# 目　　录



官方网站：http://www.bmpj.net

# 6. 并发通讯模式开发及注意事项

## 6.1 概述

并发通讯模式只能用于网络通讯设备，主要是加强通讯的并发能力，集中发送请求数据，异步接收返回数据。集中发送请求数据的间隔时间可以设置；异步接收返回数据涉及到如何分配数据到相应的设备驱动的问题，主要是通过两种方式：IP 地址的方式和设备 Code 的方式，前者适用于设备终端是固定 IP 地址的情况，后者适用于设备终端是动态 IP 的情况，例如：DTU、GPRS、3G/4G 等无线通讯方式。

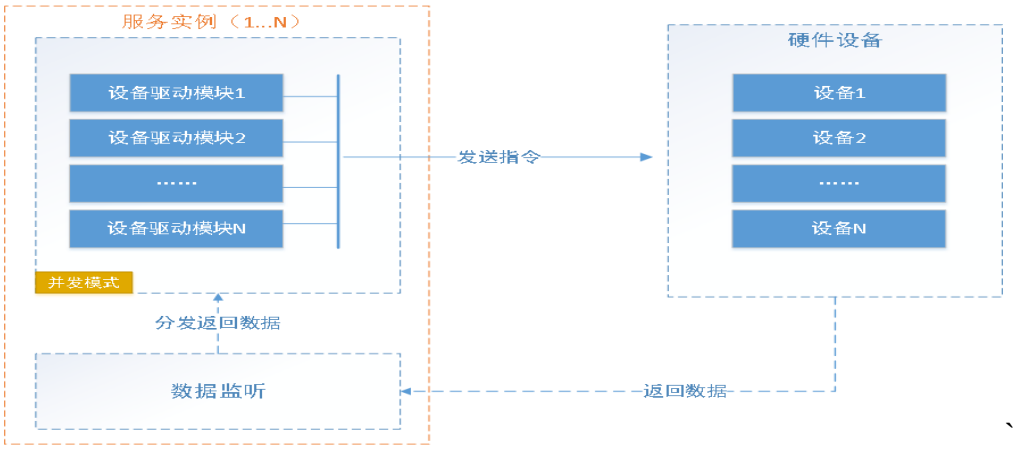
并发通讯模式本质上还是呼叫应答的通讯方式，与轮询通讯模式类似，但是比轮询通讯模式的采集数据更高效。

## 6.2 通讯机制说明

网络通讯的情况下，轮询模式显然效率比较低，那么可以采用并发模式。并发通讯模式是集中发送给所有设备请求指令，框架是采用循环同步方式发送请求命令给每个 IO 通道对应的设备，当然也可以采用并行异步方式集中发送请求命令。硬件设备接收到指令后进行校验，校验成功后返回对应指令的数据，通讯平台异步监听到数据信息后，进行接收操作，然后再进行数据的分发、处理等。

那么这里就涉及到 IO 通道接收到的数据是异步接收的，如何才能和设备驱动匹配上（把数据分发到设备驱动上），这是能过 DeviceCode 和 DeviceIP 两种方式来实现的。DeviceCode 可以是设备地址或是设备编码，DeviceIP 是预先设置好的参数，要求终端设备的 IP 地址是固定的。

通讯结构如下图：




## 6.3 设备驱动开发注意事项

### 6.3.1 实时发送数据

ServerSuperIO 框架会轮询调度所有设备，以呼叫应答的方式向设备发送请求实时数据命令，对于同一个设备的请求实时数据命令一般相对固定。在调度某一具体设备驱动的时候，会调用固定的调用 IRunDevice 驱动接口的 GetConstantCommand 函数，以获得请求实时数据的命令。代码如下：

```
public override byte[] GetConstantCommand()
{
    byte[] data = this.Protocol.DriverPackage<String>("0", "61", null);
    string hexs = BinaryUtil.ByteToHex(data);
    OnDeviceRuningLog("发送>>"+hexs);
    return data;
}
```

this.Protocol.DriverPackage 驱动调用 61 命令获得要发送的命令，并返回 byte[]数组，ServerSuperIO 获得数据后会自动通过 IO 接口下发命令数据。如果返回 null 类型，系统不进行下发操作。

### 6.3.2 优先发送其他数据

对于一个设备不可能只有一个读实时数据的命令，可能还存在其他命令进行交互，例如：读参数、实时校准等，这时就需要进行优先级调度发送数据信息。可以通过两种方式让 ServerSuperIO 框架优先调度该设备驱动。

1. 把命令增加发送数据缓存中，框架从缓存中获得数据后会自动删除，代码如下：

```
this.Protocol.SendCache.Add("读参数",readParaBytes);
```

2. 设置设备的优先级别属性，代码如下：

```
this.DevicePriority=DevicePriority.Priority;
```

### 6.3.3 如何选择 IO 通道发送数据

集中发送数据时，涉及到如何关联设备驱动与 IO 通道，框架会以 DeviceParameter.NET.RemoteIP 设置的终端 IP 参数进行选择 IO 通道发送数据。但是如果终端设备是动态 IP 地址的话，那么 RemoteIP 参数也应该是变动的。这时就需要设置服务实例是以 DeviceCode 的方式分布数据到设备驱动，终端设备先发送简单的验证数据，保证发送的 DeviceCode 与设备驱动的相对应，设备驱动接收到验证数据后需要保存临时的 RemoteIP 信息，这样保证在发送数据的时候参数准确找到要请求数据的 IO 通道到终端设备。

例如下面代码：

```
public override void Communicate(ServerSuperIO.Communicate.IRequestInfo info)
{
            this.DeviceParameter.NET.RemoteIP = info.Channel.Key;
            this.DeviceParameter.Save(this.DeviceParameter);
            ......
}
```

### 6.3.4 如何以 DeviceCode 分配数据

如果服务实例设置以 DeliveryMode.DeviceCode 模式分配数据，那么就需要在通讯协议接口里实现过滤 DeviceCode 编码的接口。

例如下面的代码：

```
internal class DeviceProtocol:ProtocolDriver
 {
     public override string GetCode(byte[] data)
     {
         byte[] head = new byte[] {0x55, 0xaa};
         int codeIndex = data.Mark(0, data.Length, head);
```

```
            if (codeIndex == -1)
            {
                return String.Empty;
            }
            else
            {
                return data[codeIndex + head.Length].ToString();
            }
        }
}
```

## 6.4 宿主程序服务实例配置注意事项

在宿主程序中创建服务实例的时候，需要把服务实例的配置参数设置为并发通讯模式，并启动服务实例，把实例化的设备驱动增加到该服务实例中。代码如下：

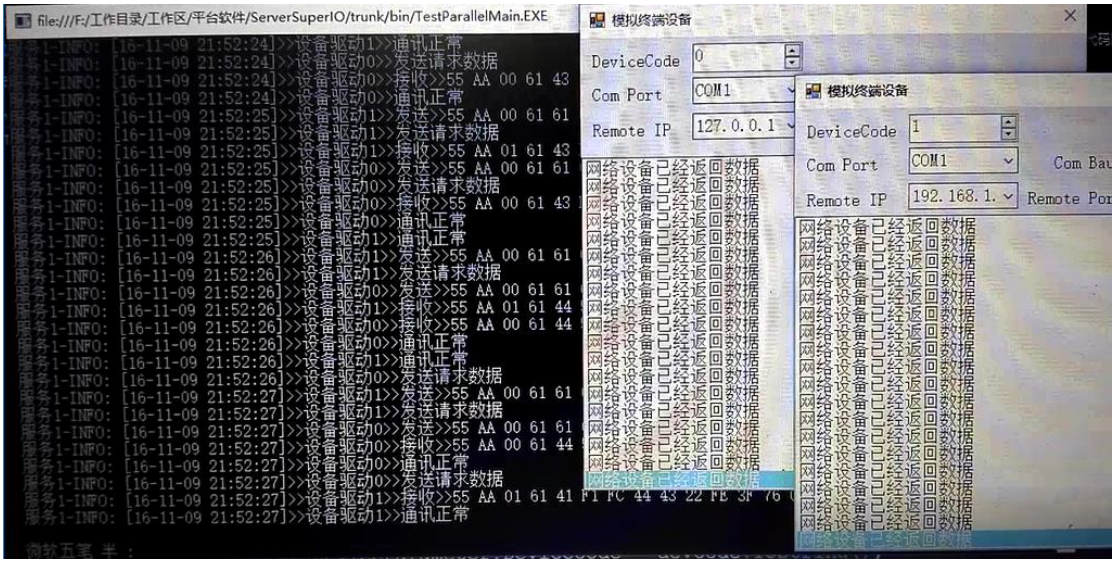
```
static void Main(string[] args)
{
            IServer server = new ServerFactory().CreateServer(new ServerConfig()
            {
                ServerName = "服务1",
                ComReadTimeout = 1000,
                ComWriteTimeout = 1000,
                NetReceiveTimeout = 1000,
                NetSendTimeout = 1000,
                ControlMode = ControlMode.Parallel,
                SocketMode = SocketMode.Tcp,
                StartReceiveDataFliter = false,
                ClearSocketSession = false,
                StartCheckPackageLength = false,
                CheckSameSocketSession = false,
                DeliveryMode = DeliveryMode.DeviceCode,
                ParallelInterval = 1000
            });
            server.AddDeviceCompleted += server_AddDeviceCompleted;
            server.DeleteDeviceCompleted += server_DeleteDeviceCompleted;
            server.Start();

            string devCode = "0";
            DeviceDriver dev1 = new DeviceDriver();
            dev1.DeviceParameter.DeviceName = "设备驱动"+ devCode.ToString();
```

```
            dev1.DeviceParameter.DeviceAddr = int.Parse(devCode);
            dev1.DeviceParameter.DeviceCode = devCode.ToString();
            dev1.DeviceParameter.DeviceID = devCode.ToString();
            dev1.DeviceDynamic.DeviceID = devCode.ToString();
            dev1.DeviceParameter.NET.RemoteIP = "127.0.0.1";
            dev1.DeviceParameter.NET.RemotePort = 9600;
            dev1.CommunicateType = CommunicateType.NET;
            dev1.Initialize(devCode.ToString());
            server.AddDevice(dev1);

            devCode = "1";
            DeviceDriver dev2 = new DeviceDriver();
            dev2.DeviceParameter.DeviceName = "设备驱动" + devCode.ToString();
            dev2.DeviceParameter.DeviceAddr = int.Parse(devCode);
            dev2.DeviceParameter.DeviceCode = devCode.ToString();
            dev2.DeviceParameter.DeviceID = devCode.ToString();
            dev2.DeviceDynamic.DeviceID = devCode.ToString();
            dev2.DeviceParameter.NET.RemoteIP = "192.168.1.102";
            dev2.DeviceParameter.NET.RemotePort = 9600;
            dev2.CommunicateType = CommunicateType.NET;
            dev2.Initialize(devCode.ToString());
            server.AddDevice(dev2);

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

ControlMode = ControlMode. Parallel 代码是设置服务实例调度设备为并发控制模式；以 DeliveryMode = DeliveryMode.DeviceCode 方式进行数据分发，当然我现在模拟的是固定的终端 IP。

## 6.5 并发模式运行效果

1.图片




2.视频

http://static.video.qq.com/TPout.swf?vid=q0344c50umz&auto=0