飞行器总体测控系统的网络通信

傅长冬　潘清　王勇
（国防科工委指挥技术学院 101407）

摘要：飞行器总体测控系统是基于网络通信的分布式测控系统。本文具体介绍总控系统的工作环境，实现网络通信的设计方案，着重介绍系统的进程关系及连接控制等关键技术。

关键词：网络通信　TCP/IP 协议　客户/服务器

一、概述

飞行器总体体控制系统是满足飞行器测试、控制要求的软件系统。工作环境中的各功能设备通过以太网(Ethernet)以总线形式相互连接(如图 1 所示)。整个系统在功能结构上表现为典型的客户/服务器结构。主测试计算机(MTP)上的控制处理系统为测试控制台(TCC)、遥测(TM)、遥控(TC)前端设备、以及各分系统专用测试设备(SCOE)上的各功能子系统提供对应的测试、处理、控制服务。它们之间通过网络通信相互传输信息，同步操作。整个系统是基于网络进程间通信和本地消息进程通信的分布式系统。

1.硬件环境

飞行器总控系统工作环境如图 1 所示。各前端设备和主测试计算机 MTP 之间通过以太网以总线形式相互连接，形成一个以 MTP 为服务器 C/S 结构的工作环境。主测试计算机的具体指标如下：N 个 SCOE 设备通过集线器(HUB)和总线相连，并且每个设备都配置 3C503 网卡、遥控、遥测，控制台计算机都为主频 100HZ 的 PC586。

2.软件环境

如图 1 所示，主测试计算机(MTP)的系统平台采用 SUN 公司的 solaris 2.4 for x86。测试控制台的系统平台可采用 SCO 的 system v version2.0 或 solaris 2.4 for X86。它们都采用 TCP/IP 协议。

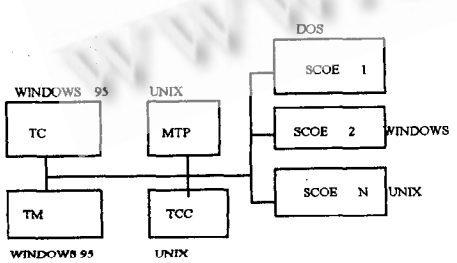

图 1　JB3 总控系统工作环境

遥控、遥测前端设备的系统平台采用 WINDOWS 95。它必须配置 TCP/IP 协议，同时需有如 borland c3.1，vc++4.0 等开发平台。

对于 SCOE，如果采用 DOS 环境，必须装配 IBM 的 TCP/IP for DOS 软件包。也需要 TC、BC 等开发平台。如果采用 WINDOWS 95 或 UNIX 系统平台，要求同上。

3.功能要求

飞行器总控系统是满足飞行器测试、控制要求的系统，是基于网络通信的分布式测控系统，它对网络通信提出了较高的要求：(1)不同系统环境能够相互通信；(2)数据能够实时、可靠的传输；(3)遵循飞行器的所有通信协议；(4)保证系统的可靠性、容错性；

二、网络通信功能的实现

1.通信协议

实现不同系统环境下的网络互连，要求所有的系统平台采用 TCP/IP 协议。TCP/IP 协议是不依赖硬件平台的网络协议，可提供面向连接的可靠的对等通信方式，它的网络通信模式主要是客户/服务器模式，系统实现时利用了传输控制协议(TCP)的服务。

2.连接方式

因为 MTP 的控制处理系统要和各前端设备的功能子系统交互数据，同步操作，因此它首先必须实现和各设备的连接。连接方式可分为集中连接方式和单独连接方式。集中连接方式就是单个服务进程等待多个连接请求。而单独连接方式是每一个连接有一个对应进程等待连接请求。实际中我们混合采用了单独连接和集中连接。对 N 个 SCOE 采用集中连接，而对 TC、TM、TCC 采用单独连接方式。集中连接方式可减少系统中进程总数目，简化处理这些请求的守护进程的编程。而单独连接方式有利于调试，而且开销较少。

3.服务方系统实现

(1)总体说明。服务方系统指的是主测试计算机的控制处理系统，它通过 UNIX 核心提供的面向连接协议服务和各前端设备进行网络间进程通信。各处理系统间通过消息和共享内存进行本地进程间的通信。整个测控系统形成基于网络通信分布式测控系统，服务方系统分为 TC、TM、TCC、SCOE 处理分系统，监控分系统，测试分系统，数据转发分系统等。TC、TM、TCC 处理分系统以单独连接方式和各自的前端功能子系统相互连接，传输数据。而 SCOE 处理系统采用集中方式和 N 个 SCOE 前端设备功能子系统相连，同步操作。监控、测试、转发分系统主要进行数据处理、系统监控、数据转发等工作。

(2)进程关系。TC、TM、TCC、SCOC 处理分系统无论集中

连接方式还是单独连接方式，都采用并发服务器模式，主进程等待对应的前端设备的连接请求，当连接请求到达时，fork()子进程，并将此子进程设置为新的进程组组头(setpgrp)，而成为新进程组组头的进程创建一定数目的子进程，它们通过消息或共享内存相互通信，共享数据来协同完成相应的服务功能。主进程继续循环等待新的连接请求，重复上述工作。对 TC、TM、TCC 处理分系统都只有二个进程组，而 SCOE 处理分系统则有(N+1)个进程组，N 个子进程组分别完成对应的 SCOE 设备的服务功能，如图 2、3 所示。

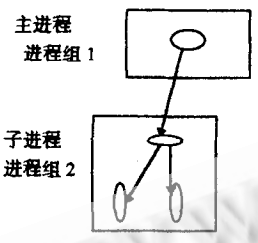

图 2　TC、TM、TCC 进程关系图

(3)连接控制。如图 2 所示，服务处理子进程和主进程分别处于不同的进程组中，因而当出现客户子系统关闭，或者其他原因导致连接中断时，主进程继续等待新的连接请求，对应的服务处理进程组的所有进程必须杀死，这是系统一致性、可靠性的要求。当同一连接重新到来时，主进程接受请求，生成功能相同的新的进程组和子进程，重复以前的处理服务过程。

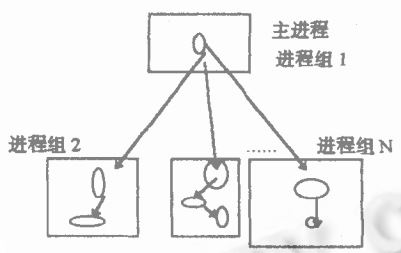

图 3　SCOE 进程关系图

实际中，我们判断 READ()系统调用，如果 read()返回数据<=0，则系统连接中断，向此进程所在的进程组的所有成员发送信号：KILL(0,SIGUSR1)。信号 SIGUSR1 这样设置：signal(SIGUSR1,action)，函数 action()进行进程退出处理。由于按功能将进程分组，一个进程组的服务进程退出，不会影响其他的进程组，因而系统是安全的，提高了系统的可靠性与一致性。

4.客户方系统实现

下面以 WINDOWS 95 环境为例，说明客户方的网络通信实现情况，开发平台为 Borlandc 3.1 或 VC++4.0。

(1)套接字(SOCKETS)。WINDOWS 套接字由 BERKELY 套接字扩展而来，为了支持 WINDOWS 的消息驱动机制，它对网络事件采用了基于消息的异步存取策略。它在异步选择机制、异步请求函数、阻塞处理方法、出错处理、启动与终止等方式作了扩展，并且支持所有的 UNIX 套接字调用。

(2)动态连接库(DLL)。WINDOWS 套接字应用程序调用的动态连接库是 WINSOCK。DLL，应用程序在执行时通过装入它实现网络通信功能。对于应用程序用到的连接库函数必须在模块定义文件(.def)中如下定义：

```
IMPORTS   WINSOCK   WSASTARTUP
          WINSOCK   CONNECT
```

上述方法是在 Borlandc 3.1 环境下采用的，在 VC++4.0 环境则按下列方法实现：

将 WSOCK32.LIB 加入到应用程序的工作项目中，去掉模块定义文件中的 IMPORTS 项，然后通过编译即可。

(3)异步方式和消息驱动。WINDOWS 中采用了使用消息队列来实现事件驱动的程序运行方式。它对网络的连接、读、写、关闭等事也是通过消息驱动相应的窗口过程函数的。实际中，我们用 WSAAsyncSelect()异步选择函数来注册应用程序感兴趣的网络事件，使得程序处于非阻塞工作方式，消息的传送采用异步方式。在窗口过程函数中，可根据消息中的 lParam 参数来判断网络事件。过程如下：

```
switch(message) {
  ...
  case WM-USER:
    initatiaze socket;
  case xxxxxx:/* 一个消息值，可由用户定义 */
    WSAAsyncSelect(SOCKET s,hWnd,UM-SOCK,
      FD-CONNECT|FD-WRITE|FD-READ);
    connect to server; /* 和服务器相连 */
  case UM-SOCK:
switch(lParam) {
      case FD-CONNECT:
        ...
      case FD-READ:
        read from socket; /* 读数据 */
      case FD-WRITE:
        write to socket; /* 写数据 */

  }
  ...
}
```

(来稿时间：1996 年 9 月)