目　　录

官方网站：http://www.bmpj.net

9. 协议过滤器，解决一包多发、粘包、冗余数据

9.1 概述

通讯中涉及到数据包的概念，是通讯协议中的数据组成形式。针对这块内容，说简单也简单，说复杂也复杂。需要我们系统性的把问题考虑全面，并用代码实现。

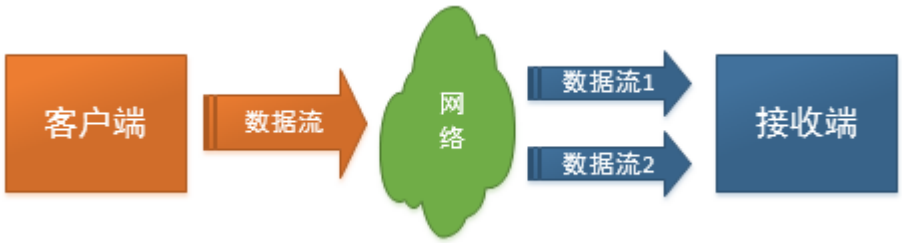
在工业领域也有极端的情况出现，早些年做通讯的时候，数据包头、包尾、数据长度、数据校验位都对，但是就是解析出来的数据不正确，这种情况不会经常出现，但是在某种特殊应用环境可能会频繁出现，后来经过分析得出结论：可能是由于地质电磁干扰引起的。但是也有技术上的设计缺陷，例如：数据校验位是累加和，改成 CRC 是不是就不会出来这个问题了；另外对于增量数据，应该有补发机制等等。

9.2 实际问题

参考协议：《连载 | 物联网框架 ServerSuperIO 教程》-4.如开发一套设备驱动，同时支持串口和网络通讯

1. 一包多发及解决

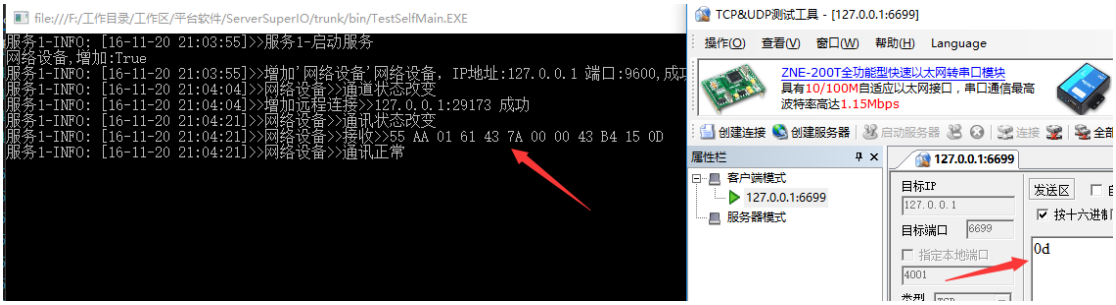
多包发送数据是应用环境中的一种情况或一个问题，并不是我们会这样实际应用，而是说在接收过程中多次接收数据才能完整接收客户端一次发送的数据，可能由于网络环境或发送数据端造成的，示意如下图：



例如实时数据的完整包为：55 AA 00 61 43 7A 00 00 43 B4 15 0D。那么接收数据的时候，第一次接收到：55 AA 00 61 43 7A 00 00 43 B4 15，第二次接到：

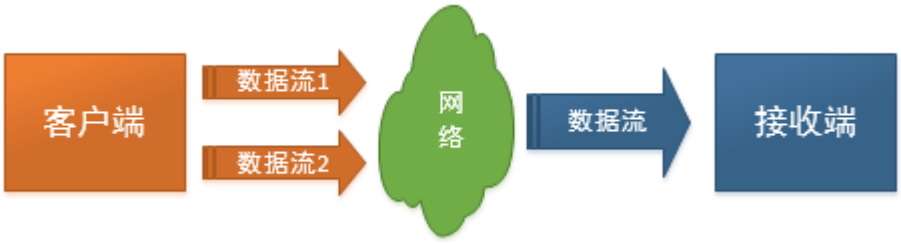
0D。按通讯协议应该能够把这两次接收的数据进行自动拼接，形成完整的数据并进行解析。

ServerSuperIO 设置了协议过滤器，可以解决这个问题，如下图：



2. 粘包及解决

我原来并不知道粘包这个概念，后来在网上看文章才明白。在通讯领域中也是经常会遇到的问题。也就是多包数据一次性的接收，那么就要合理的进行拆包。还有一种情况，就是多包半的数据一次性的接收，那半包的数据结合“1.一包多发及解决”来解决这个问题，示意如下图：



ServerSuperIO 设置了协议过滤器，可以解决这个问题，如下图：

3. 冗余数据的出现及解决

在工业领域受电缆或环境的电磁干扰，以及接头虚接等，这种情况极有可能出现。如果干扰的冗余数据夹杂在一个协议包中间，那么校验出合法的数据很困难。如果干扰的冗余数据夹杂在两个协议包中间，那么就可以通过协议过滤来实现识别出有用的数据。示意如下图：

ServerSuperIO 设置了协议过滤器，可以解决这个问题，如下图：

9.3 5 种过滤器及二次开发

FixedEndReceiveFliter：固定结尾的协议过滤器。

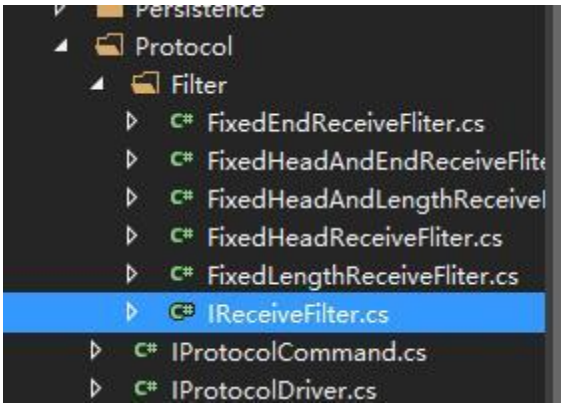
FixedHeadAndEndReceiveFliter：固定开头和结尾的协议过滤器。

FixedHeadAndLengthReceiveFliter：固定开头和长度的协议过滤器。

FixedHeadReceiveFliter：固定开头的协议过滤器。

FixedLengthReceiveFliter：固定长度的协议过滤器。

这 5 个过滤器都继承自 IReceiveFilter 接口，也可以继承这个接口进行二次开发，定制自己的协议过滤器。代码工程如下图：



9.4 设备驱动开发注意事项

对于开发设备驱动，在初始化过程中可以增加这个驱动的协议过滤器，代码如下：

```
public override void Initialize(string devid)
{
            this.Protocol.InitDriver(this.GetType(),new
FixedHeadAndEndReceiveFliter(new byte[] {0x55,0xaa},new byte[] {0x0d} ));
……
}
```

9.5 宿主程序服务实例配置注意事项

在配置参数中需要配置：StartReceiveDataFliter = true,协议过滤器才能起到作用。代码如下：

```
static void Main(string[] args)
{
            DeviceSelfDriver dev2 = new DeviceSelfDriver();
            dev2.DeviceParameter.DeviceName = "网络设备";
            dev2.DeviceParameter.DeviceAddr = 1;
            dev2.DeviceParameter.DeviceID = "1";
            dev2.DeviceDynamic.DeviceID = "1";
            dev2.DeviceParameter.DeviceCode = "1";
            dev2.DeviceParameter.NET.RemoteIP = "127.0.0.1";
            dev2.DeviceParameter.NET.RemotePort = 9600;
            dev2.CommunicateType = CommunicateType.NET;
            dev2.Initialize("1");
```

```
            IServer server = new ServerManager().CreateServer(new ServerConfig()
            {
                ServerName = "服务1",
                ComReadTimeout = 1000,
                ComWriteTimeout = 1000,
                NetReceiveTimeout = 1000,
                NetSendTimeout = 1000,
                ControlMode = ControlMode.Self,
                SocketMode = SocketMode.Tcp,
                StartReceiveDataFliter = true,
                ClearSocketSession = false,
                StartCheckPackageLength = false,
                CheckSameSocketSession = false,
                DeliveryMode = DeliveryMode.DeviceIP,
            });

            server.AddDeviceCompleted += server_AddDeviceCompleted;
            server.DeleteDeviceCompleted+=server_DeleteDeviceCompleted;
            server.Start();

            server.AddDevice(dev2);

            while ("exit" == Console.ReadLine())
            {
                server.Stop();
            }
}
```