无论是所谓的传递“字符串”，抑或传递“数值”。其归根到底都是发送二进制01码流。因为在网络上流动的只能是二进制01码流。之所以会有传递“字符串”和传递“数值“的区别，在于用户收到码流后对码流的解读模式，下面分别分析：

**传递字符串**：当前的字符编码有ascll、unicode、gbk等等。拿ascll码来说，任意一个英文字符都是由1个字节编码。并且ascll码在编码规范中没有引入位序的概念，这就意味着收端在接收到二进制位后，只要按8位组装成一个字节的形式来解读就可以了。Unicode和gbk可能会用两个或三个字节来编码某个字符，但是由于这些字节间没有定义字节序的关系，因此收端按接收到的字节序解读就行了。**综上可知，当前使用的字符编码中没有位序和字节序的概念，因此发送和接收的解读是一致的。根本上来讲，是收发双方在发送和接收字符串时都遵循统一的规范格式，因此就不存在什么字节序的问题了**。

**传递数值**：在网络上传递数值时会存在字节序的问题，这归根到底是由于不同机器（操作系统）在存储数值时没有制定统一的规范，而是不同的制造厂商依照自己的想法进行存储的。这就导致有些系统是按照大端字节序在内存中保存数值，有些系统是按照小端字节序在内存中保存数值，这种差异导致网络传输中的码流是不一样的。例如int 0x01020304在小端系统中存储为04030201，因此网络传输的二进制码流为04030201，**接收端收到码流后保存到内存中为0x04，0x03，0x02，0x01**，此时接收端如果是大端字节序，那么它收到这4个字节的码流后，将解释为0x04030201，这就和发送端发送的数值0x01020304完全不一样了。因此，在这两种系统间传递数据收发双方就必须协商好使用一种统一的字节序在网络传输，比如统一按大端传送。另外一个问题是，不同机器的类型长度可能不一致，例如，发送端系统int大小是4个字节，但接收端int大小是8个字节。这将导致接收端在收到8个字节后进行解读，我们知道这肯定是错误的。因此，字节序问题归根到底就是各个实现没有遵循统一标准的后果。

综上，在网络上传递数值时，要保证正确，必须协商好下面三点：

* **字节序**
* **类型长度**，比如long的长度统一为4
* **结构体对齐方式**

因此，protobuf对以上做了适配，因此只要收发双端用probuf对消息内容进行序列化后发送，就不用担心字节序的问题了。