本次课的主要内容

第2章 应用层

2.1 应用层协议原理

2.2 Web和HTTP

2.3 因特网中的电子邮件

- SMTP, POP3, IMAP

2.4 DNS：因特网的目录服务

2.5 P2P文件分发

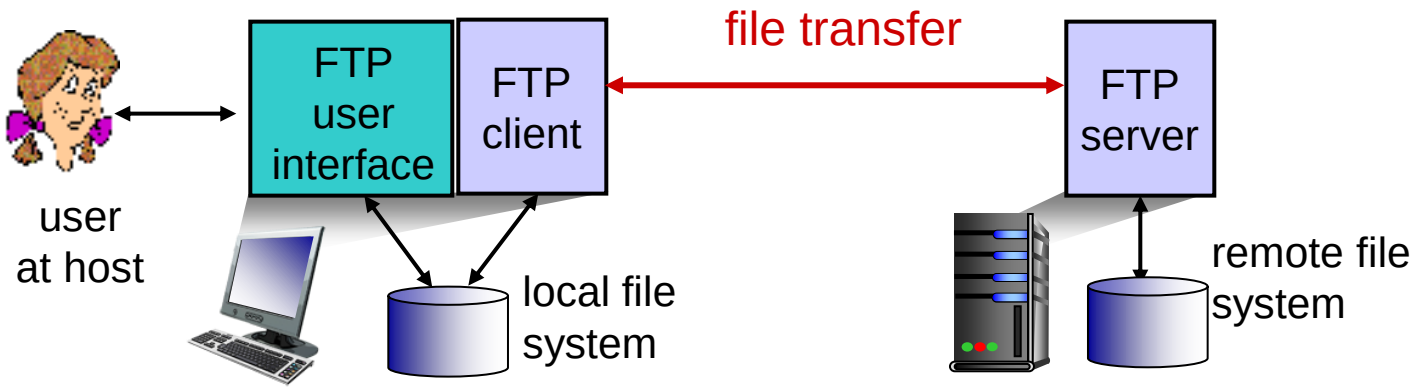
2.6 视频流和内容分发网(自学)

2.7 套接字编程：生成网络应用

2.8 小结

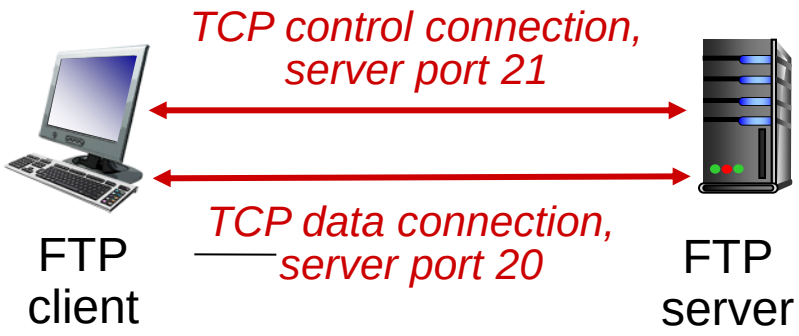
FTP: 文件传输协议

- 传输文件到/从远程的主机
- C/S架构
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- FTP：RFC 959
- FTP 服务器：端口号(21/20)



FTP: 分离的控制和数据连接

- FTP客户机向FTP服务器的21端口发起TCP连接请求；
- 客户端通过控制连接来完成认证(传输用户名/密码)；
- 客户端可以浏览远程的目录，基于控制连接发送命令；
- 当服务器收到文件传输命令，服务器打开与客户端的第二个TCP数据连接(用于传输数据)；
- 当一个文件传输完成后，服务器会关闭掉数据连接；
- 服务器需要使用另一个TCP数据连接来传输另一个文件；



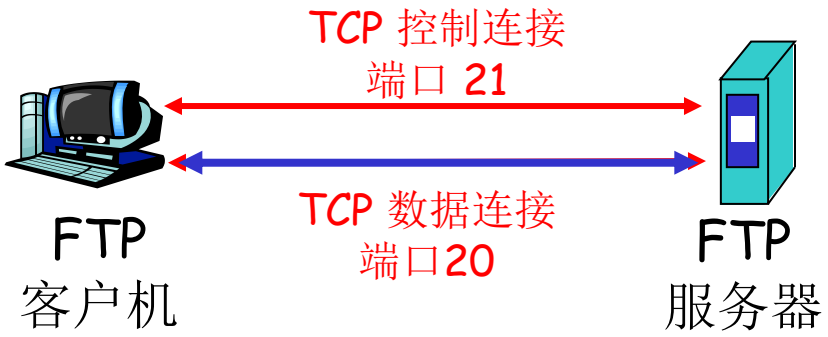
- 控制连接: “带外传输”
- FTP服务器维护“状态”：当前的目录，早期的认证信息等

FTP：控制连接

- 用于在两主机间传输控制信息（如用户标识、口令等）
  - FTP会话开始前，FTP的客户机与服务器在21号端口上建立。
  - FTP的客户机通过该连接发送用户标识和口令，或改变远程目录的命令。远程主机上的目录信息通过控制连接发送给客户机

FTP：数据连接

- 用于准确传输文件。
  - 当服务器收到一个文件传输的命令后(从远程主机上读或写)，在20端口发起一个到客户机的数据连接。
  - 在该数据连接上传送一个文件并关闭连接。
- 控制连接是持久的：在整个用户会话期间一直保持；
- 数据连接是非持久的：会话中每进行一个文件传输，都需要建立一个新的数据连接。



FTP：与HTTP的比较

- FTP的控制信息是带外传送（out-of-band）：使用分离的控制连接；
- HTTP的控制信息是带内传输(in-band)：请求和响应都是在传输文件的TCP连接中发送。
- FTP协议是有状态的：FTP服务器对每个活动用户会话的状态进行追踪，并保留；限制同时会话的总数
- HTTP协议是无状态的：不对用户状态进行追踪。

怎么理解有状态？我们传输文件的时候可以断点续传，即服务器会维护一个文件的传输的状态，已经发送的数据

FTP：命令和响应

sample commands:

- sent as ASCII text over control channel
- **USER *username***
- **PASS *password***
- **LIST** return list of file in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

2.3 因特网中的电子邮件

我(mjliu@uestc.edu.cn)通过web邮件的形式，发送一封邮件给助教李家兴(514151283@qq.com)，分析这个案例中，我的邮件的传递过程。

步骤1:

步骤2:

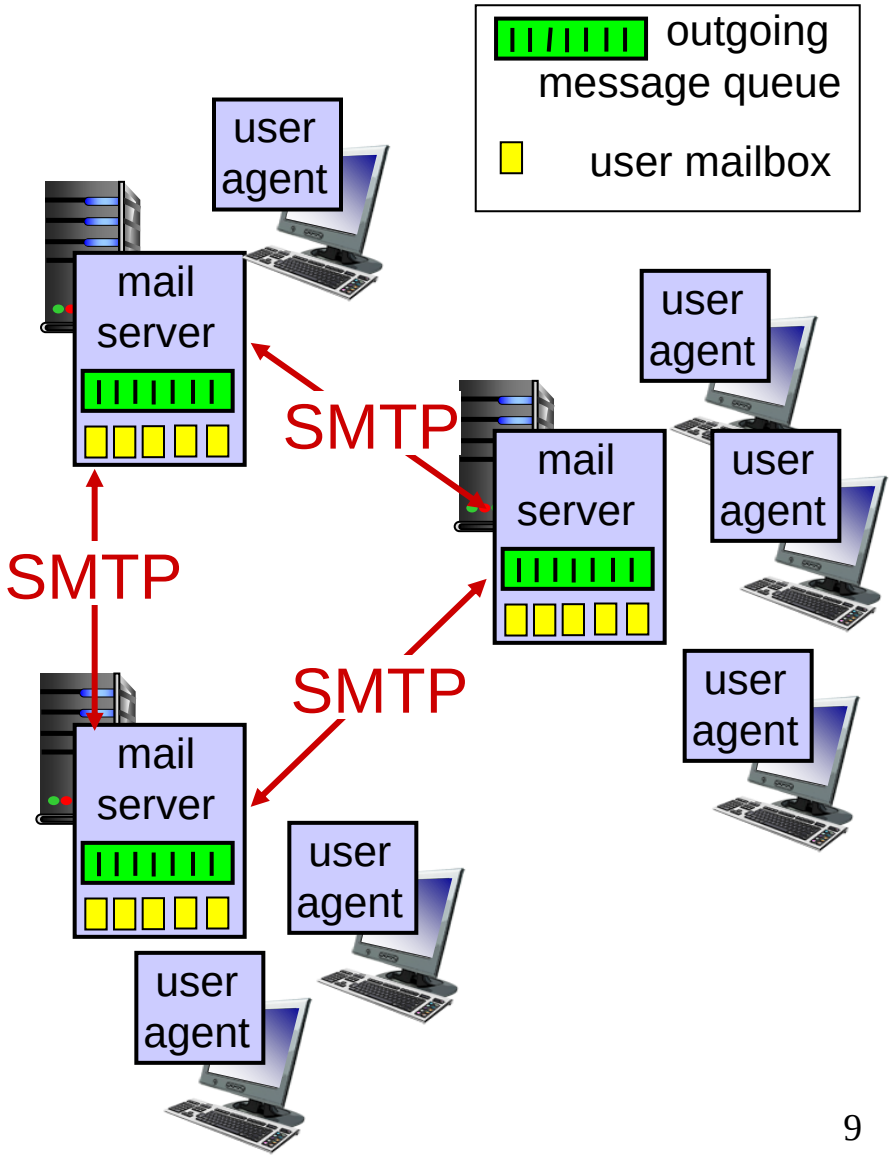
步骤3:

需要用到哪些设备?

2.3 因特网中的电子邮件

Three major components:

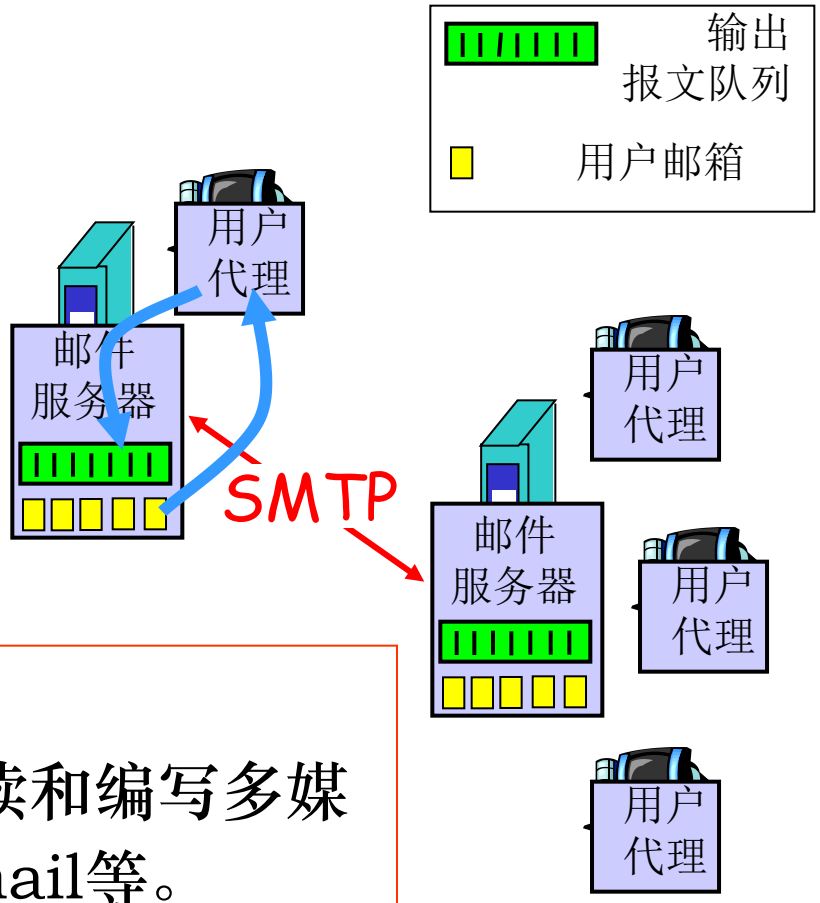
- user agents（用户代理）
- mail servers (邮件服务器)
- simple mail transfer protocol: SMTP(简单邮件传输协议)



用户代理(agent)

邮件阅读器。允许用户阅读、回复、发送、保存和撰写报文。

- 当用户完成邮件撰写时，邮件代理向其邮件服务器发送邮件，并存放在发送队列中。
- 当用户想读取一条报文时，邮件代理从其邮件服务器的邮箱中获取该报文。

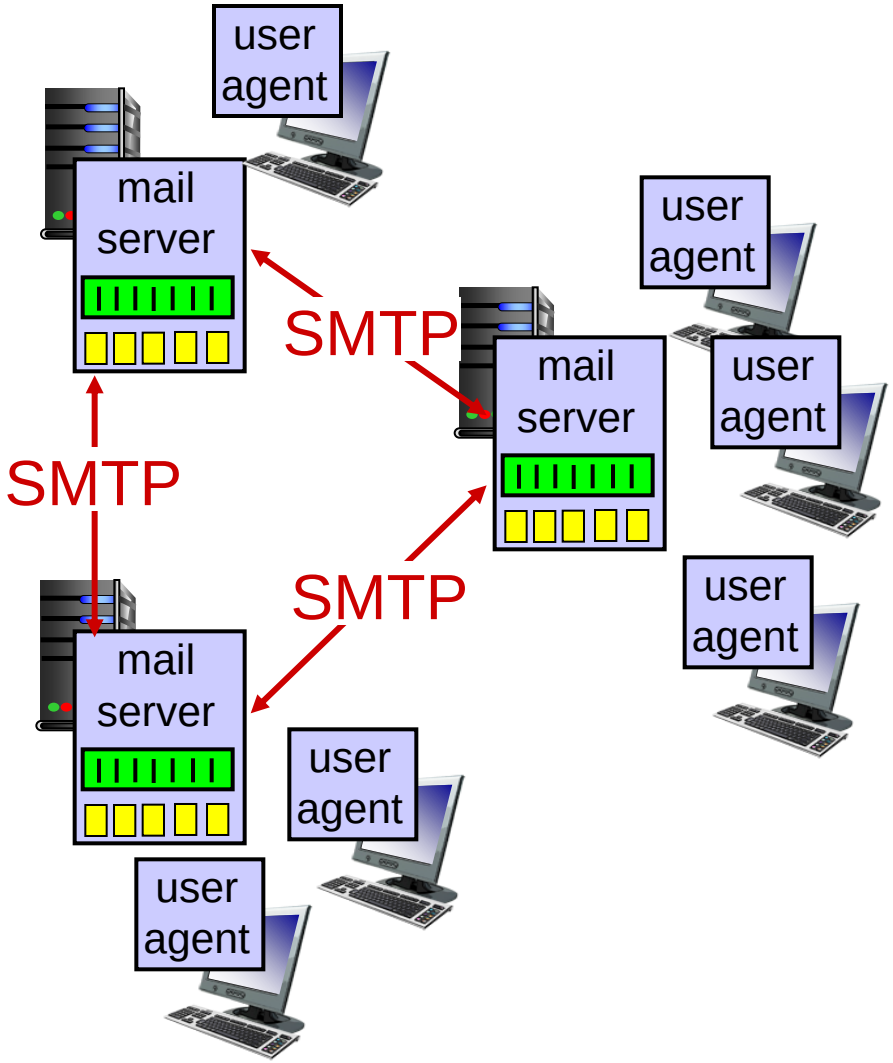


- 种类：
  - GUI（图形用户接口）：阅读和编写多媒体邮件。如Outlook、Foxmail等。

电子邮件：邮件服务器

邮件服务器:

- 邮箱：包括了用户的输入消息
- *输出的消息队列：即将发送的邮件消息*
- *SMTP协议*用于邮件服务器之间发送邮件消息
  - client: sending mail server
  - “server”: receiving mail server



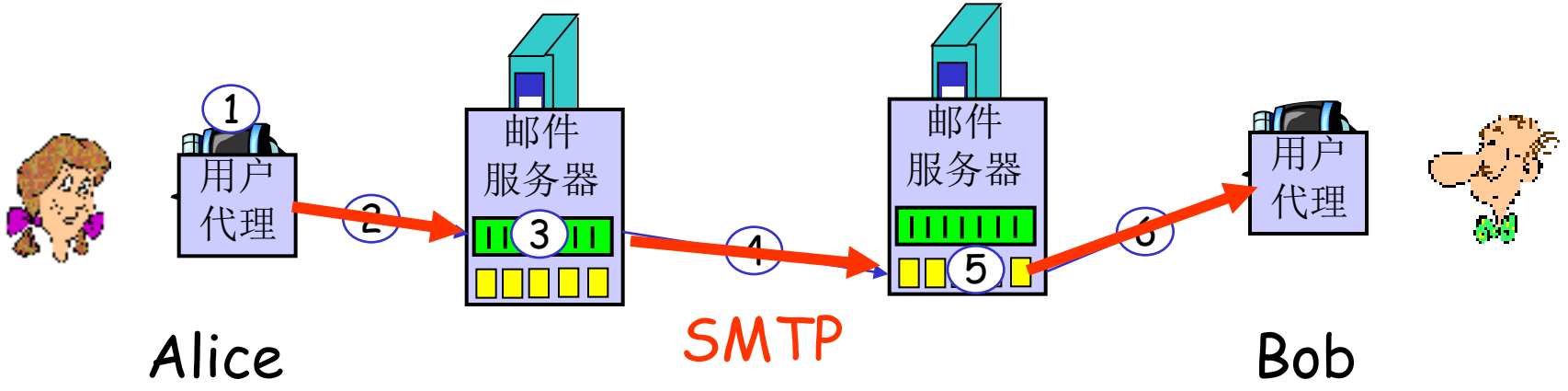
电子邮件：SMTP [RFC 2821]

- 使用TCP实现电子邮件消息从客户端到服务器的可靠传输, port 25
- 直接传输：发送服务器到接收服务器
- 传输的三个阶段:
  - handshaking (greeting)[握手阶段]
  - transfer of messages[消息传输]
  - close[关闭]
- 命令/响应交互(like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase
- 消息都使用7-bit ASCI编码

2.4.1 SMTP

把一封邮件从发送邮件服务器传送到接收邮件服务器的过程： 如Alice 向 Bob发送报文

1) Alice启动邮件代理，提供接收方的邮件地址，撰写邮件
2) 用户代理把报文发给其邮件服务器，放在发送队列中
3) SMTP的客户机侧创建与Bob的邮件服务器的TCP连接
4) SMTP通过TCP连接发送报文
5) Bob的邮件服务器接收并将该报文放入Bob的邮箱
6) Bob调用其用户代理来读报文



SMTP交互的样

C：SMTP客户
S：SMTP服务器
客户机的主机名：crepes.fr
服务器的主机名：hamburger.edu

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT
S: 221 hamburger.edu closing connection
```

TCP连接创建以后的执行过程

2.4.2 与HTTP的对比

- SMTP 使用持续连接
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP服务器使用CRLF.CRLF来作为消息的结束

comparison with HTTP:

- HTTP: pull（拉模式）
- SMTP: push（推模式）
- both have ASCII command/response interaction, status codes
- HTTP: 每个对象封装在它自己的响应消息中
- SMTP: multiple objects sent in multipart msg

HTTP把每个对象封装在它各自的HTTP响应报文中发送

SMTP把所有报文的对象封装在一个响应报文中

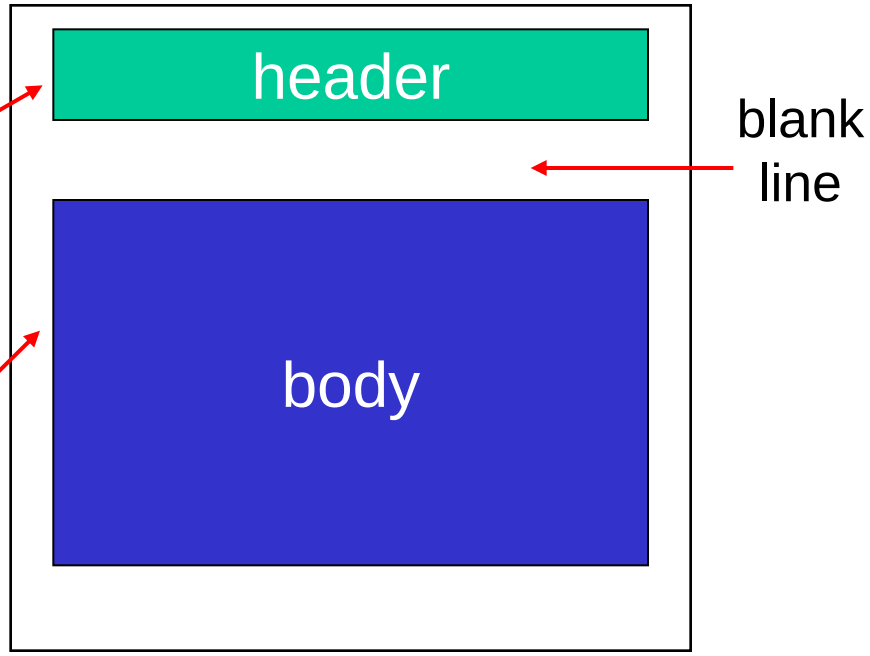
2.4.3 邮件报文格式

SMTP: protocol for exchanging email msgs

RFC 5322: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

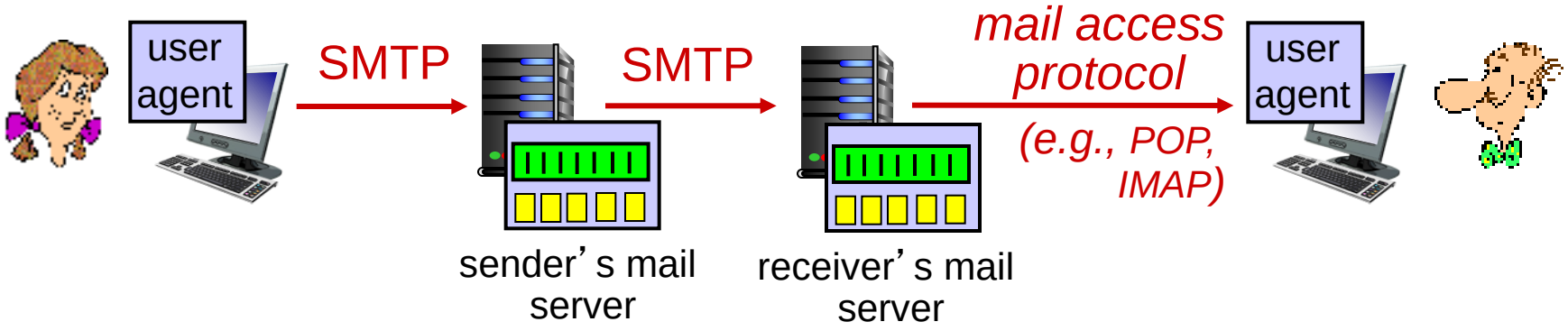
  different from SMTP MAIL FROM, RCPT TO: commands!
- Body: the "message"
  - ASCII characters only



2.4.3 邮件报文格式

- 对于非ASCII字符文本，为了能够使用SMTP进行传输，引入了MIME协议
- MIME：用于非ASCII数据传输。将非ASCII数据编码后传输，接收方再解码还原。
  - 增加新的MIME邮件首部
  - 采用某种编码

2.4.4 邮件访问协议



- SMTP: delivery/storage to receiver's server
- mail access protocol: **retrieval from server**
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.

POP3 协议

- 简单、功能有限：在用户代理打开了一个到邮件服务器(服务器)端口110上的TCP连接后，开始工作。
- 工作步骤(三个阶段)：
  - *特许阶段*：用户代理发送用户名和口令获得下载邮件的特许。（身份认证）
  - *事务处理阶段*：用户代理取回报文，可对邮件进行某些操作。如做删除标记、取消删除标记、获取统计信息等。
  - *更新阶段*：邮件服务器删除带有删除标记的报文，结束POP会话。

缺点：用户获取邮件后，服务器不再保存邮件！（早期版本）

POP3 协议

authorization phase

- client commands:
  - **`user`**: declare username
  - **`pass`**: password
- server responses
  - **`+OK`**
  - **`-ERR`**

transaction phase, client:

- **`list`**: list message numbers
- **`retr`**: retrieve message by number
- **`dele`**: delete
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3和IMAP

POP3

- previous example uses POP3 “download and delete” mode
  - Bob cannot re-read e-mail if he changes client
- POP3 “download-and-keep”: copies of messages on different clients
- POP3 is stateless across sessions

IMAP(自学)

- keeps all messages in one place: at server
- allows user to organize messages in folders
- keeps user state across sessions:
  - names of folders and mappings between message IDs and folder name

Web-Based E-Mail

- 1995年12月Hotmail 引入该技术。用户使用浏览器收发电子邮件。
- 用户代理是普通的浏览器，用户和其远程邮箱之间的通信通过HTTP进行：
  - 发件人使用HTTP 将电子邮件报文从其浏览器发送到其邮件服务器上；
  - 收件人使用HTTP从其邮箱中取一个报文到浏览器；
- 邮件服务器之间发送和接收邮件时，使用SMTP。
- 用户可以在远程服务器上以层次目录方式组织报文。

如，Hotmail 、 Yahoo、Mail等。

课堂练习

- 一个FTP的用户，发送了LIST命令来获取服务器的文件列表，这时候服务器应该通过( )端口来传输该列表。

  A、21　B、20　C、22　D、19

- SMTP协议是面向ASCII编码的，那么它使用（ ）支持非ASCII的数据传输。

  A、MAIL　B、POP3　C、IMAP　D、MIME

- FTP客户和服务器之间传递FTP命令时，使用的连接是(　　)。

A、建立在TCP之上的控制连接　B、建立在TCP之上的数据连接

C、建立在UDP之上的控制连接　D、建立在UDP之上的数据连接

课堂练习

- 电子邮件系统中，用户代理把邮件发往发送发邮件服务器、发送方邮件服务器把邮件发往接收方邮件服务器以及用户使用用户代理从接收方邮件服务器上读取邮件时，使用的协议可能是以下的哪种情形? (D)

A、IMAP 、SMTP、POP3　B、MIME、SMTP、POP3

C、SMTP、IMAP、POP3　D、SMTP、SMTP、IMAP

HOME WORK P115-R16

假定Alice使用一个基于web的电子邮件账户向Bob发送报文，而Bob使用POP3从他的邮件服务器访问自己的邮件。讨论是怎样从Alice主机到Bob主机得到该报文的。要列出在两台主机间移动该报文时所使用的各种应用层协议。

本次课的主要内容

第2章 应用层

2.1 应用层协议原理

2.2 Web和HTTP

2.3 因特网中的电子邮件

- SMTP, POP3, IMAP

2.4 DNS：因特网的目录服务

2.5 P2P文件分发

2.6 视频流和内容分发网(自学)

2.7 套接字编程：生成网络应用

2.8 小结

2.4 DNS：因特网的目录服务

people: 许多标识符:

- 身份证号, 姓名,
- 护照号

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- “name”, e.g., www.yahoo.com - used by humans

问题: **如何进行域名和IP地址的相互映射?**

域名解析服务

Domain Name System:

- *分布式数据库*

由多个层次化组织的域名服务器实现

- *应用层协议:*

主机和域名服务器通信来解析域名(address/name translation)

- 注意：核心的因特网功能，作为应用层功能实现
- 把复杂功能仍然放在网络边缘

一些概念的解释

- 运行BIND软件的UNIX机器；
- DNS协议运行在UDP之上，使用53号端口。
- DNS通常直接由其他的应用层协议 (包括HTTP、SMTP 和FTP)使用，以将用户提供的主机名解析为IP地址。用户只是间接使用。

例，某个用户主机上的一个浏览器访问某个Web页，www.someschool.edu/index.html
用户主机要将一个HTTP请求报文发送到Web服务器www.someschool.edu，需先得到相应的IP地址。

用户主机DNS客户端请求服务的过程

- 用户主机上运行 DNS应用的客户机端；
- 浏览器从URL中解析出主机地址(主机名)，传给DNS客户机端；
- DNS客户机向DNS服务器发送一个包含主机名的请求；
- DNS客户机收到含有对应主机名的IP地址的响应报文；
- 浏览器向该IP地址指定的HTTP服务器发起一个TCP连接请求。

2.4.1 DNS提供的服务

DNS services

- 主机名到IP地址的转换
- 主机别名查询 (host aliasing)
  - 规范名, 别名
  - 邮件服务器的别名
- 负载均衡 (load distribution)
  - replicated Web servers: many IP addresses correspond to one name

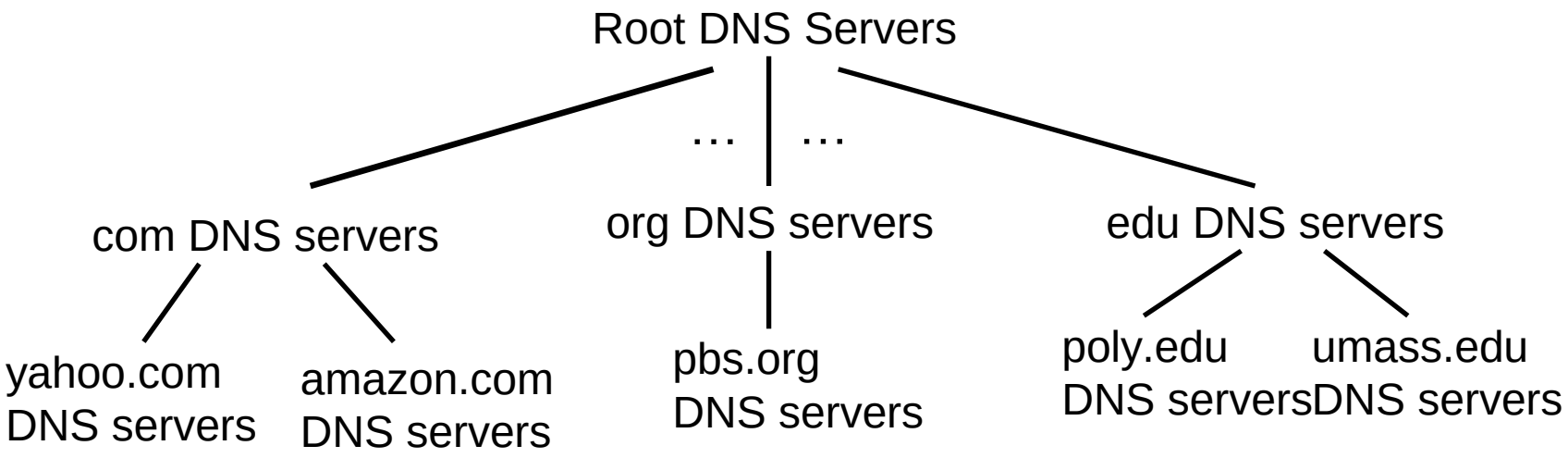
why not centralize DNS?

- 单一点失效
- 流量负担
- 远程的集中式数据库
- 维护开销大

A: doesn't scale!

规范名和别名：通过DNS可以得到主机别名对应的规范主机名及IP地址。

2.4.2 DNS工作机理概述



client wants IP for www.amazon.com; 1st approx:

- client queries **root server** to find **com DNS server**
- client queries .**com DNS server** to get **amazon.com DNS server**
- client queries amazon.com DNS server to get IP address for www.amazon.com

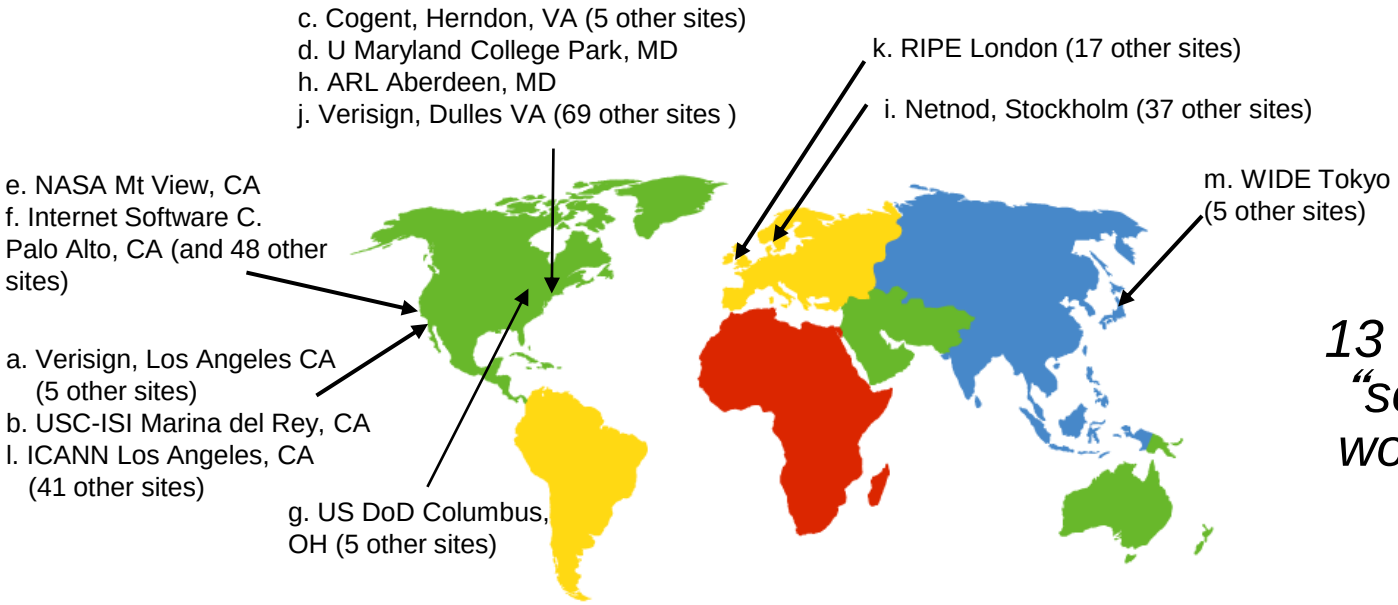
2.4.2 DNS工作机理概述

- 分布式、层次数据库
  - 根服务器：在因特网上有13（247：2011）个根服务器，主要维护的是顶级域名服务器的IP地址
  - 顶级域服务器（TLD）：维护顶级域名的IP地址

  顶级域名：com、org、net、edu、gov、jp、ca、cn
  - 权威DNS服务器：由组织机构维护的自己提供的服务器的域名到IP地址映射的DNS服务器。例如我们学校的Web服务器和电子邮件服务器的域名和IP地址的映射由我们学校自己的权威DNS服务器维护。
    - 本地DNS服务器：

DNS：根域名服务器

- 本地域名服务器如果无法解析，会向根域名服务器发起请求
- 根域名服务器(root name server):
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server



13 root name "servers" worldwide

TLD和权威域名服务器

top-level domain (TLD) servers（顶级域名服务器）:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Verisign Global Registry Services （公司） for .com TLD
- Educause (公司) for .edu TLD

authoritative DNS servers(权威域名服务器):

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

本地域名服务器

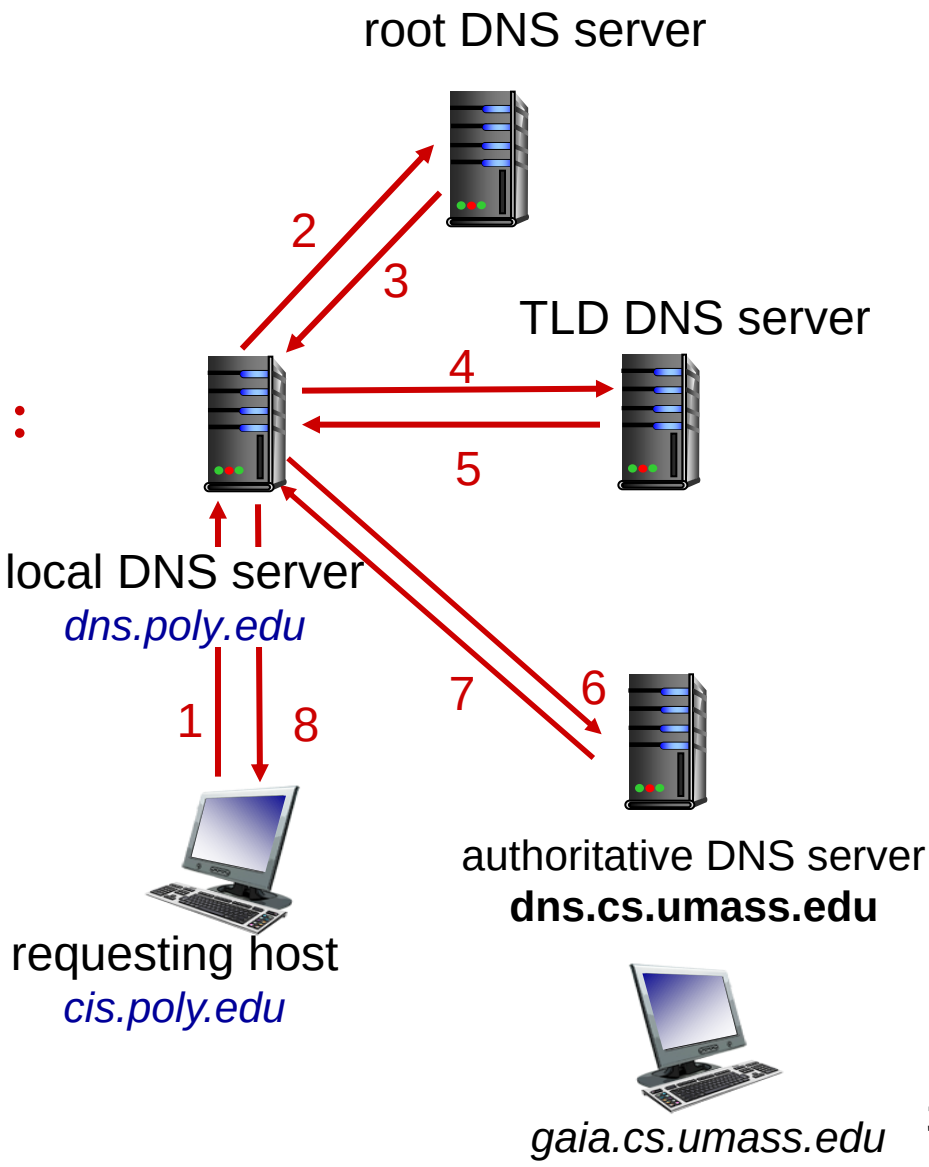
- 并不严格属于层次化结构
- 每个ISP(residential ISP, company, university)都至少有一个本地域名服务器
  - also called “默认DNS服务器”
- 当主机发送DNS请求，这个请求会被发送给本地DNS服务器
  - 会缓存最新收到的域名-地址的转换对(但是可能会过时!)
  - 作为代理，转发请求到层次化的DNS解析系统中

DNS域名解析示例

- host at **cis.poly.edu** wants IP address for **gaia.cs.umass.edu**

iterated query(迭代查询):

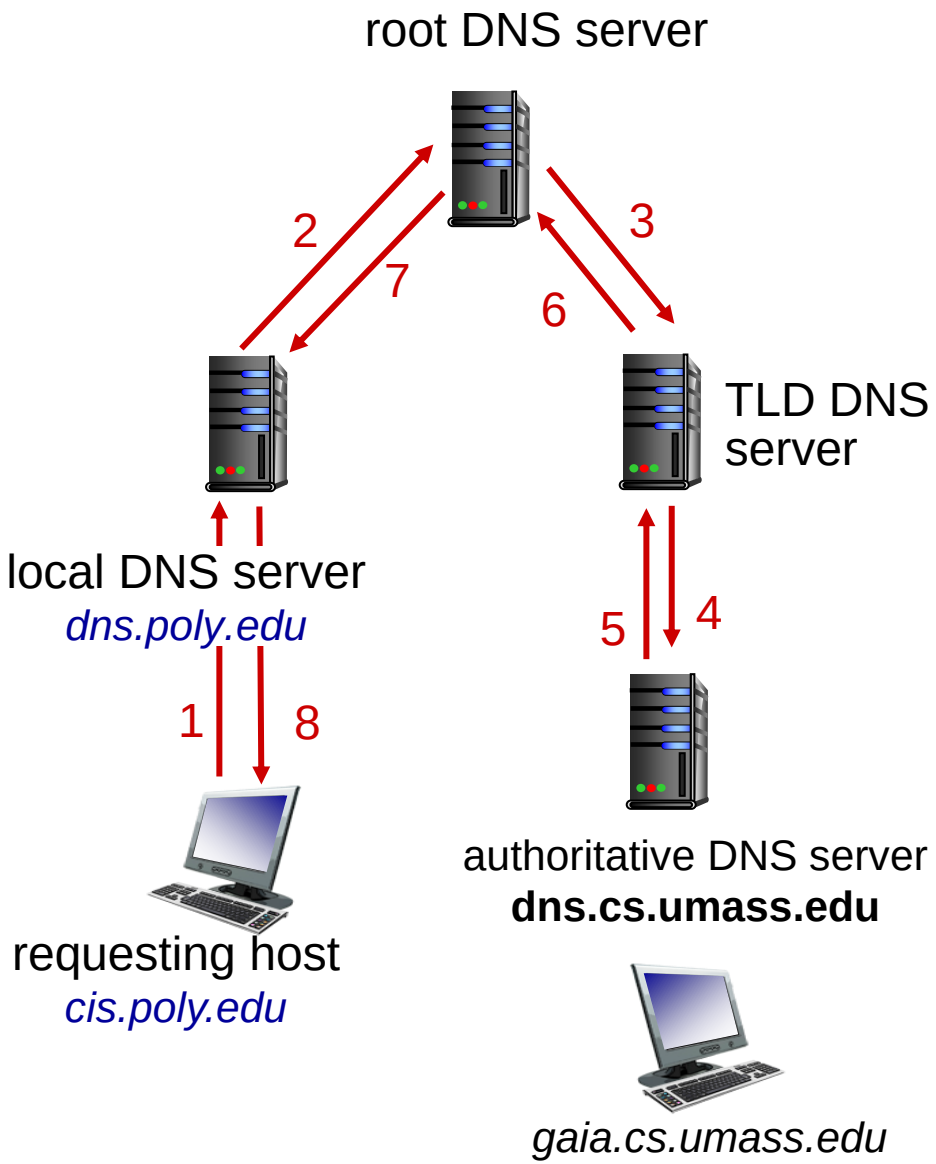
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS域名解析示例

recursive query
（递归查询）：

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



2.4.3 DNS记录和报文

- 一旦任意域名服务器学习到域名-IP地址的映射，它会缓存这个映射
  - 当TTL时间超时，缓存的映射条目会超时删除
  - TLD服务器通常会被本地域名服务器缓存
    - 从而减少根域名服务器被访问的次数
- 缓存的条目可能是过时的(best effort name-to-address translation!)
  - 如果主机的域名改变得了IP地址，直到所有的TTL超时，因特网范围的所有域名服务器得到更新
- 更新/通知机制是在IETF标准中提出的
  - RFC 2136

DNS记录

DNS: 分布式的数据库存储资源记录(resource records) (RR)

RR format: **(name, value, type, ttl)**

type=A

- **name** is hostname
- **value** is IP address

type=NS

- **name** is domain (e.g., foo.com)
- **value** is hostname of authoritative name server for this domain

type=CNAME

- **name** is alias name for some "canonical" (the real) name
- **www.ibm.com** is really **servereast.backup2.ibm.com**
- **value** is canonical name

type=MX

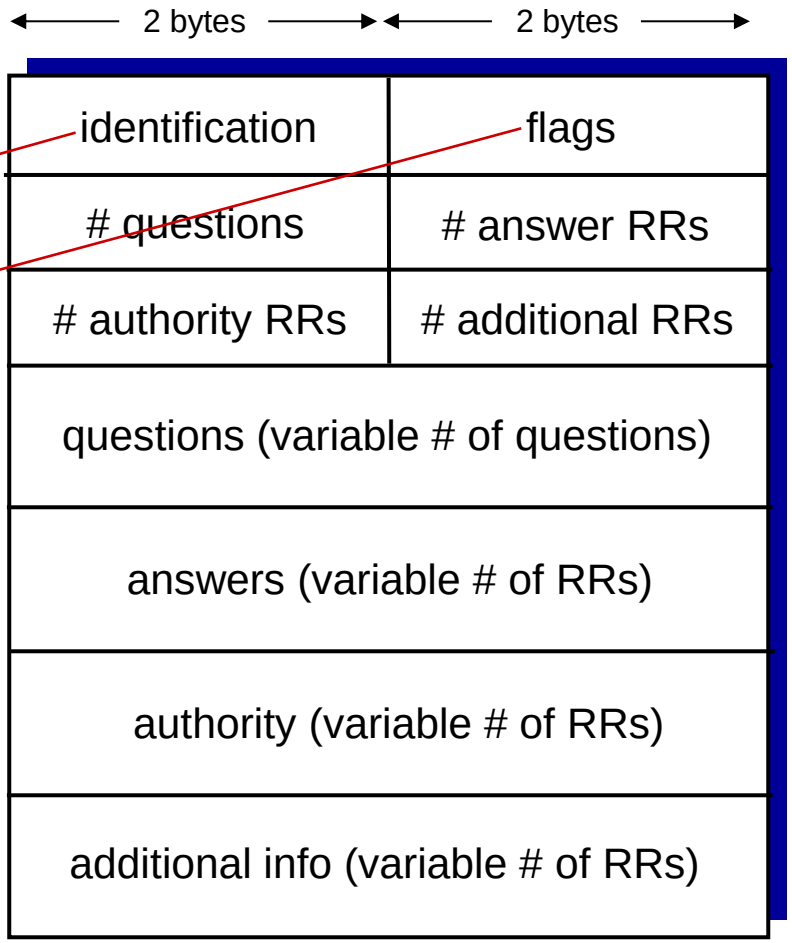
- **value** is name of mailserver associated with **name**

DNS记录和报文

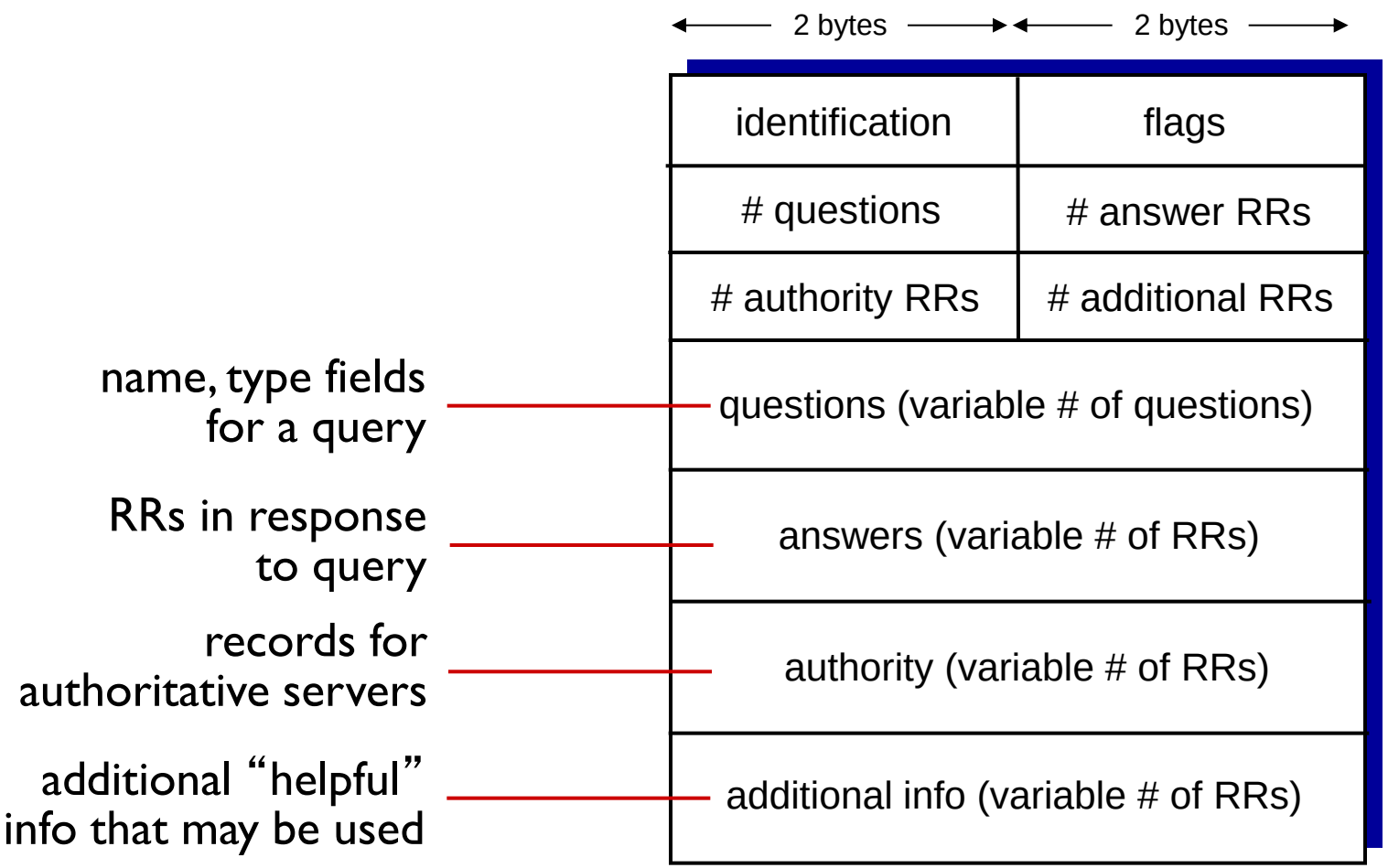
- 查询和应答报文，都是采用相同的报文格式

报文头部

- identification: 16 bit # for query, reply to query uses same #
- flags:
  - query or reply
  - recursion desired
  - recursion available
  - reply is authoritative



DNS记录和报文


2 bytes
2 bytes
identification
flags
questions
answer RRs
authority RRs
additional RRs
questions (variable # of questions)
answers (variable # of RRs)
authority (variable # of RRs)
additional info (variable # of RRs)
name, type fields for a query
RRs in response to query
records for authoritative servers
additional "helpful" info that may be used

插入记录到DNS

- example: new startup “Network Utopia”
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - **registrar inserts two RRs into .com TLD server**:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

DNS攻击

自学

DDoS攻击

- 用流量轰炸根服务器
  - 到目前为止还未成功
  - 流量过滤
  - 本地域名服务器缓存TLD服务器的IP地址，允许绕过根服务器
- 轰炸TLD服务器
  - 可能更危险

重定向攻击

- Man-in-middle(中间人)
  - Intercept queries(截获查询报文)
- DNS污染
  - Send bogus relies to DNS server, which caches

利用DNS进行DDoS攻击

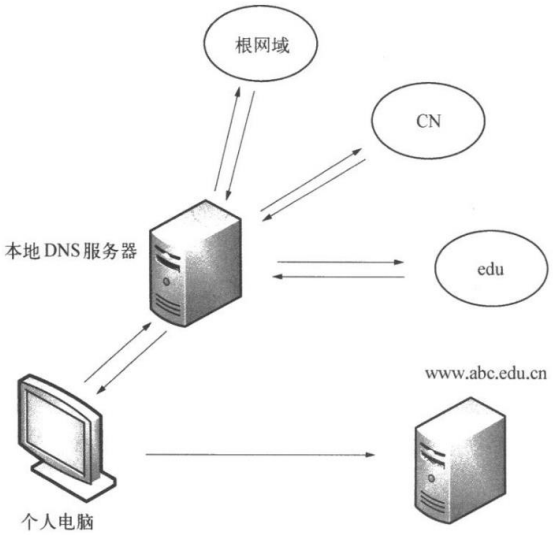
- 使用伪造的源IP地址发送查询：目标IP
- 放大需求

课堂练习

- 某公司C有一台主机h，该主机具有的Internet域名应该为（ ）。

A、com.c.h　B、h.c.com　C、com.h.c　D、c.h.com

- 一个主机申请了一个到www.abc.edu.cn的连接，为了获取服务器的IP地址，首先要进行DNS查询，下图为本次查询的过程，请回答如下问题：



- 由个人主机发送给本地DNS服务器的数据是采用什么运输层协议发送的？利用那个端口？
- 由个人主机到本地DNS服务器查询是采用了什么方式？
- 由本地DNS服务器到各个域名服务器的查询，采用了什么方式？
- 本地DNS服务器的查询顺序是什么？

本次课的主要内容

第2章 应用层

2.1 应用层协议原理

2.2 Web和HTTP

2.3 因特网中的电子邮件

- SMTP, POP3, IMAP

2.4 DNS：因特网的目录服务

2.5 P2P文件分发

2.6 视频流和内容分发网(自学)

2.7 套接字编程：生成网络应用

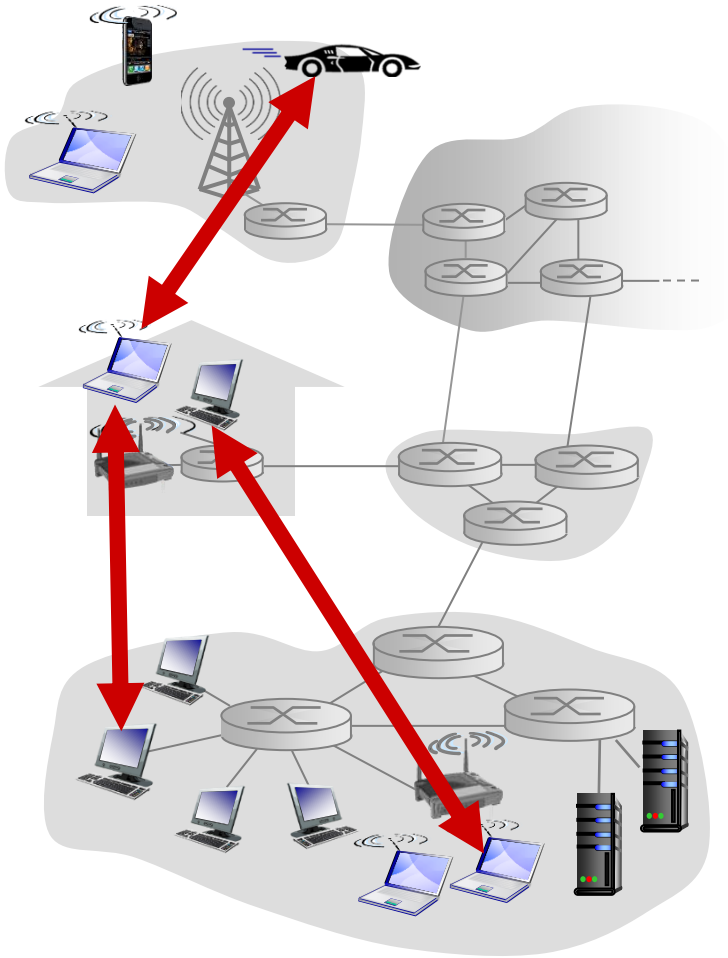
2.8 小结

2.5 P2P应用

- 服务器不需要总是开机
- 任意终端系统可以直接相互通信
- 对等节点可以间隙性互连，也可以改变IP地址

例子:

- 文件分享(BitTorrent)
- 流媒体(KanKan)
- VoIP(Skype)



2.5.1 文件分享

- 位于网络边缘的PC机(对等方peer)互相之间可以直接获取对象。

*说明：*每个参与的对等方既是内容的消费者也是内容的发布者（下载同时也向其他用户上载）。

例子，用户Alice使用P2P文件共享应用程序下载MP3。

P2P文件分享

- 在PC机上运行一个P2P文件共享应用软件(对等方);
- 通过某种方式接入因特网（无固定IP地址）；
- 使用该应用程序搜索一首MP3歌曲;
- 显示一张有该首歌曲的对等方列表:
  - 所有在线对等方，并愿意共享该首音乐的MP3拷贝
  - 对等方都是普通的PC，是因特网用户。
  - 列表中提供一些附加信息，如接入带宽和下载时间。
- 选择一个对等方（Bob）并向其请求该MP3文件;
  - 两个用户之间建立一个直接的TCP连接;
  - MP3文件从Bob的PC机向Alice的PC发送;
  - 下载期间若偶然断开，可从其他对等方继续下载。

P2P文件分享的特点

- 直接在对等方间传输：所有内容不经过第三方服务器;
- 高度的可扩展能力：利用众多对等方集合中的资源去分发内容;
- 使用客户机/服务器模式：请求的对等方是客户机，被选中的对等方是服务器;
  - 服务器对等方使用文件传输协议向客户机对等方传送。
  - 通过传送“HTTP请求”和“HTTP响应”报文进行。
  - 所有的对等方必须既能运行文件传输协议的客户机端程序，又能运行服务器端程序。
  - 对等方既是一个客户机，又是一个瞬时Web服务器。

P2P中的内容定位

一个对等方如何确定哪个对等方有所需要的内容。

（1）集中式目录(Napster)

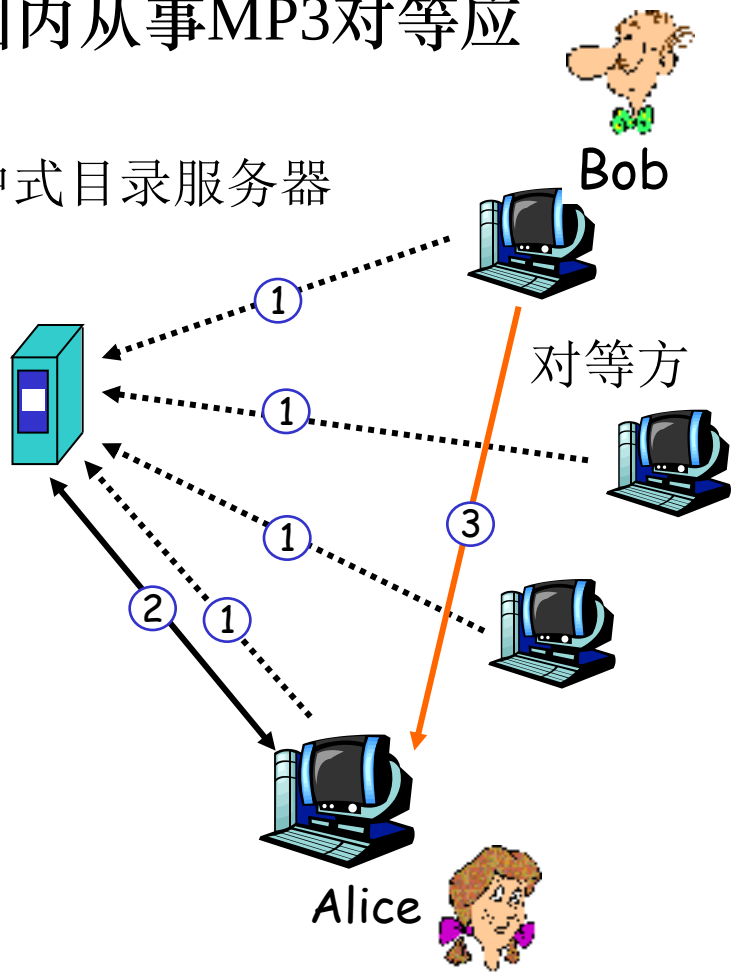
（2）查询洪泛(Gnutella)

（3）利用不均匀性(KaZaA)

集中式目录

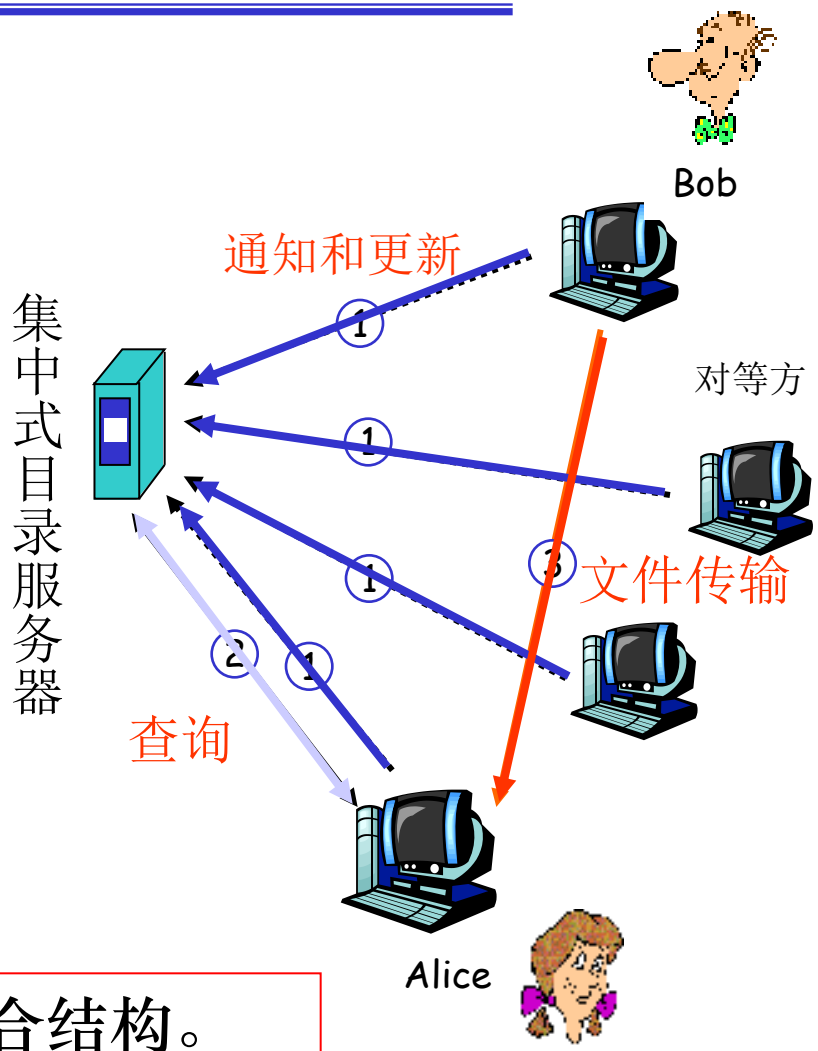
源于Napster公司（第一家世界范围内从事MP3对等应用程序）设计。

- 目录服务器(大型服务器或服务器场)：提供目录服务。收集可共享的对象，建立集中式的动态数据库(对象名称到IP地址的映射)。



集中式目录的工作过程

- 通知：对等方启动时，将其IP地址及可共享内容通知目录服务器；
- 查询内容：用户查询需要共享的对象。如 Alice 查询某首歌。
- 获取内容：来自Bob。
- 更新：当对等方获得新对象或删除对象时，通知目录服务器更新。



属于客户机/服务器与P2P的混合结构。

集中式目录存在的问题

- 单点故障
- 性能瓶颈
- 侵犯版权
- 可靠性：文件传输是分散的，但定位内容的过程是高度集中的。

查询洪泛：Gnutella

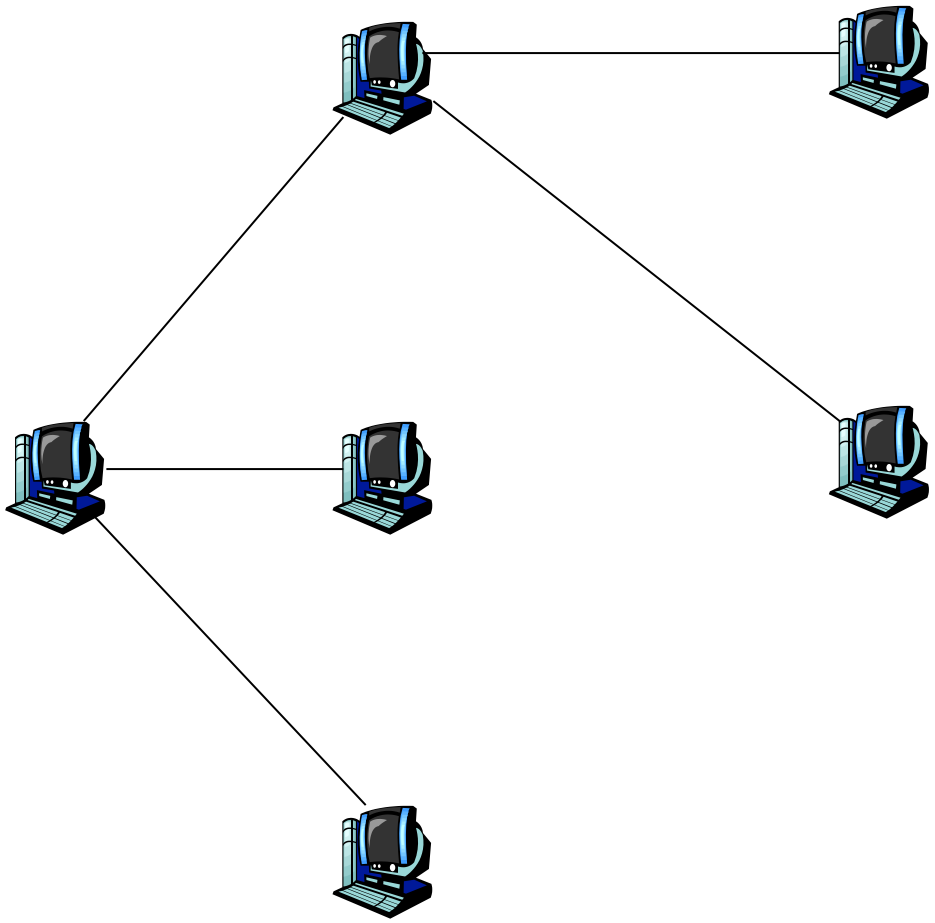
- Gnutella是一个公共域文件共享应用程序。
  - 全分布方式：无中心服务器
  - 许多Gnutella客户机实现协议：运行在对等方。
- 实现
  - 对等方先形成一个抽象的逻辑网络（覆盖网络）：
  - 通过洪泛查询，找到所需对象：如，某个对等方Alice想得到一个对象。
- 特点
  - 设计简单。
  - 扩展性差。
  - “查询报文”在网络中产生很大的流量。

限范围的洪泛查询：

在“查询报文”中设置一个计数字段，并给定一个特定值。可能减少查询的对等方数量

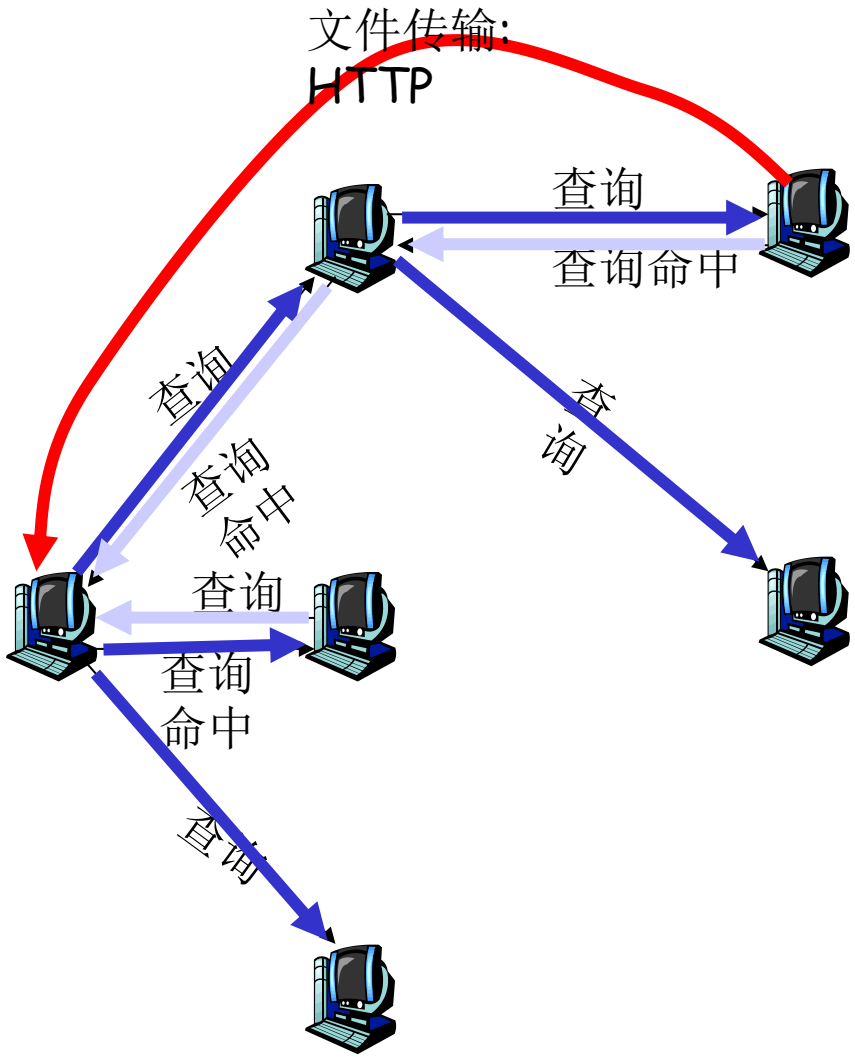
覆盖网络(重叠网络/应用层网络)

- 如果对等方X和Y间有一条TCP连接，则存在一条边；
- 所有活动对等方和边形成覆盖网络；
- 边不是物理链路；
- 一个对等方所连接的节点少于10个。



洪泛查询

- 向覆盖网络中的每个邻居发送“查询报文”；
- 每个邻居再向邻居转发，使覆盖网络上的每个对等方都能收到该查询；
- 如果收到查询的对等方中有被请求对象，沿反向路径回发“查询命中”报文；
  - 可能收到多个对等方发回的“查询命中”报文
  - 选择一个对等方，双方建立一条直接的TCP连接，并通过HTTP报文得到内容。



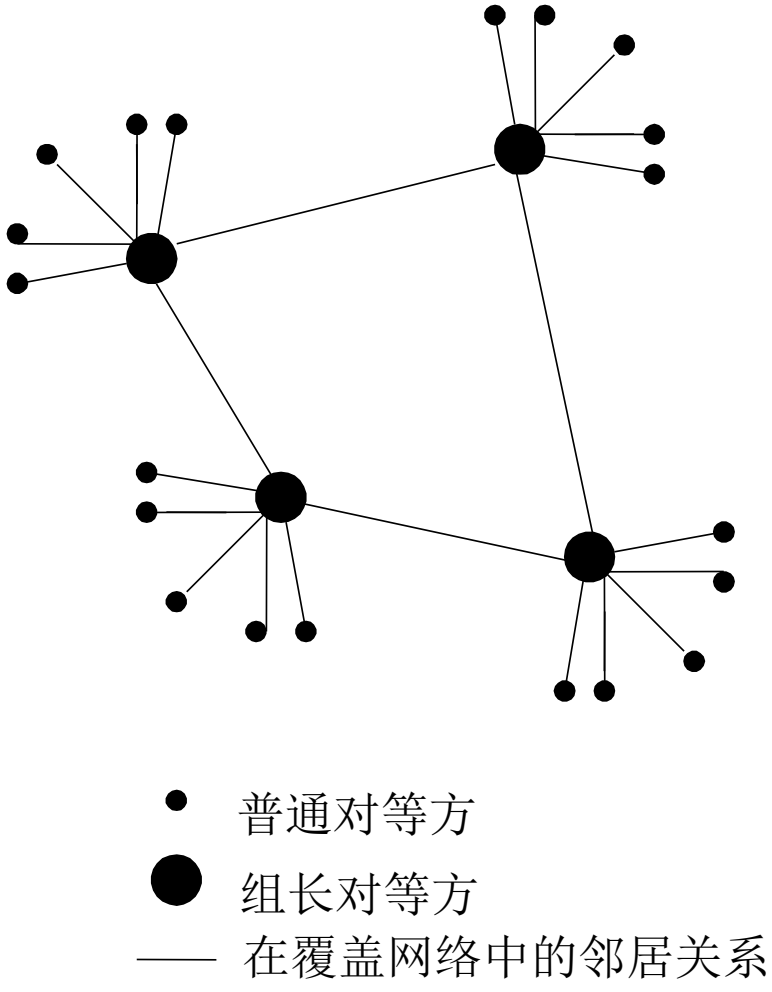
利用不均匀性：KaZaA

- 与Gnutella类似，KaZaA无专用服务器，但对等方地位不完全平等。
- 实现：
  - 对等方先形成一个层次的覆盖网络
  - 通过查询，找到所需对象。
- 特点：
  - 层次覆盖网络
  - 查询流量不大
  - 设计技巧
    - 请求排队：限制并行上载数量
    - 激励优先权：上载文件比下载文件多的用户优先。
    - 并行下载：从多个对等方请求并下载同一个文件的不同部分。

层次覆盖网络

对等方根据通信关系划分若干组，组成层次结构。

- 每组包括若干个组员，一个组长
  - 组员与其组长有一个TCP连接，将共享内容告诉组长
  - 组长维护一个数据库，该组的共享内容及相关对等方的IP地址。
  - 相关组长之间建立TCP连接
- 组长追踪其所有子节点上的内容



KaZaA查询

对等方确定到某个特定对象的方法：

- 向组长发出查询，组长用本组中具有该对象的对等方列表响应；
- 或与其他组长联系，请它们向该对等方发送具有该对象的对等方列表。

课堂练习

- 以下关于P2P概念的描述中，错误的是（ ）。

A、P2P是网络结点之间采取对等的方式直接交换信息的工作模式。

B、P2P通信模式是指P2P网络中对等节点之间的直接通信能力。

C、P2P网路是指与互联网并行建设的，由对等结点组成的物理网络。

D、P2P实现技术是指为实现对等节点之间直接通信的功能所需要设计的协议、软件等。

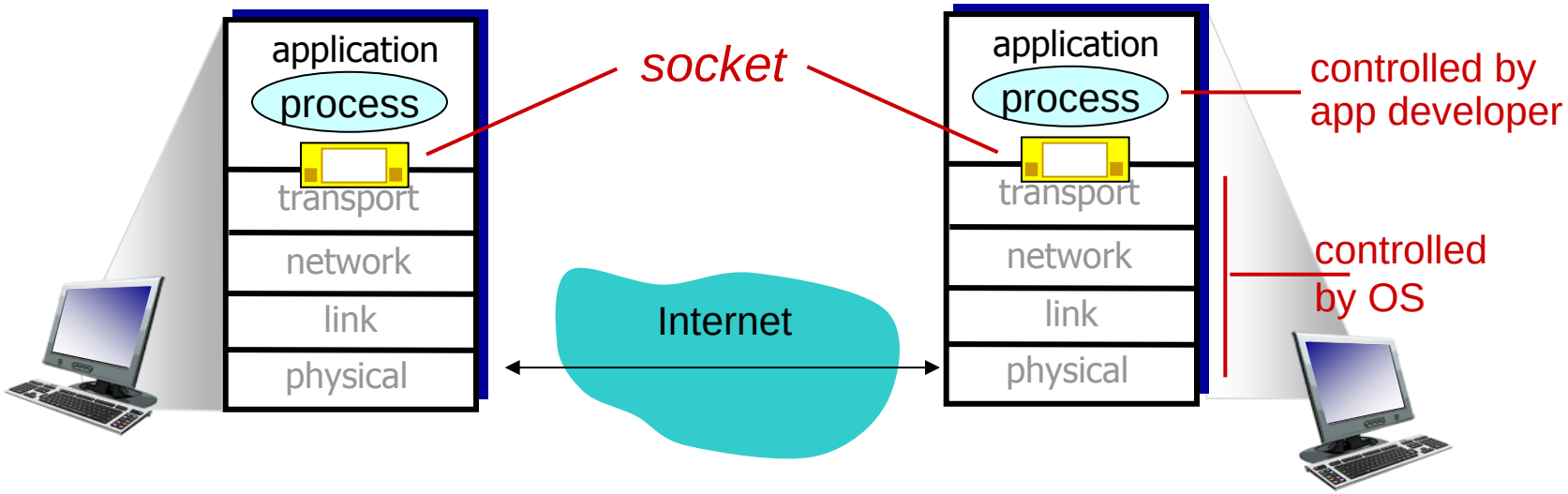
实验1—思考

任务1：在本实验中，我们要求学生能够根据实验要求自主设计实验方案和步骤，通过捕捉数据包，分析HTTP的各种报文格式、报文的发送顺序，以及各种规定动作。我们将分析和验证HTTP协议的以下几个方面：基本的GET/回答交互，HTTP报文格式，检索大HTML文件，检索具有内嵌URL的HTML文件，持续和非持续连接，HTTP鉴别和安全性；我们会为这个实验提供一个基本的实验指导，以给学生以引导。

任务2：在本实验中，我们要求学生通过抓包对DNS协议的运行过程进行观察和分析。由于我们只能从DNS客户(DNS是用于将因特网主机名转换为IP地址的协议)的角度进行观察，客户向它的本地DNS服务器发送一个请求，并接收返回的响应。在此过程中发生的很多事情均不为DNS客户所见，如等级结构的DNS服务器互相通信递归地或迭代地解析该客户的DNS请求。然而，从DNS客户的角度而言，该协议是相当简单的，即向本地DNS服务器发送一个请求，从该服务器接收一个响应。

2.7 Socket套接字编程

- 目标: 学习如何使用sockets 通信来建立C/S架构的应用
- socket: 位于应用进程和端到端传输协议之间的门



套接字编程

- 针对两种运输层服务有两种类型的套接字
  - *UDP:* 不可靠的数据报服务
  - *TCP:* 可靠的面向字节流的服务

应用示例:

1. 客户端从键盘读取一行字符串(数据)，发送数据到服务器.
2. 服务器收到数据，将字符转化为大写字符.
3. 服务器发送修改后的数据给客户端.
4. 客户端接收修改后的数据，在显示器上显示这行字符。

2.7.1 使用UDP套接字编程

UDP: 在客户端和服务器之间不建立连接

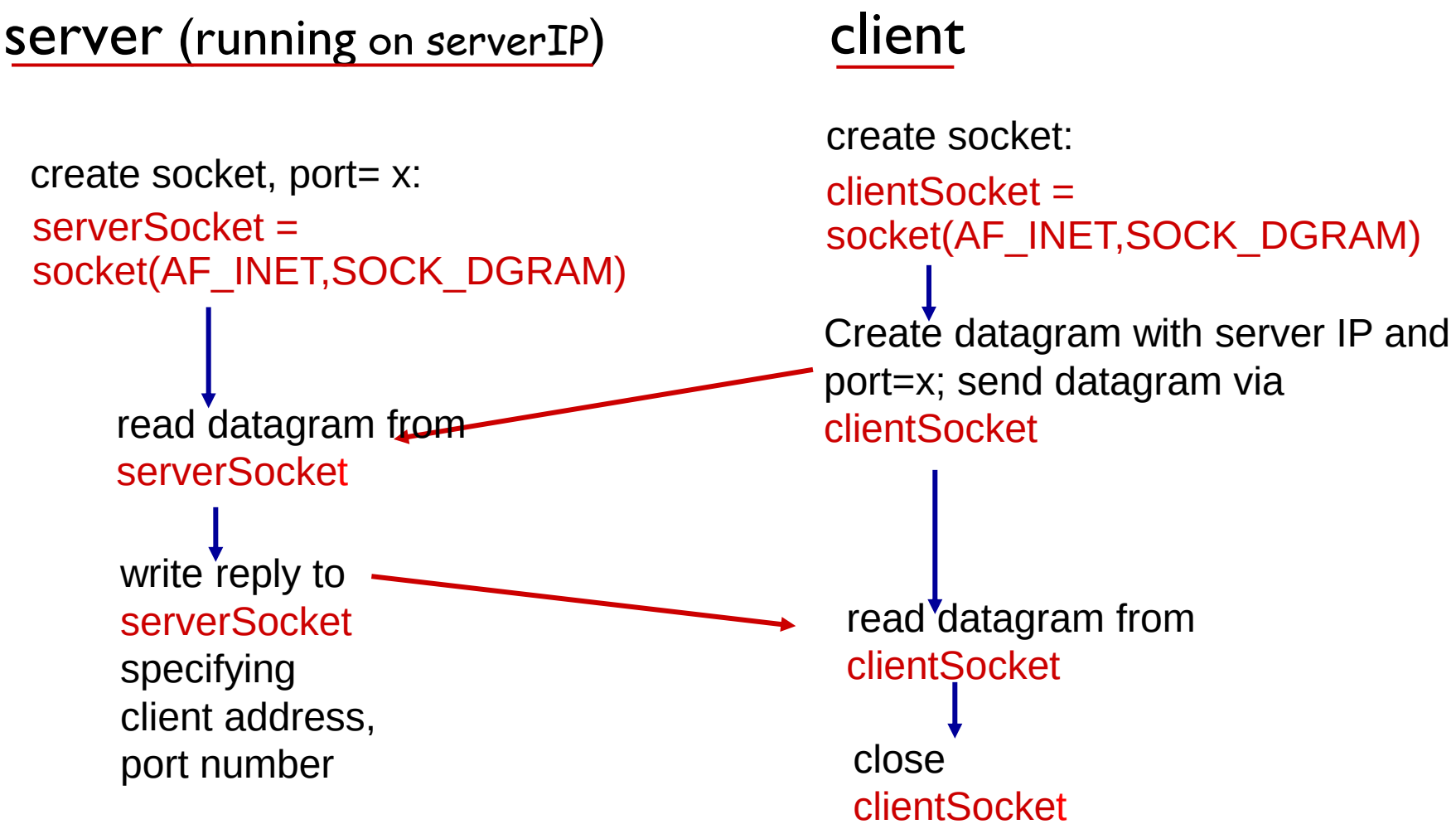
- 发送数据前不“握手”
- 发送方明确地粘贴IP目的地址和目的端口号给每个发送的分组
- **rcvr** 从收到的分组中抽取发送方IP地址

UDP: 发送的数据可能会丢失，到达接收方时可能会乱序

应用层的观点:

- UDP在客户端和服务器之间提供不可靠的字节组传输（“数据报”）。

C/S套接字交互：UDP



Python套接字
https://www.cnblogs.com/itogo/p/5910706.html

UDP套接字编程的示例

Python UDPClient

include Python's socket library

```
from socket import *
serverName = 'localhost'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input('Input lowercase sentence:')
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress =
clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()
```

create UDP socket for server

get user keyboard input

Attach server name, port to message; send into socket

read reply characters from socket into string

print out received string and close socket

注意这里应该写一个真实的域名或者localhost表示本机的，或者写IP地址

modifiedMessage：放收到的消息

serverAddress：放接收的数据包的源IP地址和源端口号

UDP套接字编程的示例

Python UDPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', serverPort))
print "The server is ready to receive"
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)
```

把本机IP地址和端口号和server套接字绑定

create UDP socket

bind socket to local port number 12000

loop forever

Read from UDP socket into message, getting client's address (client IP and port)

send upper case string back to this client

modifiedMessage：放收到的消息

serverAddress：放接收的数据包的源IP地址和源端口号

2.7.2 TCP套接字编程

客户端必须联系服务器

- 服务器进程必须首先运行
- 服务器必须创建套接字(socket)用于欢迎客户端进程的联系

客户端进程联系服务器：

- 创建TCP套接字，指定服务器的IP地址和端口号
- 客户端创建套接字：客户端进程创建到服务器的TCP连接

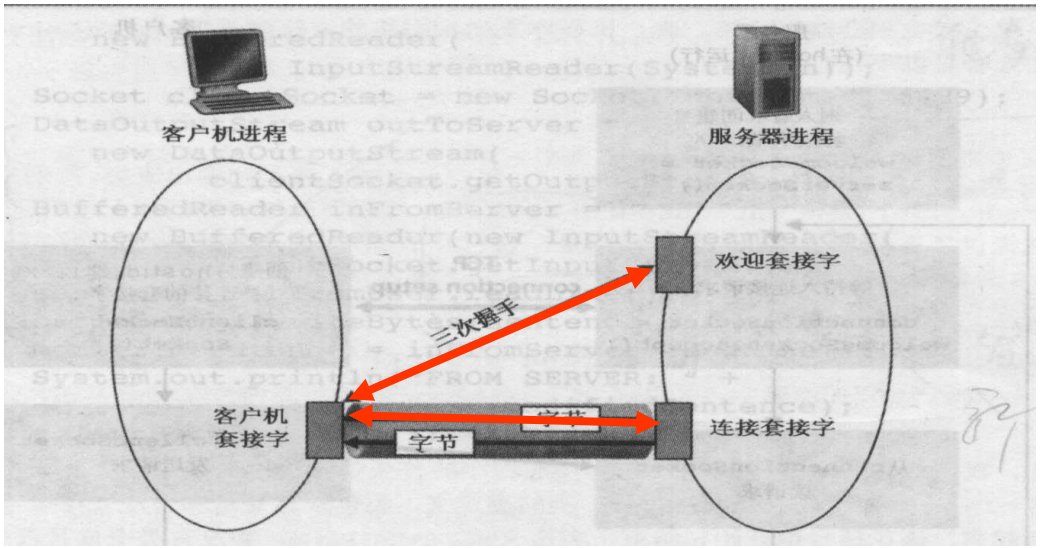
- 当收到客户端进程的联系，服务器TCP为服务器进程创建新的套接字，以实现与某个客户端进程通信
- 允许服务器与多个客户端进程通信
  - 源端口号被用于区分客户端进程

应用观点：

TCP 提供可靠的、面向字节流的数据传输

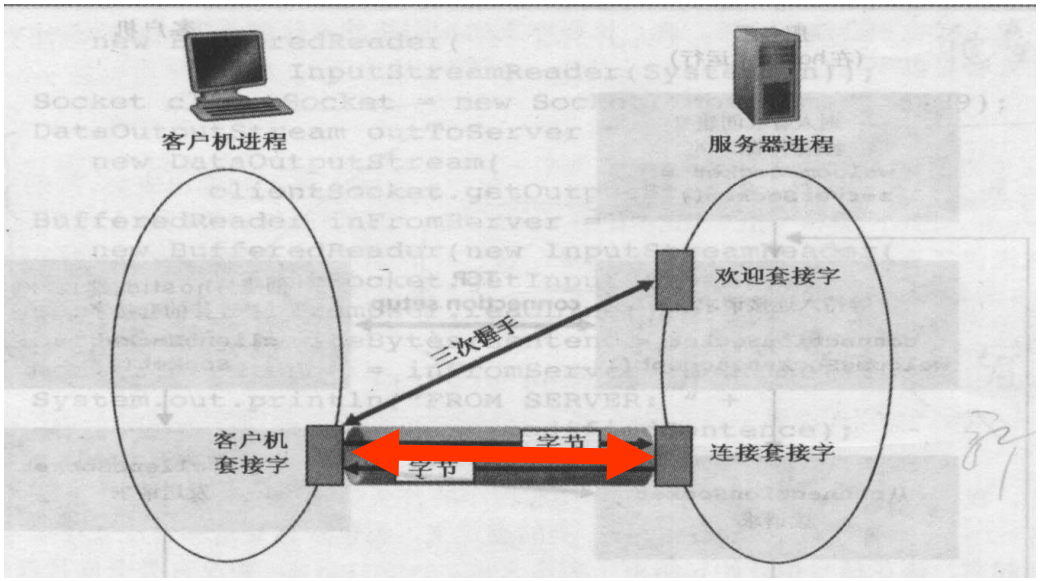
建立TCP连接

- 客户机进程向服务器发起一个TCP连接：创建一个本地套接字，指定相应服务器进程的地址（IP地址和端口号）。
- 建立一个TCP连接：当服务器听到客户机的连接请求（敲门）时，创建一个新套接字，经过“三次握手”，客户机套接字和服务器套接字之间建立一个TCP连接（直接的虚拟管道）。

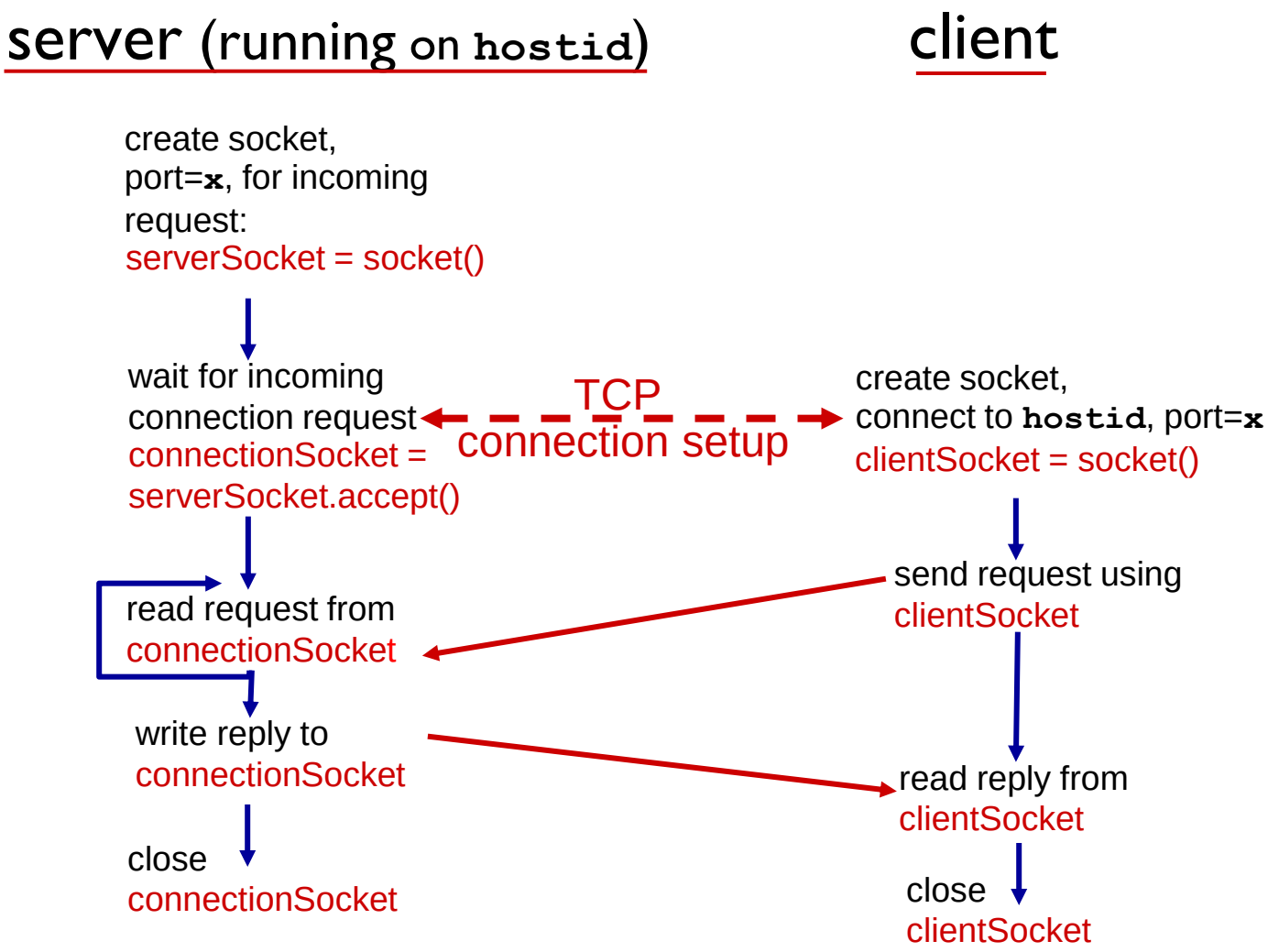


传送数据

- TCP连接为客户机和服务器了一个直接的传输管道。
- 可靠的, 顺序的,字节流的传输



C/S套接字交互：TCP


server (running on hostid)
client
create socket,
port=x, for incoming
request:
serverSocket = socket()
wait for incoming
connection request
connectionSocket =
serverSocket.accept()
TCP
connection setup
create socket,
connect to hostid, port=x
clientSocket = socket()
send request using
clientSocket
read request from
connectionSocket
write reply to
connectionSocket
read reply from
clientSocket
close
connectionSocket
close
clientSocket

TCP客户端编程的示例

Python TCPClient

```
from socket import *
serverName = 'servername'
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input('Input lowercase sentence:')
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print 'From Server:', modifiedSentence
clientSocket.close()
```

这里仍然要注意服务器的域名，或者写为localhost或者写明IP地址

create TCP socket for server, remote port 12000

No need to attach server name, port

TCP服务器编程的示例

Python TCPServer

```
from socket import *
serverPort = 12000
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind((‘’,serverPort))
serverSocket.listen(1)
print ‘The server is ready to receive’
while 1:
    connectionSocket, addr = serverSocket.accept()

    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()
```

create TCP welcoming socket

server begins listening for incoming TCP requests

loop forever

server waits on accept() for incoming requests, new socket created on return

read bytes from socket (but not address as in UDP)

close connection to this client (but *not* welcoming socket)

案例分析：给学生描述这样一个案例，某天你的电脑QQ能登录，但是各种网页打不开了？应该怎么处理？

首先判断IP地址和DNS配置是否正确；然后ping网关地址看是否能够ping通，如果能够ping通，则继续ping一个公网的IP地址，如果能够ping通则说明网络是通的，如果不能ping通则说明内网到外网没有通，需要查看网关路由器的状态(这个时候通常需要重新启动一下设备，在后续网络层再详细讲解)；如果能ping通外网IP，但是不能ping通外网的URL(或者浏览器无法打开网页)，则考虑可能是DNS的解析出现问题，这个时候我们需要重新设置本机的DNS服务器(换一个)，如果仍然不行，则可能判断接入网的DNS服务器被攻击了，无法进行正常的域名解析。

注：一个非常有意思的真实DNS故障案例，扩展了上述场景

引自(http://blog.csdn.net/ityouknow/article/details/58588459)