USB 遥控终端 RmcsRecThread5050()线程判断 bRecStatus=DATA_ERROR,即直接 break 跳出数据接收循环，进入上一层循环并调用显示函数 NetStatusDisplay(),显示 5050 OFF,即网络链路已断开,并重新调用 accept()函数等待新的连接请求(图 2 中第 4 步);由于采用了阻塞模式,函数 accept()必须等到操作完成后才返回，因此 RmcsRecThread5050()线程就一直在 accept()处等待新的 TCP 连接请求,不会走到第二层循环调用函数 RmcsRecData()接收从端口 5050 上发来的数据。在该过程中,USB 遥控终端不会向中心机发送链路断开的消息(因为开始建立的 TCP 链路只是不再接收数据,实际并未断开,使用 netstat 命令在 USB 遥控终端甲机上可以查看到其 TCP 状态未变)，从而导致 USB 遥控终端显示 TCP 数据链路已断开，而中心机实战软件判断 TCP 连接未断开的现象。RmcsRecThread5050()线程相关代码如下：

```
void RmcsRecThread5050()
{ BOOL   bRecStatus;    //判断数据正确标志
  BYTE   bRecData [MAX_RECEIVE_LENGTH]; //收数据缓冲区
  DWORD  dwDataLength=0;    //数据长度
  SOCKET  ServerSock;     //服务器端 Socket
  SOCKADDR_IN  Client_sin;  //地址
  int   iClient_sin_len = sizeof(Client_sin);
  BindLocal(ServerSock,TCPORT1);//绑定 5050 端口,Listen()
  while(TRUE) //第一层循环
```

```
{bC5050=FALSE; //用于标志网络链路是否连通
NetStatusDisplay（）; //显示网络链路断开:5050 OFF
memset （&Client_sin,0,iClient_sin_len）; //zero memory
Sock5050 = accept （ ServerSock,(struct sockaddr FAR *) &Client_sin,
(int FAR *) &iClient_sin_len ）; //accept 连接
 int iSetOpVal = 1;
setsockopt(Sock5050,SOL_SOCKET, SO_ KEEP-ALIVE,(char FAR *)&iSetOpVal,sizeof （int）);//设置 Socket 属性
bC5050 = TRUE;　　//网络链路已连通
NetStatusDisplay（）; //显示网络链路连通:5050 ON
while(TRUE) //第二层循环
{ memset(bRecData,0,MAX_RECEIVE_LENGTH); //zero memory
bRecStatus = RmcsRecData(Sock5050,bRecData, dwDataLength);//收数据
if(bDisplayOut==TRUE)
SendDisOutOri(bRecData, dwDataLength);//显示
if(bRecStatus==DATA_RIGHT) //数据正确,进行下一步处理
DataPreProcess(bRecData,lpRmcsData,lpSaveRm-csData,
dwDataLength);
else break; //否则 DATA_ERROR，跳出第二层循环
}}}
```

3.1.4 解决方法

（1）现有条件下的应急方案

在保持现有软件状态不变的条件下，如果出现中心机遥控软件收到中心错误的遥控类数据转发至遥控终端导致中心机与 USB 遥控终端的 TCP 连接断开的情况（例如密指令发令字节数填错），则中心机拆除与终端的连接，重新与终端建立 TCP 连接。

（2）更改 USB 遥控终端软件代码

以 RmcsRecThread5050（）线程为例，最简单的方法是将第二层 While 循环的第二个 if 语句中的“else break;”改为“else continue;”，使之在收到错误数据时不会跳出 While 循环。

（3）完善中心机遥控软件

从目前看来，只有错误的密指令数据能够骗过中心机遥控软件发送到 USB 遥控终端，导致中心机与 USB 遥控终端的 TCP 连接断开，因此，中心机遥控软件只要进行相应的完善，增加对密指令数据长度 L 和遥控帧序列字节数 N 的比判，使错误的密指令数据在中心机这里被拦截掉，不会发送到 USB 遥控终端，也就杜绝了 TCP 连接断开故障的发生。

3.2 中心机非正常重启后出现假连接故障分析

3.2.1 故障现象

中心机与 USB 遥控终端建立 TCP 连接后，中心机由于软件故障、死机等原因非正常重新启动后启动 TCAS 软件重新建立与 USB 遥控终端的连接，USB 遥控终端无法收到中心机发送的数据，也就是出现了假连接。

3.2.2 故障原因

当中心机非正常重启时，TCAS 无法向 USB 遥控终端发送连接断开消息，从而导致 USB 遥控终端不知道中心机客户端已经与它断开连接。而当中心机重启后再与 USB 遥控终端连接时，USB 遥控终端在保持原有 TCP 链路的同时又重新建立了一个 TCP 链路，从而导致数据收发不正常。

3.2.3 故障分析

（1）以甲机互连为例，中心机与 USB 遥控终端建立正常的 TCP 连接。

USB 遥控终端甲机的 TCP 连接状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.5:5050	129.23.22.33:1030	ESTABLISHED

中心机遥控甲机的 TCP 连接状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.33:1030	129.23.22.5:5050	ESTABLISHED

（2）中心机正常的拆除连接。

USB 遥控终端甲机的 TCP 连接状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.5:5050	129.23.22.33:1030	CLOSE_WAIT

注：CLOSE_WAIT 表示正在等待本地用户发来的断开请求。

中心机遥控甲机的 TCP 连接状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.33:1030	129.23.22.5:5050	FIN_WAIT_2

注：FIN_WAIT_2 表示正在等待从 USB 遥控终端发来的断开请求。

（3）如果中心机非正常重启（直接按下电源开关），查看 USB 遥控终端的 TCP 连接状态，发现未发生改变。

（4）重启后打开 TCAS 实战软件，建立与 USB 遥控终端甲机的 TCP 连接，此时，查看 USB 遥控终端的 TCP 连接状态为：

Proto	Local Address	Foreign Address	State
TCP	129.23.22.5:5050	129.23.22.33:1029	ESTABLISHED
TCP	129.23.22.5:5050	129.23.22.33:1030	ESTABLISHED

可以发现，USB 遥控终端甲机出现了两个与中心机遥控甲机的连接，一个是原先的端口 1030，一个是新端口 1029，而且状态都是 ESTABLISHED。通过查看 USB 遥控终端的软件可知，终端只建立了一个线程 RmcsRecThread5050()在端口 5050 上收发数据，因此，从远地端口 1029 上来的数据不会被终端收到，导致了假连接现象的发生。

3.2.4 解决方法

（1）现有条件下的应急方案

在保持现有软件状态不变的条件下，如果出现中心机由于软件故障、死机等原因非正常重新启动，则遥控终端软件也要重新启动，然后再建立 TCP 连接。

（2）更改 USB 遥控终端软件代码

还是以 5050 端口为例，更改 USB 遥控终端软件代码，使得每当服务器上有用户连接成功，服务器都会为其创建两个线程：接收线程（RecvData）和发送线程（SendData），并且接收线程在创建后处于可执行状态，而发送线程则阻塞，等待服务器将其唤醒。这两个线程都执行一个无限循环的过程，只有当通信出现异常或用户端关闭连接时，线程才被自身所结束。

4 采用 UDP 协议的改进构想

4.1 UDP 协议与 TCP 协议

4.1.1 两种协议的特点

用户数据报协议（UDP）是一个简单的面向数据报的传输层协议，提供给用户进程的是一种非连接、高效率但却“不可靠”的数据包通信方式。数据报套接字使用 UDP 协议。采用 UDP 协议的应用层协议有：Time 协议、域名服务协议（DNS）、简单文件传输协议（TFTP）等等。

而传输控制协议（TCP）是一种面向全双工字节流的传输层协议，提供给用户进程面向连接的“可靠”的数据传输方法。流式套接字使用 TCP 协议。采用 TCP 协议的应用层协议有：文件传输协议（FTP）、简单邮件传送协议（SMTP）、终端协议（TELNET）、超文本传输协议（HTTP）等。

4.1.2 两种协议的适用范围

UDP 协议的适用范围：在高可靠性、低延迟的局域网络（LAN）上运行很好，但在通信子网的服务质量（QOS）相对低下的 Internet 上，它就可能无法正常运行了。因此，基于 UDP 的应用程序必须自己提供可靠性保障，例如丢失数据报的重传、失序数据报的组合等等。虽然 UDP 协议不可靠，但是由于它传输数据的高效率，并且具有发送广播数据报的能力，使得它在实际网络应用中被广泛采用，例如交易型应用程序（一次交易过程往往只有一来一去两次数据报交换）。

TCP 协议的适用范围：TCP 协议的应用范围要比 UDP 协议广得多，特别是在 Internet 上，它也可以正常运行，提供可靠的服务。但是，相对于简单而高效率的 UDP 协议，TCP 协议则复杂很多，建立连接和撤销连接的开销比较高。TCP 协议适用于发送大批量数据并且需要让发出的数据按顺序无重复的到达目的地的应用程序，如大多数的 Internet 应用程序。

4.2 采用 UDP 协议的改进构想

4.2.1 可行性分析

中心机与 USB 遥控终端在测量船试验指挥网中通过遥测网交换机互相连接，远望号船试验指挥网的网络拓扑结构比较简单，属于标准的高可靠性、低延迟的局域网络（LAN）；同时，中心机与 USB 遥控终端的数据交换量很小，数据链路大部分时间都处于空闲状态；TCAS 与 USB 遥控终端应用程序都能够提供一套数据可靠性保障方法（发送方连续发送三帧相同数据，收数据方进行三取二比对，比对正确后返回正确应答，如果返回错误应答，则要求进行数据重传，如果未收到应答，则说明该数据链路不通），所有这些都符合采用 UDP 协议的特性。

事实上，在测量船试验指挥网中可以找到很多采用UDP 协议通信的例程，如中心接口模拟器与中心

机的网络接口、中心机与遥测终端的网络接口、185G 遥测遥控处理机与 185G 遥控终端的网络接口等等，它们运行完全正常。

综上所述，中心机与 USB 遥控终端的网络链路采用 UDP 协议可行。

3.2.2 可以采用的两种网络通信模式

(1)对等模式

由于 UDP 不用建立连接，所以中心机和 USB 遥控终端可以处在对等的状态进行通信，对等模式的 UDP 通信的步骤如下：

①建立一个数据报方式的套接字；

②绑定地址；

③进行数据传输；

④关闭套接字。

对等模式的 UDP 通信模型程序流程如图 3 所示：

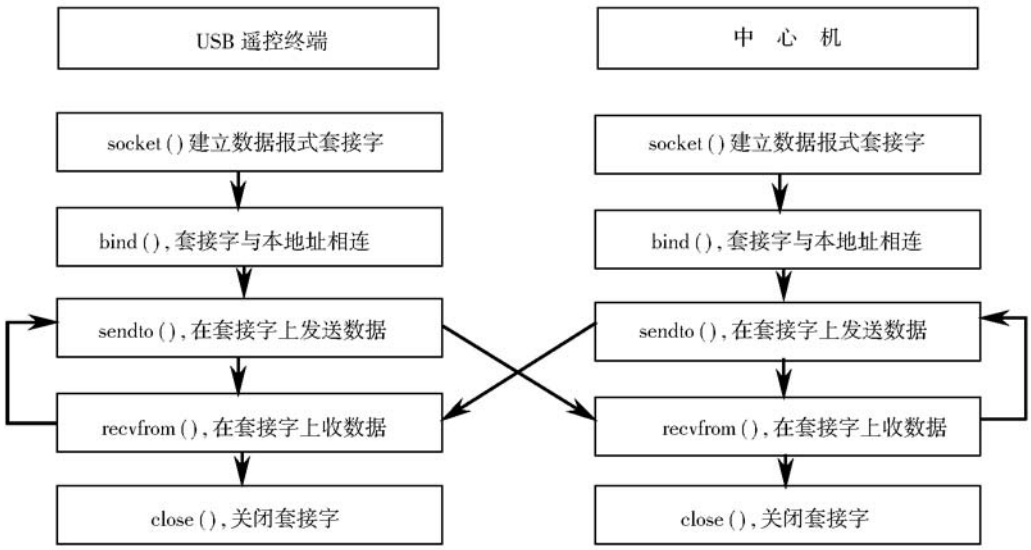

图 3　对等模式的 UDP 通信模型

(2)客户端—服务器模式

中心机和 USB 遥控终端也可以建立客户端—服务器模式的 UDP 通信，客户端—服务器模式的 UDP 通信的步骤如下：

服务器端

①建立一个数据报方式的套接字；

②绑定地址；

③阻塞，等待客户端数据报；

④处理服务请求，进行数据传输；

⑤关闭套接字。

客户端

①建立一个数据报方式的套接字；

②绑定地址；

③向服务器发送服务请求；

④进行数据传输；

⑤关闭套接字。

客户端—服务器模式的 UDP 通信模型程序流程如图 4 所示：

4.2.3 基于组地址多播的对等通信方案

组地址多播是指向多个目的地址发送数据，多播的组地址全部是 D 类地址，范围包括：224.0.0.0 到 239.255.255.255。能够接收发往一个特定多播数据的主机集合称为主机组，一个主机组可以跨越多个网络，主机组中成员可随时加入或离开主机组，主机组中对主机的数量没有限制，同时不属于某一主机组的主机也可以向该组发送信息。

在测量船全封闭式的试验指挥局域网中，采用组地址多播方式的优点显而易见：第一，不用建立复杂的连接，数据收发的效率很高。第二，同一系统的主机使用一样的组地址，便于管理。第三，不同组的主机也可以很方便的互发信息，并且同一主机组的主机都可以收到该信息，极大简化了应用程序。

因此在中心机与 USB 遥控终端网络通信方案的改进构想中，我们采用组地址多播方式，建立对等的 UDP 通信模式（程序流程见图 3）。中心机或者

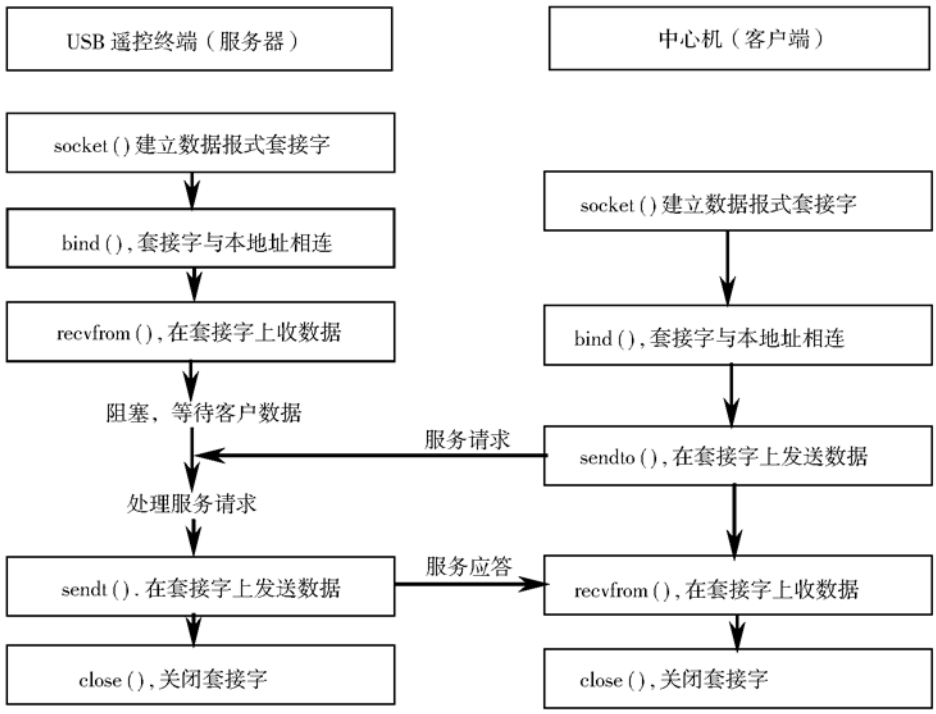

图 4　客户端–服务器模式的 UDP 通信模型

USB 遥控终端进程采用多线程技术，需要建立下列两个线程：接收数据线程（RecvDataThread）和发送数据线程（SendDataThread）。接收数据线程（RecvDataThread）负责创建数据报套接字并绑定相应的接收组地址，然后调用 recvfrom( )函数在相应的组上接收网络数据；发送数据线程（SendDataThread）绑定相应的发送组地址，然后调用 sendto( )函数在相应的组上发送网络数据。该改进构想需要中心机与 USB 遥控终端同时更改部分代码。

5 结束语

通过 TCP 连接故障的分析以及采用 UDP 改进构想的讨论，对 TCP 和 UDP 这两大传输层协议的特点有了一个大致的了解。然而对于“远望”号船试验指挥网各应用进程之间的网络通信，究竟是使用 TCP 协议还是 UDP 协议，确实是一个难以抉择的问题。事实上，只要应用程序足够完善，不论是采用 TCP 协议还是 UDP 协议都能够很好地工作。 ◇

The analysis and amelioration of the malfunction about TCP connection between the central computer and the telecontrol terminal of USB

BAI Wenliang　WANG Linna

（China Satellite Maritime Tracking and Control Department）

Abstract: Based on the characteristic of the TCP/IP protocol technology, this paper analyzes the principle of causing the malfunctions of connecting between the central computer and the telecontrol terminal of USB, and presents a relevant ameliorative method. An ameliorative thought that using UDP protocol to build that network data chain is also discussed.

Keywords: TCP/IP protocol; TCP; UDP; Socket; The central computer; The telecontrol terminal of USB