

第二章 应用层

本层原理、流行应用原理和相关协议、TCP 套接字（Socket）编程、UDP 套接字编程

目标：

1. 网络应用的**原理**：网络应用协议的概念和实现。服务模型、两种服务模式、内容分发。
2. 网络应用的**实例**：流行的应用层协议。
3. **编程**-Socket API：原语。

互联网中应用层协议最多：可以自定义功能所以协议特别多。很快地设计和部署新的业务模式，鼓励创新。存在各种各样的网络应用。

创建一个新的网络应用：编程、租服务器、装环境、搭服务器、提供客户端下载。**网络核心中没有应用层软件**，仅在端系统部署，非常块。

2.1 应用层协议原理

体系结构

1. **客户-服务器**模式：（C/S:client/server）

服务器一直运行、IP固定；客户端可以下线、IP可变。一般仅客户端主动与服务器相互通信，间歇性连接，客户端之间不相互通信。——可扩展性差、设备增加性能断崖式下降、可靠性差：服务器坏掉直接无法服务。

2. **对等**模式：（P2P:Peer To Peer）

可以平滑扩展，但是服务能力不稳定、管理困难。

3. **混合体**模式：客户-服务器和对等体系结构。

Naspter（客户端有音乐文件，上线后向服务器发送消息，服务器告诉请求者从哪里下载文件，用户越多下载速度越快——查询是集中式、文件分发是p2p）。

集中通信 QQ 等，在线检测：集中；用户通信：P2P。

进程通信

进程：在主机上运行的应用程序。——同一主机使用操作系统管道即可。

客户端进程、服务器进程。——P2P架构每个会话也有这两种分别。

需要解决的问题：

1. 进程**标示和寻址**。
2. 传输层向应用层提供**服务形式**（接口）。位置：层间界面的SAP 形式：应用程序接口API。
3. 定义**协议**，如何实现应用功能。

问题1：标示和寻址

主机IP、传输层协议（TCP or UDP）、端口号（进程跑在哪里）。

一个进程：用**IP+port**标示 端节点，一对主机进程之间的通信由**2个端节点**构成。

问题2：服务形式

- 穿过的信息：

要传输的报文（本层SDU）、谁传的、传给谁。

传输层实体封装的内容：源端口号，目标端口号，数据等。将IP地址往下交IP实体，用于封装IP数据报。

- 减少信息量（代表层间信息）——**socket：一个整数**

TCP代表四元组（源、目的IP端口）UDP代表二元组（本地的IP和端口）。

socket是**本地**为了便于表示使用的，不是公用的。穿过层间信息少、便于管理。

socket和端口的区别！——其实就是上一章讲的ICI。

- **TCP-socket**

4元组：(源IP, 源port, 目标IP, 目标port)

唯一的指定了一个会话

- **UDP-socket**

但是传输报文时：必须要提供**对方IP, port**。接收报文时：传输层需要上传对方的IP, port。

也就是这两部分不在socket表示，有单独的数据结构来表示。

指定了应用所在的一个端节点（end point）

进程向套接字发送报文或从套接字接收报文——类比于门。

问题3：如何实现（编制协议）

应用层协议：报文格式，解释，时序等。可以进行传输、解析、实现时序。

应用协议仅仅是应用的一个**组成部分**。实体仅是涉及到层间通信的处理部分（运行中的软硬件模块），其他业务部分不是实体。

常见协议：HTTP, SMTP（公用）、Skype（私用）。

其他知识点

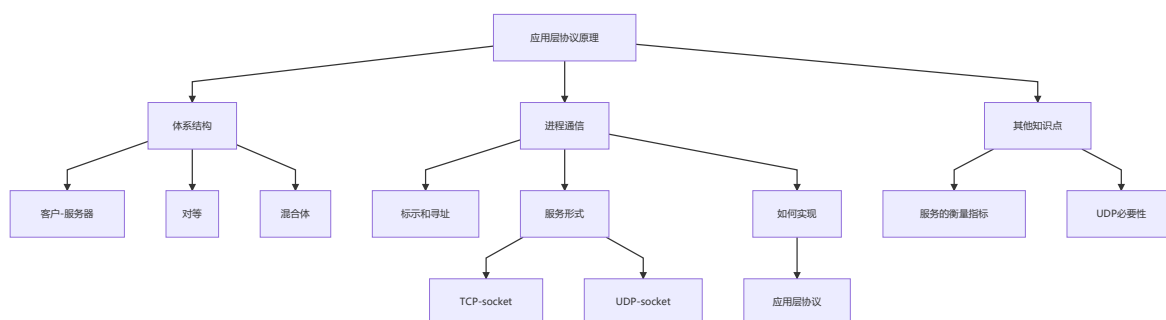
传输层服务的**衡量指标**：数据丢失率、延迟、吞吐、安全性。——常见应用对传输服务的要求

传输层提供的服务：TCP（可靠、流量控制、拥塞控制、面向连接；时间保证、最小吞吐保证和安全无法提供）、UDP（不可靠，没有各种保证、没有链接建立过程）。

UDP存在的必要性：

区分不同的进程、无需建立连接、不做可靠性的工作、能够按照设定的速度发送数据。

安全性：TCP & UDP**都不安全**——明文传输。可以使用SSL，在TCP和应用之间，属于应用层的功能。私密性、完整性鉴别和可认证性。



2.2 Web and HTTP

一些术语

web: 一种**应用**，由对象组成——支持**协议**: **HTTP**。

对象: 各种网页中的资源（使用url来唯一表示）。

基本的HTML文件+引用连接（url）。各种对象通过相互引用构成了网状的信息空间。

url: 统一资源定位系统。

```
Prot://user:psw@www.someSchool.edu/someDept/pic.gif:port
协议名 :// 用户:口令 @ 主机名 / 路径名 :端口
```

注: url支持匿名访问, 所以用户: 口令可以省略。http默认80端口, ftp默认21端口, 所以url可以不指定。

HTTP概况

HTTP: **超文本传输协议**。应用层Web服务的实现协议。

客户/服务器模式: 客户端发送request, 服务器响应response。

两类: 1.0:RFC 1945/ 1.1:RFC 2068

跑在**TCP**协议之上:

1. 客户发起与服务器的 TCP连接 (建立套接字)。
2. 服务器本身有一个守候 (waiting) socket, 客户端发送请求后就建立另一个通信socket。
3. 客户端拿到响应内容之后在本地渲染网页。
4. 完成以后关闭连接。

HTTP是**无状态**的: 服务器并不维护关于客户的任何信息。

——支持更多用户、避免死机同步、避免维护历史信息。

HTTP连接

需要**下层实体进行一次交互**才能完成。

区别:

- 非持久HTTP: 1.0版本, **最多只有一个**对象在 TCP连接上发送。
- 持久HTTP: 1.1版本, **多个对象**可以在一个TCP连接上发送。

非持久:

- 每次都要重复连接建立、接收、请求、响应。
- 往返时间RTT (分组从客户端到服务器再回来-忽略小分组传输时间)。
- 总共2RTT+传输时间。

持久:

- **非流水线** (non-pipeline): 收到前一个响应后 才能发出新的请求, 收到一个对象花费一个RTT。
- **流水线**: 一个引用对象就立即产生一个请求, 甚至可以达到仅用一个RTT。

好处: 可以省掉每次建立连接的一个RTT, 避免并行建立TCP连接占用资源。

HTTP请求报文

分类: **请求**、**响应**。ASCII码编码, 人类可读。

组成：

1. 请求行（GET-请求/POST-上载/HEAD-拿头，文件路径，协议版本号）。
2. 首部行/headline（key:value格式：主机名、浏览器版本、可以设置非持久连接）。
3. 后接回车空行。
4. 实体行（请求报文通常没有）。

提交表单输入：

1. Post方式：输入内容放在实体行
2. url方式：在url后边 ?key=value&key2=value2

方法类型：

- HTTP/1.0: GET POST HEAD
- HTTP/1.1: GET POST HEAD; PUT DELETE (网页管理员使用)

HTTP响应报文

组成：

1. 状态行（协议版本号，状态码和解释）。
2. 首部行（key:value格式：连接方式、服务器版本、上次修改日期-版本号、内容长度-tcp不维护界限、内容类型）。
3. 回车空行。
4. 实体行。

响应状态码：

200 OK; 301 Moved Permanently; 400 Bad Request; 404 Not Found; 505 HTTP Version Not Supported

可以在自己电脑上测试手动发送客户端请求。

cookies

含义：用户-服务器状态——4个组成部分

1. HTTP响应报文有一个cookie的首部行-服务器状态;
2. HTTP请求报文有一个cookie的首部行-客户端状态;
3. 用户端系统存储cookie文件，用于浏览器管理;
4. 在Web站点维护一个后端数据库存各客户cookie。

例子，初始请求无cookie，维护cookie-ID，以后请求就带了这个cookie-ID。

作用：

1. 用户验证
2. 购物车
3. 推荐
4. 用户状态

问题：隐私问题。

Web缓存-cache

目标：不访问原始服务器，就满足客户的请求。

代理服务器将内容缓存下来，客户之间请求代理服务器。命中直接返回——**客户端快，服务器端负载轻，网络负担也减弱**。未命中继续请求原始服务器。

通常缓存是由ISP安装，既是客户端又是服务器。互联网二八定律——20%热点内容80%人访问，使得缓存成为可能。

示例：没有缓存，排队延时接近无限大。两种方式

1. 增大带宽，更大的接入互联网速度。
2. 安装本地缓存，在40%命中率的条件下平均响应时间甚至还减小了。

问题：远程服务器内容变了。解决方法：**条件GET方法。If-modified-since**

