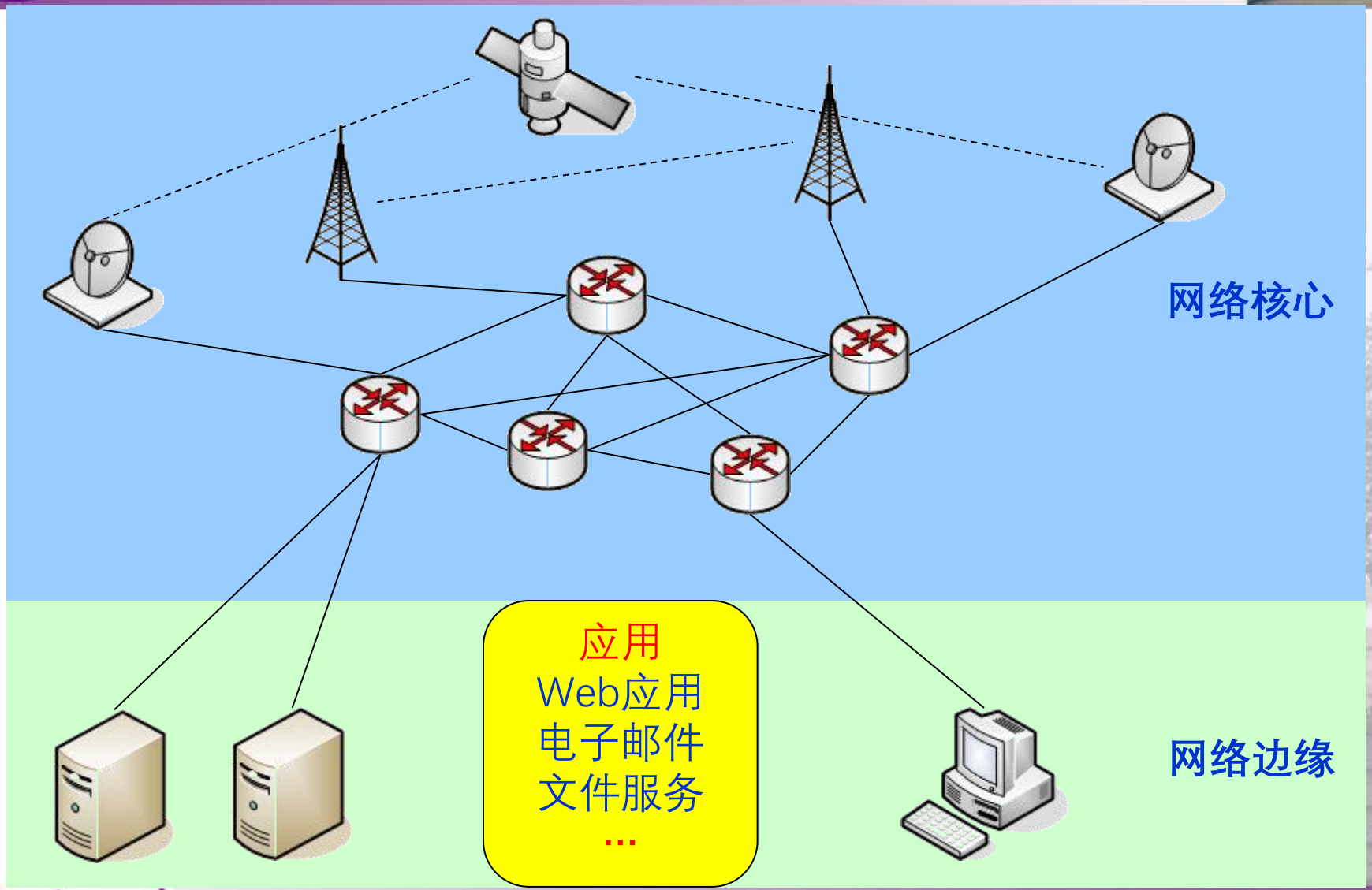
第四讲 网络通信的基本编程

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
- 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

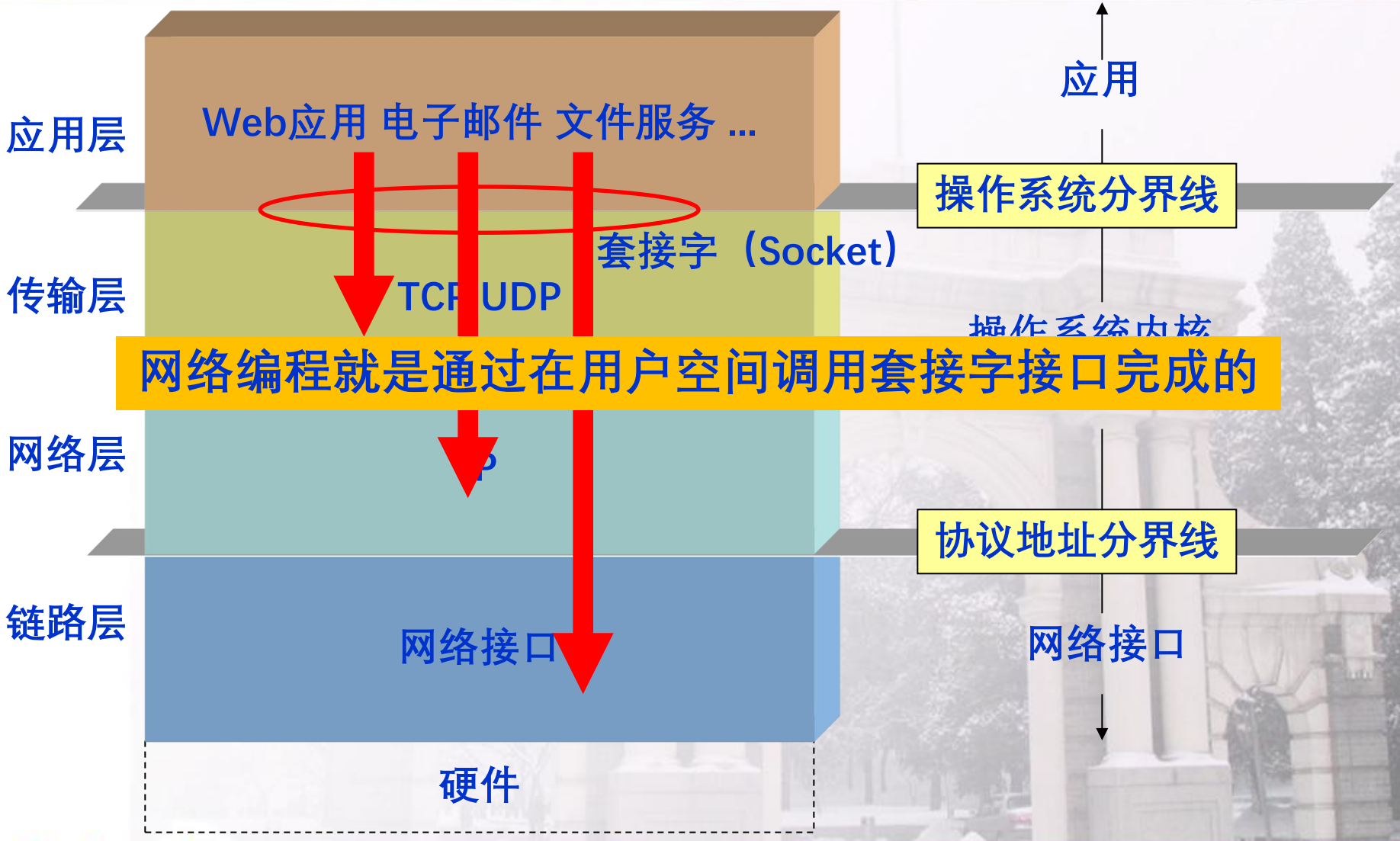
课堂演示：即时通信系统

- 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。

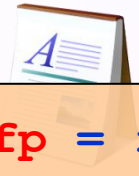
TCP/IP 模型中的两个分界线



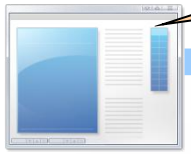
Socket的引入

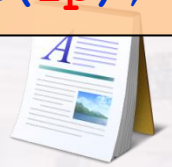
应用程序

?



```
FILE *fp = fopen("a.txt", "r");
...
fscanf(fp,...);
...
fclose(fp);
```





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　　QHostAddress::toIPv4Address
  - 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　　QHostAddress::toString()

1、基本概念

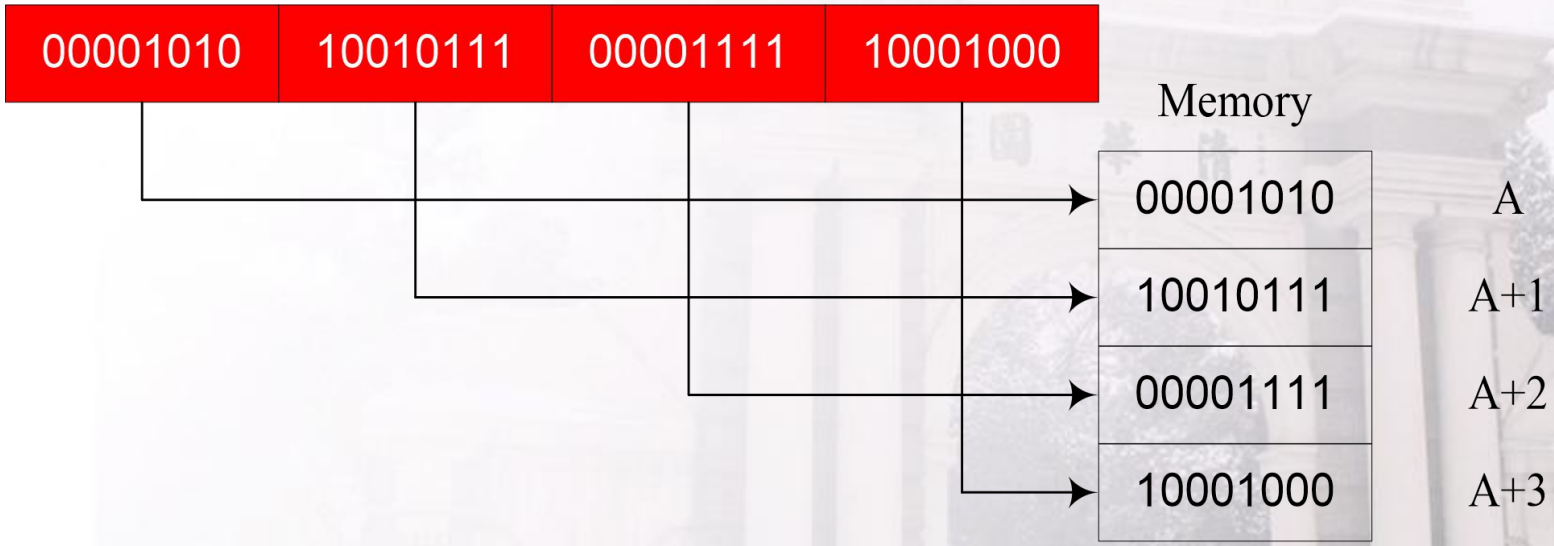
- 端口号
  - 为了区分一台主机接收到的数据包应该递交给哪个进程来进行处理，使用端口号
  - TCP端口号与UDP端口号独立
- 端口号一般由IANA (Internet Assigned Numbers Authority) 管理
  - 众所周知端口：1~1023，1~255之间为大部分众所周知端口，256~1023端口通常由UNIX占用
  - 注册端口：1024~49151
  - 动态或私有端口：49152~65535

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)：高位字节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：进行I/O操作时，函数要等待相关的操作完成以后才能返回。
  - 非阻塞方式：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 网络编程

- QT 网络模块提供了用于编写网络服务客户端和服务器端程序的各种类：
  - QHostInfo：用于查询主机信息
  - QTcpSocket 和 QTcpServer ：用于 TCP 通信
  - QUdpSocket ：用于 UDP 通信
  - 实现HTTP、FTP等普通网络协议的高级类，以及用于网络代理、网络承载管理的类，等等
- 要在程序中使用 QT 网络模块，需要在项目配置文件（*.pro）中增加一条配置语句：

  Qt += network

2、主机信息查询

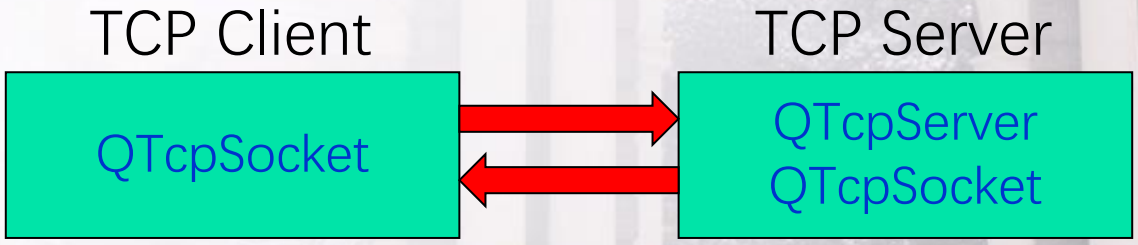
- 查询一个主机的MAC地址或IP地址是网络应用程序中常用到的功能，Qt提供了QHostInfo类可以用于此类信息的查询

QHostInfo 类的主要函数（省略了const关键字）

类别	函数原型	作用
静态函数	void abortHostLookup(int id)	中断主机查找
	QHostInfo fromName(QString &name)	返回指定主机名的IP地址
	QString localHostName()	返回本机主机名
公共函数	QList<QHostAddress> address()	返回主机关联的IP地址列表
	QString hostName()	返回通过IP查找的主机名

3、TCP通信

- TCP 通信概述
  - TCP（Transmission Control Protocol）是一种被大多数Internet网络协议用来数据传输的传输层网络协议
  - 可靠、面向流、面向连接的传输协议，特别适合用于连续数据的传输
  - TCP通信需要首先建立 TCP 连接，服务使用 C/S 模式将通信端分成客户端和服务器端



客户端与服务器端 TCP 通信示意图

3、TCP通信：QTcpServer

- QTcpServer
  - 主要用于构建服务器，建立网络监听，创建网络Socket连接

- 构建一个QTcpServer对象
  - QTcpServer::QTcpServer(QObject * parent = 0)
  - 返回一个表示此TCP Server的对象

3、TCP通信：QTcpServer（续）

- 监听到来的连接请求
- 服务器端程序通常使用QTcpServer::listen()开始服务器监听，可以指定监听的IP地址和端口，一般一个服务程序只监听某个端口的网络连接
  - 公共函数接口：
    - bool listen(const QHostAddress & address = QHostAddress::Any, quint16 port = 0)
  - 在给定IP地址和端口上开始监听，如果成功返回true
  - 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

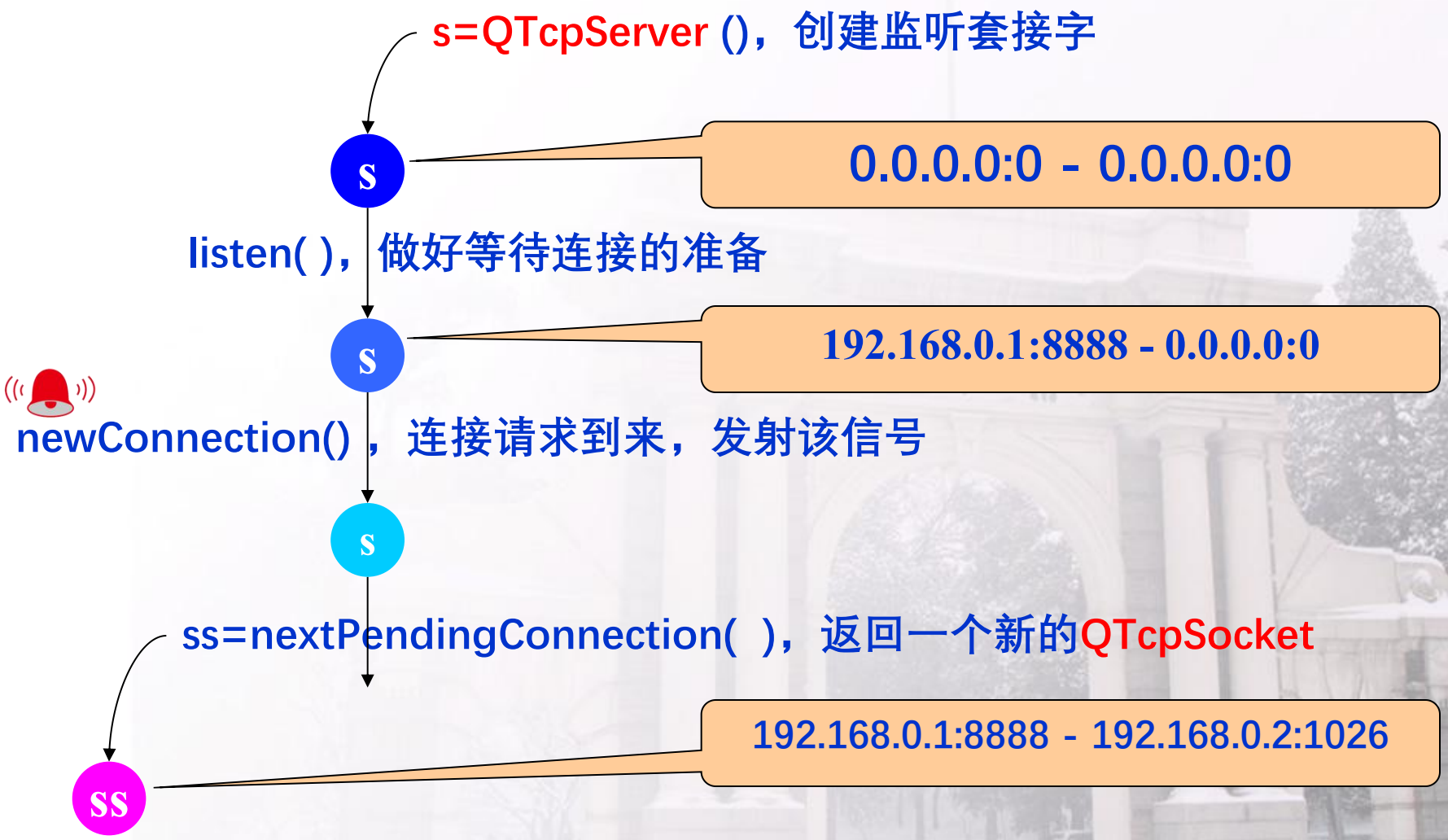
3、TCP通信：QTcpServer（续）

- 处理新客户端接入
  - 当正在监听状态的服务器端程序有客户端接入时，QTcpServer内部的创建一个与客户端连接的QTcpSocket对象，然后发射信号
    - signal：void newConnection()：当有新的连接时此信号发射
  - 在 newConnection() 信号的槽函数中，可以通过 nextPendingConnection() 接受客户端的连接，并获得相关联的 QTcpSocket 对象用于客户端通信
    - 公共函数接口：QTcpSocket* nextPendingConnection()：返回下一个等待接入的连接

3、TCP通信：QTcpServer（续）

- 停止监听
  - QTcpServer 可以直接调用 close() 函数停止监听
    - 公共函数接口：void close()：关闭服务器，停止监听
  - QTcpServer 可以通过 isListening() 判断当前 QTcpServer 是否处于监听状态
    - 公共函数接口：bool isListening()：返回 true 表示服务器处于监听状态
  - 一种推荐的“停止监听”实现方式是先通过 isListening() 判断当前 QTcpServer 是否处于监听状态，如果处于监听状态再通过 close() 函数结束监听
    - if (tcpServer->isListening()) { tcpServer->close(); }

3、TCP通信：QTcpServer（续）



3、TCP通信：QTcpSocket

- **QTcpSocket**
  - **客户端与服务器建立 TCP 连接后，具体的数据通信是通过 QTcpSocket 完成，QTcpSocket 类提供了 TCP 协议的相关接口**

QTcpSocket 类连接相关的主要接口函数（省略了const关键字）

类别	函数	功能
公共函数	void connectToHost(QHostAddress & address, quint16 port)	以异步的方式连接到指定 IP 地址和端口的 TCP服务器
	bool waitForConnected()	等待直到成功建立连接
	void disconnectFromHost()	断开 socket 连接
信号	void connected()	connectToHost() 成功连接后会发射此信号
	void disconnected()	当 socket 断开连接后发射此信号
	void stateChange(QAbstractSocket::SocketState socketState)	当 socket 状态发生变化时需要发射此信号，socketState 表示了 socket 当前的状态

3、TCP通信：QTcpSocket（续）

- 创建 QTcpSocket 连接
  - 客户端首先可以通过调用 QTcpSocket 构造函数创建一个未连接状态的 QTcpSocket 对象
    - 构造函数：QTcpSocket::QTcpSocket(QObject *parent = nullptr)
  - QTcpSocket 对象随后通过 connectToHost 函数尝试连接到服务器，需要指定服务器的地址和端口
  - connectToHost 以异步方式连接服务器，不会阻塞程序运行，如果需要使用阻塞方式连接服务器，则使用 waitForConnect 函数阻塞程序运行
  - 连接成功后会发射 connected() 信号

3、TCP通信：QTcpSocket（续）

- QTcpSocket 状态变化
  - socket 连接状态变化时会发射 stateChange 信号
  - socket 状态包括：UnconnectedState, HostLookupState, ConnectingState, ConnectedState, BoundState, ClosingState, ListeningState

- 断开 QTcpSocket 连接
  - 客户端通过调用 disconnectFromHost() 断开 socket 连接
  - 断开成功时会发送 disconnected() 信号

3、TCP通信：QTcpSocket（续）

- QTcpSocket 数据通信
  - Socket 之间的数据通信协议一般有两种方式，基于数据块或基于行
  - 基于数据块的通信协议用于一般的二进制数据传输，需要自定义具体的格式
    - 以写为例，相关的公共函数接口举例：

qint64 write(const QByteArray &byteArray)：将byteArray中的数据写出，返回真实写出的字节数量或者返回-1标志着出现错误

3、TCP通信：QTcpSocket（续）

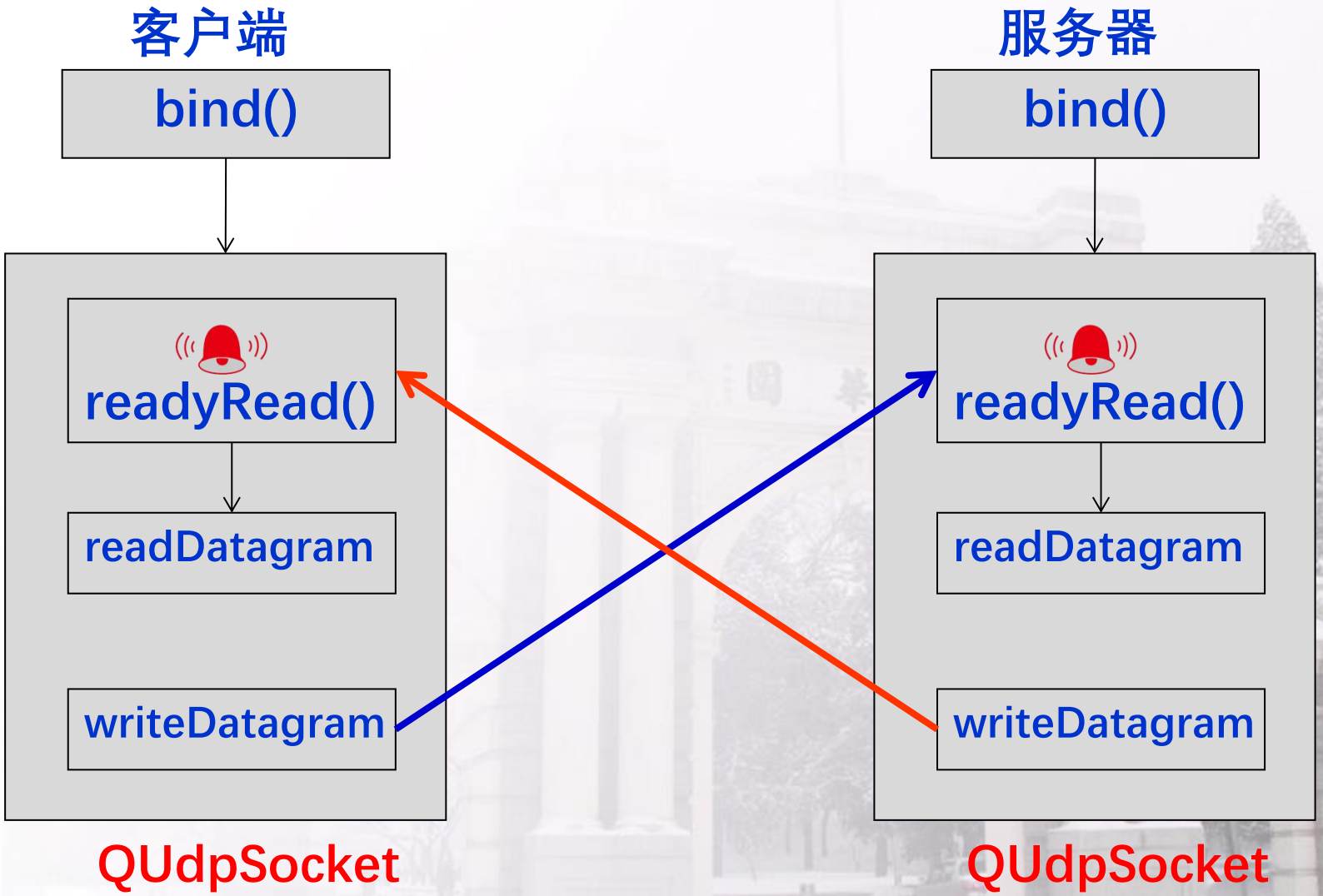
- QTcpSocket 数据通信（续）
  - 基于行的通信协议一半用于纯文本数据的通信，每一行数据以一个换行符结束
  - 以读为例，当QTcpSocket接收到数据后，会发射 readyRead() 信号并在相关联的槽函数里实现数据读操作，相关接口函数：
    - 公共函数接口：bool canReadLine()：如果有行数据需要从Socket缓冲区读取，返回true
    - 公共函数接口：QByteArray readLine()：从缓冲区读取一行数据
    - signal：void readyRead()：当缓冲区有新数据需要读取时发射此信号

3、TCP通信：QTcpSocket（续）

- QTcpSocket 数据通信（续）
  - QTcpSocket类继承或间接继承于一些其他的Qt类，相关联的读写方式也比较多，更多函数接口可以从QT官方文档中获得：
    - Qt官方文档链接

QTcpSocket和QUdpSocket的类继承关系

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

数据传输编写规范

- 发送时：
  - 先将4字节的信息长度，转成网络字节序
  - 然后发送4字节的信息长度内容
  - 最后，发真实内容

- 接收时：
  - 先收4字节的信息长度，并将其转成主机字节序
  - 根据得到的信息长度，收取真实内容

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。

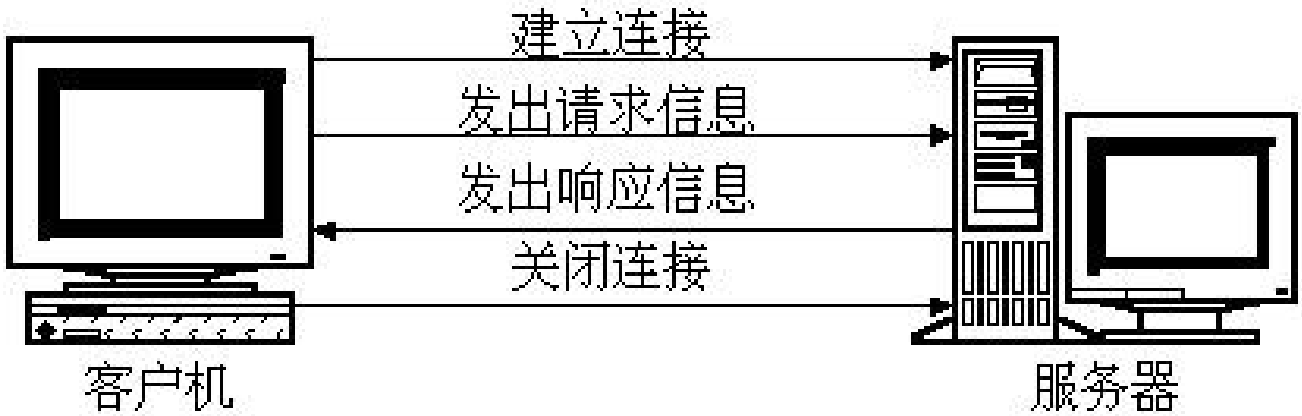
- HTTP：Hypertext Transfer Protocol
- FTP: File Transfer Protocol
- POP3和SMTP:邮件接收和发送协议
- Telnet:远程登录协议

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.cs.tsinghua.edu.cn/index.htm

5、HTTP 协议基本原理

- 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，返回所请求的响应信息。



HTTP协议是什么

- Web浏览器和Web服务器之间通过HTTP协议基于 B/S 架构进行通信
- 它不仅保证计算机正确快速地传输超文本文档，还可以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，服务器通过分析URL定位需要的资源。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.cs.tsinghua.edu.cn/index.htm
- 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.cs.tsinghua.edu.cn是装有网页的服务器的域名，或者站点服务器名称
- 3. Index.htm 是服务器上的一个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 请求有多种类型：
  - HTTP1.0定义了三种请求方法：GET, POST 和 HEAD
  - HTTP1.1新增了五种请求方法：OPTIONS, PUT, DELETE, TRACE 和 CONNECT 方法
- 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

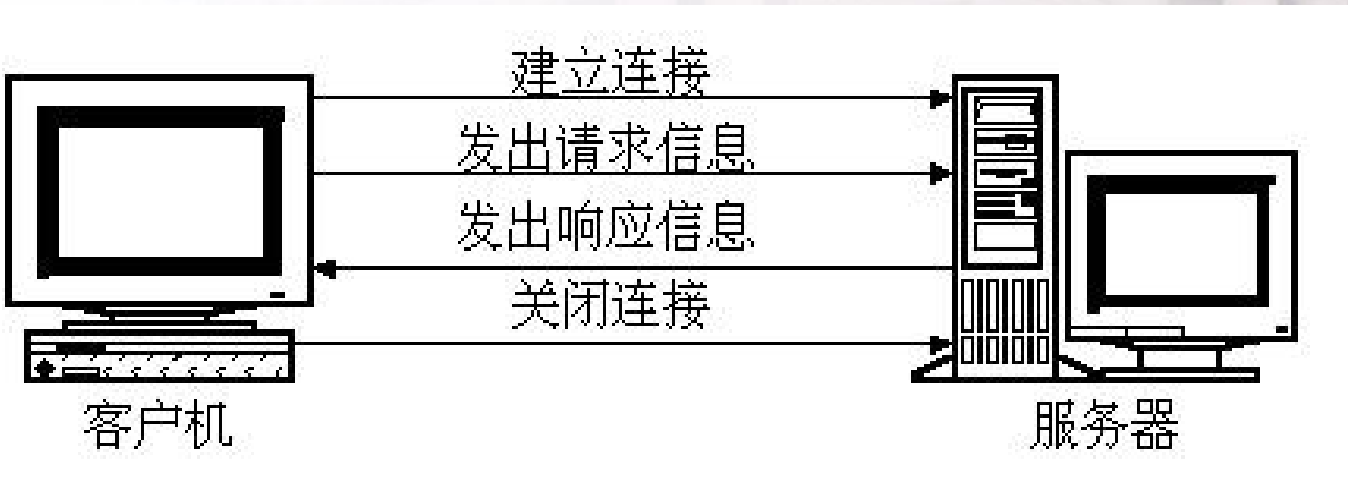
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 75

<?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?