现代计算机网络

李伟明 lwm@hust.edu.cn

Ch4. P2P原理与技术

4.1 P2P网络基本概念

4.2 混合式P2P网络（第一代）

4.3 无结构P2P网络（第二代）

4.4 结构化P2P网络（第三代）

4.1 P2P网络基本概念

What is P2P ? (Peer-to-Peer)

- 对等(网络，计算)…;端到端…
- 经系统间直接交换来共享计算资源和服务的应用模式
- 以非集中方式使用分布式资源来完成关键任务的一类系统和应用
  - 资源包括计算、存储、带宽、场景（计算机、人和现场）和信息等资源
  - 关键任务可能是分布式计算、数据/内容共享，通信和协同、或平台服务
- 典型位置：因特网边界

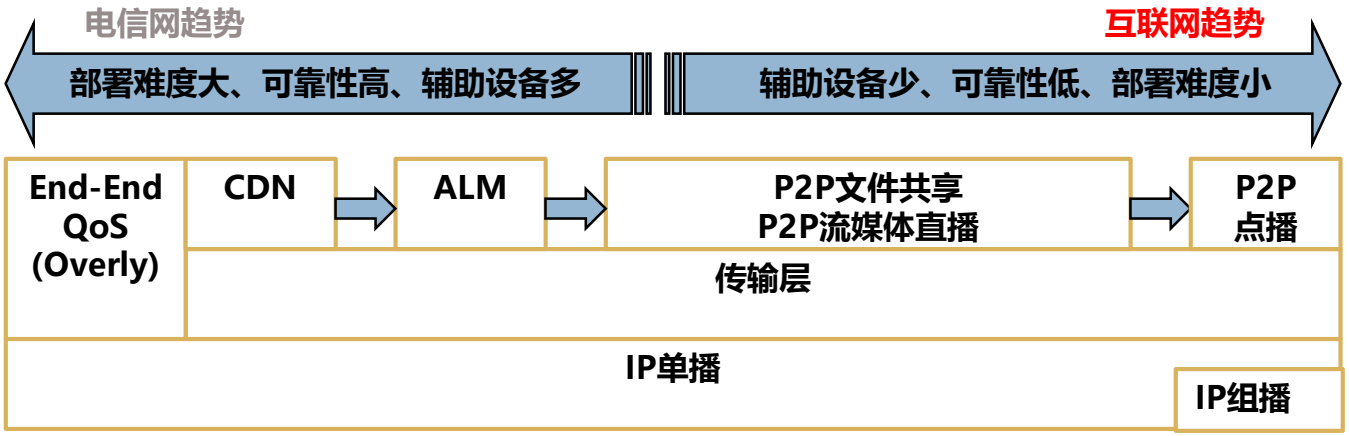
典型定义

- Intel 工作组：通过在系统之间直接交换来共享计算机资源和服务的一种应用模式
- A.Weytsel：在因特网周边以非Client方式使用的设备
- R.l.Granham：通过3个关键条件定义
  - 具有服务器质量的可运行计算机
  - 具有独立于DNS的寻址系统
  - 具有与可变连接合作的能力
- C.Shirky：利用因特网边界的存储/CPU/内容/现场等资源的一种应用。访问这些非集中资源意味着运行在不稳定连接和不可预知IP地址环境下，P2P节点必须不依赖DNS系统，对中心服务器来说具备有效的或全部的自治

- P2P是另一种应用模式：
  - 相对集中式、和C/S模式，纯P2P：没有服务器的概念，所有成员都是对等端
- 并不是新的概念
  - 早期分布式系统：如UUCP（ Unix-to-Unix Copy ）和交换网络
  - 电话通信
  - 计算机网络中的通信、网络游戏中的P2P玩家

互联网内容分发技术演变与比较

- 单播C/S：不适合同时多用户高带宽传输；
- IP组播：需网络层设备支持，难以普遍部署；
- ISP-CDN（Content Delivery Network）：成本高、部署难、大规模
- ALM（Application-Level Multicast）树拓扑应用层组播，抗扰动性差、下游节点延迟大、节点间难同步，难满足用户需求



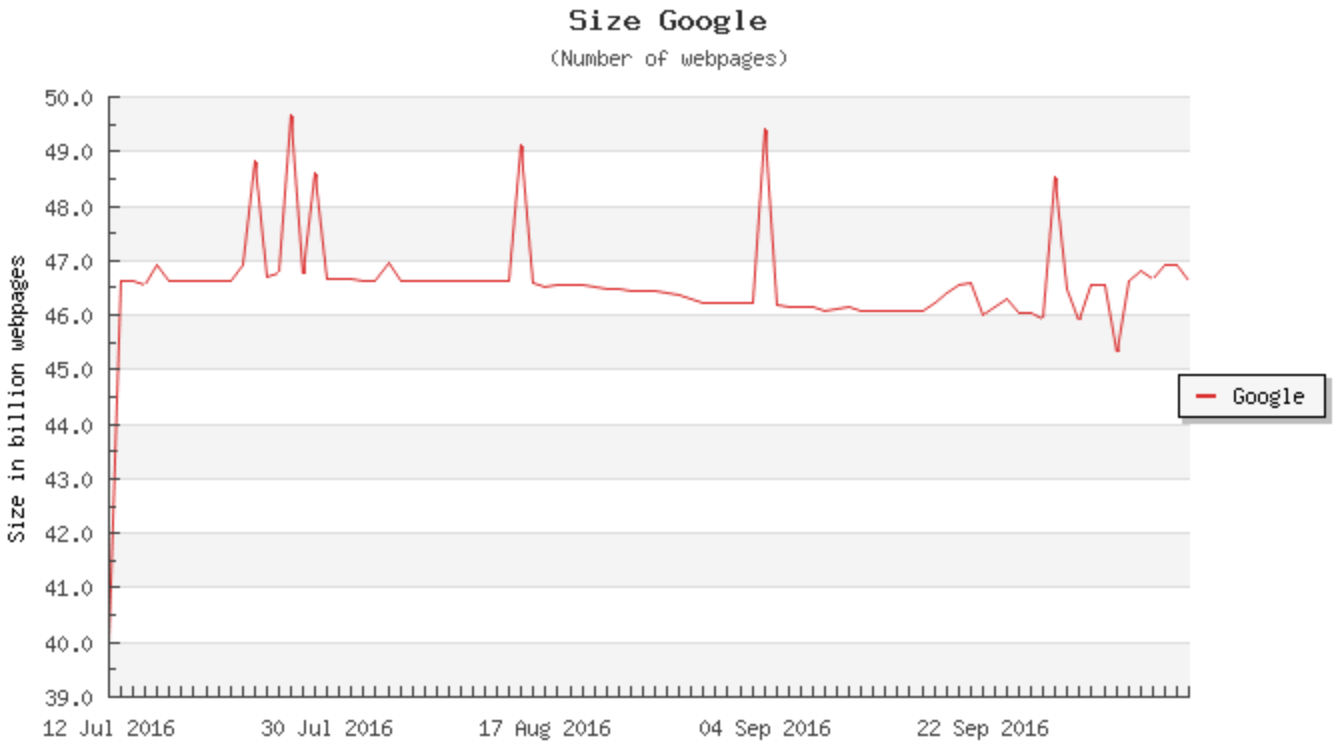
C/S 模式

最直接的服务模式，分为Client和Server，但是面临挑战：

- 单服务器或搜索引擎已不能满足或覆盖日益增长的Web内容需求
- 2×10^{18} Bytes/year Internet上增长
- 但仅 3×10^{12} Bytes/year 可被公众利用 (0.00015%)

C/S 模式的挑战

Google 可搜索的网页



C/S

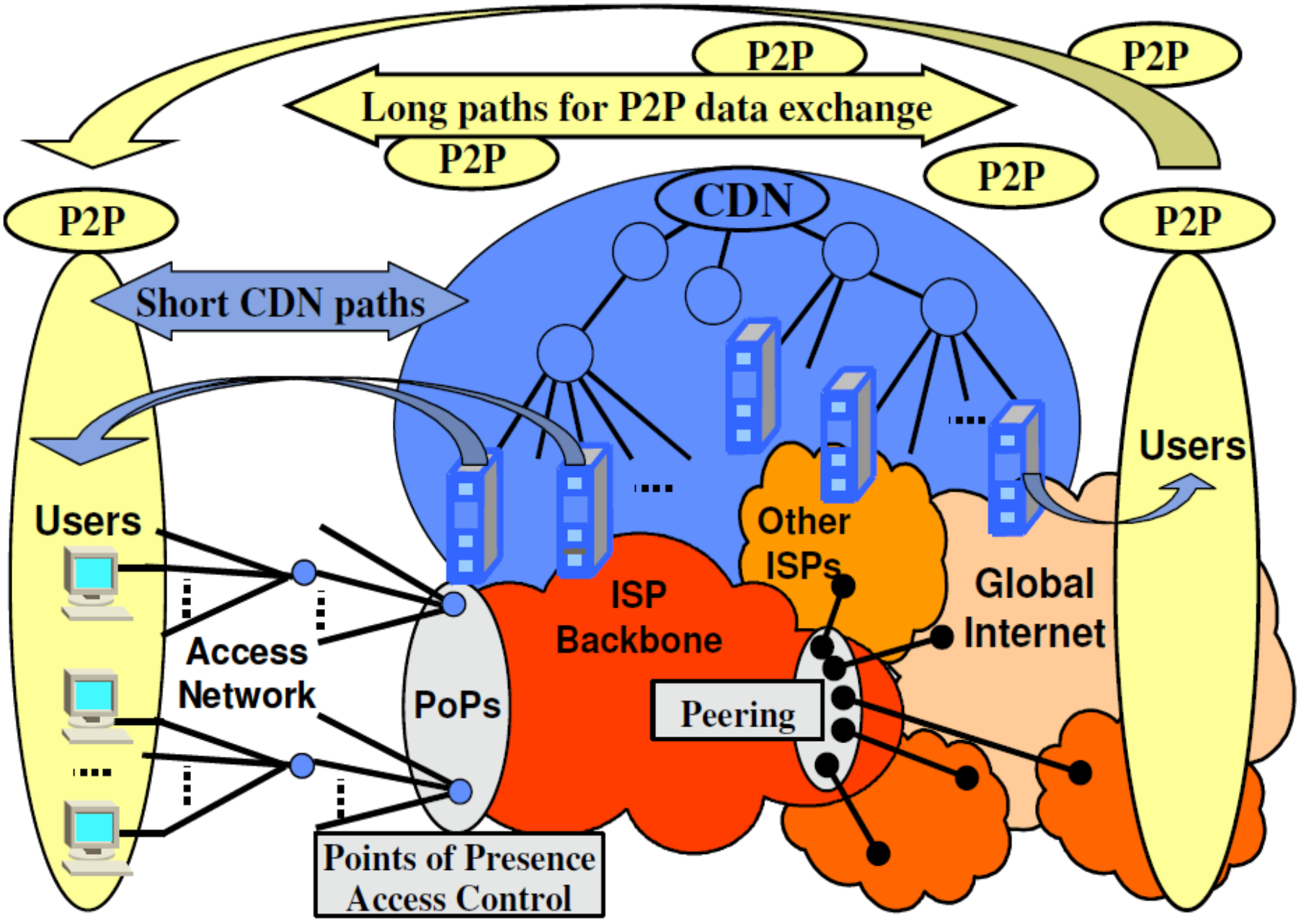
C/S 模式严重限制可用带宽和服务的利用

- 流行的服务器和搜索引擎已成为流量瓶颈
- 但许多高速网络连接的客户端却很空闲
- 客户端的计算能力与信息被忽视

Content Delivery Networks模式

- 服务器在因特网上分散部署（内容重复）
- 分布部署的服务器由总部中心授权控制
- Examples: Internet content distributions by Akamai, Overcast, and FFnet.
- C/S和CDN 模式都有单点失效问题

Transmission paths in CDN and P2P networks

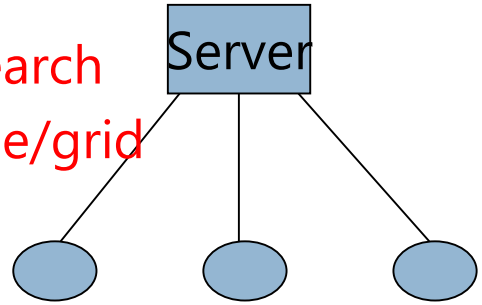


P2P系统

- 既是客户端consumer也是服务器端 producer+consumer=prosumer
- 任何时候都有加入或离开的自由
- 无限的peer diversity: 服务能力、存储空间 、网络带宽和服务需求
- 开放的广域无中心分布系统

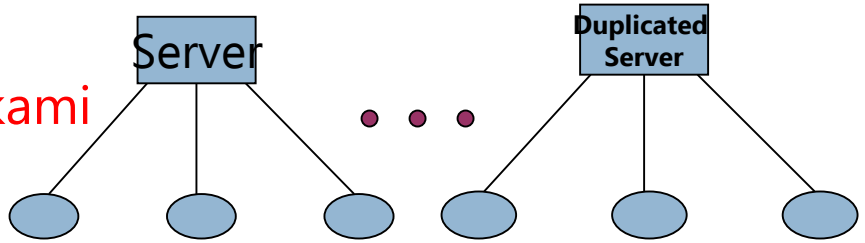
C/S模式

a search engine/grid

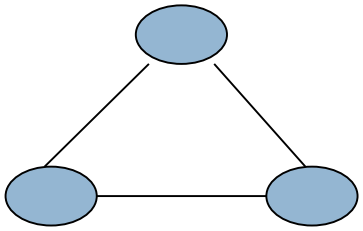


Content Delivery Networks

e.g. Akami



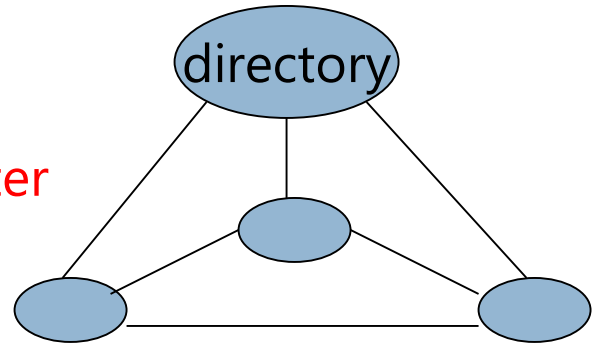
Pure P2P



e.g. Freenet & Gutella

Hybrid P2P

e.g. Napster



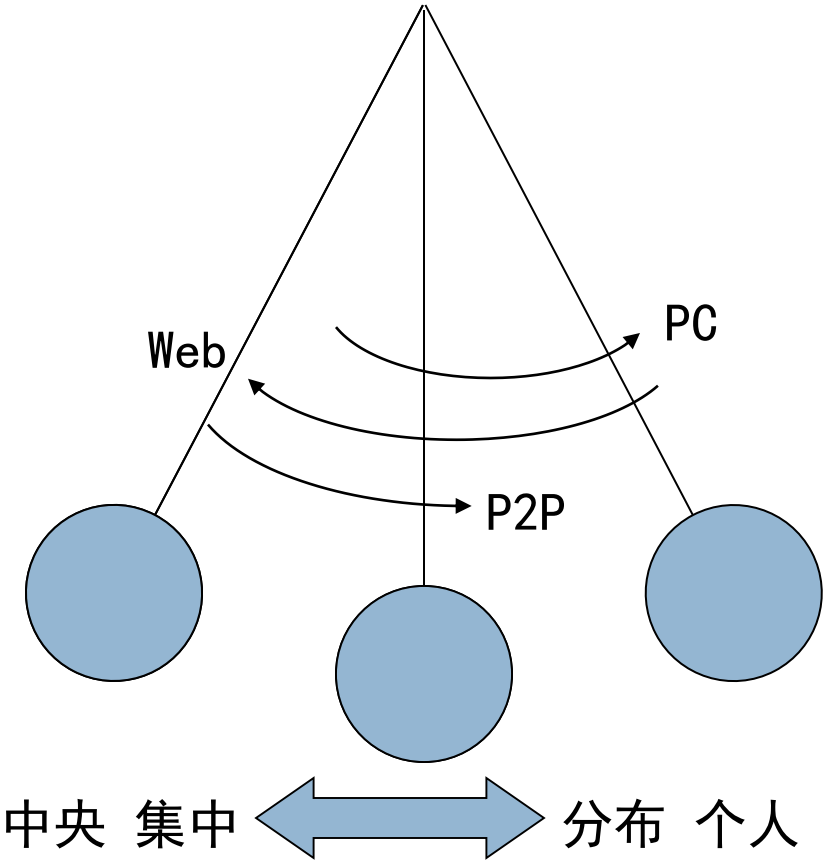
P2P 的目标与优势

- 只要不存在网络的物理断开，目标文件总是可以找到
- 信息可扩展：往P2P系统加入更多内容将不影响其性能
- 系统可扩展：部分节点加入或离开，将不影响P2P 系统的性能

P2P 的目标与吸引力

- P2P是一类发挥互联网边缘资源（存储、处理能力、内容、带宽、用户现场）可用性的应用
- 依赖个人设备（去中心服务）
- 非脆弱的健壮性（无单点故障）
- 柔软性/快速恢复（内建冗余）
- 抗DOS攻击（无中心服务器）
- 更高的可扩展性（随时增加节点）
- 改进的高峰请求服务（提出需求的设备越多，意味着服务器资源也越多）

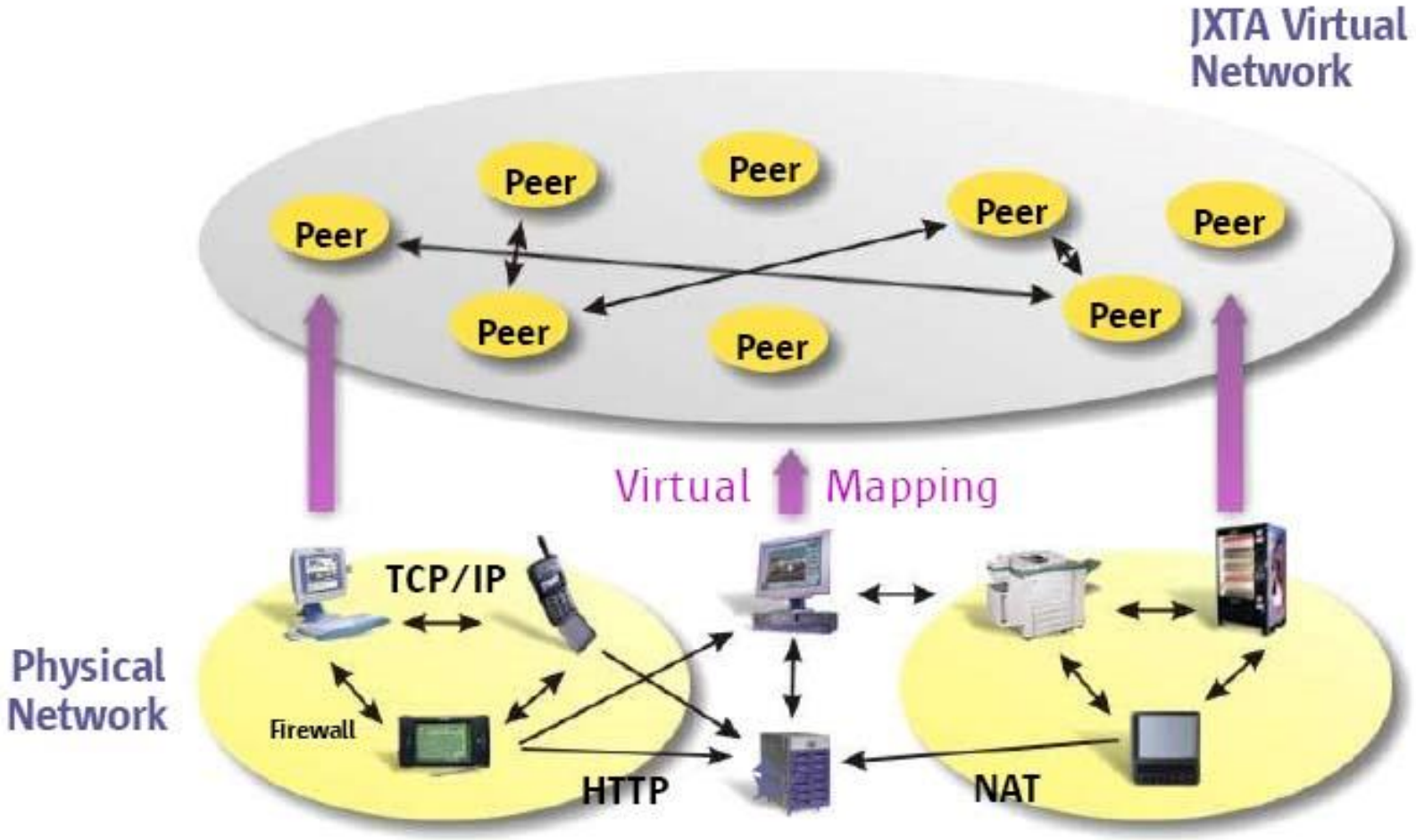
P2P-从集中向分布的演化



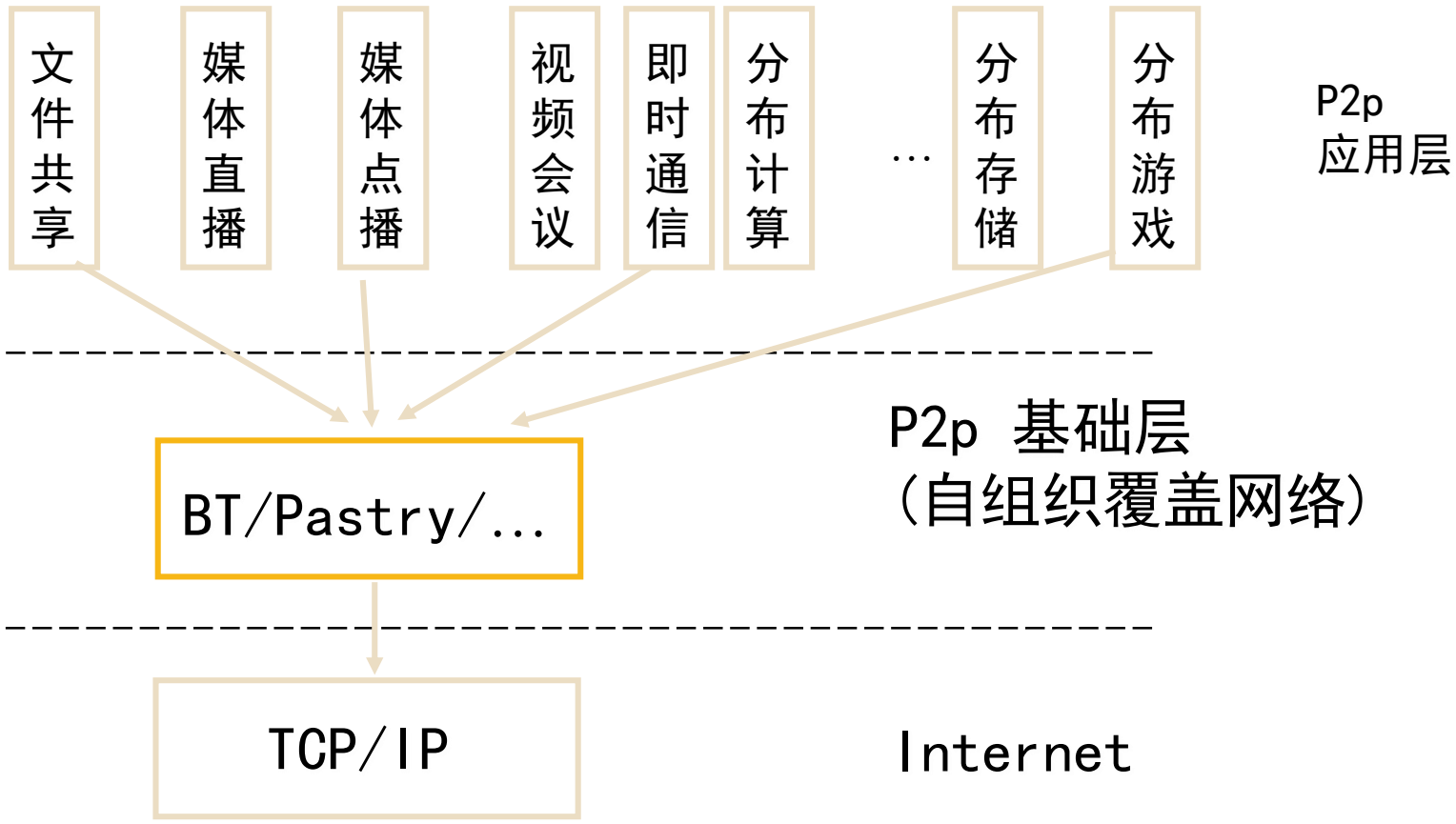
- 将每一个 PC 变成服务器
- 适合自组织 ad-hoc组工作
- 推动采用IPv6,用户直接连接网络
- IPv6可以提供无服务器DNS，“局部无服务器的DNS（LSLDNS）” 也称为多播DNS

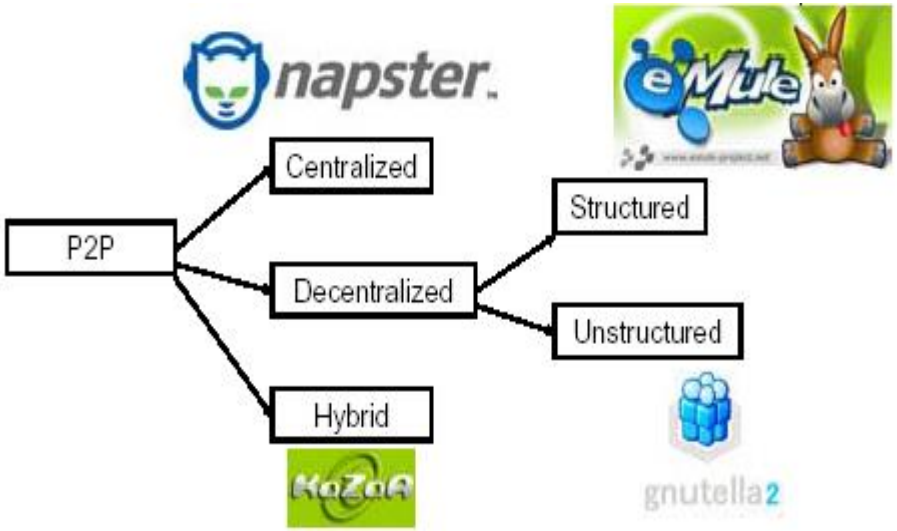
充分发挥互联网无所不在的优势

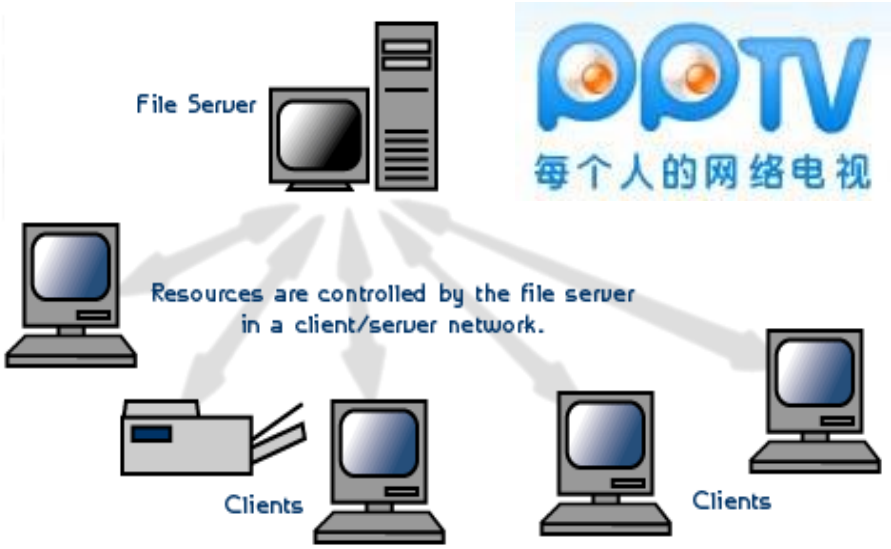
P2P也是一种重叠网络



P2P的应用


文件共享
媒体直播
媒体点播
视频会议
即时通信
分布计算
…
分布存储
分布游戏
P2p
应用层
P2p 基础层
（自组织覆盖网络）
BT/Pastry/...
TCP/IP
Internet





即时通讯软件

skype

流媒体

PPLive

PPStream

下载软件

Maze
天网 Tianwang

KaZaA

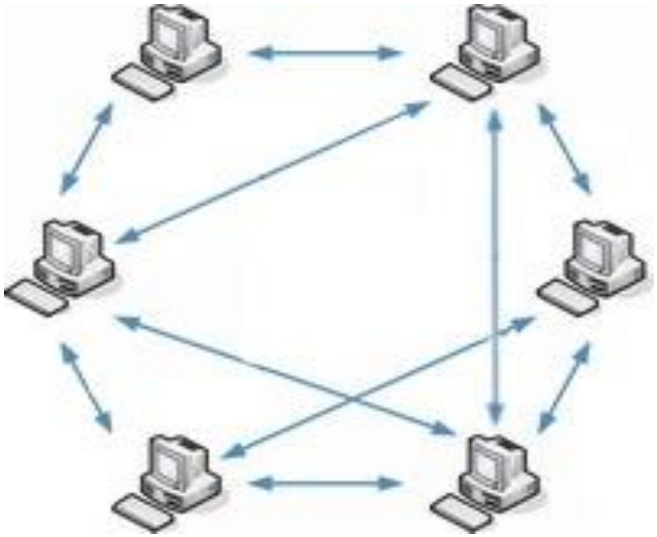
匿名访问

Tor

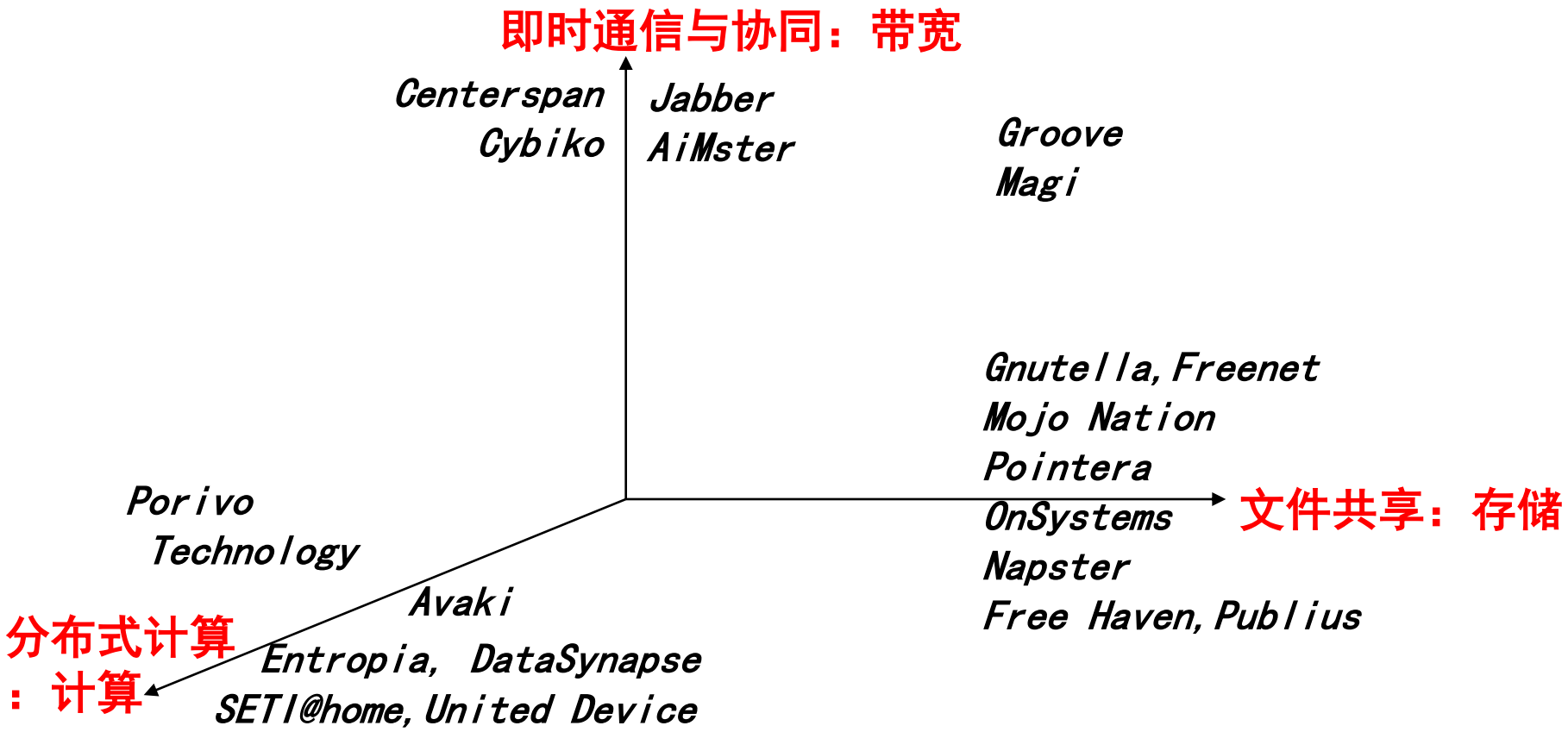
科学研究

SETI HOME

P2P游戏等



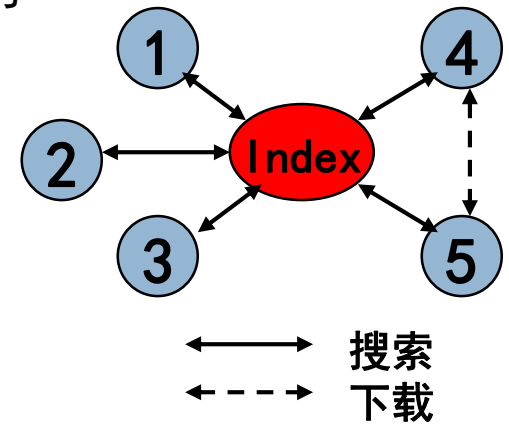
P2P应用的多维视图


即时通信与协同：带宽
Centerspan
Cybiko
Jabber
AiMster
Groove
Magi
Gnutella, Freenet
Mojo Nation
Pointera
OnSystems
Napster
Free Haven, Publius
文件共享：存储
Porivo
Technology
Avaki
Entropia, DataSynapse
SETI@home, United Device
分布式计算
：计算

4.2 混合式P2P网络（第一代）

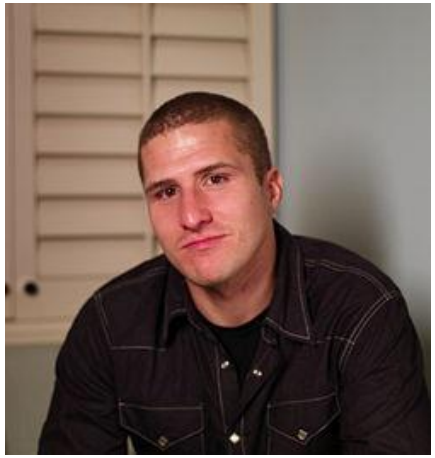
4.2.1 集中目录模式：Napster

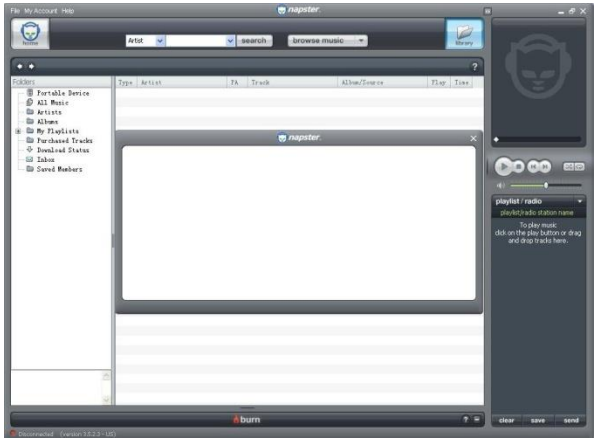
- C/S式集中目录查询，P2P式下载
  - Peers连接到能提供共享内容的中心目录上，匹配请求与索引
  - 文件直接在两个Peers间交换
- 需要一些目录服务器
  - 目录服务器：记载群组所有参加者的信息
- 限制了规模的扩大：
  - 大量用户增加—>大量请求->大量服务器
- 然而Napster经验表明：
  - 除开法律问题外,该模式还很有效和强大
  - 1999.12被多家唱片公司起诉、并败诉



P2P网络先驱 Napster

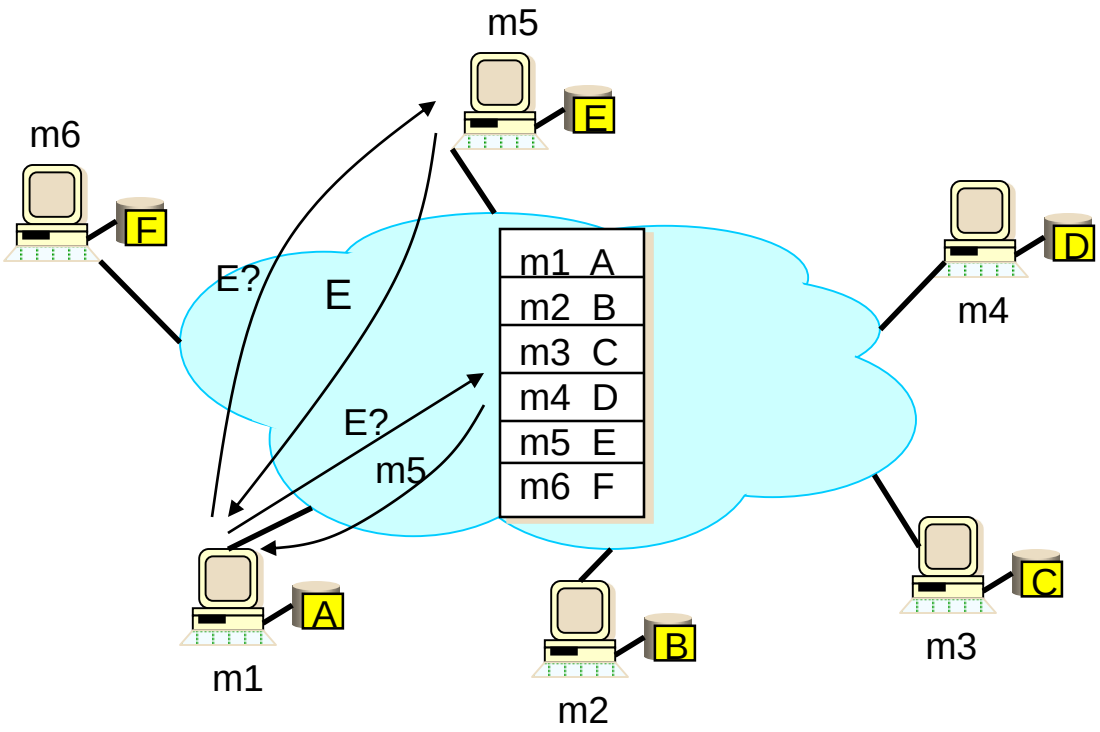
- 1999，18岁Shawn Fanning开发
  - 让音乐迷交流MP3文件
  - 服务器只提供索引，无任何歌曲，MP3文件在Peers自己的硬盘上
- 第一个在互联网上不经过服务器直接交换文件的应用体系
  - 发布半年后，吸引到5000--6100万注册用户





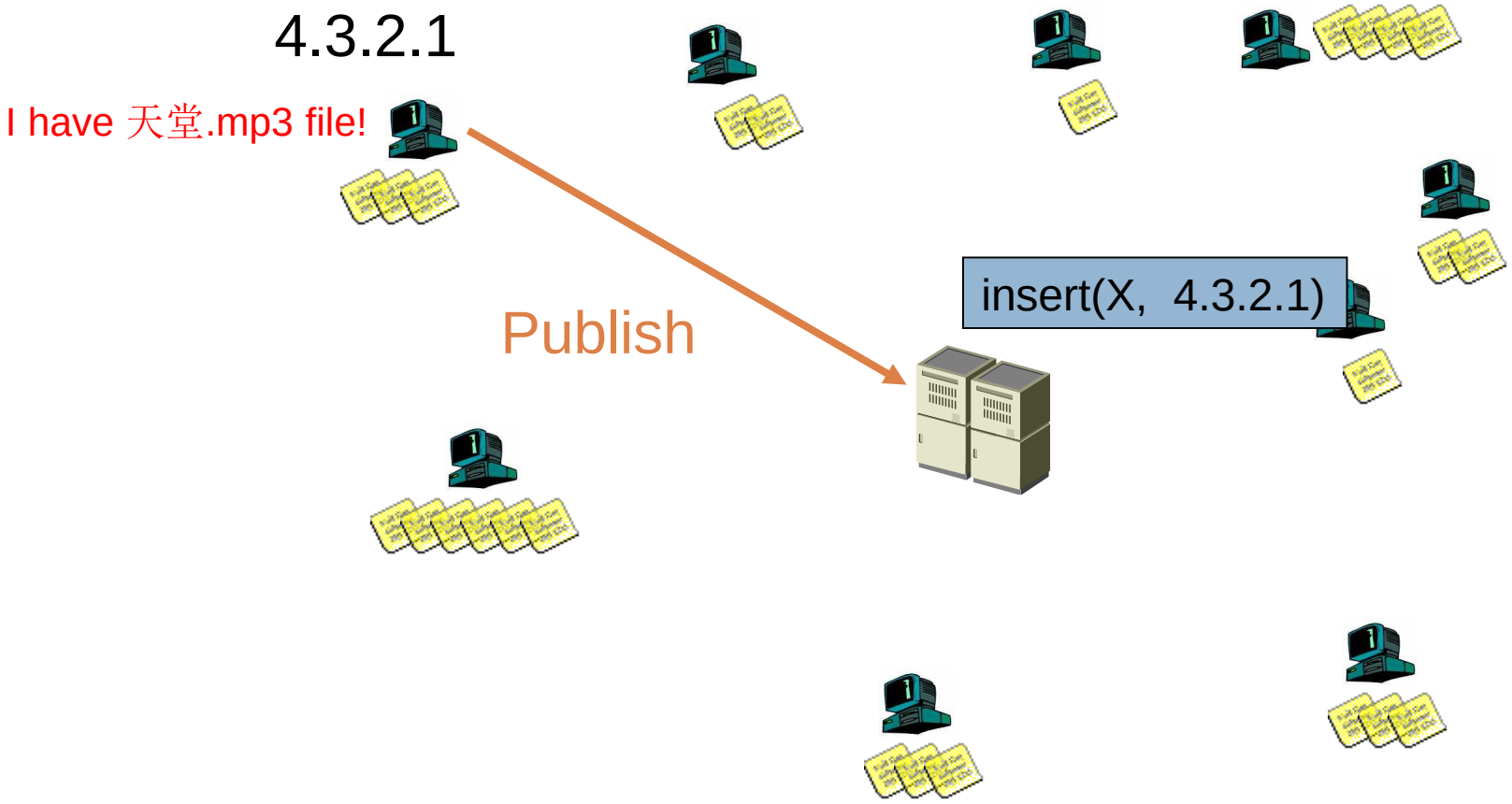
Napster界面

Napster: Example

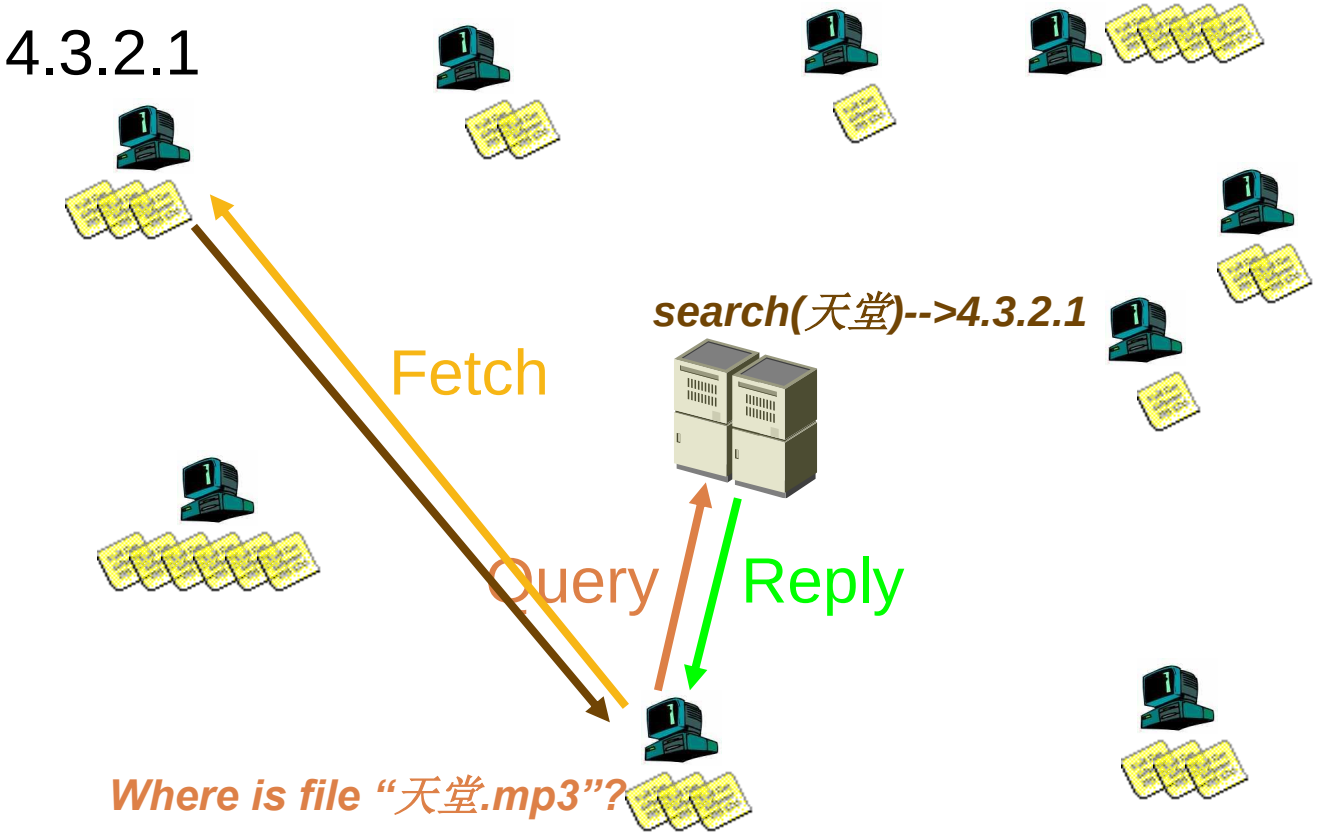


1st Generation

Napster原理



Napster原理



Napster遇到的官司及缺点

- 官司
  - 1999.12.7,美国唱片协会代表7大唱片公司指控Napster侵犯音乐版权，要求法院关闭该公司并赔偿1亿美元；
  - 2000.2月，旧金山第九巡回上诉法院的3名法官裁决，认为Napster一直知道并纵容用户侵权，但没有应要求立即关闭Napster，而是把初判送到低一级地方法院
  - 2001.7.12,法官Patel命令Napster继续关闭直到它可以证明能够有效阻止对版权文件的使用。
  - Napster背后的技术和思想给互联网带来的极大影响，并带来互联网新的Long Tail 现象

Napster官司及缺点

- Napster的缺点：
  - C/S的单点故障、系统瓶颈、可扩展性低
  - 未考虑不同用户的能力差异、无鼓励机制
  - 版权问题

Governments are good at cutting off the heads of a centrally controlled networks like Napster, but pure P2P networks like Gnutella and Tor seem to be holding their own.

---- Satoshi

4.2.2 分片优化的BitTorrent

- BitTorrent
  - 2002.10，穷困潦倒的Bram Cohen发明BT，免费软件企业家John Gilmore帮助他解决了部分生活费
  - 2003初，在Internet2(IPv6连接200多所美国大学的主干)用BT发行一个新版的Linux和日本卡通，速度比ADSL快3500多倍，10月流量超过Internet2流量的10%

4.2.2 分片优化的BitTorrent

- BitTorrent
  - Valve软件公司的董事Gabe Newell正在游戏分发网络，在Seattle给他一个职位，开发Half Life 2中使用的数据传输系统，叫做Steam（Steam主要基于CDN技术）
  - BT既指一个混合式P2P网络，也指其对应的协议及支持该协议的应用软件--BitTorrent/BitComet/BitSpirit/FlashBT

BitTorrent原理

BT下载，首先需要一个种子文件，种子文件包含以下数据：

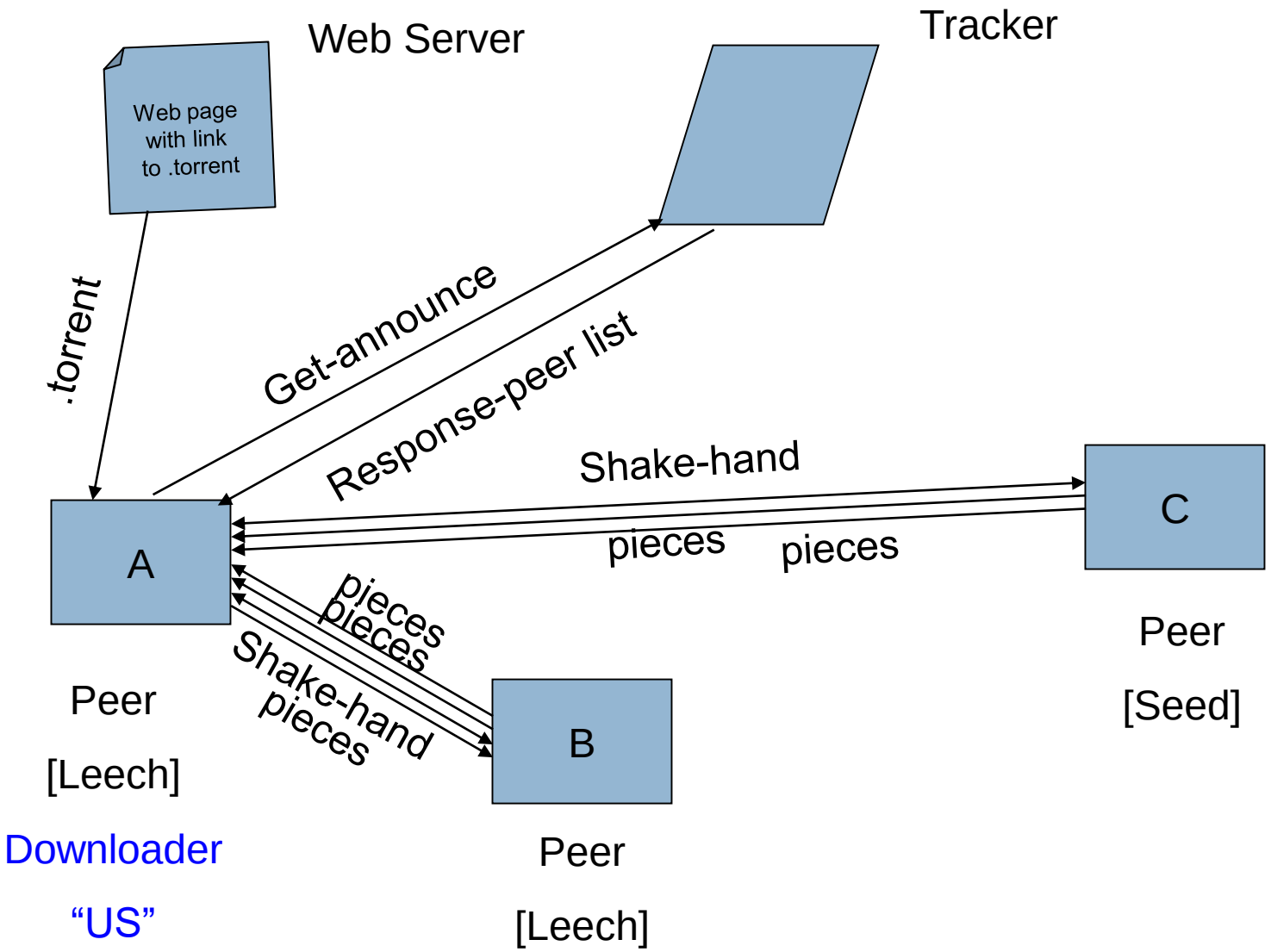
- announce - tracker的URL（tracker是特殊的服务器，帮助peers之间联系）
- info - 该条映射到一个字典，该字典的键将取决于共享的一个或多个文件：
  - name - 建议保存的文件名（一个文件）和目录名称（多个文件）
  - piece length - 每个文件块的字节数。通常256KB = 262144B
  - pieces - 每个文件块的SHA-1的集成Hash。因为SHA-1会返回160-bit的Hash，所以pieces将会得到1个160-bit的整数倍的字符串。
  - length - （只有一个文件时）文件的大小
  - files - （多个一个文件时）一个字典的列表（每个字典对应一个文件）与以下的键：
    - path - 一个对应子目录名的字符串列表，最后一项是实际的文件名称
    - length - 文件的大小（以字节为单位）

BitTorrent原理

- 上载
  - 某Peer想要共享文件或目录，则为该文件或目录生成一“种子”文件
  - 然后把该“种子”上传到BT网站上
- 下载
  - 需下载的Peer到BT网站上找到所需种子
  - 根据种子信息进行下载

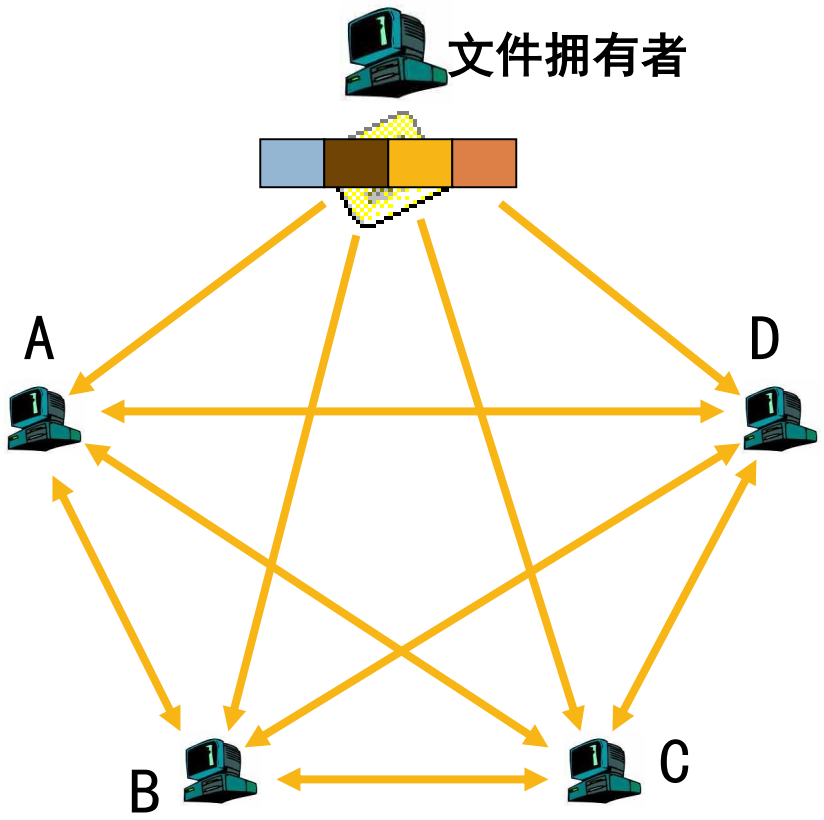
1. 用户通过BT网站搜索文件，得到.torrent文件列表
2. 用户选择列表中的一项或多项
   - 每个被选.torrent文件会启动一项下载任务
   - 通常.torrent会指向该文件对应的Tracker，而Tracker会把一部分用户信息给请求者
3. 用户根据Tracker用户信息
   - 与其它用户建立连接
   - 从它们下载文件分片，也提供分片
4. 高速高效
   - 并不总是和最初邻居交换分片，而是每隔一段时间从Tracker获得新的邻居
   - 采用“阻塞算法”主动停止那些对自己无贡献的邻居，寻求更好的邻居

BT 的结构


Web Server
Tracker
Web page with link to .torrent
.torrent
Get-announce
Response-peer list
Shake-hand
pieces
pieces
A
C
pieces
pieces
Shake-hand
pieces
B
Peer
[Leech]
Downloader
"US"
Peer
[Seed]
Peer
[Leech]

BitTorrent的分片与分发

- 分割分片
  - BT把文件分割成相同大小的块，典型256KB，为了并发下载再分为16KB的子分片
  - 用SHA-1对每个分片或子分片生成校验值或文件块ID;保存在.torrent中，
  - 只有在验证其唯一性完整性后才通知其它peer自己拥有该片
- 流水分发
  - 在TCP之上，可流水地同时发送多个请求，通常5个，以避免两个分片间的延迟



BitTorrent的分片与分发

- 分片选择的阶段规则
  - 最初：随机选择一个分片
  - 分片下载中：整分片优先：一旦请求了某个分片的子分片，则该分片剩下的子分片优于其他分片被请求，以尽快获得一个整分片
  - 文件下载中间阶段/平稳期：最少者优先：选择Peers拥有最少的分片，最新的分片（非常关键）
  - 最后：尽快取消。为防止最后阶段的潜在延迟，多发请求；一旦得到最后子分片，就向其它用户发出Cancel消息。

BitTorrent原理

- 协议
  - 种子文件上传下载、Peers和Tracker间通信都是使用HTTP协议
  - 各Peers间通信使用BitTorrent Peer协议（TCP）
- 问题
  - Tracker的单点失效
  - 未对Peers身份认证
- 开发
  - BitTorrent协议公开，任何人可开发服务器端或客户端
  - 国内流行BitComet，用Python语言开发

BitTorrent的阻塞算法

- Tracker并不集中分配资源，而由用户控制
  - Peer尽可能提高自己的下载效率
  - Peer根据下载方决定对其上传回报（tit-for-tat）针锋相对
    - 对合作者，提供上传服务
    - 对投机者，阻塞对方，暂时拒绝上传服务
- 阻塞算法（Choking algorithm）
  - 经济学背景-Pareto efficient:当系统中资源配置已达到这样一种境地：任何重新改变资源配置的方式，都不可能使一部分人在没有其它人受损的情况下受益。
  - 一个Peer使用Choking会暂时停止一个P2P邻居从该Peer下载资源

BitTorrent的阻塞算法

- 阻塞算法（Choking algorithm）
  - 防止邻居仅仅下载不上传：每个用户一直保持4个邻居的疏通。每隔20秒（10+10）轮询，每隔10s决定将上传数据给自己速率最低的邻居设置为Choking状态，并保持该状态10s，10s足够TCP调整传输率到最大。如果还是处于Choking，就换邻居。
  - 新的邻居加入后，可能会提高下载带宽，就找到了更好的邻居。
  - 那么一个新的用户，不拥有任何分片也就不能上传，会不会被一直Choking？
  - 优化疏通（Optimistic unchoking）：在任何时候，每个peer都拥有一个称为“optimistic unchoking”的连接，这个连接总是保持疏通状态，而不管它的下载速率是怎样。每隔30秒，重新计算一次哪个连接应该是“optimistic unchoking”。30秒足以让下载能力也相应的达到最大，这样即使是个新加入者，也有机会下载分片。

磁力链接

- 随着BT服务器的大量关闭，导致很多种子文件无法获取。既然BT网站上每个种子都不同，那么就有不同的Hash值，种子文件便可以简化成一个链接，这便是磁力链接（Magnet links）。
- 磁力链接是以“magnet:?xt=urn:btih:”开头的字符串，后面则是资源文件的infohashes值，例子：magnet:?xt=urn:btih:D3F7987EDF395104240AA16B30564B7372E55FFA
- 该链接为变形金刚4的磁力链接，是从迅雷下载中直接复制得到。由于Hash是不可逆的，所以从磁力链接上不能倒推出种子文件的信息，需要借助第三方种子库完成该转换。

磁力链接

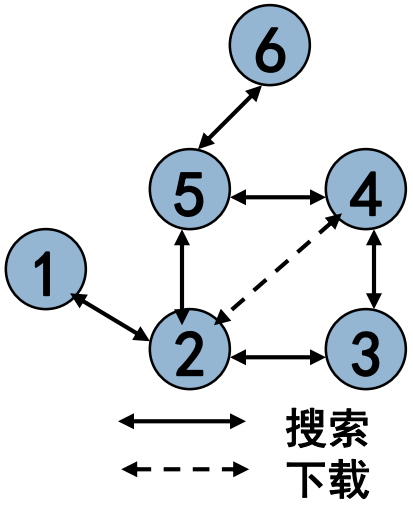
- 迅雷种子库hxxp://bt.box.n0808.com提供该转换服务：
  - 转换时需要访问：hxxp://bt.box.n0808.com/D3/FA/D3F7987EDF395104240AA16B30564B7372E55FFA.torrent
  - 即可下载对应的种子文件，其中的/D3/FA分别为infohashes的第一个字节和末尾字节。
- 我们在使用迅雷打开磁力链接进行下载时，迅雷会先从种子库中获取种子，得到具体文件信息后再进行下载。

优缺点总结

- 拓扑结构：服务器仍然是网络的核心
- 底层协议：全部使用TCP，限制了链接的Peer数量
- 查询与路由简单高效：
  - Napster和BT的用户访问服务器；服务器返回文件索引或种子文件；用户再直接同另一Peer连接
  - 故路由跳数为O(1)，即常数跳
- 容错、自适应和匿名性
  - 服务器单点失效率高
  - 自组织和自适应主要依靠服务器
  - 服务器的存在使匿名性实际不可能
- 用户接入无安全认证机制

4.3 无结构P2P网络（第二代）

- 过程：洪泛请求模式
  - 每个Peer的请求直接广播到连接的Peers
  - 各Peers又广播到各自连接的Peers
  - 直到收到应答或达到最大洪泛步数(典型5-9步)
- 特点
  - 可智能发现节点，完全分布式
  - 大量请求占用网络带宽,可扩展性并不一定最好,
- 协议
  - Gnutella/KaZaA/eDonkey/Freenet使用该模式
  - 消息协议：用于节点间相互发现和搜索资源
  - 下载协议：用于两节点间传输文件（使用标准的HTTP协议:GET）
- 改进
  - Kazaa 设立Super-Peer客户软件,以集中大量请求
  - Cache最近请求



- 何为无结构或结构化P2P
  - 根本区别在于每个peer所维护的邻居是否能够按照某种全局方式组织起来以利于快速查找。
- 无结构网络优点
  - 将重叠网络看作一个完全随机图，结点之间的链路不遵循某些预先定义的拓扑来构建。
  - 容错性好，支持复杂查询
  - 结点频繁加入和退出，但对系统的影响小。

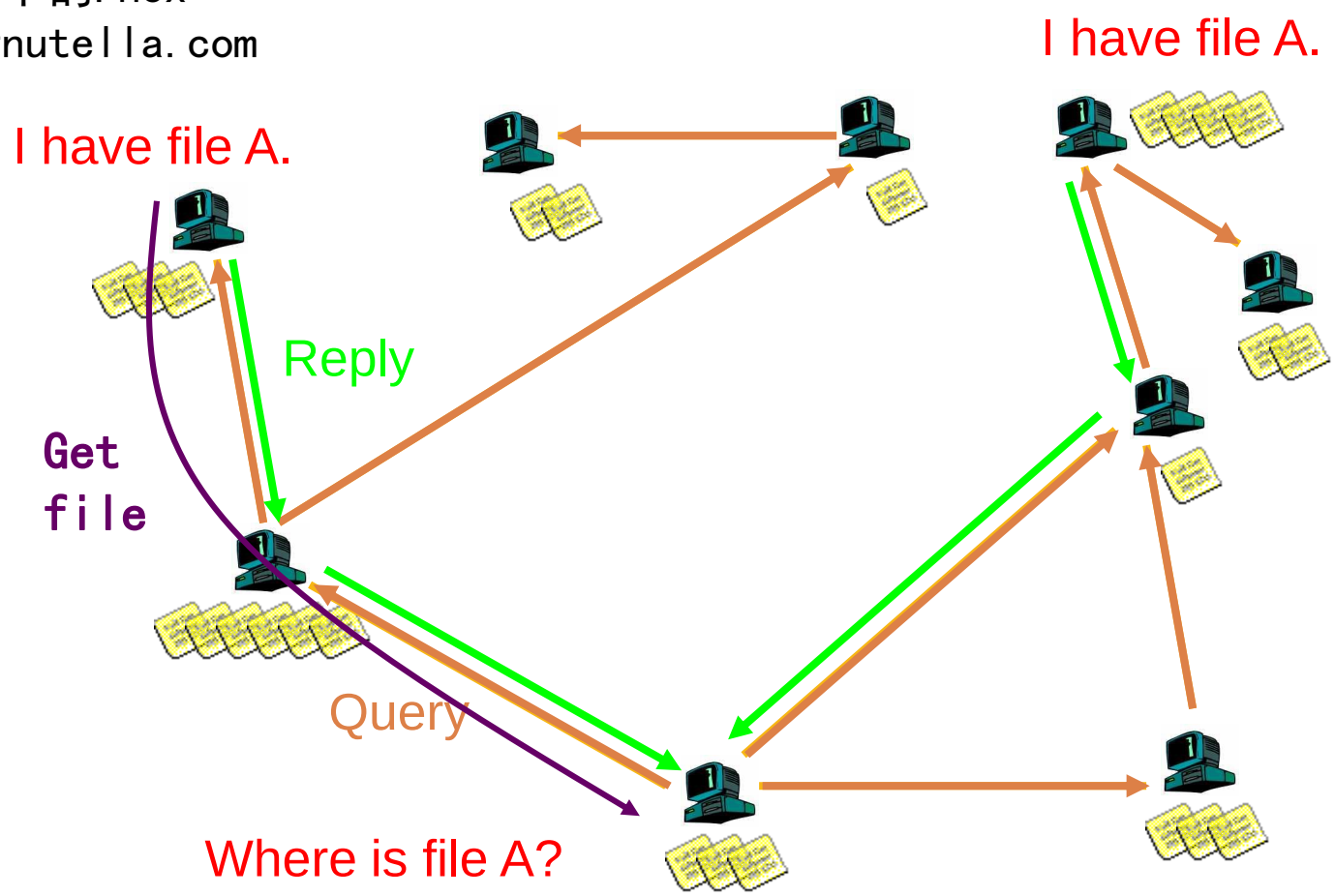
4.3.1 Gnutella

- 背景
  - 2000.3诞生于Nullsoft公司，由MP3播放软件WinAmp的设计者Justin Frankel和Tom Peper发明
  - 3.14日Napster版权案出现后， Nullsoft的母公司AOL（America On Line）在该软件于Gnutella网站上公布仅1.5小时后就关闭了该网站
  - 但就在这1.5小时间几千用户MP3迷下载了Gnutella，并将其公开、改造和克隆，保留下来。

- 现状：Gnutella更多指这种无结构P2P网络协议
  - gnutella.wego.com:改造或克隆软件
  - www.gnutellaworld.net:交换各种相关信息
  - Rfc-gnutella.sourceforge.net:协议文档0.4，0.6建议使用UltraPeer
  - www.Gnutella.com：商业网站，各种应用软件集合和联盟

认识Gnutella

- Gnutella是开源软件
- 衍生出Windows/Linux平台下诸多客户端
- 最新Windows平台下的Phex
- 下载地址：www.gnutella.com



Gnutella原理

- 纯分布式对等
  - 每个Peer即使服务器也是客户机
  - 每个Peer监视网络局部的状态信息，相互协作
- 工作过程
  - 原始加入：必须首先连接到一个“众所周知”几乎总是在线的【或称中介、自举、入口】节点（功能同一般Peer），进入Gnutella网
  - 查询和应答消息采用广播或回播（Back-propagate）机制
    - 每条消息被一个全局唯一16字节随机数编码为GUID（128 bit）
    - 每个节点缓存最近路由的消息，以支持回播或阻止重广播
    - 每条消息都有一个TTL数，每跳一次减少一次

Gnutella典型消息

- 组成员消息（找到新邻居）
  - Ping：新节点加入网络时用=I am here !；探测其它节点=Are you there?
  - Pong：收到ping后决定是否回播pong消息，然后将ping广播给邻居节点（含IP地址、端口号、共享文件数量和大小）=Yes，I am here
- 查询消息
  - Query：指定查询文件和响应速度等信息，每个查询有唯一ID= I am looking for…
  - Query Response：回应包含IP地址、端口号、位置和带宽等信息，以及命中节点的NodeID，本消息沿来路回播=Your wanted file is here…

Gnutella典型消息

- 文件传输消息
  - Get:获取文件 = I’ll get…from you
  - Push:请求Firewall后的文件拥有节点，主动建立连接把文件上传给自己 = I can’t get…from you ,so you push it to me

Gnutella的问题与改进

- 基本呈幂率分布：有连接数L的节点数占网络总节点数的%比，正比于L-a（显然L越大，占比越低） Gnutella的a=2.3＜3，节点随机失效的容错性比较高，但是不能防止某些节点失效而使Gnutella分裂。

- 幂律：哈佛的语言学家zipf在研究语料库的时候发现的，所以也叫齐普夫定律，如果把单词出现的频率按由大到小的顺序排列,则每个单词出现的频率与它的名次的常数次幂存在简单的反比关系，或者说，二者乘积为一个常数：

  其公式为：P(r) = C / r^α （r为rank值，P(r)为出现频率）

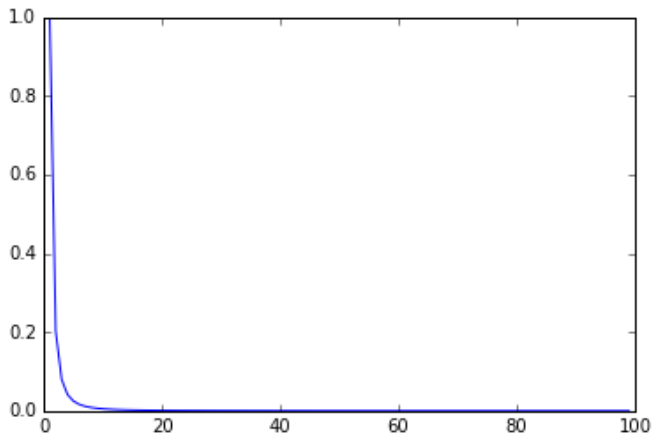
- 在zipf分布中，提高α即可使分布迅速降低为零，就是常见的“80/20法则”。80%的资源掌握在20%的人手里。前20%的单词出现频率占所有单词的80%。

- 如果α值比较小就是长尾分布

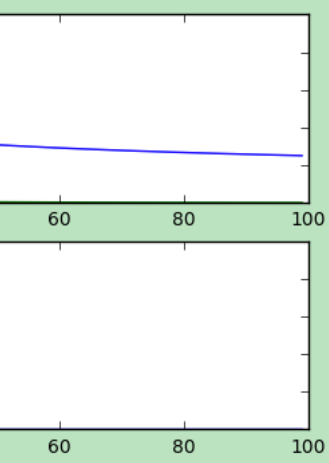
Gnutella的问题与改进

可以画出Gnutalla的客户端有连接数L的节点占到的百分比，左图可以看出绝大部分在20以下；右图为α不同值的比较。

```
In [7]: %pylab inline
        L = range(100)[1:]
        print L
        plot(L, [i**(-2.3) for i in L])

Populating the interactive namespace from numpy and matplot
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52
 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87

Out[7]: [<matplotlib.lines.Line2D at 0x7844c50>]
```



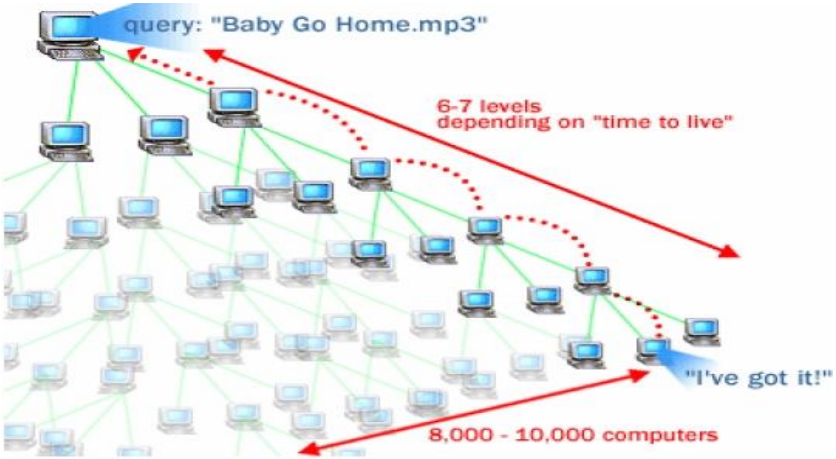

```
In [31]: %pylab inline
         L=range(100)[1:]
         subplot(2,1,1)
         plot(L,[i**(-0.3) for i in L])
         plot(L,[i**(-1.3) for i in L])
         subplot(2,1,2)
         plot(L,[i**(-3.3) for i in L])

Populating the interactive namespace from numpy and matplotlib

Out[31]: [<matplotlib.lines.Line2D at 0xb1ebeb8>]
```



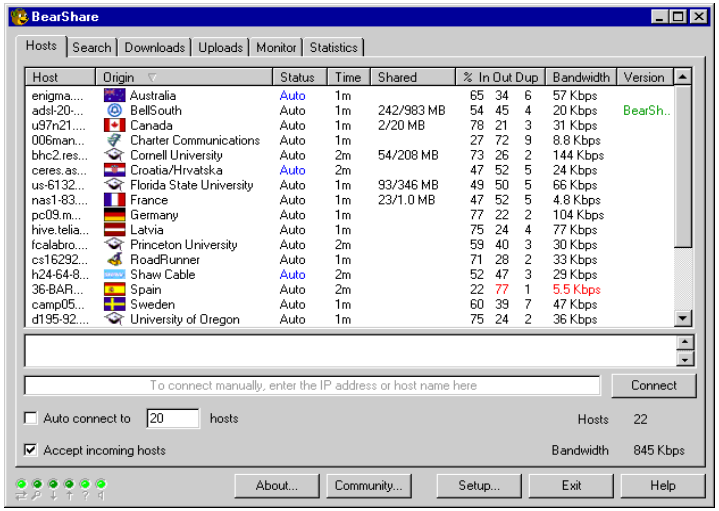
Gnutella的问题与改进

由于Gnutella每个节点链接数量有限：

- 洪泛广播加重网络带宽负担，受TTL限制，消息只能达到一定范围，这又导致有些文件不能查询到
- 改进：分层P2P=增加超级节点负责查询消息的路由，构成P2P骨干网，叶节点只是通过超级节点代理接入



最初的Gnutella采用的Flooding搜索算法示意图

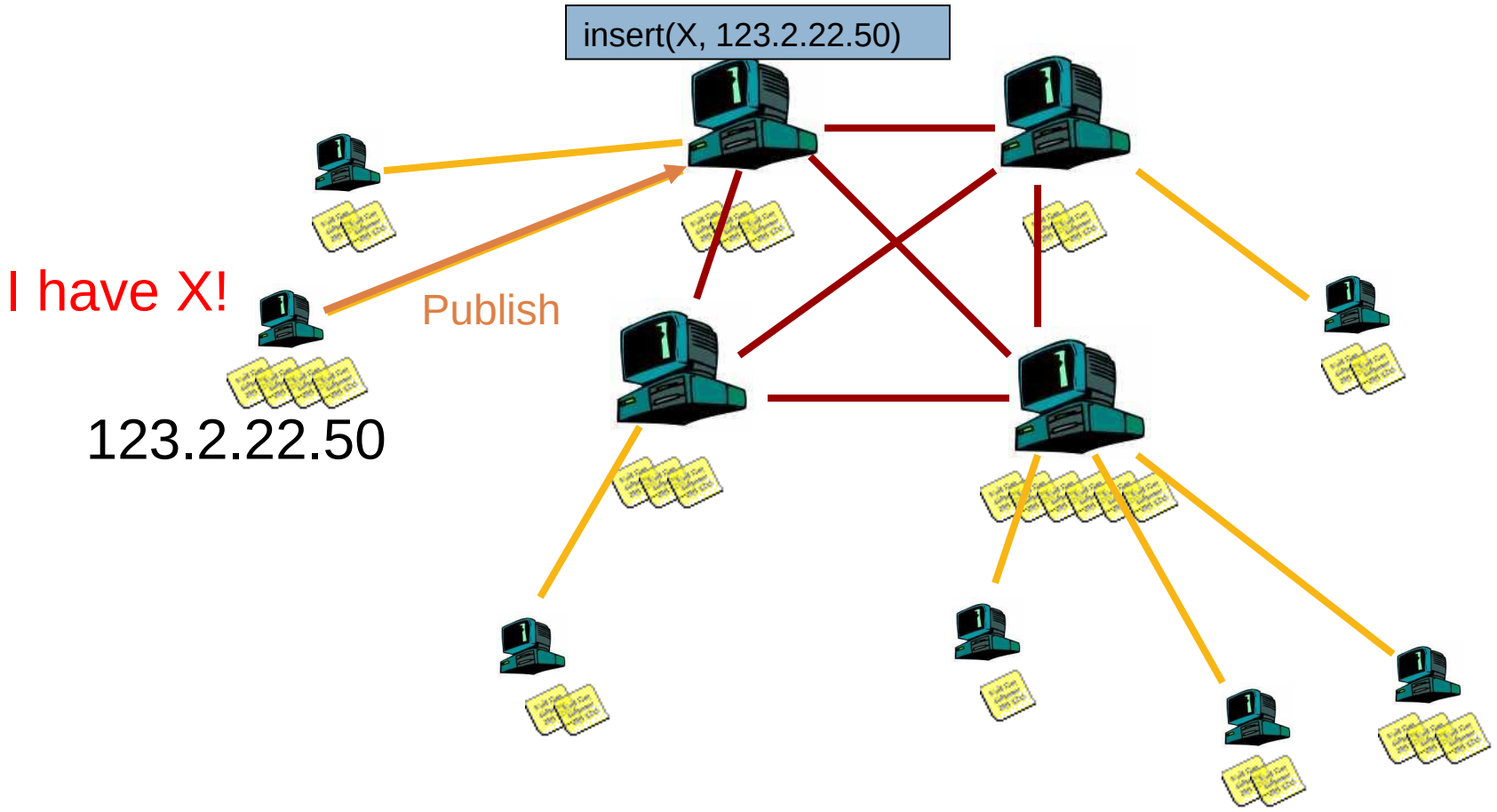


采用第二代Gnutella协议最经典的软件-Bearshare

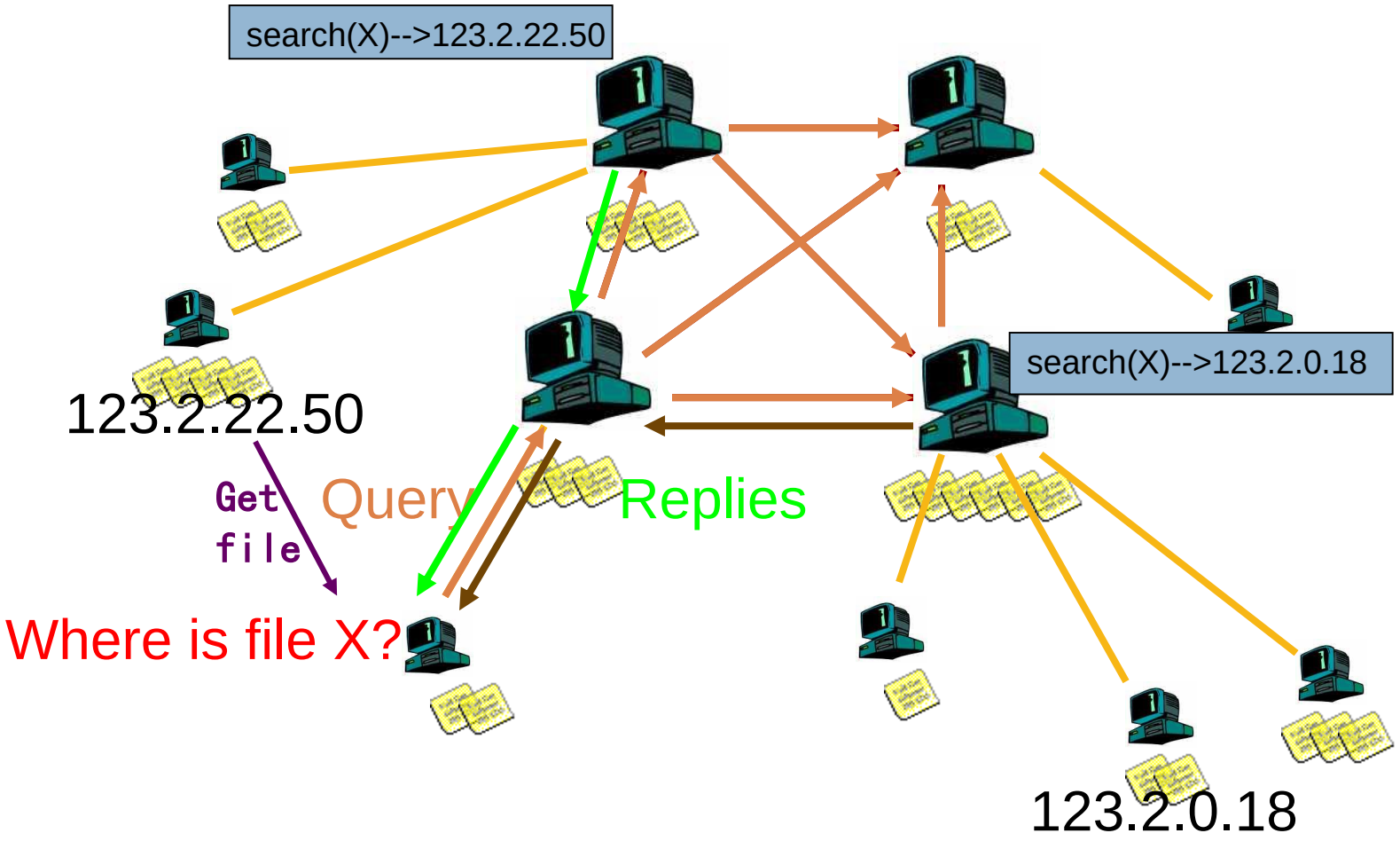
4.3.2 基于超节点的KaZaA

- 2000.7，斯堪的纳维亚的Niklas和丹麦的Friis开发
  - Niklas是著名P2P企业家，在KaZaA之后，创办了：
    - Joltid公司：推广P2P解决方案和P2P流量优化技术
    - Altnet公司：第一个安全应用P2P网络，发行数字版权管理许可证
    - Skype公司：全球第一家实时语音通信公司
- 基于FastTrack协议，主要用于MP3 music files共享
  - 比Gnutella更早引入SuperNode
  - KaZaA是专有协议，对消息加密，存在超级和普通两类节点
  - 超级：高带宽、高处理能力、大存储容量、不受NAT限制
  - 普通：低带宽、低处理能力、小存储容量、受NAT限制
- 加入、上载与查询
  - 普通节点选择一超级节点作为父节点加入，并维持半永久TCP连接
  - 将自己贡献的文件元数据、描述符上传给它，并生成Hash值
  - 父超节点根据文件描述符关键字查询，返回文件所在IP地址+元数据

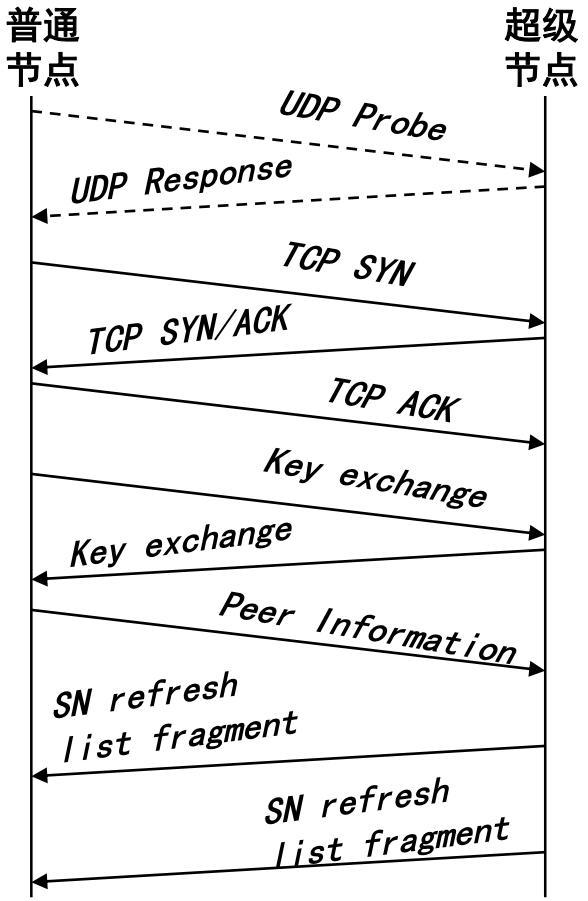
KaZaA共享文件过程



KaZaA原理

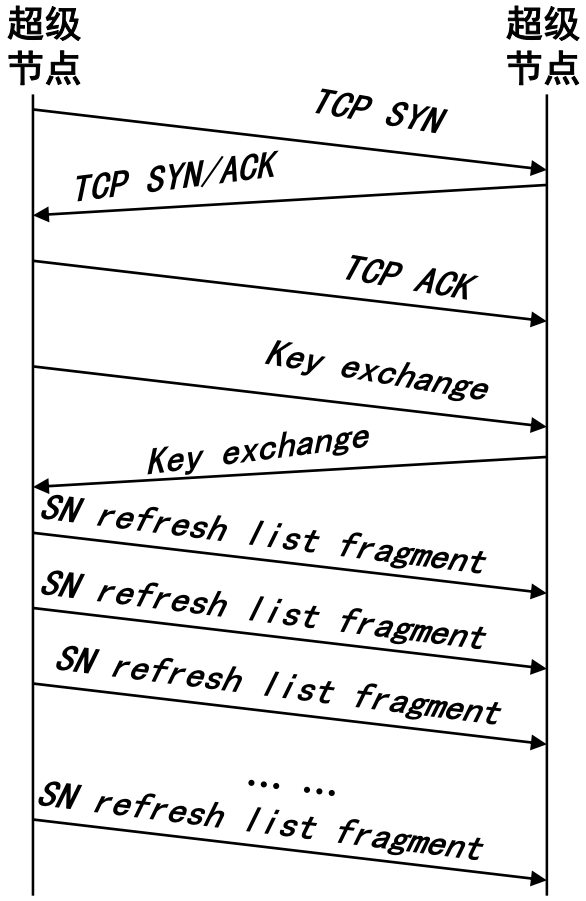

search(X)-->123.2.22.50
search(X)-->123.2.0.18
123.2.22.50
Get file
Query
Replies
Where is file X?
123.2.0.18

KaZaA连接的建立和消息格式

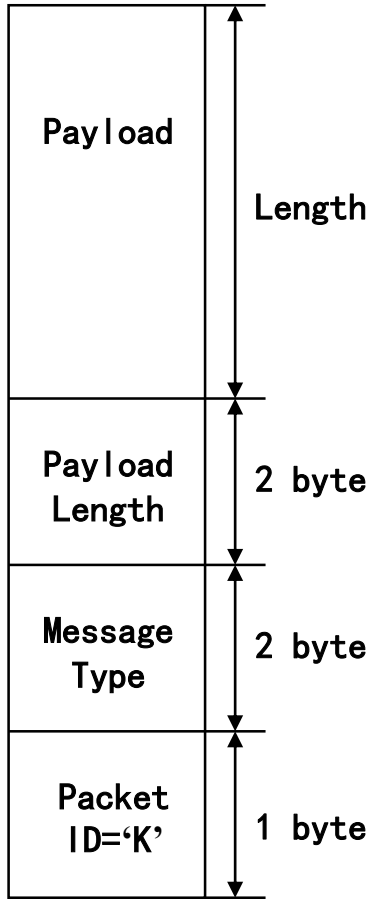


普通到超级连接的建立

SN=Super Node



超级节点间连接的建立



消息格式

KazaA的问题

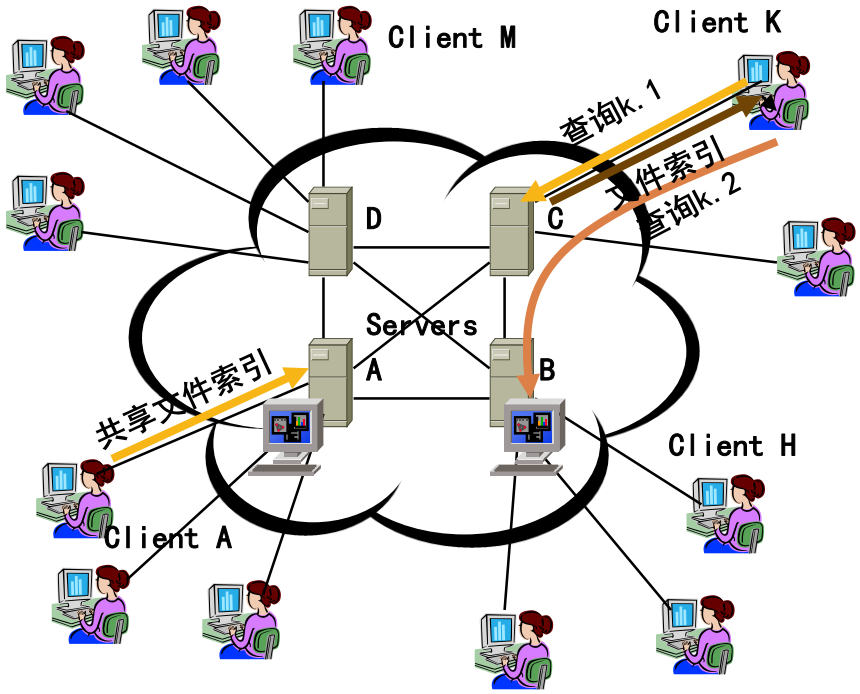
- KazaA是由Consumer Empowerment公司运营的，所以同样遭到了多个国家的法律诉讼
- KazaA后来出售给其他公司，只接受用户按月订阅的音乐服务
- 少部分用户仍然维护原来的KazaA网络

4.3.3 eDonKey/eMule/Overnet

- 背景
  - eDonKey，2000年, Jed McCaleb创立专注文件下载
    - 与BT类似，文件分块下载；内容Hash作完整性验证，服务器为核心
    - BT是基于文件、由 Tracker服务器来查询、搜索和跟踪用户；但eDonKey是基于用户的类似KaZaA的超级节点。
  - eMule，2002.5.13， Merkey因不满eDonKey客户端功能创建
    - 在eDonKey上加入新功能、优化图形界面
  - Overnet是一个独立的分布式搜索应用
    - 被eDonKey整合到自己的体系中
- 特点
  - 分块下载的双层无结构P2P网络

eDonKey结构与过程

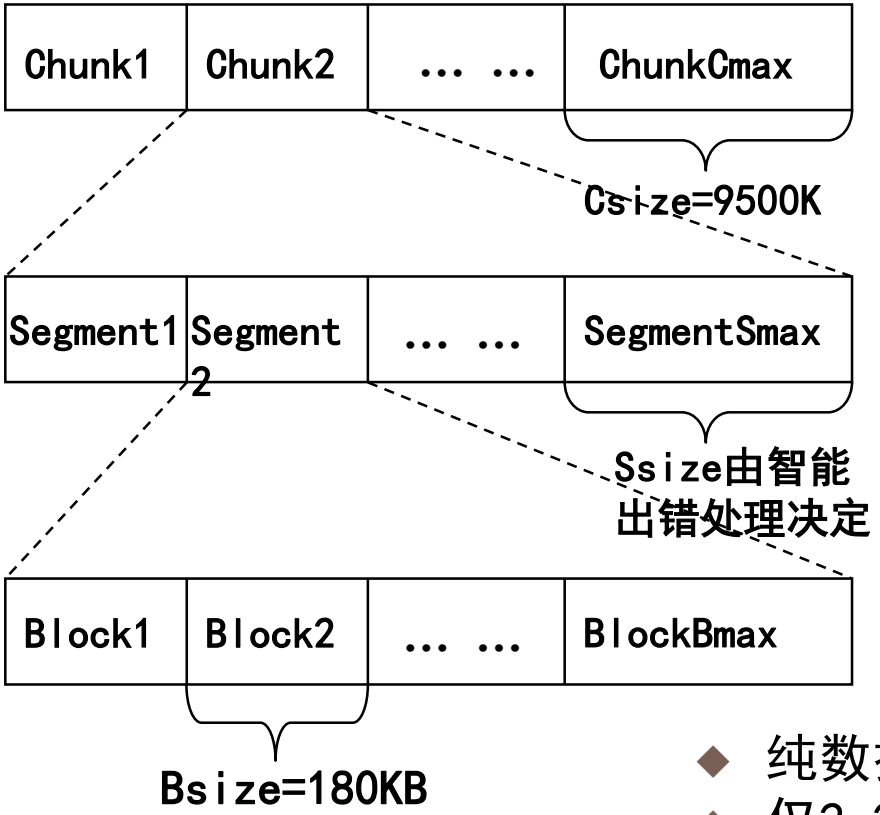
- 结构
  - 服务器层Server或超节点+客户端层Client
  - Server间交换文件索引和服务器列表
  - 每个C连接到一个S进行文件查询和S列表更新
- 加入与查询
  - 连接最适合S
    - 与曾经的“入口S”列表中的S建立连接，从中选择离自己延时最小的那个
    - 通过入口S获得普通S列表，从此表中选择最适合S连接，并断开原入口S
  - 上载共享文件信息:客户A上载索引到A，A和B、C、D交换文件信息
  - 查询：客户k向C/D发出查询
  - 回答：返回索引=文件名+大小+位置



eDonKey结构与过程

- 连接方式
  - C←→S间TCP/4661;深层查询UDP/4665
  - C←→C间TCP/4662
  - 动态自适应：下载者每40s向上传者重发下载请求，否则关闭连接
  - S←→S间周期性交换服务器、文件列表

eDonKey分块及性能测量



eDonkey文件分块细节

C←→C间一个小时TCP/4662上的网络参数测量

‘4662’端口所有连接	343.1743万
主机数	25万
所有流的传输总量	295 G Byte
下载连接的传输总量	208 G Byte（70.5%）
下载连接数	7.7111万（2.24%）

- 纯数据下载比70.5%并不高
- 仅2.24% 的数据连接却承担了70.5%的通信量
- 总体不如BT

4.3.4 无结构P2P网络总结

- 覆盖网络的拓扑特性
  - 用户自发形成的、随机松散、任意形状的普通拓扑
  - 但也符合内在某些规律
    - 小世界模型：5-6跳找到（人、信息）
    - 幂率模型：互联网中有连接数L的节点数占网络总节点数的份额正比于L^{-a}, a是网络本身的常数因子
- 路由和定位方法
  - 洪泛法
    - 预先不知道数据在何处？路由存在很大随机性
    - TTL 控制洪泛半径，大于半径的可能存查不到
  - 扩展环：试探性洪泛，不断增加TTL
  - 随机走：随机选择一个邻居行走，直到TTL耗尽
  - 超节点路由

4.3.4 无结构网络P2P总结

- 容错性与自适应
  - 幂率特性对随机节点失效有高容错性
  - 自适应：检测邻居在线否
  - 超级节点列表定期更新
- 可扩展性
  - 改造洪泛提高可扩展性
- 安全性与匿名性
  - 无结构不易追踪
- 增强机制—复制
  - 查询分布：均匀、Zipf
  - 复制份数：均匀、依查询概率比例、方根复制
- 优势和缺陷
  - 高容错性和良好自适应性，较高安全性和匿名性
  - 路由效率低/可扩展差/准确定位差