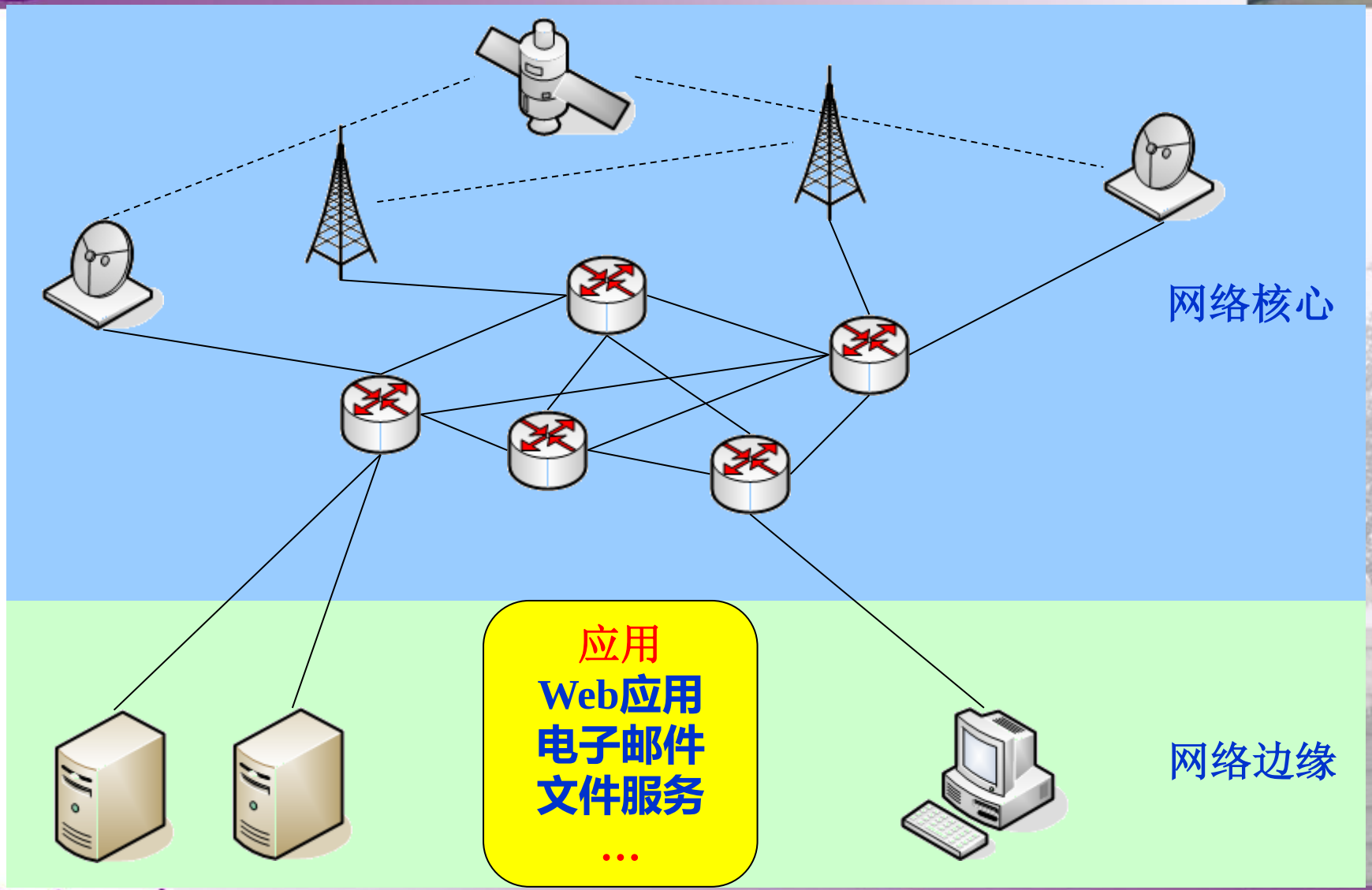
第四讲 网络通信的基本编程

清华大学计算机系

主要内容

- 背景与基本概念
- Qt Socket 简介
- 有连接的TCP通信
- 无连接的UDP通信
- 有代表性的网络应用协议

网络结构



B/S模式和C/S模式

- C/S（Client/Server）结构，即客户机和服务器结构
  - 将任务（存储、操作或计算的任务）分配到客户端或服务器中
  - 客户端和服务器通过网络通信来协作

- B/S（Browser/Server）结构，即浏览器和服务器结构。
  - 客户端使用标准的浏览器，不需要专门开发、部署客户端

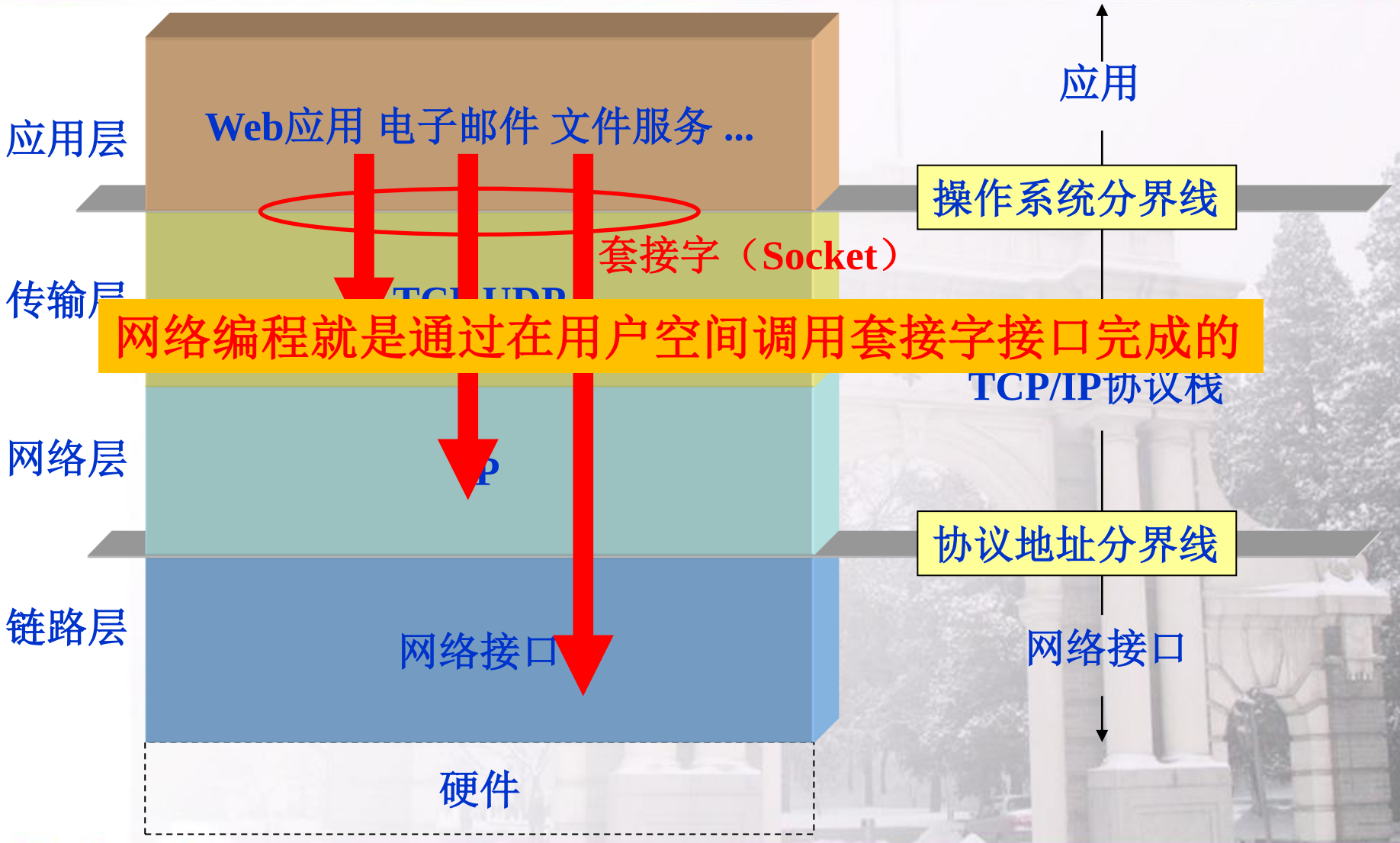
课堂演示：即时通信系统

- 即时通信系统（IMS）是最常见的网络应用软件
  - 如QQ、MSN、微信、飞信等
- 开发IMS，需要实现最简单的“发送”和“接收”功能。
  - 方便初学者掌握TCP/IP网络程序设计
- 开发IMS，读者可以学会C/S模式的网络通信软件的开发。
  - 包括服务器端程序设计和客户端程序设计。

即时通信系统：功能需求

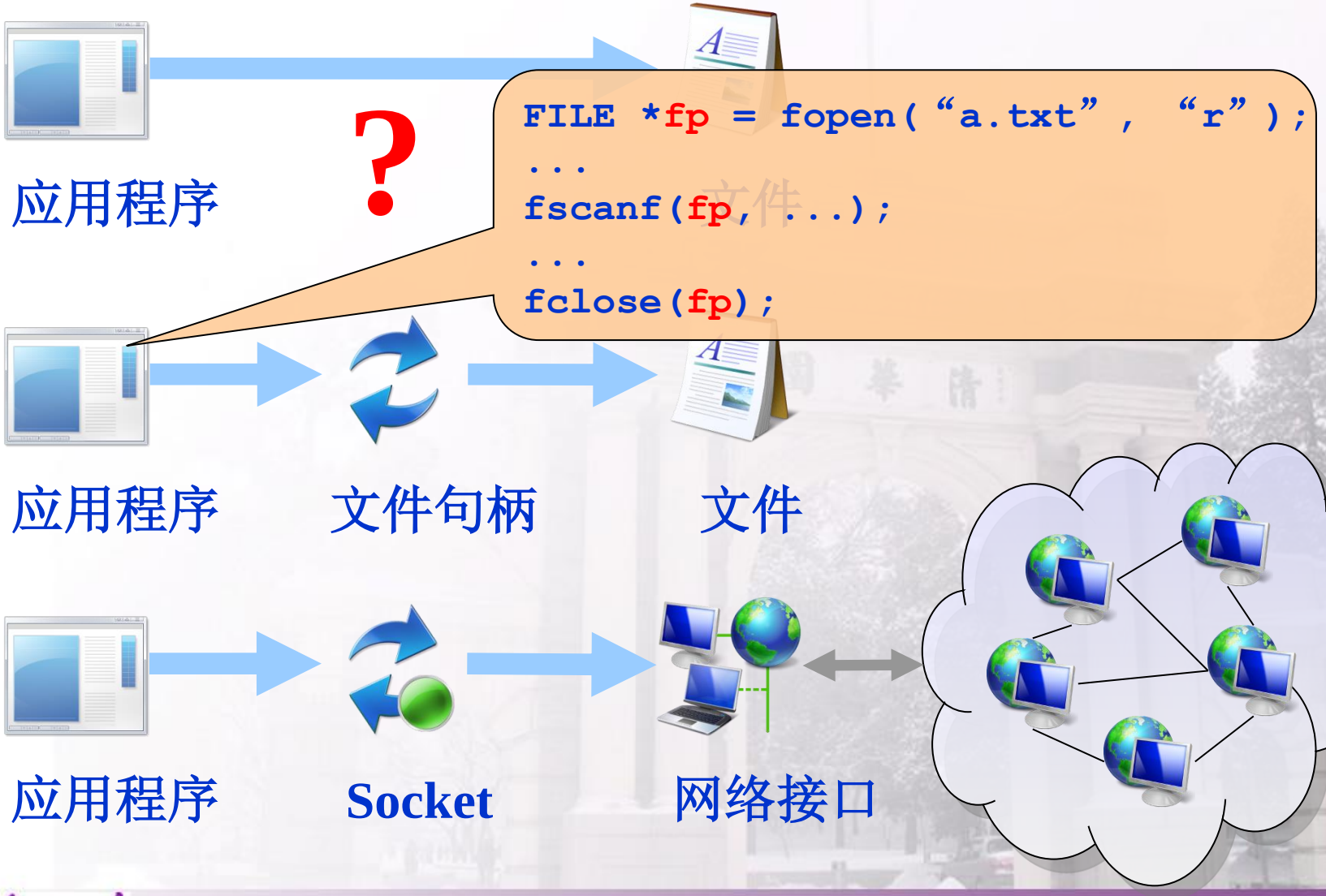
- **启动服务器，处于监听状态**
  - 服务器建立之后，等待客户机的连接申请。
- **启动客户端，尝试对服务器进行连接操作**
- **一个连接建立之后，其他客户机还可以再连接到上面**
  - 这样可以进行多用户的信息交互。
- **成功建立连接之后，开始进行对话操作**
  - 实现只有消息的接收方可以看到，保护隐私。
- **聊天结束之后，客户机断开连接，退出聊天的过程。**
  - 如果是服务器关闭，连接在上面的所有客户机将会断开。

TCP/IP 模型中的两个分界线



网络编程就是通过在用户空间调用套接字接口完成的

Socket的引入


?
FILE *fp = fopen(“a.txt”, “r”);
...
fscanf(fp, ...);
...
fclose(fp);
应用程序
应用程序
文件句柄
文件
应用程序
Socket
网络接口

什么是Socket

- 文件I/O操作 - 句柄（Handle）
- 网络I/O操作 - 套接字（Socket）
- Socket提供了一个通信接口，应用程序在网络上发送、接收的信息都通过这个接口来实现。
- Socket和句柄一样，是操作系统的资源

1、基本概念

- IP地址：
  - Internet中的主机要与别的机器通信必须具有一个IP地址，IP地址是Internet中主机的标识。
  - 表示形式：常用点分形式，如166.111.8.28，最后都会转换为一个32位的整数。

- IP地址转换函数
  - inet_addr()：点分十进制数表示的IP地址转换为网络字节序的IP地址
  - inet_ntoa()：网络字节序的IP地址转换为点分十进制数表示的IP地址

1、基本概念

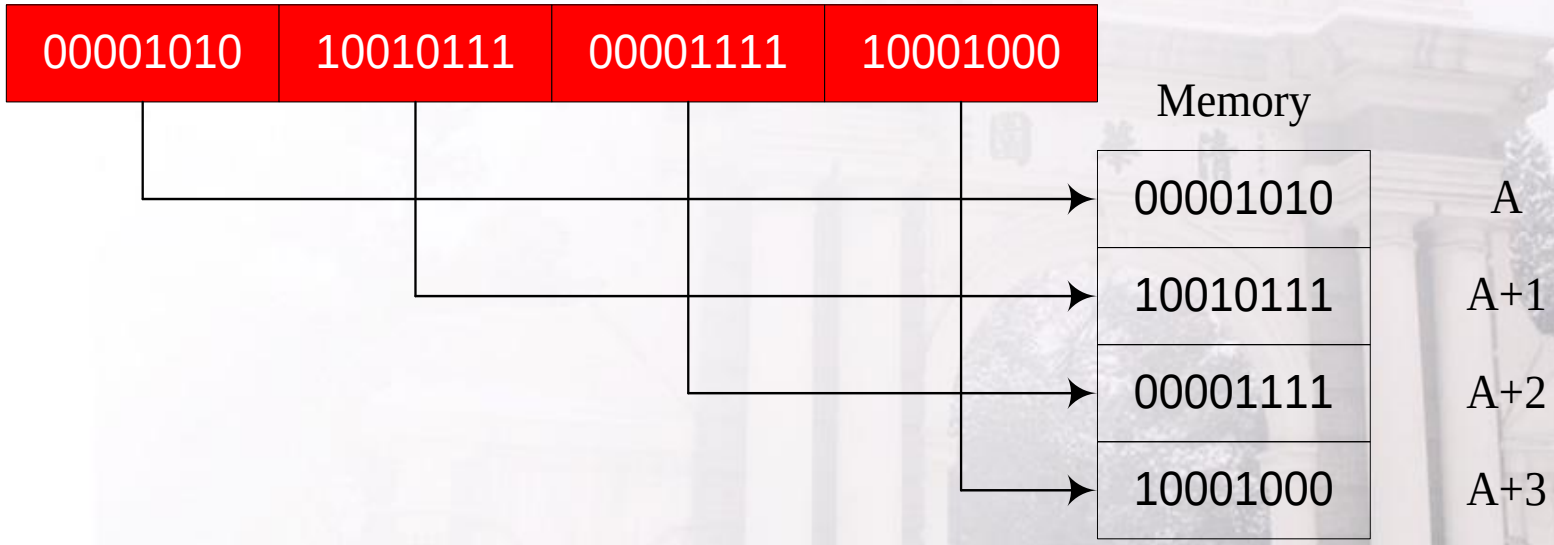
- 端口号
  - 为了区分一台主机接收到的数据包应该递交给哪个进程来进行处理，使用端口号
  - TCP端口号与UDP端口号独立
- 端口号一般由IANA (Internet Assigned Numbers Authority) 管理
  - 众所周知端口：1~1023，1~255之间为大部分众所周知端口，256~1023端口通常由UNIX占用
  - 注册端口：1024~49151
  - 动态或私有端口：49152~65535

1、基本概念

- 使用socket实现网络通信
  - 配置一个socket需要五种信息：
    - 本地的IP地址、本地的协议端口
    - 远程的IP地址、远程的协议端口
    - 连接所使用的协议
- 打个比方：
  - 如果把IP数据包的投递过程看成是给远方的一位朋友寄一封信，那么：
  - IP地址就是这位朋友的所在位置，如北京清华大学计算系（依靠此信息进行路由）
  - 端口号就是这位朋友的名字（依靠这个信息最终把这封信交付给这位收信者）

1、基本概念

- 字节序
  - 大尾端(Big-Endian):字节的高位在内存中放在存储单元的起始位置



  - 小尾端(Little-Endian):与大尾端相反

1、基本概念

- 网络字节序：Network Byte Order
  - 使用统一的字节顺序，避免兼容性问题
- 主机字节序：Host Byte Order
  - 不同机器的HBO与CPU的设计有关，可能不一样
  - Motorola 68K系列，HBO与NBO是一致的
  - Intel X86系列，HBO与NBO不一致
- 字节排序函数
  - #include <QtEndian>
  - qFromBigEndian(const uchar * *src*)：大端字节序转换为主机字节序
  - qToBigEndian(T *src*, uchar * *dest*)：主机字节序转换为大端字节序

1、基本概念

- 阻塞通信与非阻塞通信
  - 阻塞方式：套接字进行I/O操作时，函数要等待到相关的操作完成以后才能返回。
  - 非阻塞方式：套接字进行I/O操作时，无论操作成功与否，调用都会立即返回。
- 缺省处于非阻塞方式，也就是事件编程
  - 好处：可以在一个线程中实现多路tcp链接，节省资源
  - 缺点：编程难度比较大。
- 在满足要求的情况下，还是阻塞方式的socket编程比较容易理解
  - waitForConnected() 等待链接的建立
  - waitForReadyRead() 等待新数据的到来
  - waitForBytesWritten() 等待数据写入socket
  - waitForDisconnected() 等待链接断开

2、QT Socket 简介

- 不同平台的 Socket的发展
  - UC Berkeley为UNIX系统开发出了一套套接字（BSD socket）
  - 在此基础上扩展形成了windows套接字。
  - Windows Socket 规范是一套开放的、支持多协议的 Windows 下的网络编程接口，它规范了Internet协议族（一般为TCP/IP）的API使用.
- QT Socket的作用
  - 针对多个的操作系统，QT Socket 统一了网络编程接口，简化了编程，使两个进程、两种平台之间易于实现连接、通信

Socket的分类

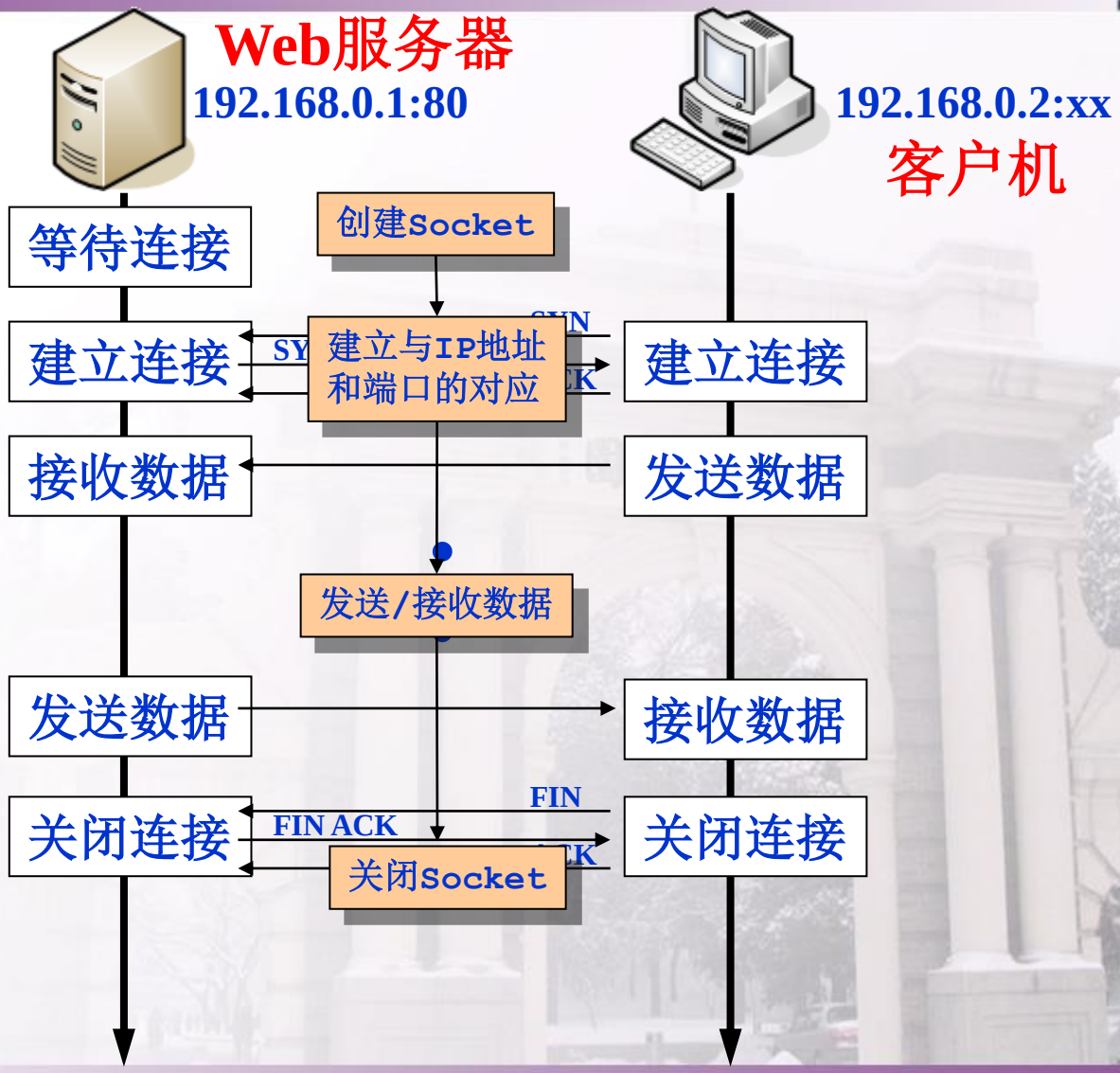
- 套接字有三种类型
  - 流式套接字(SOCK_STREAM)：一种可靠的面向连接的服务，实现了无差错无重复的顺序数据传输
  - 数据报套接字（SOCK_DGRAM）：一种无连接的服务，数据通过相互独立的报文进行传输，是无序的，并且不保证可靠、无差错
  - 原始套接字(SOCK_RAW)：允许对底层协议如IP或ICMP（因特网控制消息协议）直接访问，主要用于新的网络协议实现的测试等

2、QT Socket 简介

- 编程时的两点注意事项
  - 需要包含头文件QtNetwork：
    #include <QtNetwork>
  - 在QT Creator环境中，需要在.pro文件中增加一行
    QT += network

- 不同的传输协议
  - 有连接的TCP协议，使用如下两个类：
    QTcpServer
    QTcpSocket
  - 无连接的UCP协议，使用如下一个类：
    QUdpSocket

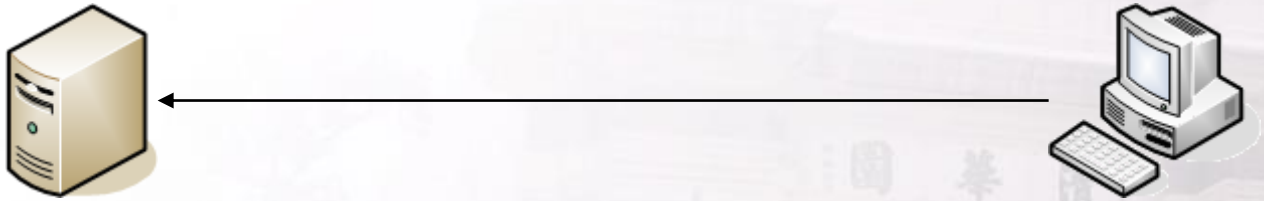
3、有连接的C/S网络通信程序 (TCP)



问题

- 如何在套接字和IP地址/端口之间建立关联？

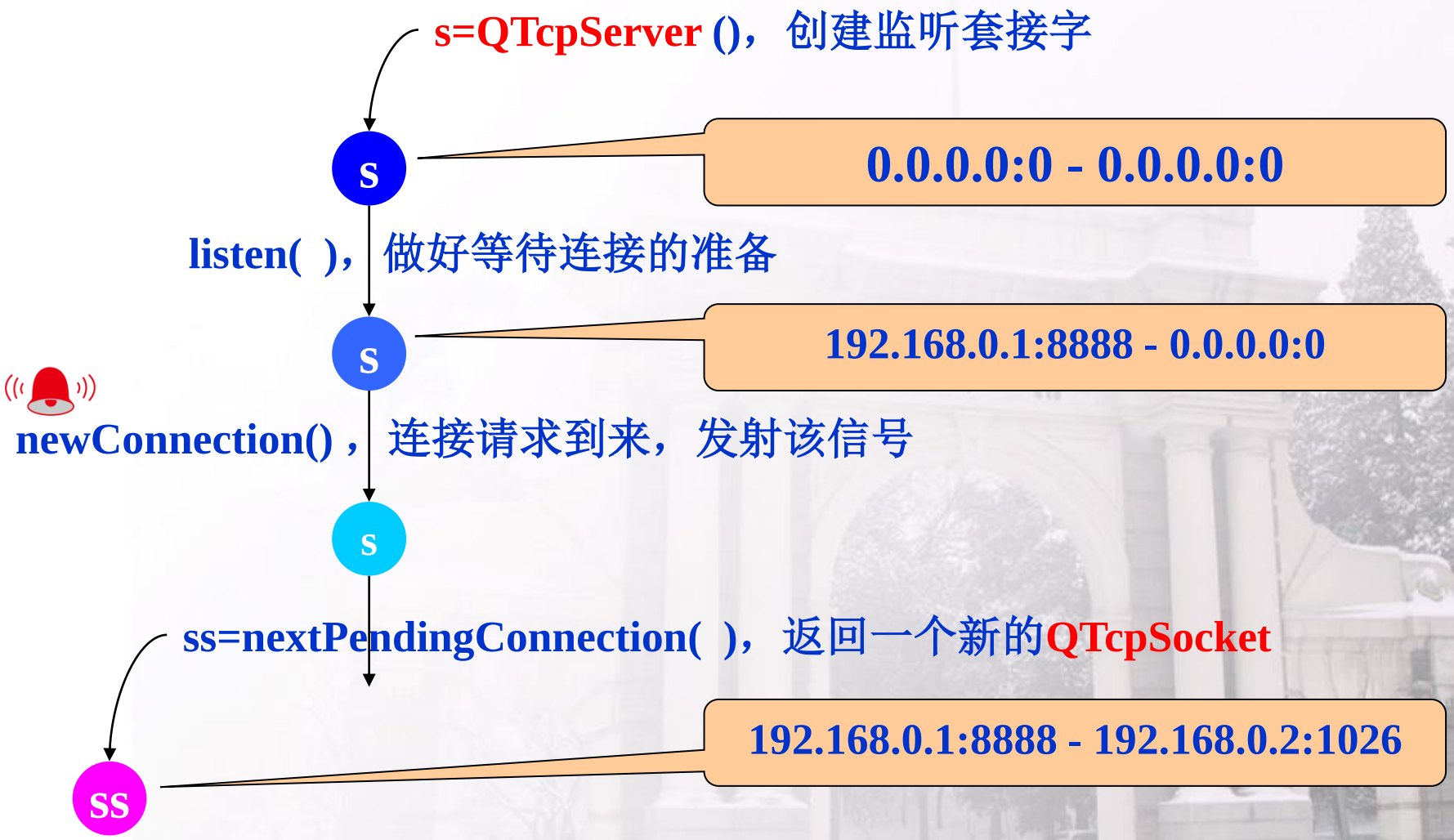
TCP流程：观察地址绑定



服务器
IP地址：192.168.0.1
在8888端口监听

客户端
IP地址：192.168.0.2
连接服务端

TCP流程 - 服务器建立连接

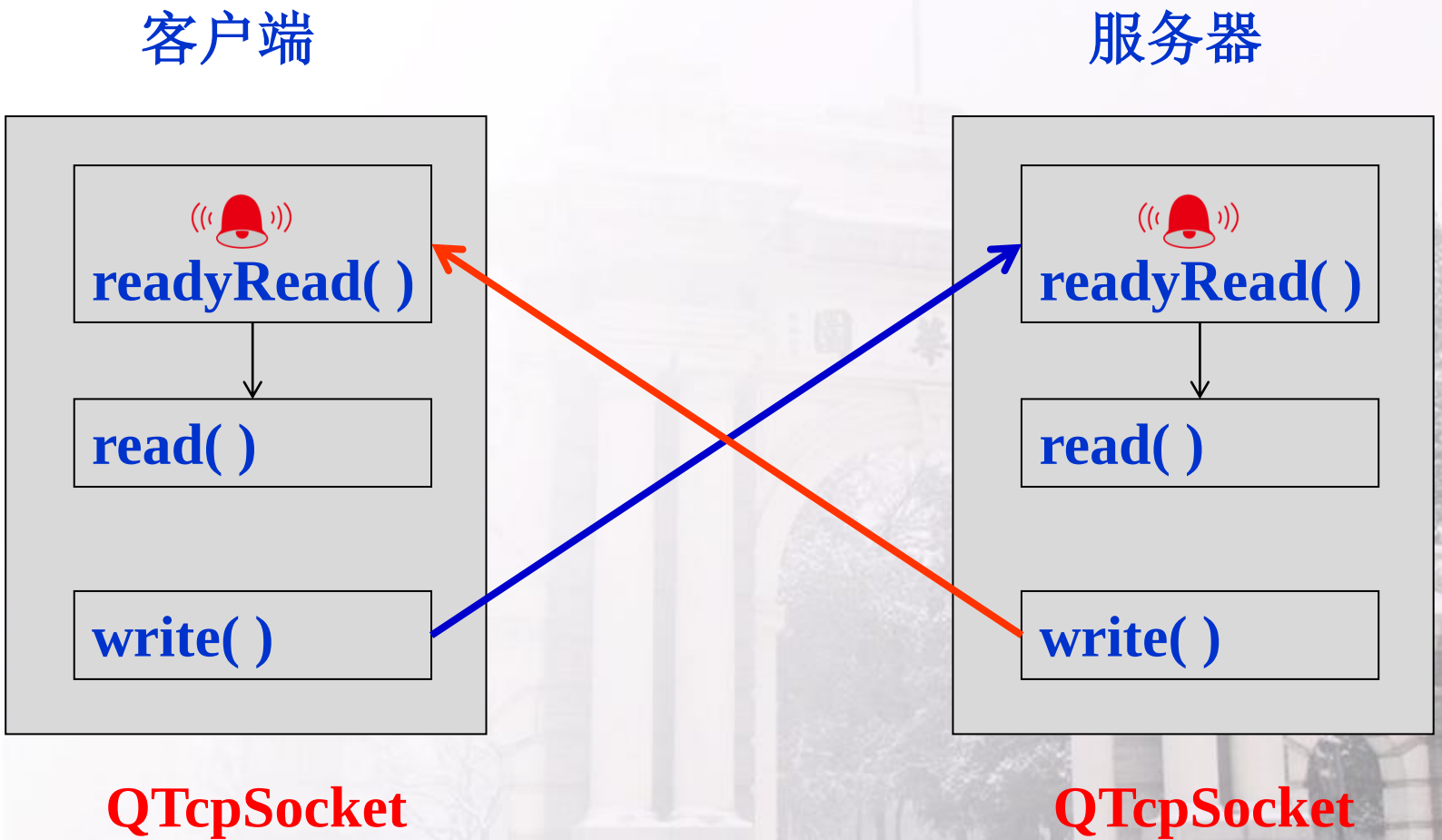


TCP流程 - 客户端建立连接

问题

- 现在，数据的发送和接收还要不要直接指定IP地址和端口了？

TCP流程 – 数据传输


客户端
服务器
readyRead( )
read( )
write( )
readyRead( )
read( )
write( )
QTcpSocket
QTcpSocket

4、QT Socket网络编程——TCP

- QTcpServer常用网络连接函数及信号：

QTcpServer	创建监听套接字
listen	监听连接请求
newConnection	有连接请求到来的信号
nextPendingConnection	接受建立连接请求
close	关闭套接字
waitForNewConnection	阻塞，等待连接请求到来

4、QT Socket网络编程——TCP

- QTcpSocket常用网络连接函数：

QTcpSocket	创建读写套接字
connectToHost	请求与服务器建立连接
connected	连接已经建立，发射信号
readyRead	有新的数据到来，发射信号
read，readData	接收数据
write，writeData	发送数据
close	关闭套接字
waitForConnected	阻塞，等待连接成功
waitForReadyRead	阻塞，等待数据到来

（1） QTcpServer

- **构建一个QTcpServer对象**
  - **QTcpServer::QTcpServer(QObject * parent = 0)**
  - **返回一个监听socket**

- 监听到来的连接请求
  - **Bool listen(const QHostAddress & *address* = QHostAddress::Any, quint16 *port* = 0)**
  - **如果IP地址是缺省值，将监听所有网络接口**
  - **如果端口设定是0，系统将自动选择一个**
  - **成功返回true，失败返回false.**

（1） QTcpServer

- **新的连接请求到来，发射信号**
  - **void QTcpServer::newConnection()**
  - **可以通过QObject::connect建立“signal-slot”连接**

- **接受建立连接请求**
  - **QTcpSocket * QTcpServer::nextPendingConnection()**
  - **返回下一个pending的连接，作为QTcpSocket 对象**
  - **该QTcpSocket 对象是QTcpServer的子对象**

（1） QTcpServer

- 关闭套接字
  - void QTcpServer::close()
  - 关闭监听套接字， QTcpServer将不再监听建立连接请求

- 阻塞，等待连接请求到来
  - bool QTcpServer::waitForNewConnection(int *msec* = 0, bool * *timedOut* = 0)
  - 阻塞等待，直到（1）有连接请求到来；（2）超时
  - 若（1），返回true；否则，返回false

（2）QTcpSocket

- **构建一个QTcpSocket对象**
  - **QTcpSocket::QTcpSocket(QObject * parent = 0)**
  - **该读写套接字，处于未连接状态**

- 向指定服务器发送连接请求
  - **Void connectToHost(const QHostAddress & address, quint16 port, OpenMode openMode = ReadWrite)**
  - 必须指定服务器的**IP**地址和端口

（2）QTcpSocket

- **新的数据到来，发射信号**
  - **void QIODevice::readyRead()**
  - 可以通过**QObject::connect**建立“**signal-slot**”连接

- 接收数据
  - **QByteArray QIODevice::read(qint64 *maxSize*)**
  - **qint64 QIODevice::readData(char * *data*, qint64 *maxSize*)**
  - 最多读**maxSize**个字节

（2）QTcpSocket

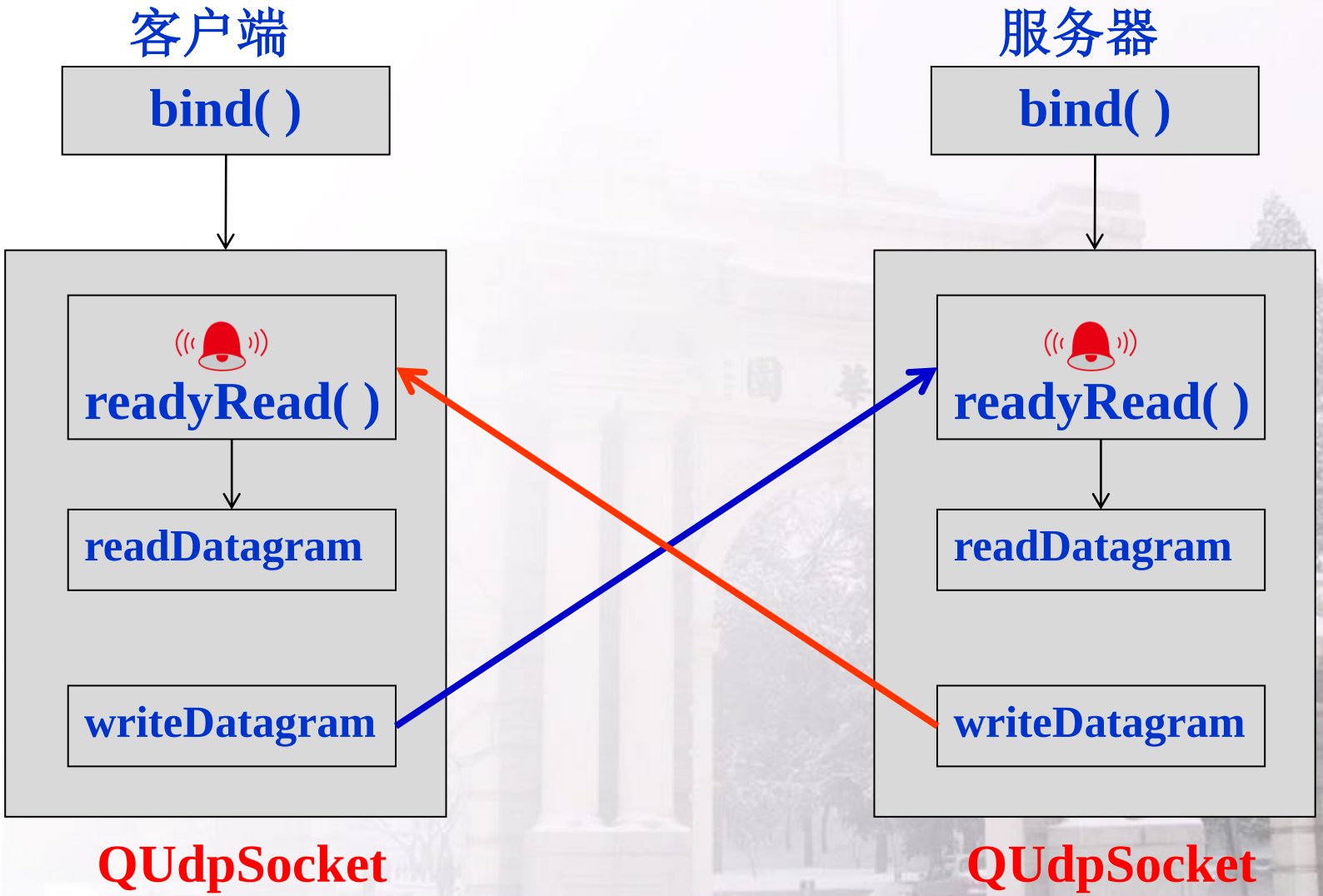
- 关闭套接字
  - void QAbstractSocket::close()
  - 关闭读写套接字， 已有连接将被断开

- 阻塞，等待对方发送的数据到来
  - bool QAbstractSocket::waitForReadyRead(int *msecs* = 30000)
  - 阻塞等待，直到（1）有对方数据到来，ReadyRead信号触发； （2）超时
  - 若（1），返回true；否则，返回false

TCP流程 – 数据传输

- **QTcpServer的基本操作:**
  - 调用**listen**监听端口。
  - 连接信号**newConnection**，在槽函数里调用**nextPendingConnection**获取连接进来的**socket**。
- **QTcpSocket的基本操作:**
  - 调用**connectToHost**连接服务器。
  - 调用**waitForConnected**判断是否连接成功。
  - 连接信号**readyRead**槽函数，异步读取数据。
  - 调用**waitForReadyRead**，阻塞读取数据。

4、无连接的C/S网络通信程序 (UDP)



思考题

- 考虑到**Qapplication**中的事件处理不可阻塞
- 如何在**QT**图形界面的同时实现阻塞的网路通信？

课程要求

- 对于**TCP**通信，网络通信方面只能使用**QTcpServer**和**QTcpSocket**两个类

- 对于**UDP**通信，网络通信方面只能使用**QUdpSocket**这个类

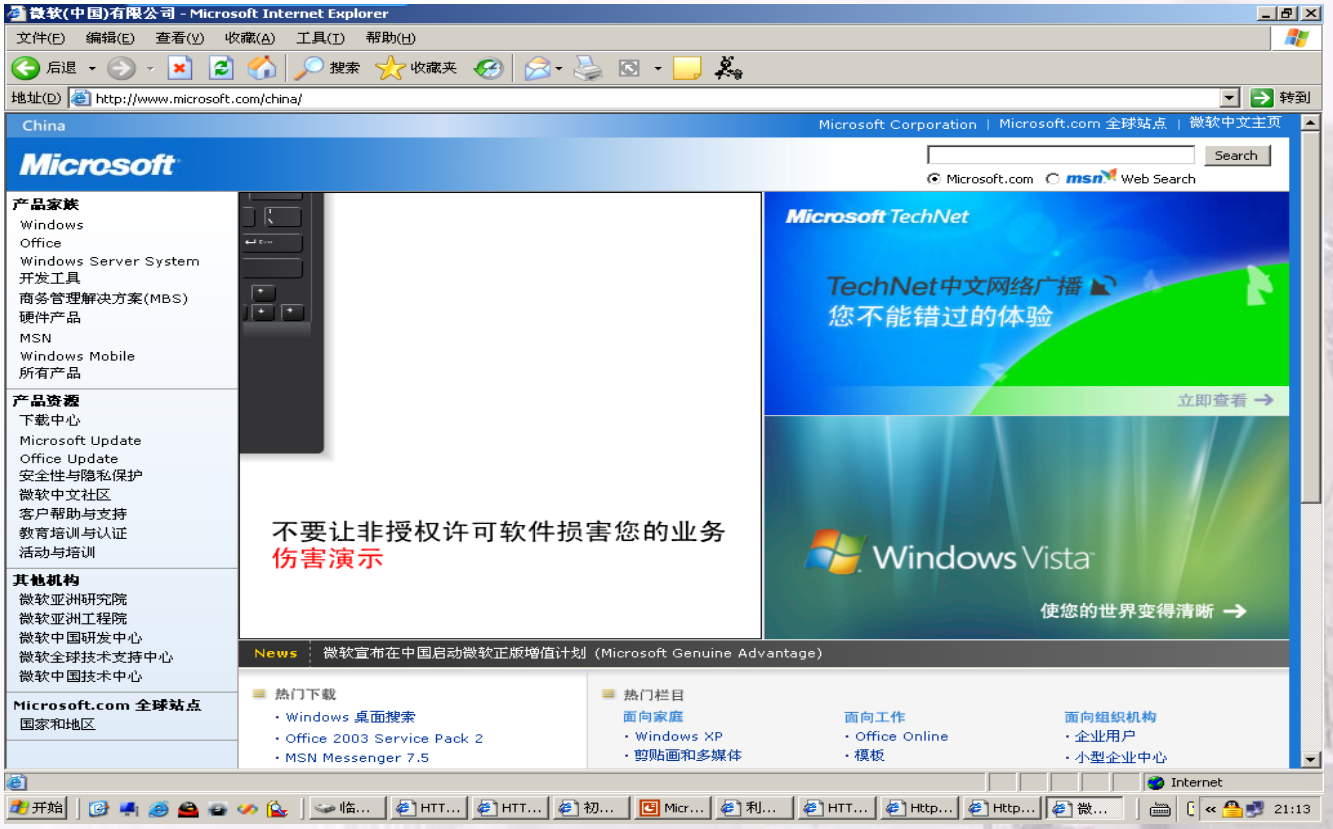
5、用户层网络通信协议

- 网络协议三要素：语法、语义和时序。
  - 语法：规定“如何讲”，即确定数据和控制信息的格式。
  - 语义：规定“讲什么”，即确定通信双方要发出的控制信息，执行的动作和返回的应答。
  - 时序：规定了信息交流的次序。

- HTTP：Hypertext Transfer Protocol
- FTP: File Transfer Protocol
- POP3和SMTP:邮件接收和发送协议
- Telnet:远程登录协议

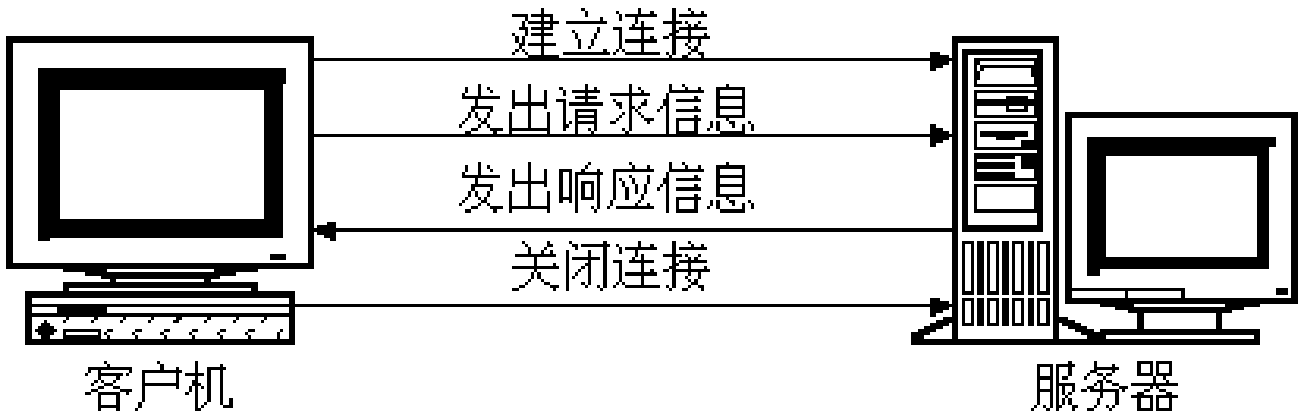
5、HTTP 协议基本原理

- 键入如下网址后，在浏览器中看到如下网页：
  http://www.microsoft.com/china/index.html



5、HTTP 协议基本原理

- 它的原理是：浏览器通过超文本传输协议HTTP,将Web服务器上站点的网页代码提取出来，并翻译成漂亮的网页。
- 一个客户机与服务器建立连接后，发送一个请求给服务器，服务器接到请求后，给与请求的响应信息。



HTTP协议是什么

- Web浏览器和Web服务器之间通过HTTP协议进行通信。
- 它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部份，以及哪部分内容先显示(如文本先于图形)等。

URL是什么

- 在浏览器地址栏里输入的网站地址叫做URL (Uniform Resource Locater,统一资源定位符)。就像每家每户都有一个门牌地址一样，每个网页也都有一个Internet地址。
- URL对网络资源的位置提供了一种抽象的识别方法，并用这种方法给资源定位。这里的资源是指Internet上可以被访问的任何对象，包括文件、文档、图像、声音等等，以及与Internet相连的任何形式的数据。URL是一个字符串

URL的组成格式

- 先看一下刚才打开的**URL**的组成格式：

http://www.microsoft.com/china/index.html

- **1. http:// 代表超文本传输协议，通知服务器显示Web页，通常不用输入；**
- **2. www. microsoft.com是装有网页的服务器的域名，或者站点服务器名称**
- **3. China 为该服务器上的子目录，就好像我们的文件夹**
- **4. Index.html 是文件夹中的一个HTML文件**

超文本标记语言HTML

- HTML文档通过标记（Tag）和属性（Attribute）对超文本的语义进行描述。
- HTML虽然本质上并不是编程语言，但它却是在开发HTML文档时必须遵守的一套严格而且简明易懂的语法规则。
- 也就是说，如果一个文档是基于HTML标准的，则可以解释某些标记的含义。

HTTP服务器活动

- HTTP协议是基于请求/响应范式的。
- HTTP请求分为两种类型，一种是GET请求，另一种是POST请求。
- Web服务器接收到客户请求之后，将根据配置信息执行一定数量的活动。
- 当Web服务器应用程序完成客户请求之后。必须构造一个HTML页面或其他WEB内容，并传输给客户。

5、HTTP 协议基本原理

- 请求的结构

```
Method URL Version
Headers

Message body
```

注意空行

例子

```
POST /TheStockExchange/Trading/GetStockPrice.asp HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

Symbol=MSFT
```

注意空行

HTTP 的 GET 和 POST 方法

- **HTTP-GET**

例子

```
GET /Trading/GetStockPrice.asp?Symbol=MSFT HTTP/1.1
Host: localhost
```

- **HTTP-POST**

例子

```
POST /Trading/GetStockPrice.asp HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 11

Symbol=MSFT
```

GET 和 POST 方法的区别

- GET 方法通常没有消息主体
- GET 方法支持最大1024个字节的查询字符串，POST 方法没有限制
- POST 方法把查询字符串放在消息主体中传输，因此比 GET 方法支持更多的数据类型

5、HTTP 协议基本原理

- 响应的结构

Version Status-Code Description
Headers

Message body

注意空行

例子

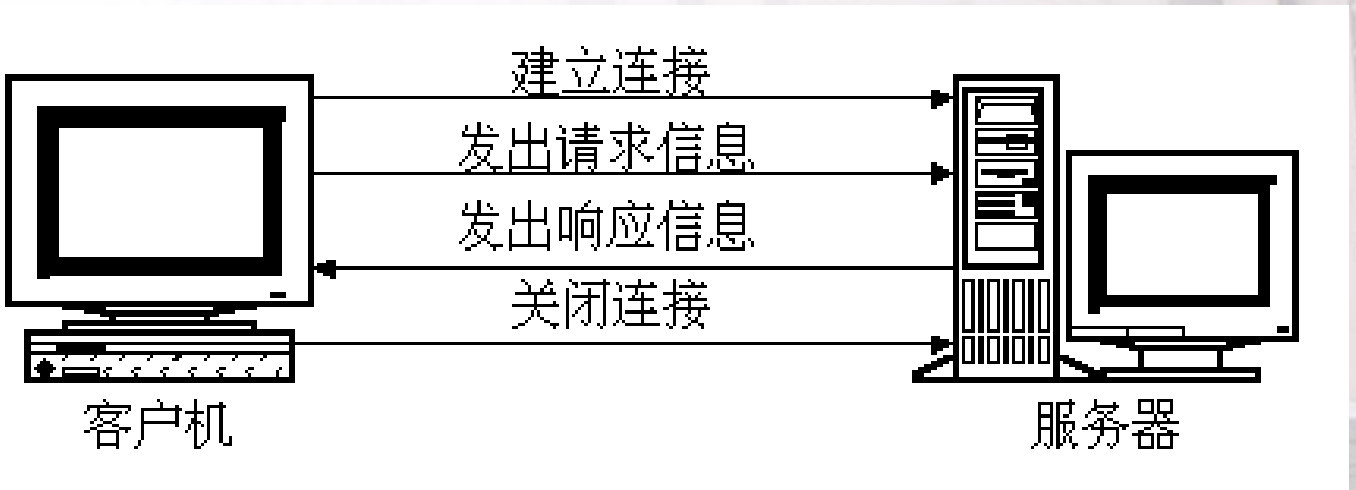
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 75

<?xml version="1.0" encoding="utf-8"?>
<stock symbol="MSFT" Price="71.50" />

注意空行

简单Web Server的工作流程

- 等待Client的连接请求，建立连接；
- 接收Client发来的请求信息
- 解析Client请求信息，并打开所请求文件
- 构建HTTP协议响应头
- 发送响应头和请求文件



Thank you!

Questions?