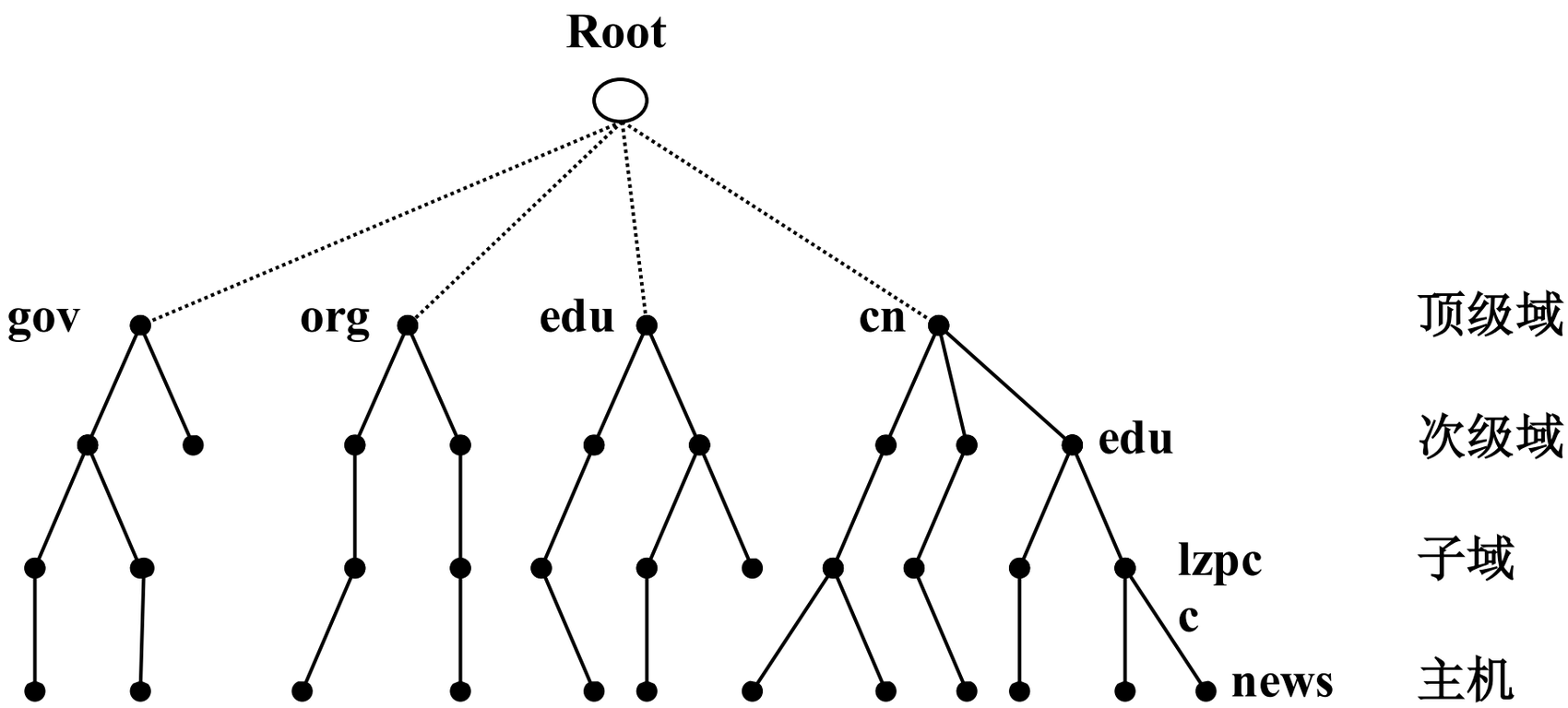
TCP/IP网络基本编程与典型应用

DNS与地址信息查询

- 在IP网络中，IP地址标识不便于记忆和理解，在很多情况下人们使用直观的域名(Domain Name)来标识网络站点
- 在TCP/IP协议中，必须使用IP地址来标识站点，而不能使用域名
- 若使用域名来标识网络站点，通信时须要将域名使用DNS（Domain Name System）映射成IP地址。RFC 1034和RFC 1035文档中详细描述了DNS的功能
- DNS由三部分组成：域名空间、域名服务器和解析器
- 域名空间采用层次结构来组织和管理



域名空间层次结构

- 域名服务器/名字服务器
  - 主名字服务器：每个区域至少有两个主名字服务器来保存所有的区域内信息。一个是主名字服务器，另一个是备份名字服务器，它从主名字服务器那里得到数据
  - 高速缓存（Caching Only）服务器：名字服务器将接收到的信息保存在缓存区中，直到数据失效。此外，还有一种高速缓存名字服务器，它将接收到的信息转发到其它名字服务器上

- 域名服务器续
  - 转发服务器(Forwarding Server)：它建立一个“转发服务表”，记录它的上级名字服务器。当服务器收到地址映射请求时，该服务器按一定的次序依次查询上一级名字服务器，直到查到该数据为止
  - 如果查不到，则返回无此数据的出错信息

DNS与地址信息查询：Socket实现

- Winsock API提供了一组信息查询函数，让我们能方便地获取套接口所需要的网络地址信息以及其它信息，

- gethostname()
  - 用来返回本地计算机的标准主机名。
  - int gethostname(char* name, int namelen);

- （Winsock）调用函数出错的处理
  - 如果没有错误发生，gethostname()返回0
  - 否则它返回SOCKET_ERROR。应用程序可以通过WSAGetLastError()来得到一个特定的错误代码

- WSAEFAULT：名字长度参数太小
- WSANOTINTIALISED：在应用这个API前，必须成功地调用WSAStartup()
- WSAENTDOWN：Windows Sockets实现检测到了网络子系统的错误
- WSAEINPROGRESS：一个阻塞的Windows Sockets操作正在进行

- gethostbyname()
  - 返回对应于给定主机名的主机信息。
  - struct hostent* gethostbyname(const char* name);
- gethostbyaddr()
  - 根据一个IP地址取回相应的主机信息。
  - struct hostent* gethostbyaddr(const char* addr, int len, int type);
    - len：IP地址的长度；type：AF_INET/AF_INET6/…

- （Linux）调用gethostbyname函数出错的处理
  - 如果有错误发生，返回NULL指针
  - 否则它返回hostent指针

- struct hostent结构体（Linux为例）
- struct hostent {

    char *h_name;

    char **h_aliases;

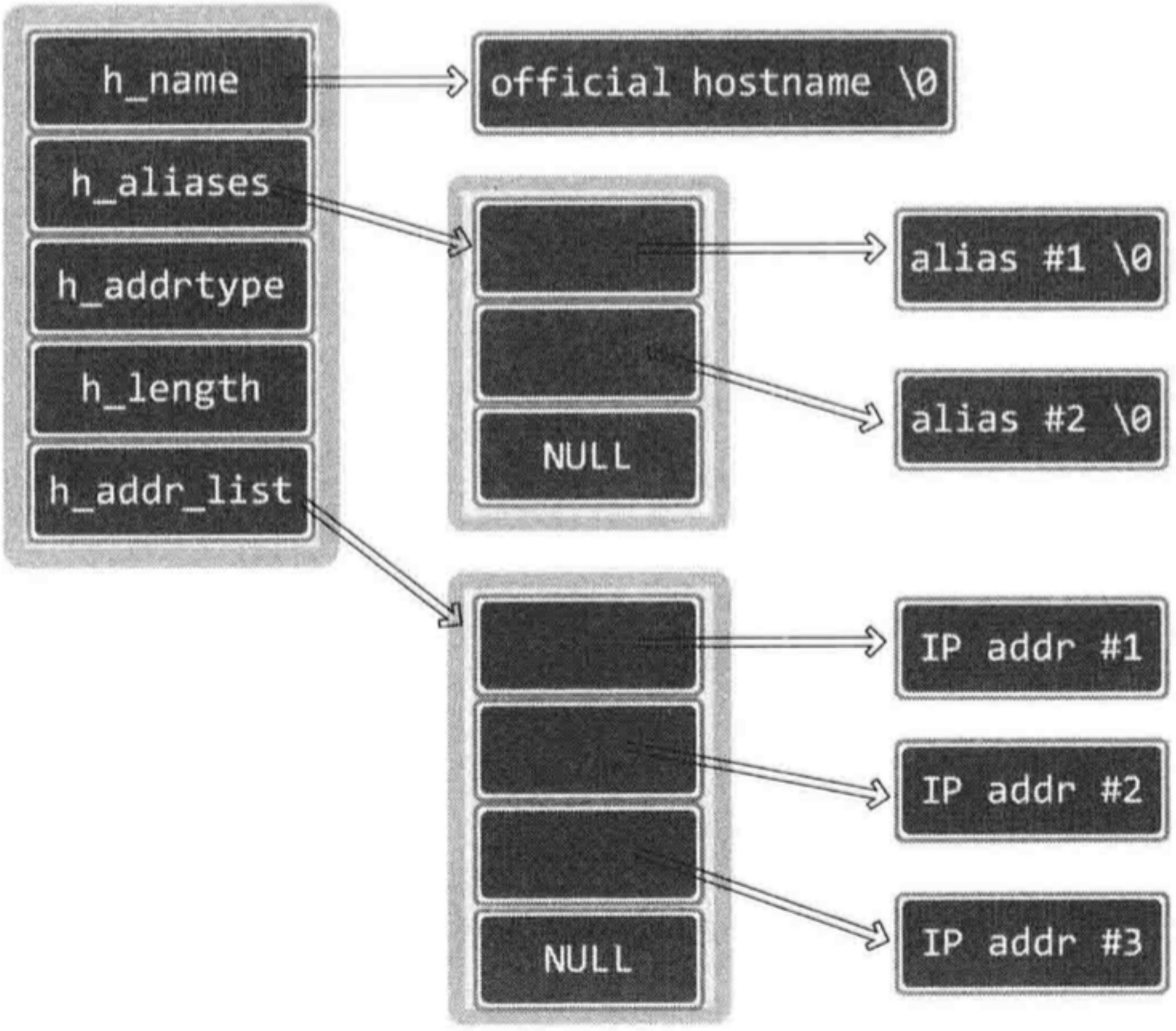
    int h_addrtype;

    int h_length;

    char **h_addr_list;

}

- h_name：official domain name，官方域名
- h_aliases：同一个IP绑定的多个域名别名
- h_addrtype：IP地址族信息，如IPv6是AF_INET6
- h_length：IP地址的长度（字节）
- h_addr_list：保存域名所对应的IP地址们


h_name
h_aliases
h_addrtype
h_length
h_addr_list
official hostname \0
alias #1 \0
alias #2 \0
NULL
IP addr #1
IP addr #2
IP addr #3
NULL

- getprotobyname(…)
  - 返回对应于给定协议名的相关协议信息。
- getprotobynumber (…)
  - 返回对应于给定协议号的相关协议信息。

- 此类函数特点 [除特例以外， 如gethostname()]
  - 函数名都采用getXbyY的形式
  - （winsock）函数成功地执行，返回一个指向包含所需信息的某种结构的指针
  - （winsock）函数执行错误返回空指针。调用WSAGetLastError()得到错误代码。
  - 函数执行时，可能在本地计算机上查询，也可能通过网络向域名服务器发送请求，来获得所需要的信息，取决于用户网络的配置方式

- Winsock中的异步版地址信息查询函数
  - 为了能让程序在等待响应时能作其他的事情，Winsock API扩充了一组作用相同的异步查询函数，不会引起进程的阻塞
  - 使用Windows的消息驱动机制。函数形状与getXbyY各函数对应，在每个函数名前面加上了WSAAsync前缀，名字采用WSAAsyncGetXByY()的形式

IP配置信息管理：GetNetworkParams函数

- （winsock）GetNetworkParams

- DWORD GetNetworkParams(

  Out PFIXED_INFO pFixedInfo,

  _In_ PULONG   pOutBufLen

);

- GetNetworkParams(…)
  - 获取本地计算机的网络参数
  - 参数pFixedInfo
    - 一个指向缓冲的指针，它包含一个FIXED_INFO 结构，如果函数成功的话，该结构接收本地计算机的网络参数。该缓冲必须在之前由GetNetworkParams 函数分配。
  - 参数pOutBufLen
    - 一个指向一个ULONG变量的指针，该变量指定FIXED_INFO结构的大小。如果不够大，则该函数会以合适的大小来填充这个变量，并返回一个ERROR_BUFFER_OVERFLOW的错误代码

IP路由表信息查询与管理

- GetIpAddrTable(…)
- 查询IPv4路由表（接口-IP地址）
- DWORD GetIpAddrTable(

  Out PMIB_IPADDRTABLE pIpAddrTable,

  Inout PULONG pdwSize,

  _In_ BOOL bOrder

  );

- 其中PMIB_IPADDRTABLE的定义
- typedef struct _MIB_IPADDRTABLE {

```
  DWORD         dwNumEntries;

  MIB_IPADDRROW table[ANY_SIZE];

} MIB_IPADDRTABLE, *PMIB_IPADDRTABLE;
```

- AddIPAddrress(...)和DeleteIPAddrress(…)
- DWORD AddIPAddress(

  _In_ IPAddr Address,

  _In_ IPMask IpMask,

  _In_ DWORD IfIndex,

  Out PULONG NTEContext,

  Out PULONG NTEInstance

);

- DWORD DeleteIPAddress(

  _In_ ULONG NTEContext

  );

- 注意：自己建立的才可以删

获取网络管理接口

- DWORD GetNumberOfInterfaces(_Out_ PDWORD pdwNumIf);
  - 获取本地计算机网络接口（Interface）数量的函数

- DWORD GetInterfaceInfo(_Out_ PIP_INTERFACE_INFO pIfTable, _Inout_ PULONG dwOutBufLen);
  - 获取本地主机名、域名和DNS服务器信息的函数

- 获取和设置特定的接口
  - 取本机所有接口的信息
  - DWORD GetIfTable(

```
  _Out_   PMIB_IFTABLE pIfTable,
  _Inout_ PULONG       pdwSize,
  _In_    BOOL         bOrder
);
```

- 获取和设置特定的接口
  - 取一个特定Index接口的信息
  - DWORD GetIfEntry(

    Inout PMIB_IFROW pIfRow

    );

```
typedef struct _MIB_IFROW {
  WCHAR wszName[MAX_INTERFACE_NAME_LEN];
  DWORD dwIndex;
  DWORD dwType;
  DWORD dwMtu;
  DWORD dwSpeed;
  DWORD dwPhysAddrLen;
  BYTE  bPhysAddr[MAXLEN_PHYSADDR];
  DWORD dwAdminStatus;
  DWORD dwOperStatus;
  DWORD dwLastChange;
  DWORD dwInOctets;
  DWORD dwInUcastPkts;
  DWORD dwInNUcastPkts;
  DWORD dwInDiscards;
  DWORD dwInErrors;
  DWORD dwInUnknownProtos;
  DWORD dwOutOctets;
  DWORD dwOutUcastPkts;
  DWORD dwOutNUcastPkts;
  DWORD dwOutDiscards;
  DWORD dwOutErrors;
  DWORD dwOutQLen;
  DWORD dwDescrLen;
  BYTE  bDescr[MAXLEN_IFDESCR];
} MIB_IFROW, *PMIB_IFROW;
```

- 获取和设置特定的接口
  - 设置管理一个特定Index接口的信息
  - DWORD SetIfEntry(

    _In_ PMIB_IFROW pIfRow

    );

ARP操作

- GetIpNetTable(…)
- DWORD GetIpNetTable(

```
  _Out_   PMIB_IPNETTABLE pIpNetTable,
  _Inout_ PULONG          pdwSize,
  _In_    BOOL            bOrder
);
```

- SetIpNetEntry(…)
  - 修改本机ARP表的表项
- DeleteIpNetEntry(…)
  - 删除本机ARP表的表项

- SendARP(…)
- DWORD SendARP(

  _In_ IPAddr DestIP,

  _In_ IPAddr SrcIP,

  Out PULONG pMacAddr,

  Inout PULONG PhyAddrLen

);

应用层网络工作模式

- Client/Server结构
- 典型的两层(double tier)结构的计算机应用模式
- 出现的背景：
  - PC机和工作站的计算能力越来越强
  - 一部分业务处理代码从服务器端移到工作站上，形成了一个单独的层

- C/S结构的系统在性能上有局限性，可表现在以下几个方面：
  - C/S结构的系统受LAN地域范围的限制只能在小范围内使用
  - 系统管理和维护复杂，客户端需要单独安装程序，管理和维护不便
  - 对客户端的要求高，使用系统的用户要经过培训
  - 对网络的性能要求高，有大量的数据要经过网络传输
  - 封闭式系统，不同系统之间无法交流
  - 不同系统用户界面风格不统一，操作复杂不利于推广使用

- Browser/Server模式
  - Internet技术的飞速发展，基于Web的服务被大量的应用
  - Web技术走向统一框架，基于浏览器的应用软件大量涌现，客户端的计算机上只要求安装浏览器（Browser）
  - 用户通过浏览器提出服务请求后由远程的Web服务器响应用户的请求
  - Web服务器需要的数据存储在数据库服务器上——浏览器、Web服务器和数据库服务器组成了一个三层应用系统。


Web页面请求
数据操作请求
浏览器
Web服务器
数据库服务器
页面请求应答
数据获取

- B/S模式的优势

  - 易用性好：客户端用户主要使用单一的浏览器软件，通过鼠标即可访问文本、图像、声音、视频等信息，系统的操作使用简单，特别适合非计算机人员使用。

  - 易于维护：由于客户端除了标准浏览器之外无需专用的软件，系统的维护工作简单

  - 信息共享度高：使用HTML数据格式开放标准，被多数浏览器支持

- B/S模式的优势续
  - 扩展性好：B/S模式使用标准的TCP/IP 协议，通过HTTP能够直接接入Internet，具有良好的扩展性
  - 安全性好：网络安全即软件安全
  - 支持各种网络结构：PSTN 、DDN 、帧中继、X25 、ISDN 、CATV、ADSL等……只要能运行浏览器，均能很好的使用B/S结构
  - 兼容性：由于采用标准的TCP/IP协议，它可以与现有网络很好的结合，尤其对企业节省成本意义很大。

B/S应用程序工作的基本过程

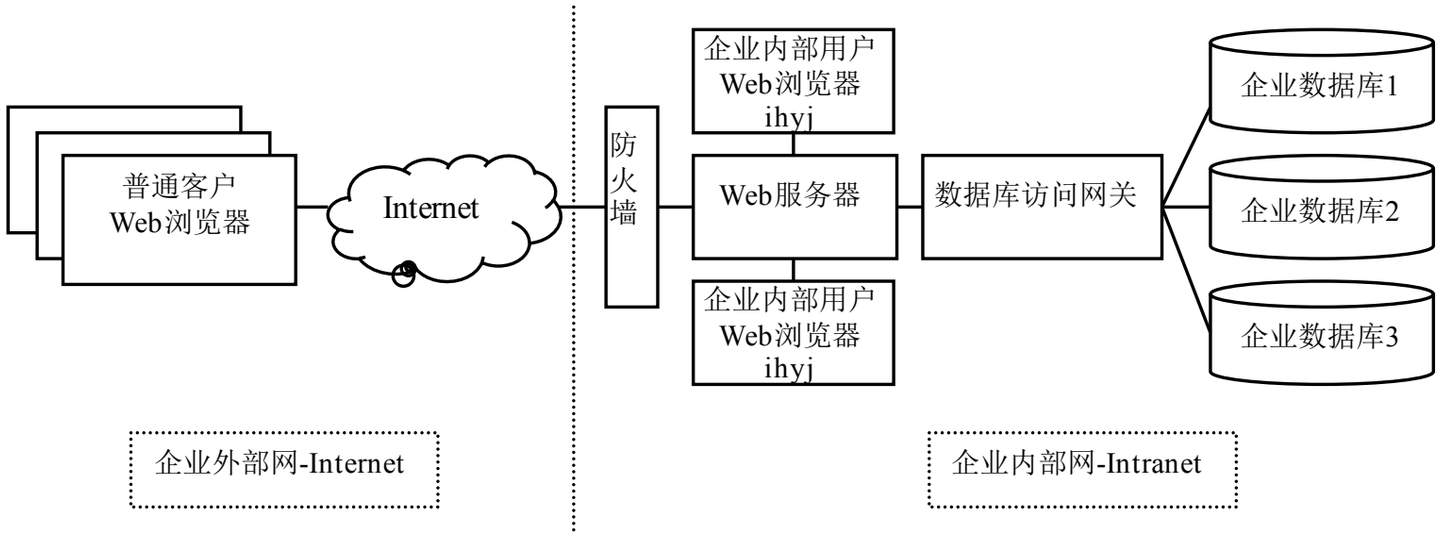
- 第1步：当用户需要某种Web服务时，通过Web浏览器向Web服务器提出请求。请求一般以HTTP协议的形式传输到Web服务器。
- 第2步：Web服务器根据用户的请求，调出相应的HTML、XML、ASP或JSP文件，如果请求的是HTML文档（或XML）则转到第4步，如果请求的是ASP或JSP文档则执行第3步。

- 第3步：Web服务器执行ASP或JSP脚本程序，如果所执行的脚本程序要使用数据库服务器中的数据，则首先要建立Web服务器与数据库服务器之间的连接，然后由脚本程序向数据库服务器的DBMS系统发出操作请求（如SELECT、UPDATE、INSERT和DELETE等）。数据库服务器中的DBMS系统根据请求信息找到所要操作的数据表，并执行相应的操作，然后将操作取得的结果传送到脚本程序。服务器端的脚本程序在取得数据后将生成用户所需的HTML文档，然后执行如下的第4步。

- 第4步：Web服务器将对应的HTML文档以HTTP协议传输到客户端。

- 第5步：客户端的浏览器对接收到的HTML文档进行解释，并通过浏览器将请求得到的信息呈现给用户。

B/S系统的部署结构



- 一个机构使用B/S结构的应用系统时，可根据机构的规模和地理分布情况，使用星形拓扑结构建立内部通信网络Intranet，或利用Internet虚拟专网（VPN）进行通信。

- 内部网Intranet可以通过防火墙接入Internet，整个网络全都使用TCP/IP协议

B/S与C/S：联系与区别

- 从本质上来说，B/S结构还是一种客户机/服务器结构
- 客户机端没有安装专门的应用软件，而是使用浏览器来代替了客户端软件，并且客户端与服务器端的通信协议一般只限于HTTP协议
- 而服务器端的软件则由通用、商业软件公司开发的Web服务器软件和特定的Web应用软件组成
- 因此可以说B/S结构是一种特殊的C/S结构

Telnet

- TELecommunication NETwork protocol，电信网络协议
- 现在一般认为Telnet为远程登录
- 我们主要对应用层的Telnet协议的工作原理进行分析与说明
- Telnet的设计思想对于我们进行网络应用程序的设计具有一定的借鉴意义

Telnet概述

- 一套远程登录的系统需要具备如下条件：

- 具有一个本地系统，它可以是一台终端或一台主机（我们主要讨论主机时的情况）。

- 具有一个远程系统，它是运行着某种操作系统的一台独立的主机。

- 本地系统和远程系统可以互相通信。

- 本地系统的用户在远程多用户系统中有用户账号。

- 本地主机和远程主机可能使用的是不同的操作系统，Telnet需要能够运行在不同操作系统的主机上。
- Telnet由RFC 854规定
  - 工作在应用层的Telnet以TCP为传输层协议，端口23
  - 在本地系统和远程主机之间以半双工的方式进行通信
  - TCP/IP协议簇中提供的标准远程登录协议，几乎每个TCP/IP的实现都提供这个功能

Telnet设计与工作原理

- C/S模式，客户端是本地系统，运行Telnet客户程序。服务器是远程主机系统，运行Telnet服务器进程
- 本地客户Telnet进程首先提出远程登录的请求，远程Telnet服务器进程在23端口进行守候，通过三次握手就在客户传输层与服务器传输层之间建立了一条TCP连接，在此连接上进行它们之间的交互通信

- Telnet设计与工作原理
  - Telnet客户进程工作在应用层，它把收到的数据送到客户TCP,把数据传输到服务器的对等层（即服务器TCP层），再由服务器TCP层将收到的数据送到其应用层的Telnet服务器进程。
  - Telnet服务器进程不能直接处理（解释或执行）收到的数据。，Telnet服务器进程起的作用，就好像是远程主机的一个终端一样，它接收客户端传入的数据，送到服务器端内核（相当于输入设备），同时也接收服务器端内核要送到客户的结果，然后通过它传输到客户端（相当于输出设备）。

Email

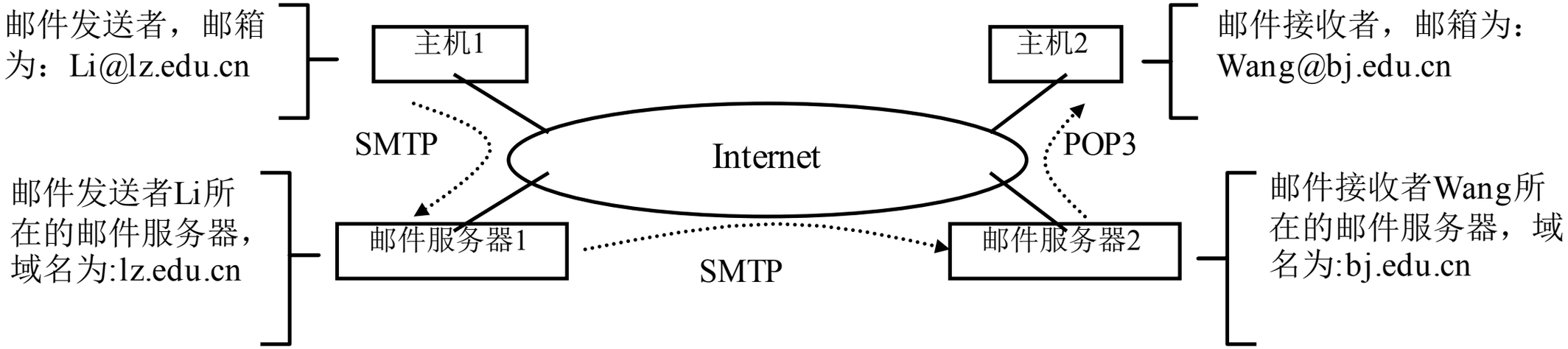
- 电子邮件的内容其实也是以文件的形式存在的，文件传输系统就可以进行电子邮件的传输了

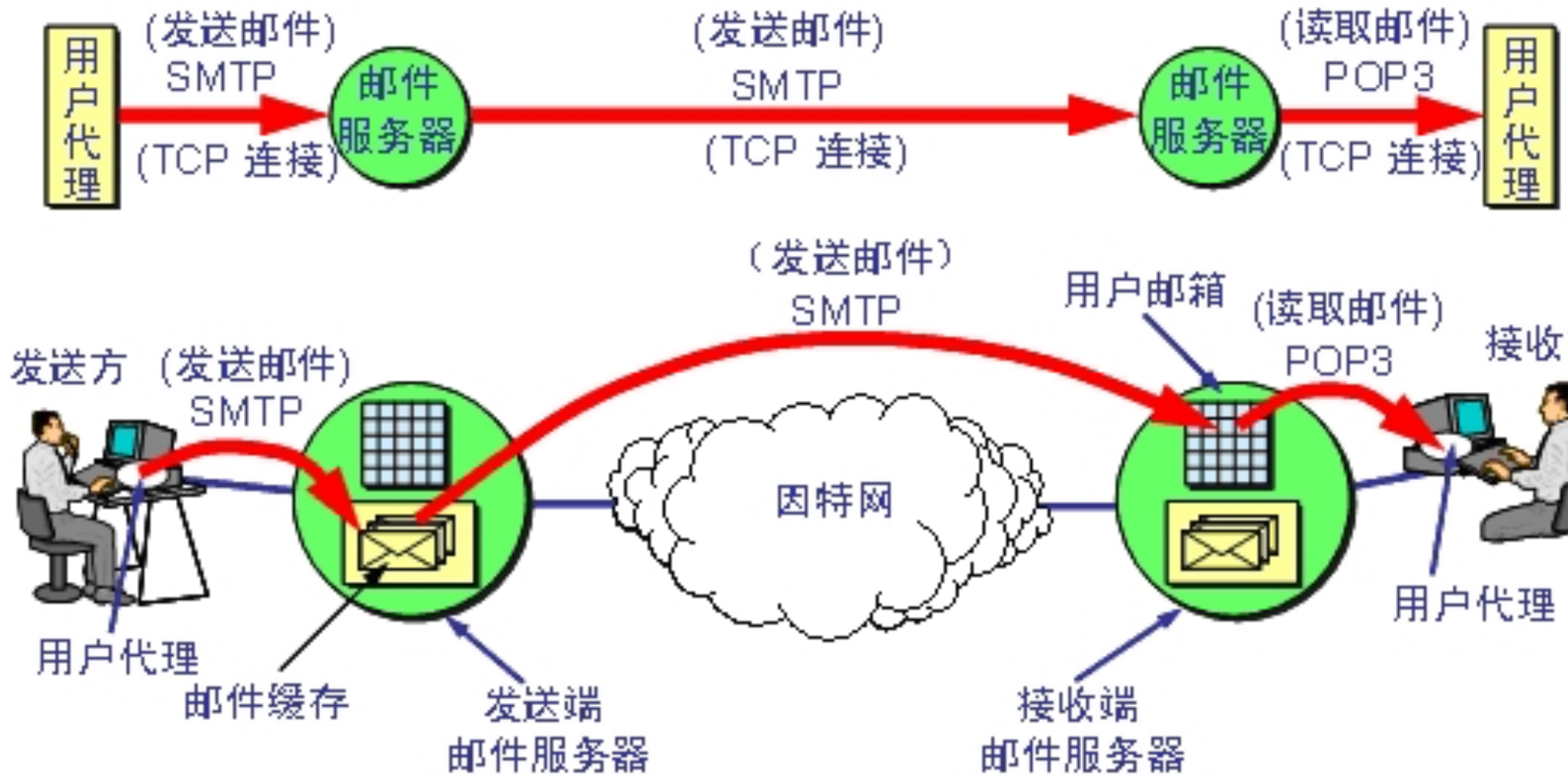
- 为什么还要专门设计电子邮件系统呢?

  - 电子邮件的使用对象是人，因此，email系统应该便于人们对邮件的编写

  - 电子邮件文档具有非常显著的结构化特点，它与我们日常使用的一般邮件类似，有收件人、主题、信件内容、发件人等；三是电子邮件要便于把邮件同时发送给多个用户。另

  - 在Internet中并没有使用文件传输系统来传输电子邮件），而是使用用户操作起来非常简单的专门设计的电子邮件系统进行邮件传输。（网络发展的初期用文件传输协议来传输电子邮件的


邮件发送者，邮箱为：Li@lz.edu.cn
主机1
主机2
邮件接收者，邮箱为：Wang@bj.edu.cn
SMTP
Internet
POP3
邮件发送者Li所在的邮件服务器，域名为:lz.edu.cn
邮件服务器1
邮件服务器2
SMTP
邮件接收者Wang所在的邮件服务器，域名为:bj.edu.cn


用户代理
(发送邮件)
SMTP
(TCP 连接)
邮件服务器
(发送邮件)
SMTP
(TCP 连接)
邮件服务器
(读取邮件)
POP3
(TCP 连接)
用户代理
发送方
(发送邮件)
SMTP
用户代理
邮件缓存
发送端
邮件服务器
(发送邮件)
SMTP
因特网
用户邮箱
接收端
邮件服务器
(读取邮件)
POP3
接收方
用户代理

- SMTP协议
  - 发送端邮件服务器（以下简称客户端）与接收端邮件服务器（以下简称服务器）的25号端口建立TCP连接。
  - 客户端向服务器发送各种命令，来请求各种服务（如认证、指定发送人和接收人）。
  - 服务器解析用户的命令，做出相应动作并返回给客户端一个响应
  - 上两步交替进行，直到所有邮件都发送完或连接被意外中断

C：telnet smtp.126.com 25　/*　以telnet方式连接126邮件服务器　*/

S：220 126.com Anti-spam GT for Coremail System (126com[071018])　/* 220为响应数字，其后的为欢迎信息，会应服务器不同而不同*/

C：HELO smtp.126.com /* HELO 后用来填写返回域名（具体含义请参阅RFC821），但该命令并不检查后面的参数 */

S：250 OK

C: MAIL FROM: bripengandre@126.com /* 发送者邮箱 */

S：250 … /* "…"代表省略了一些可读信息 */

C：RCPT TO: bripengandre@smail.hust.edu.cn　/* 接收者邮箱 */

S：250 … /* "…"代表省略了一些可读信息 */

C：DATA /* 请求发送数据 */

S：354 Enter mail, end with "." on a line by itself

C：Enjoy Protocol Studing

C：.

S：250 Message sent

C：QUIT /* 退出连接 */

S：221 Bye

- 常用SMTP命令
  - HELO <domain> <CRLF>。向服务器标识用户身份发送者能欺骗，说谎，但一般情况下服务器都能检测到。
  - MAIL FROM: <reverse-path> <CRLF>。<reverse-path>为发送者地址，此命令用来初始化邮件传输，即用来对所有的状态和缓冲区进行初始化。
  - RCPT TO：<forward-path> <CRLF>。 <forward-path>用来标志邮件接收者的地址，常用在MAIL FROM后，可以有多个RCPT TO。
  - DATA <CRLF>。将之后的数据作为数据发送，以<CRLF>.<CRLF>标志数据的结尾。

- 常用SMTP命令续
  - REST <CRLF>。重置会话，当前传输被取消。
  - NOOP <CRLF>。要求服务器返回OK应答，一般用作测试。
  - QUIT <CRLF>。结束会话。
  - VRFY <string> <CRLF>。验证指定的邮箱是否存在，由于安全方面的原因，服务器大多禁止此命令。
  - EXPN <string> <CRLF>。验证给定的邮箱列表是否存在，由于安全方面的原因，服务器大多禁止此命令。
  - HELP <CRLF>。查询服务器支持什么命令。

- POP3协议
  - 用户运行客户端软件（如Foxmail, Outlook）
  - 客户端与邮件服务器的110端口建立TCP连接
  - 客户端向服务器端发出各种命令，来请求各种服务（如查询邮箱信息，下载某封邮件等）。
  - 服务端解析用户的命令，做出相应动作并返回给客户端一个响应。
  - 以上两步交替进行，直到接收完所有邮件，或两者的连接被意外中断而直接退出
  - 用户代理解析从服务器端获得的邮件，以适当地形式（如可读）的形式呈现给用户

C：telnet pop3.126.com 110 /* 以telnet方式连接126邮件服务器 */

S：+OK Welcome to coremail Mail Pop3 Server (126coms[3adb99eb4207ae5256632eecb8f8b4855]) /* +OK,代表命令成功，其后的信息则随服务器的不同而不同*/

C：USER bripengandre /* 采用明文认证 */

S：+OK core mail

C：PASS Pop3world /* 发送邮箱密码 */

S：+OK 654 message(s) [30930370 byte(s)] /* 认证成功，转入处理状态 */

C：LIST 1 /* 显示第一封邮件的信息 */

S：+OK 1 5184 ./* 第一封邮件的大小为5184 字节 */

C：UIDL 1 /* 返回第一封邮件的唯一标识符 */

S：+OK 1 1tbisBsHaEX9byI9EQAAsd /* 数字1 后的长字符串就是第一封邮件的唯一标志符 */

C：RETR 1 /* 下载第一封邮件 */

S：+OK 5184 octets /* 第一封邮件的大小为5184字节 */

S：Receive… /* 第一封邮件的具体内容 */

S：…

C：QUIT /* 转入更新状态，接着再转入认证状态 */

S：+OK

C: QUIT /* 退出连接 */

S：+OK core mail /* 成功地退出了连接 */

FTP

- 在Internet上有大量的FTP服务器，提供了非常丰富的各种资源。即使WWW广泛使用的现在，FTP仍然是Internet最主要的应用之一

- 文件传输协议FTP（File Transfer Protocol，由RFC959描述）工作在TCP/IP协议簇的应用层，其传输层使用的是TCP协议，基于C/S模式工作

- FTP用来把一台主机上的文件要传到另外一台主机上

- FTP可传输的文件类型
  - ASCII码文件：这是FTP默认的文本文件格式
  - EBCDIC码文件：同样为文本类型文件，用8位代码表示一个字符，传输时要求两端都使用EBCDIC码
  - 图像（Image）文件：为二进制文件类型，以字节为单位进行传输
  - 本地（Local）文件：字节的大小由本地发送主机定义，对于使用8位字节的系统来说，本地文件以8位字节传输就等同于图像文件传输

WWW和HTTP

- WWW的工作过程
  - 用户首先要确定网页文件所在的URL，由URL惟一确定用户要访问的文件在Internet上的位置
  - 浏览器（相当于客户）向DNS（域名服务器）发出请求，要求把域名www.bing.com转化为它所对应的IP地址。
  - DNS进行查询后，向浏览器发出应答，应答内容为IP地址
  - 在查询得到网页所在的服务器IP地址后，就进入HTTP协议的工作阶段。浏览器向IP地址为23.76.34.3的主机发出与端口80建立一条TCP连接的请求。80端口是服务器提供Web服务的端口

- WWW工作过程（HTTP协议部分）
  - 连接建立成功后，浏览器发出一条请求传输网页的HTTP命令，格式为：GET /somefolder/somepage.html
  - 当服务器收到请求后，向浏览器发送somepage.html文件。
  - 文件发送完成后，由服务器主动关闭TCP连接。至此HTTP的工作过程也结束了。浏览器显示收到的网页文件somepage.html
  - 如果somepage.html文件中包含有图片，还要与服务器建立TCP连接以下载图片。

- 超文本传输协议HTTP（Hypertext Transfer Protocol）1990年提出。目前在WWW中使用的主要是HTTP/1.0和HTTP/1.1

- HTTP协议工作于C/S模式，浏览器就是客户，接受连接并对请求返回信息的应用程序是Web服务器，Web服务器在TCP端口80监听客户的请求。

- 常用的浏览器相当于一个用户代理(User agent)，用户要求完成的各种操作是由用户代理向Web服务器提出，并处理由Web服务器传输到客户的响应。

- 在客户与服务器建立连接后，客户发送请求服务的方法，服务器进行应答。

完