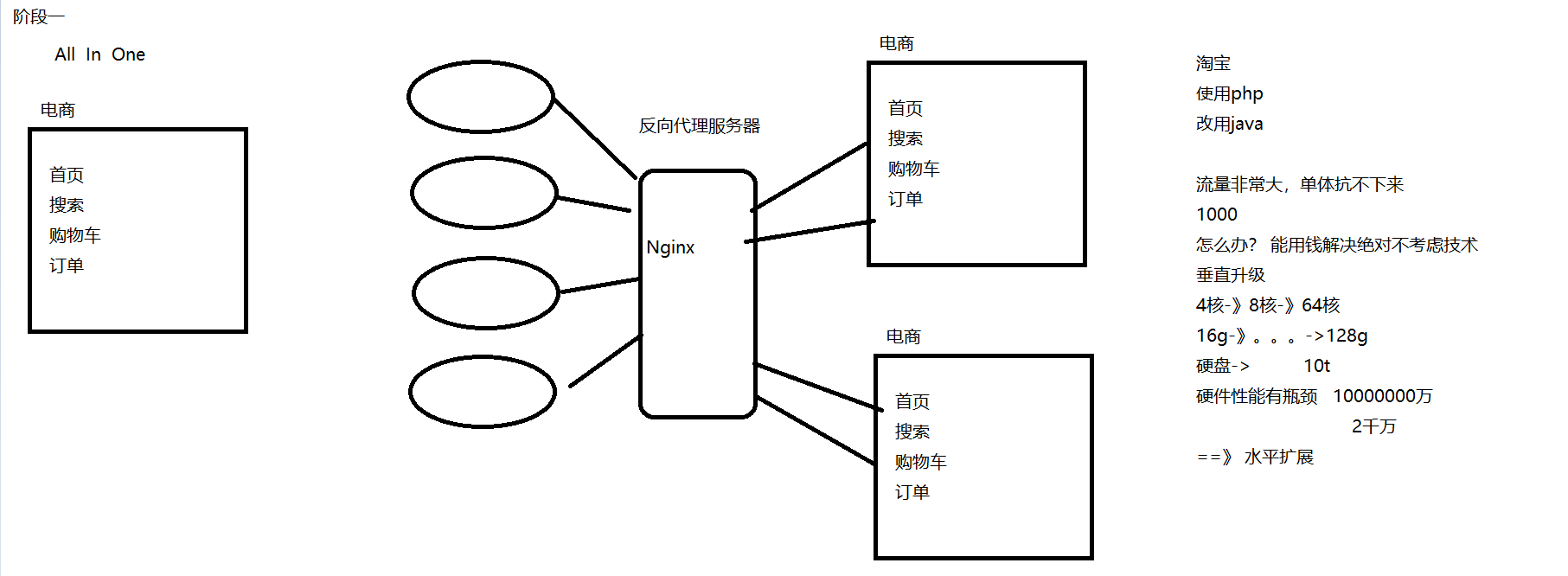
# 一、架构演进

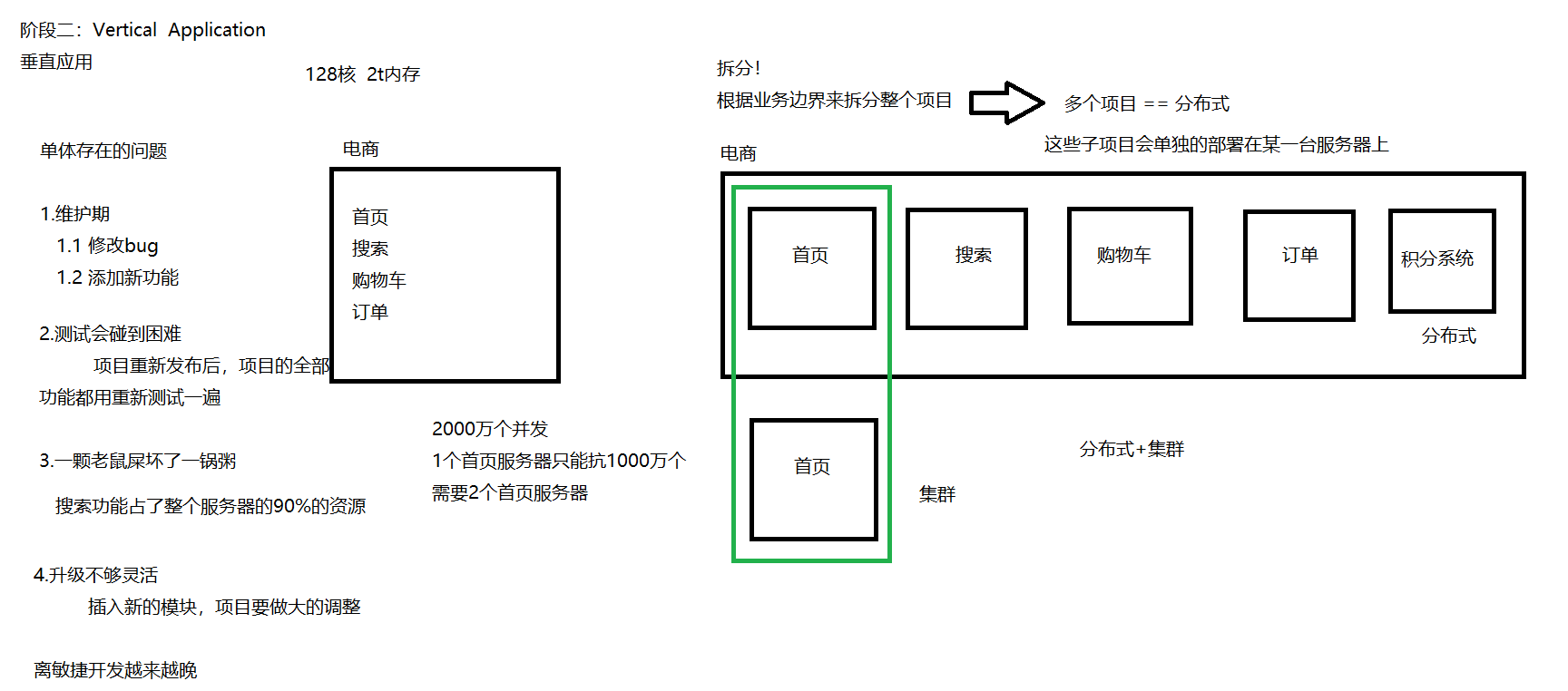
## 1.阶段一： All In One



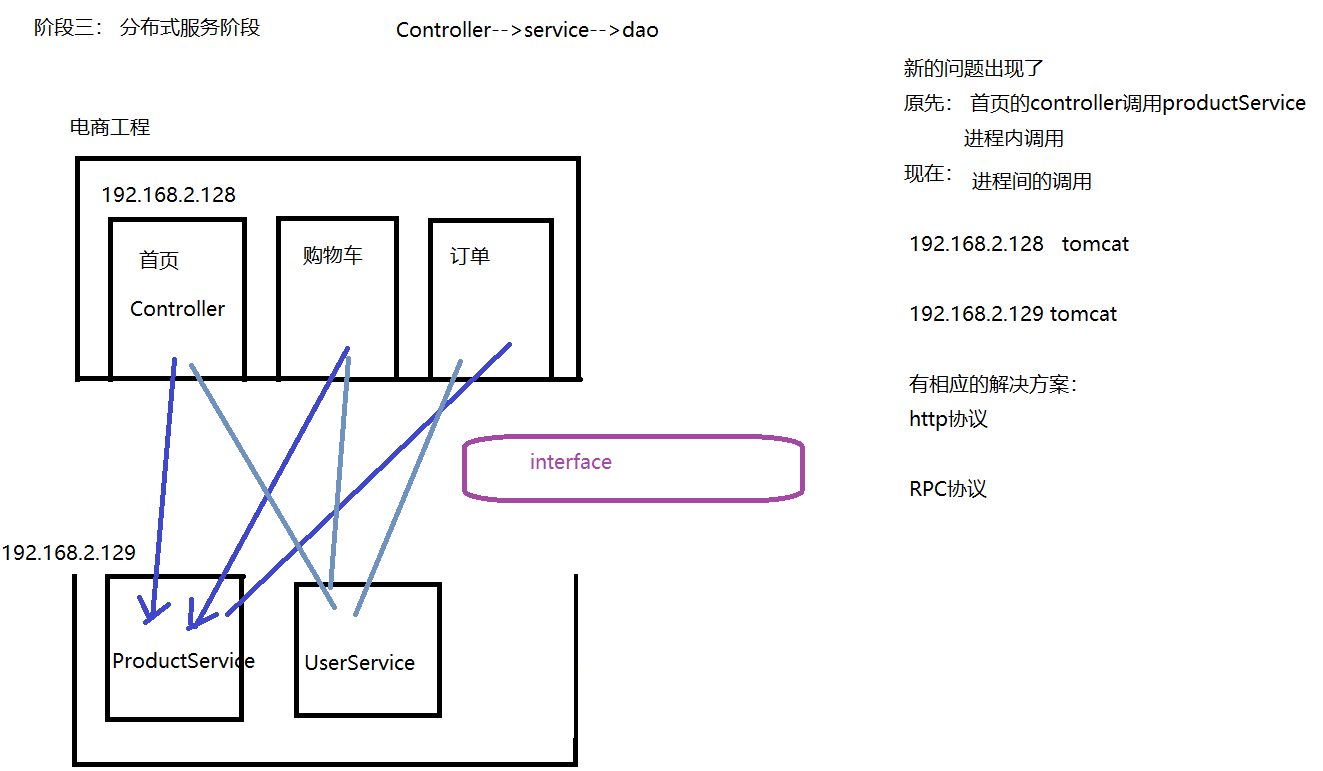
## 2.阶段二：垂直应用阶段

项目： 全部在一起

工程： 有很多项目 功能==》 项目

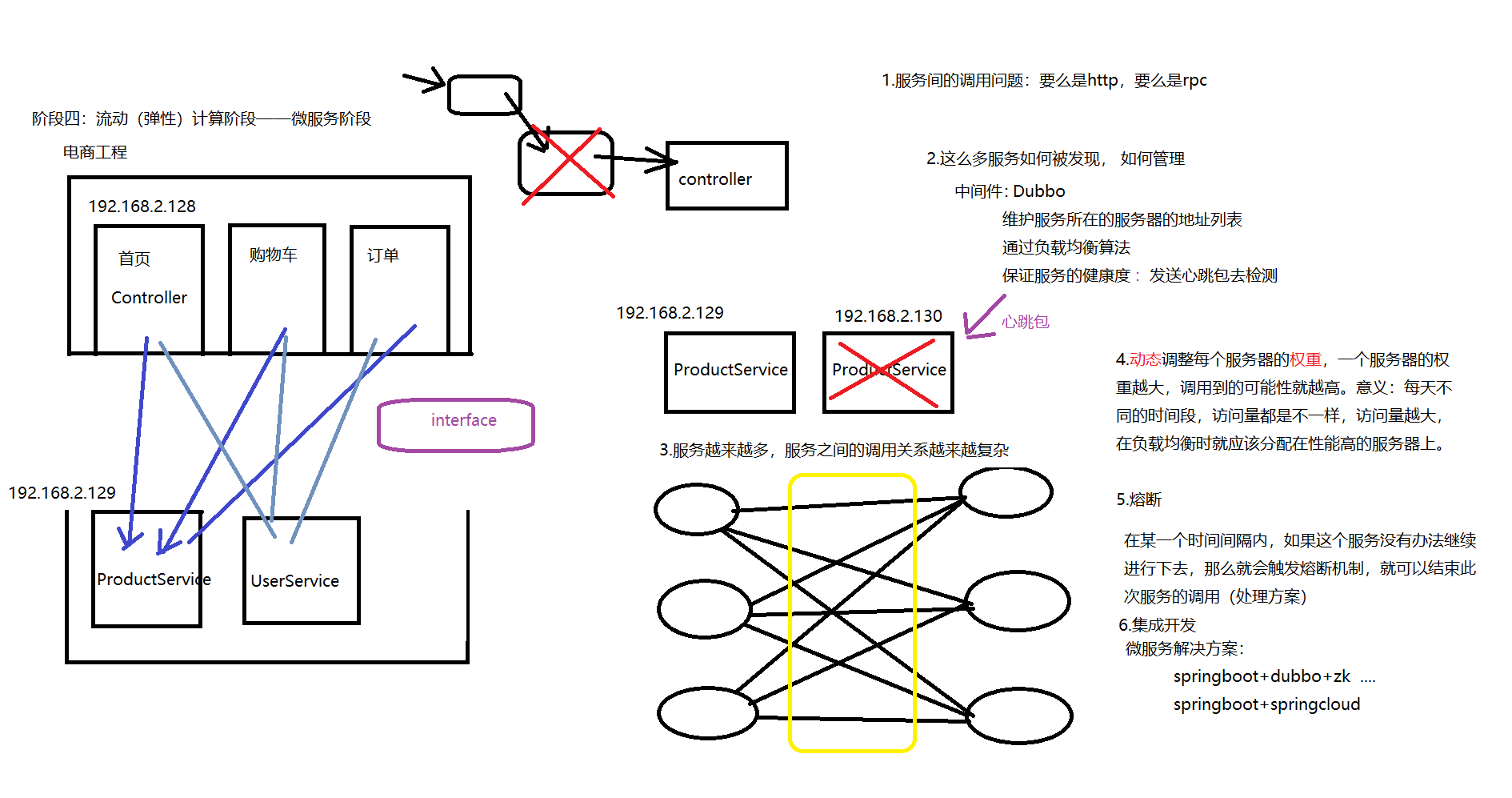


## 3.阶段三：分布式服务阶段



## 4.阶段四：微服务阶段

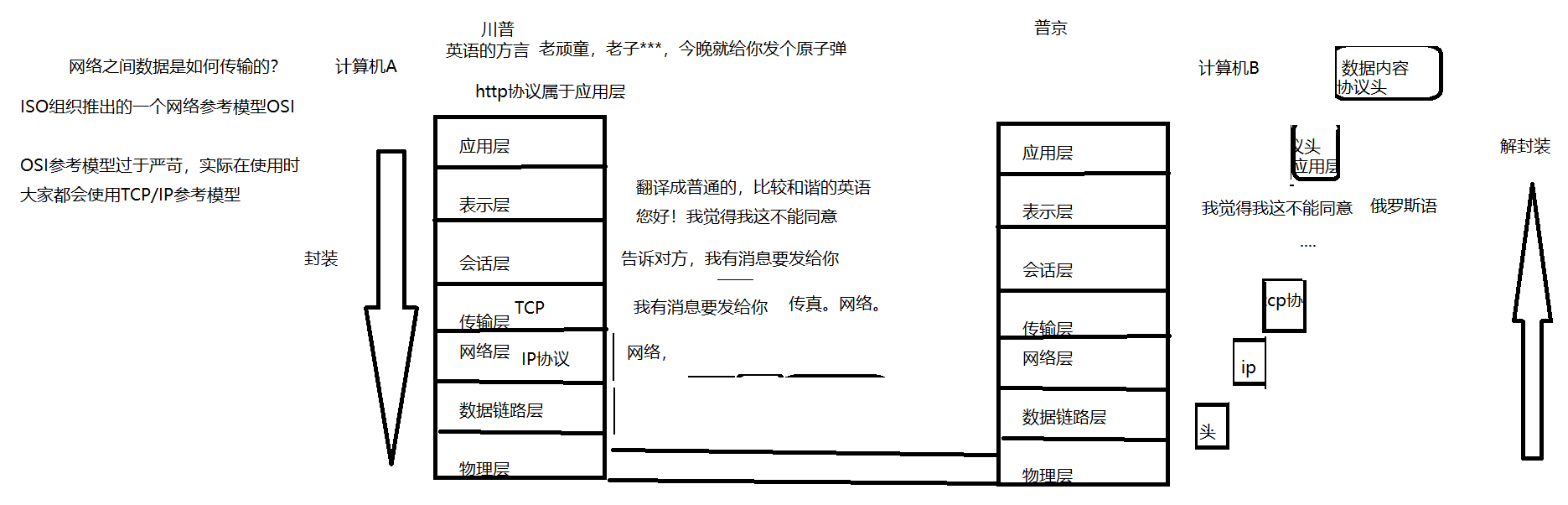
负载均衡



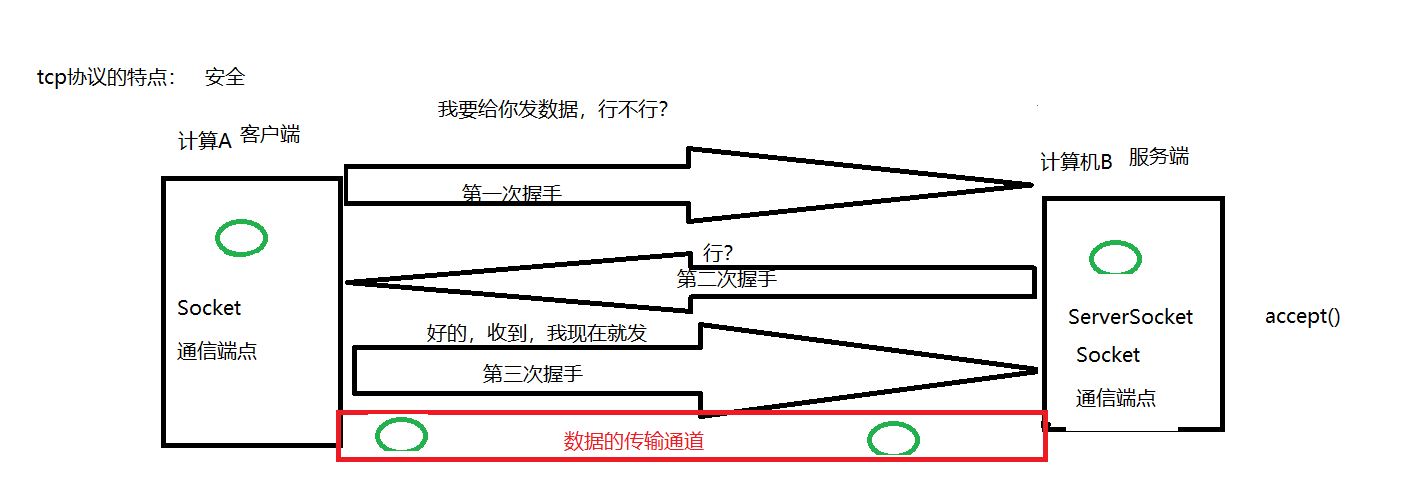
# 二、RPC协议解释

通过RPC协议，在调用远程服务器上的服务时，感觉上就像是调用本地服务一样，那么的方便。也就是说，现在微服务中服务的调用，性质是进程间的调用，但是有了RPC以后，就感觉像是进程内的调用。

# 三、网络参考模型



# 四、如何使用TCP进行编程



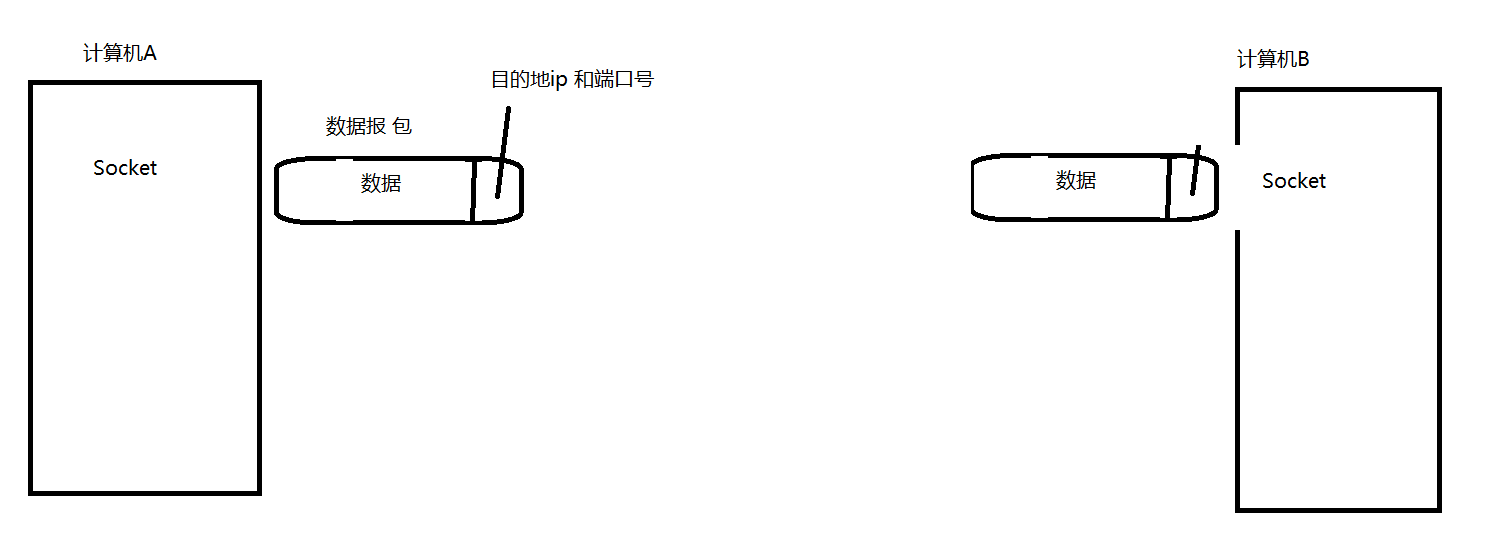
客户端

|  |
| --- |
| **public class** MyClient {  **public static void** main(String[] args) **throws** IOException {    *//获得表示127.0.0.1的IP地址对象* InetAddress ip = InetAddress.*getByName*(**"127.0.0.1"**);   *//创建一个客户端Socket,并指明要连接的服务端的socket的ip地址和端口号* Socket socket = **new** Socket(ip,9091);   OutputStream outputStream = socket.getOutputStream();  *//传输数据* outputStream.write(**"晚上约吗？"**.getBytes());   *//关闭端点* socket.close();    } } |

服务端：

|  |
| --- |
| **public class** MyServer {  **public static void** main(String[] args) **throws** IOException {  *//服务端* ServerSocket serverSocket = **new** ServerSocket(9091);   *//serverSocket调用accept，使得该程序处于阻塞状态，直到得到一个客户端连接，于是创建一个Socket对象* Socket socket = serverSocket.accept();   InputStream is = socket.getInputStream();   **byte**[] b = **new byte**[1024];  **int** len = 0;  **while**((len = is.read(b))!=-1){  System.***out***.print(**new** String(b,0,len));  }   *//关闭端点。* socket.close();  serverSocket.close();    } } |

# 五、如何使用UDP进行编程



特点：快，但不安全，因为网络是不稳定的。

客户端：

|  |
| --- |
| **public class** MyClient {   **public static void** main(String[] args) **throws** IOException {   *//创建好了端口（码头）* DatagramSocket socket = **new** DatagramSocket();  *//创建数据报包* **byte**[] data = **"晚上约吗？"**.getBytes();   DatagramPacket dp = **new** DatagramPacket(data,0,data.**length**,  InetAddress.*getByName*(**"127.0.0.1"**),9099);    *//发送数据包* socket.send(dp);   socket.close();   }    } |

服务端：

|  |
| --- |
| **public class** MyServer {   **public static void** main(String[] args) **throws** IOException {    DatagramSocket socket = **new** DatagramSocket(9099);    **byte**[] b = **new byte**[8192];  *//创建的是一个空的数据报包* DatagramPacket p = **new** DatagramPacket(b,b.**length**);  *//于是socket会去收取一个数据报包将数据封装在p（刚才创建的空的数据包）里。* socket.receive(p);   *//拆包  //获得数据* **byte**[] data = p.getData();*//data的长度是8192  //获得真实数据的长度* **int** length = p.getLength();  System.***out***.println(**new** String(data,0,length));   socket.close();    } } |

# 六、dubbo的工作流程

