Introduction to Computer Networks

COSC 4377

Lecture 5

Spring 2012

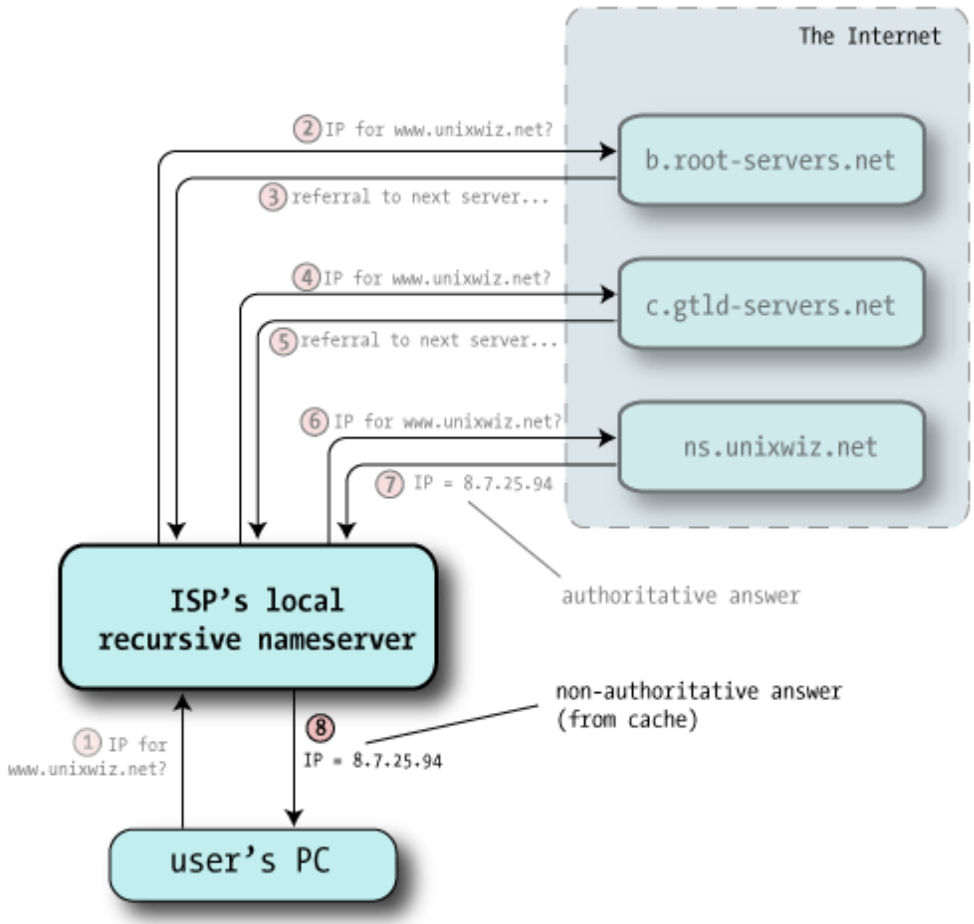
February 1, 2012

Announcements

- HW2 due today
- Start working on HW3, HW4
- HW deadlines
- No TA office hours next week
  - For tomorrow: 1030am-100pm
- Fast HTTP transfer competition

Today’s Topics

- Domain Name System (DNS)
- Peer to Peer (P2P) Networks



http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Example

```
dig +norec bayou.cs.uh.edu @a.root-servers.net

dig +norec bayou.cs.uh.edu @a.edu-servers.net

dig +norec bayou.cs.uh.edu @ns1.uh.edu

dig +norec bayou.cs.uh.edu @dns.cs.uh.edu

    ;; ANSWER SECTION:
    bayou.cs.uh.edu. 3600  IN A  129.7.240.18
```

Structure of a DNS Message

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

- Same format for queries and replies
  - Query has 0 RRs in Answer/Authority/Additional
  - Reply includes question, plus has RRs
- Authority allows for delegation
- Additional for glue, other RRs client might need

Header format

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

- Id: match response to query; QR: 0 query/1 response
- RCODE: error code.
- AA: authoritative answer, TC: truncated,
- RD: recursion desired, RA: recursion available

Other RR Types

- CNAME (canonical name): specifies an alias

```
www.google.com.       446199  IN  CNAME   www.l.google.com.
www.l.google.com. 300 IN  A   72.14.204.147
```

- MX record: specifies servers to handle mail for a domain (the part after the @ in email addr)
- SOA (start of authority)
  - Information about a DNS zone and the server responsible for the zone
- PTR (reverse lookup)

```
18.240.7.129.in-addr.arpa. 3600 IN  PTR  bayou.cs.uh.edu.
```

http://en.wikipedia.org/wiki/List_of_DNS_record_types

Inserting a Record in DNS

- Your new startup httpserver.com
- Get a block of addresses from ISP
  - Say 212.44.9.128/25
- Register helpme.com at GoDaddy.com (for ex.)
  - Provide name and address of your authoritative name server (primary and secondary)
  - Registrar inserts RR pair into the com TLD server:
    - helpme.com NS dns1.httpserver.com
    - dns1.helpme.com A 212.44.9.129
- Configure your authoritative server (dns1.helpme.com)
  - Type A record for www.httpserver.com
  - Type MX record for httpserver.com

Inserting a Record in DNS, cont

- Need to provide reverse PTR bindings
  - E.g., 212.44.9.129 -> dns1.httpserver.com
- Normally, these would go into the 9.44.212.in-addr.arpa zone
- Problem: you can't run the name server for that domain. Why not?
  - Your block is 212.44.9.128/25, not 212.44.9.0/24
  - Whoever has 212.44.9.0/25 wouldn't be happy with you setting their PTR records
- Solution: [RFC2317, Classless Delegation]
  - Install CNAME records in parent zone, e.g:

129.9.44.212.in-addr.arpa CNAME 129.ptr.httpserver.com

DNS Security

- You go to starbucks, how does your browser find www.google.com?
  - Ask local name server, obtained from DHCP
  - You implicitly trust this server
  - Can return any answer for google.com, including a malicious IP that poses as a man in the middle
- How can you know you are getting correct data?
  - Today, you can't
  - HTTPS can help
  - DNSSEC extension will allow you to verify

DNS Security 2 – Cache Poisoning

- Suppose you control evil.com. You receive a query for www.evil.com and reply:

```
;; QUESTION SECTION:
;www.evil.com.                 IN      A

;; ANSWER SECTION:
www.evil.com.          300     IN      A       212.44.9.144

;; AUTHORITY SECTION:
evil.com.              600     IN      NS      dns1.evil.com.
evil.com.              600     IN      NS      google.com.

;; ADDITIONAL SECTION:
google.com.              5     IN      A       212.44.9.155
```

- **Glue record pointing to your IP, not Google's**
- **Gets cached!**

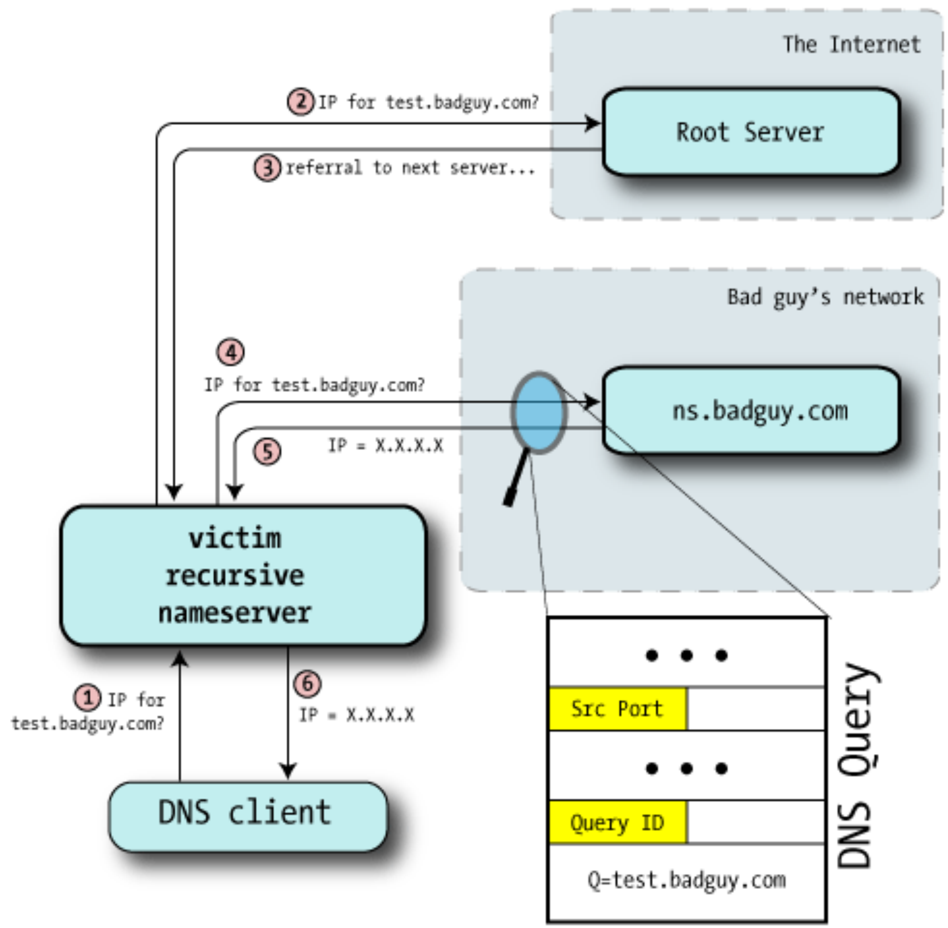
Cache Poisoning # 2

- But how do you get a victim to look up evil.com?
- You might connect to their mail server and send
  - HELO www.evil.com
  - Which their mail server then looks up to see if it corresponds to your IP address (SPAM filtering)
- Mitigation (bailiwick checking)
  - Only accept glue records from the domain you asked for

Cache Poisoning

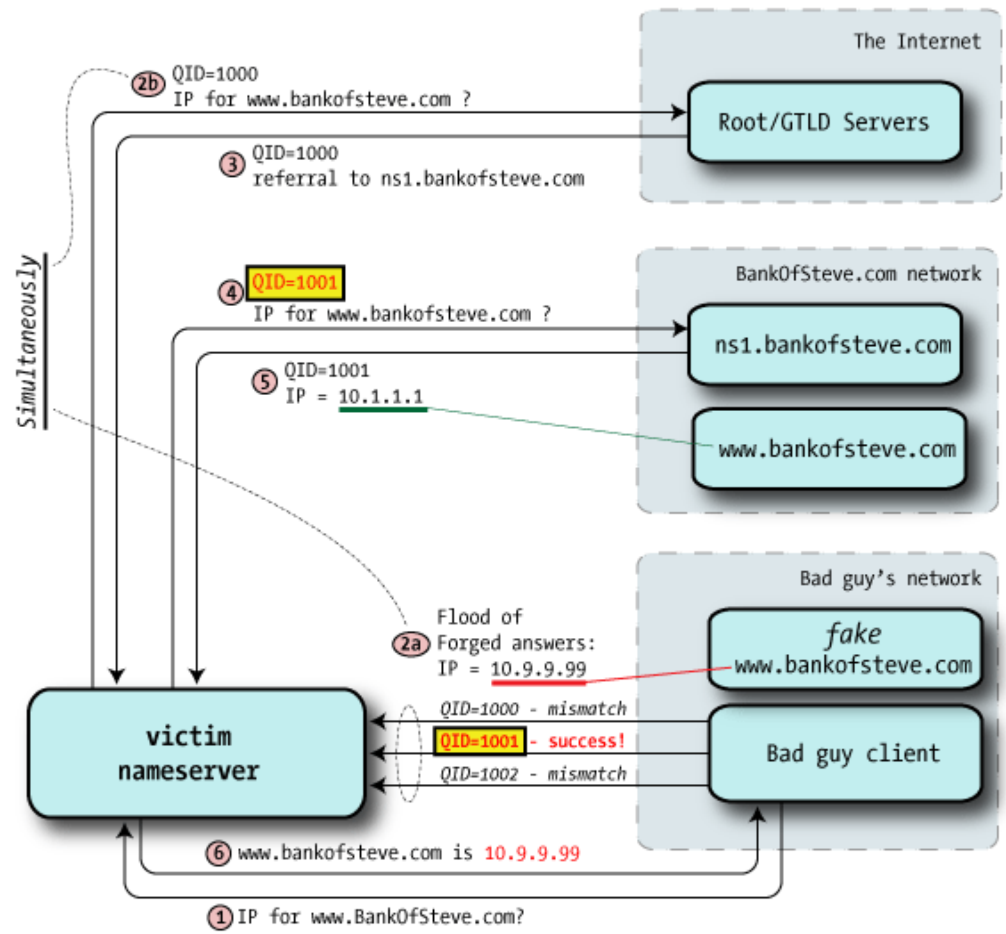
- Bad guy at Starbucks, can sniff or guess the ID field the local server will use
  - Not hard if DNS server generates ID numbers sequentially
  - Can be done if you force the DNS server to look up something in *your* name server
  - Guessing has 1 in 65535 chance (Or does it?)
- Now:
  - Ask the local server to lookup google.com
  - Spoof the response from google.com using the correct ID
  - Bogus response arrives before legit one (maybe)
- Local server caches first response it receives
  - Attacker can set a long TTL

Guessing Query ID



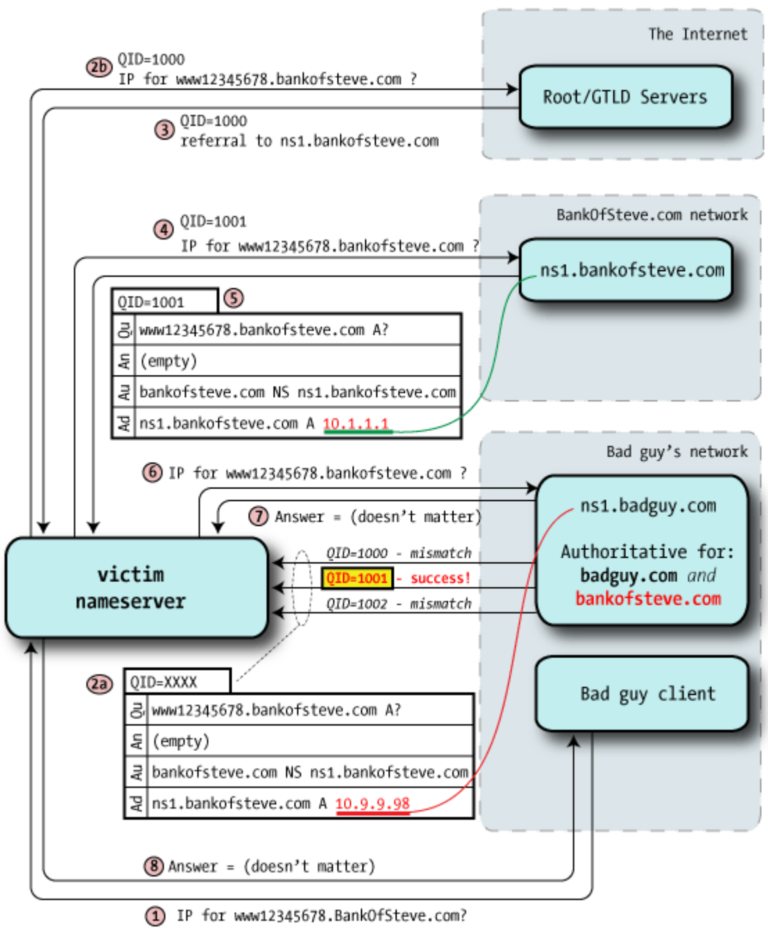
http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Cache Poisoning



http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Hijacking Authority Record



http://www.unixwiz.net/techtips/iguide-kaminsky-dns-vuln.html

Kaminsky Exploit

- If good guy wins the race, you have to wait until the TTL to race again
- But…
  - What if you start a new race, for AAAA.google.com, AAAB.google.com, …?
  - Forge CNAME responses for each
  - Circumvents bailiwick checking

Countermeasures

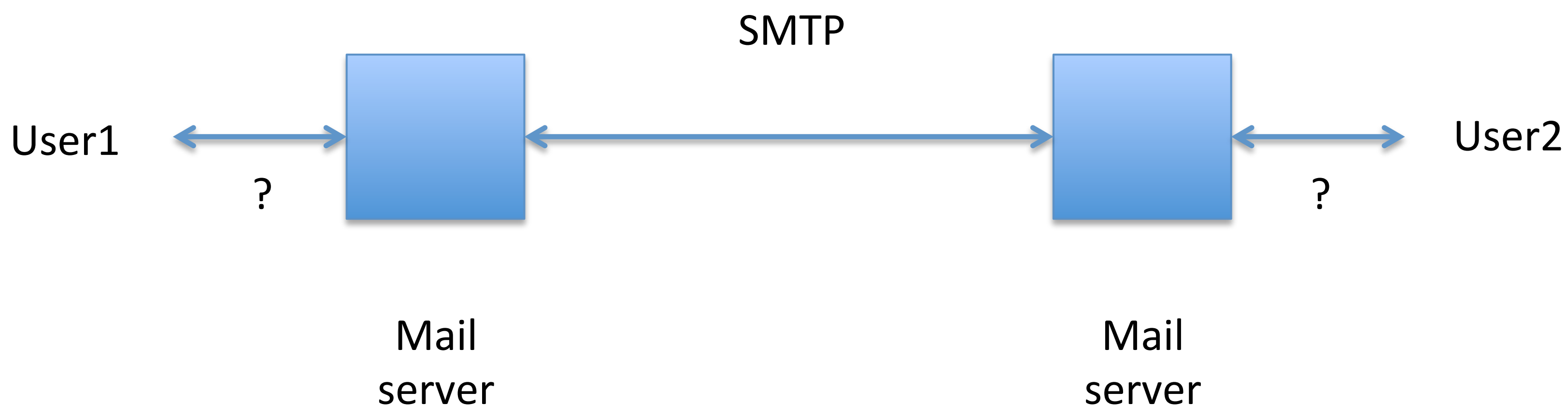
- Randomize id
  - Used to be sequential
- Randomize source port number
  - Used to be the same for all requests from the server
- Offers some protection, but attack still possible

Load Balancing using DNS

- Return multiple IP addresses (“A” records) for a name
- Benefits
  - Spread the load evenly across the IP addresses
- Problems
  - Caching, no standard on which address to use, …
- How to solve these problems?
  - Poll load to compute return list
  - http://en.wikipedia.org/wiki/Round-robin_DNS

```
dig www.google.com

; <<>> DiG 9.7.3-P3 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9457
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.google.com.            IN    A

;; ANSWER SECTION:
www.google.com.       575157      IN    CNAME www.l.google.com.
www.l.google.com.     57    IN    A     74.125.227.49
www.l.google.com.     57    IN    A     74.125.227.50
www.l.google.com.     57    IN    A     74.125.227.51
www.l.google.com.     57    IN    A     74.125.227.52
www.l.google.com.     57    IN    A     74.125.227.48

;; Query time: 26 msec
;; SERVER: 10.0.1.1#53(10.0.1.1)
;; WHEN: Wed Feb  1 09:31:04 2012
;; MSG SIZE  rcvd: 132
```

Emails



Client-Server Bottlenecks

- Download time can scale O(n) with n clients
- Scaling up server bandwidth can be expensive
- Too expensive to provision for flash crowds

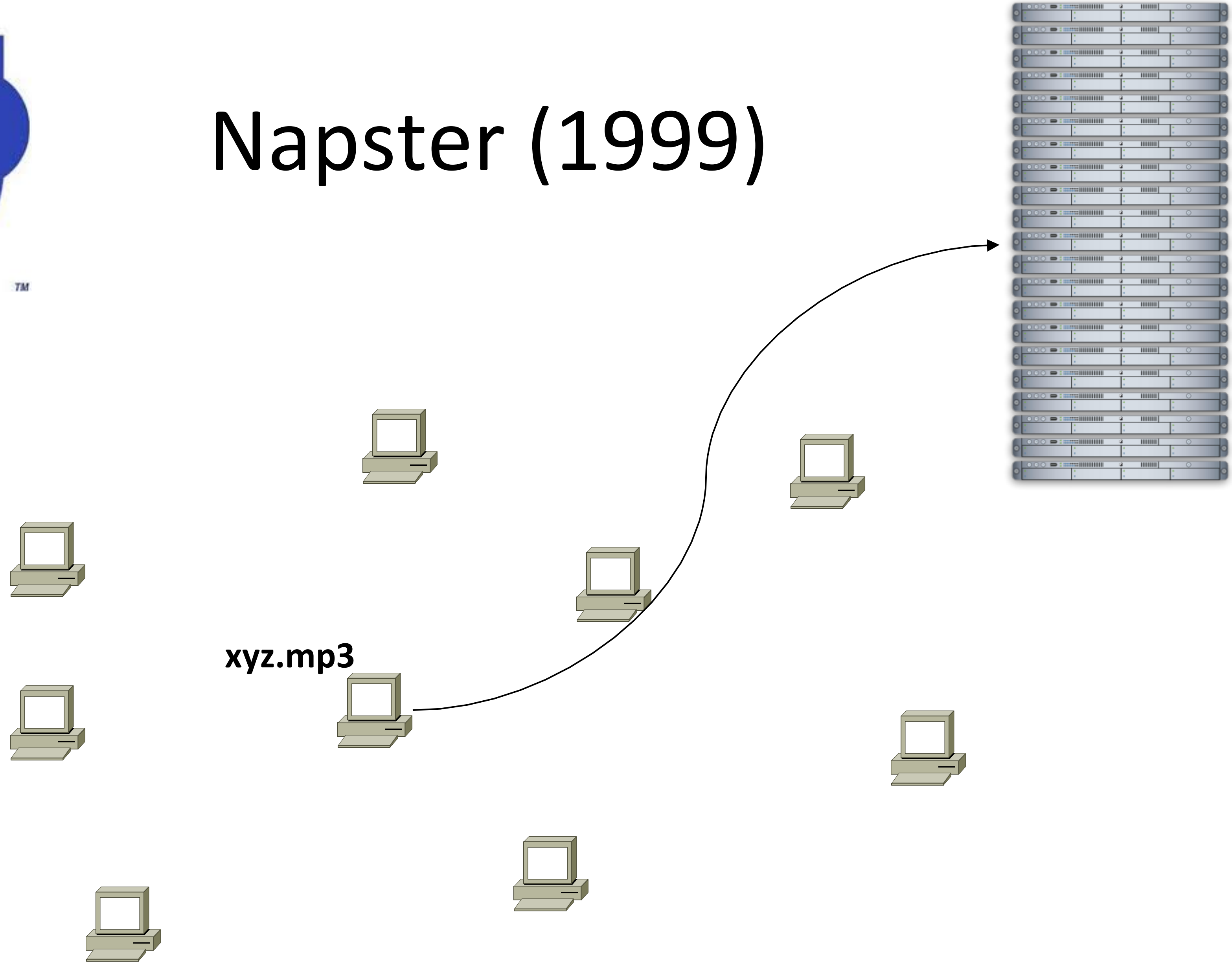
Peer-to-Peer Systems

- How did it start?
  - A killer application: file distribution
  - Free music over the Internet! (*not exactly legal…*)
- Key idea: share storage, content, and bandwidth of individual users
  - Lots of them
- Big challenge: coordinate all of these users
  - In a scalable way (not NxN!)
  - With changing population (aka *churn*)
  - With no central administration
  - With no trust
  - With large heterogeneity (content, storage, bandwidth,…)

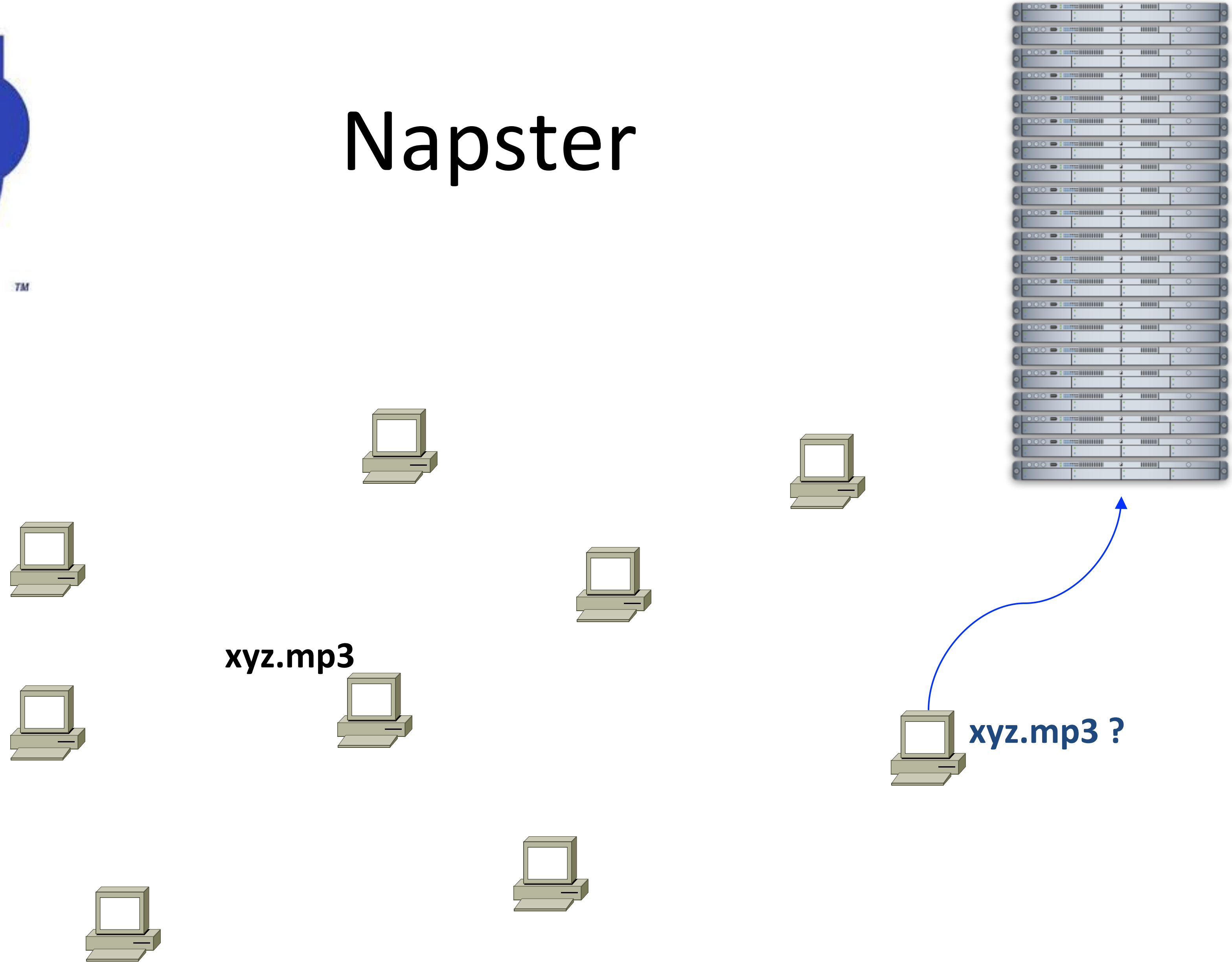
3 Key Requirements

- P2P Systems do three things:
- Help users determine what they want
  - Some form of search
  - P2P version of Google
- Locate that content
  - Which node(s) hold the content?
  - P2P version of DNS (map name to location)
- Download the content
  - Should be efficient
  - P2P form of Akamai

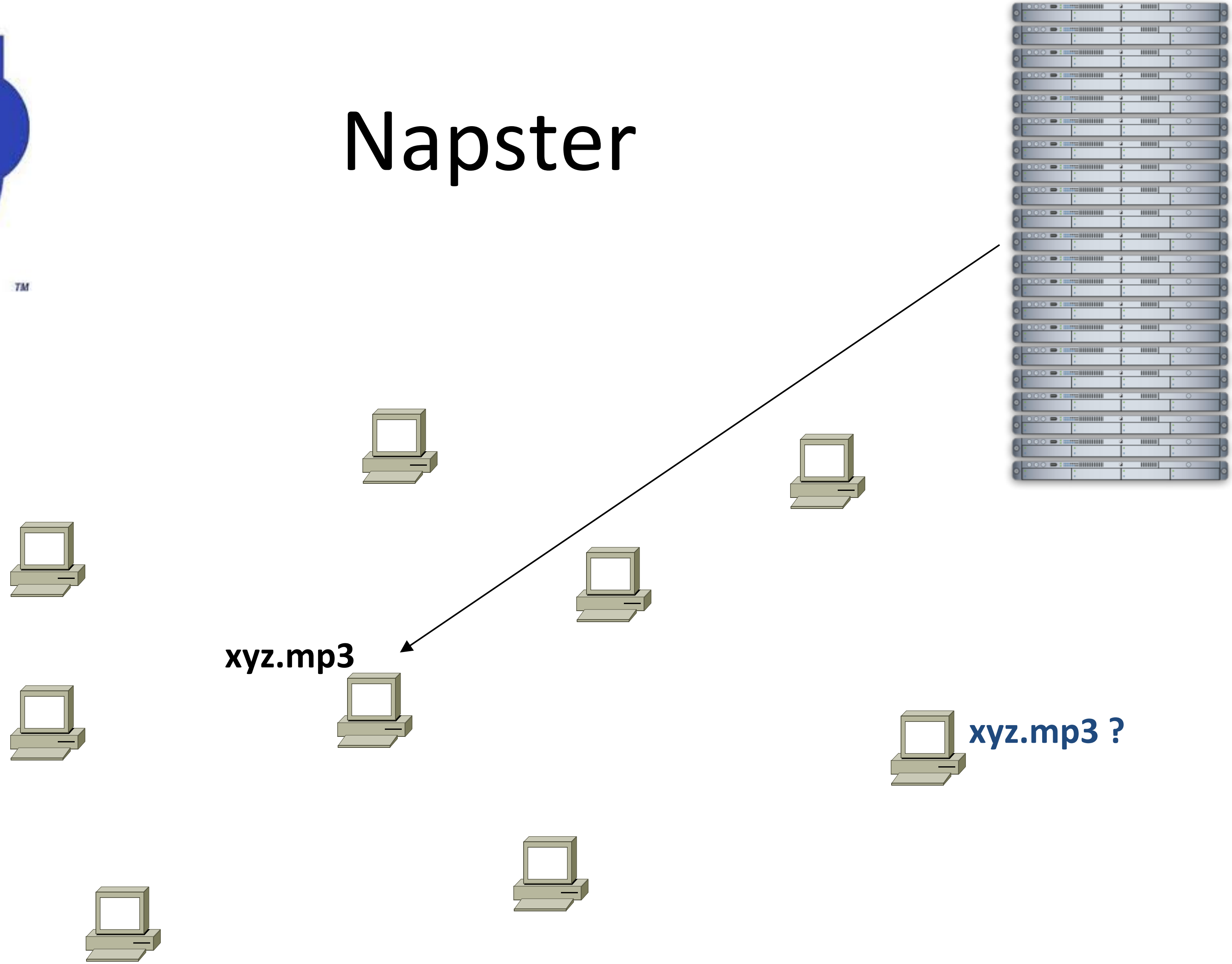
Napster (1999)



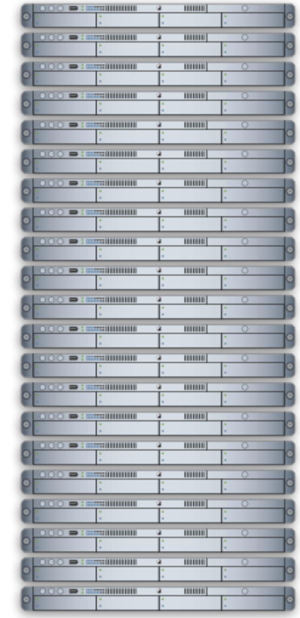
Napster

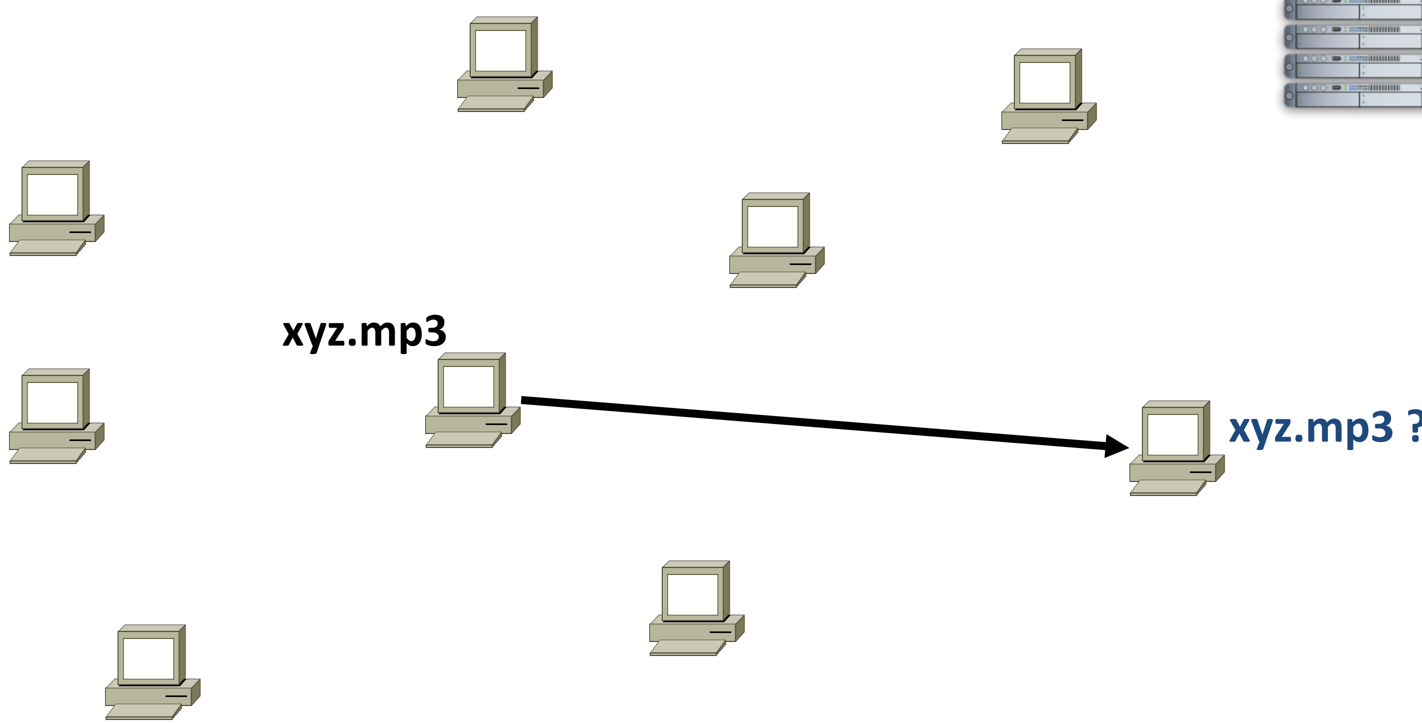


Napster



Napster



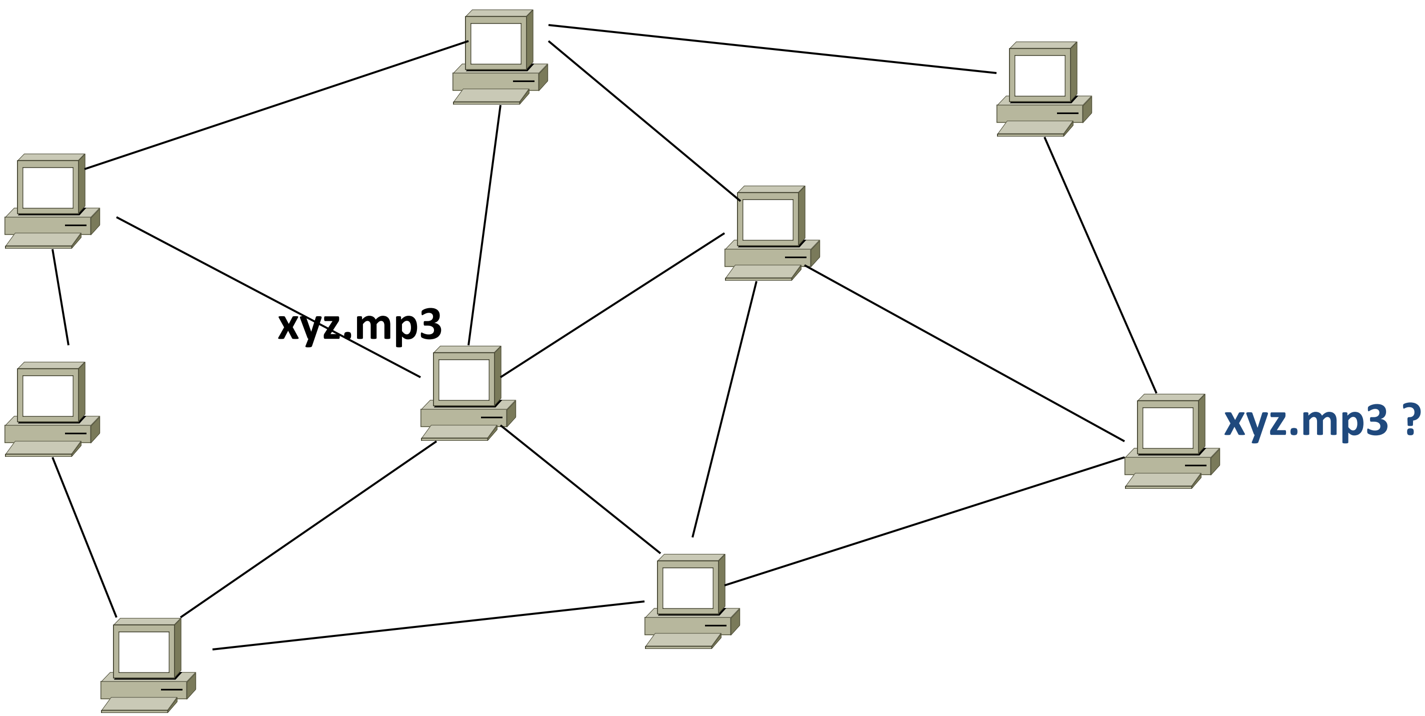


Napster

- Search & Location: central server
- Download: contact a peer, transfer directly
- Advantages:
  - Simple, advanced search possible
- Disadvantages:
  - Single point of failure (technical and … legal!)
  - The latter is what got Napster killed

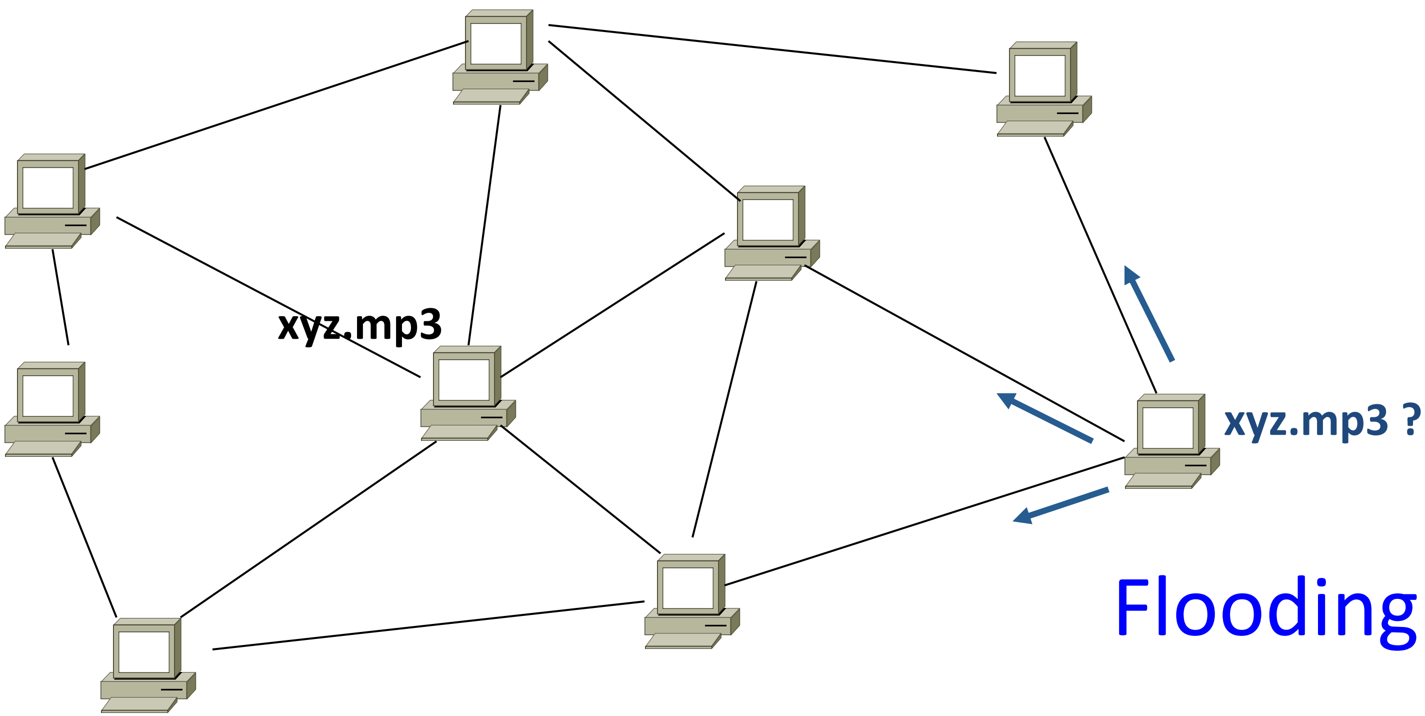
Gnutella: Flooding on Overlays (2000)

- Search & Location: flooding (with TTL)
- Download: direct

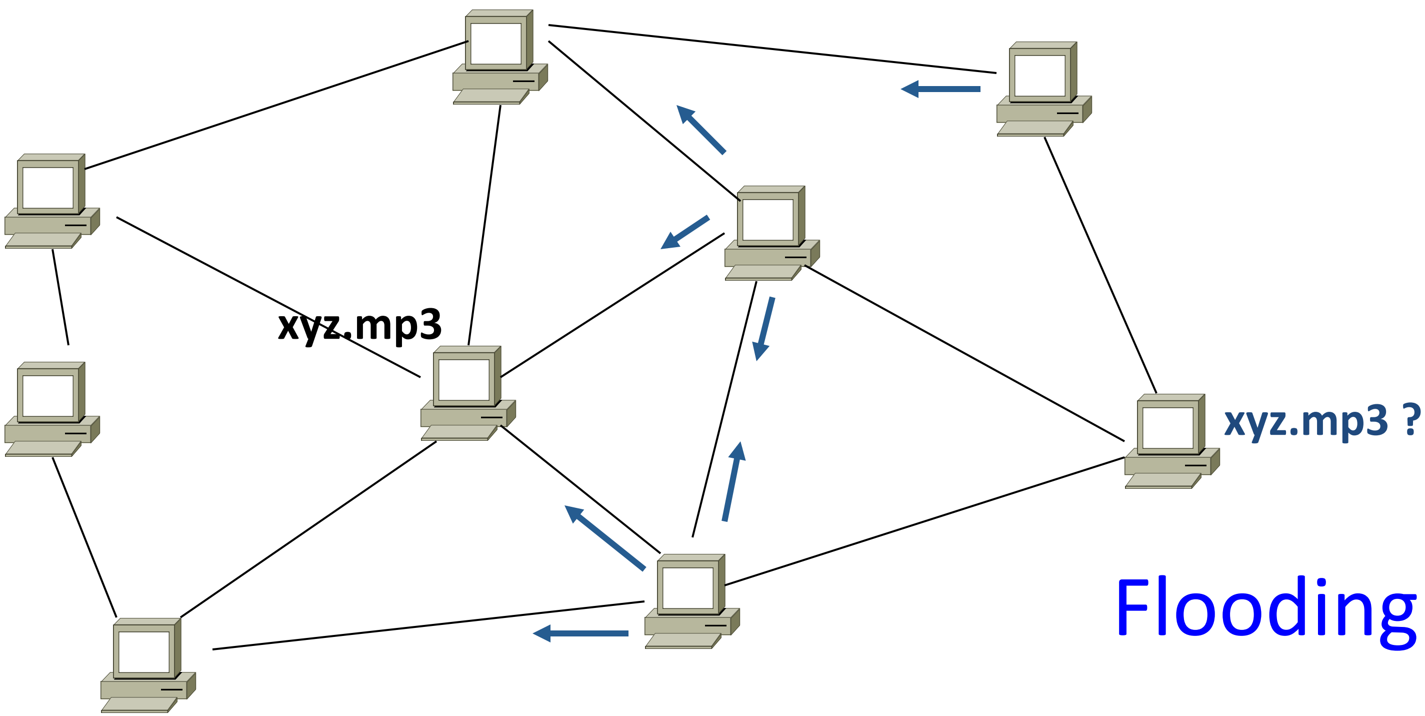


An "unstructured" *overlay network*

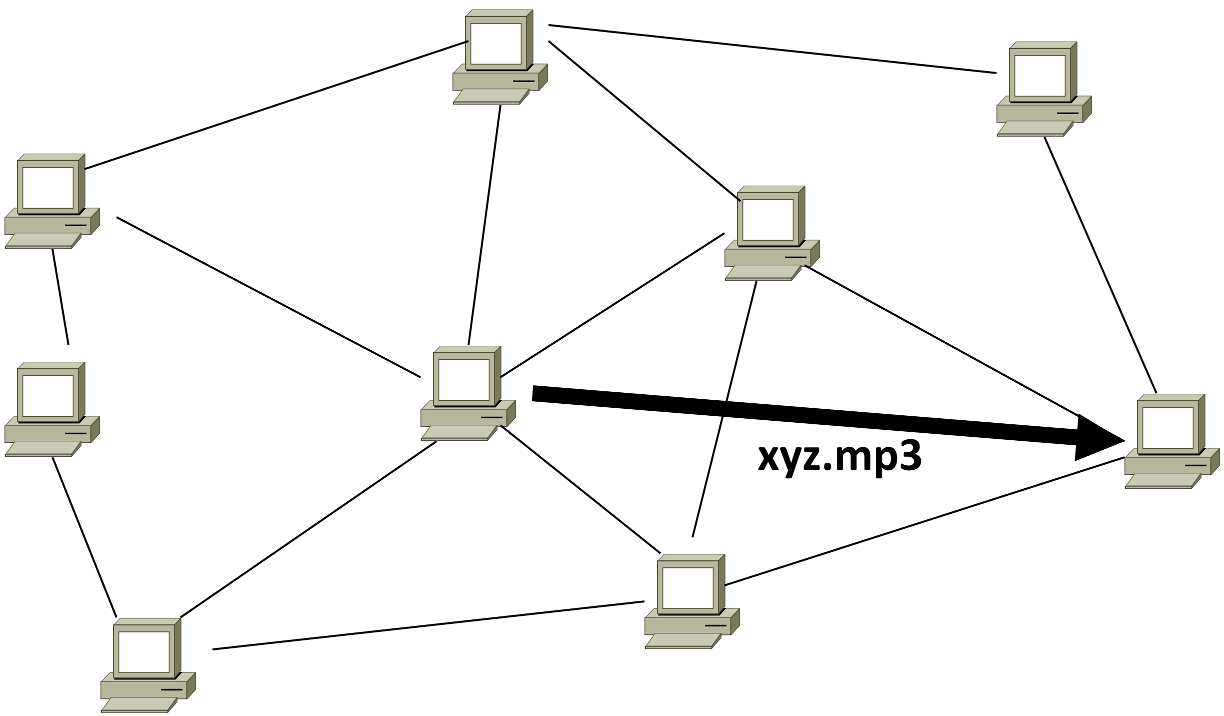
Gnutella: Flooding on Overlays



Gnutella: Flooding on Overlays



Gnutella: Flooding on Overlays



BitTorrent

- One big problem with the previous approaches
  - Asymmetric bandwidth
- BitTorrent
  - Search: independent search engines (e.g. PirateBay, isoHunt)
    - Maps keywords -> .torrent file
  - Location: centralized *tracker* node per file
  - Download: chunked
    - File split into many pieces
    - Can download from many peers

BitTorrent

- How does it work?
  - Split files into large pieces (256KB ~ 1MB)
  - Split pieces into subpieces
  - Get peers from tracker, exchange info on pieces
- Three-phases in download
  - Start: get a piece as soon as possible (random)
  - Middle: spread pieces fast (rarest piece)
  - End: don't get stuck (parallel downloads of last pieces)

BitTorrent Tracker Files

- Torrent file (.torrent) describes files to download
  - Names tracker, server tracking who is participating
  - File length, piece length, SHA1 hash of pieces
  - Additional metadata
- Client contacts tracker, starts communicating with peers


```
d8:announce39:http://torrent.ubuntu.com:6969/announce13:announce-
listll39:http://torrent.ubuntu.com:6969/announceel44:http://
ipv6.torrent.ubuntu.com:6969/announceee7:comment29:Ubuntu CD
releases.ubuntu.com13:creation
datei1272557944e4:infod6:lengthi733837312e4:name29:ubuntu-10.04-
netbook-i386.iso12:piece lengthi524288e6:pieces28000:…
```


Example tracker from ubuntu.com

BitTorrent

- Self-scaling: incentivize sharing
  - If people upload as much as they download, system scales with number of users (no free-loading)
- Uses *tit-for-tat*: only upload to who gives you data
  - *Choke* most of your peers (don't upload to them)
  - Order peers by download rate, choke all but P best
  - Occasionally unchoke a random peer (might become a nice uploader)

Skype

- Real-time communication
- Two major challenges
  - Finding what host a user is on
  - Being able to communicate with those hosts

Skype

- Uses Superpeers for registering presence, searching for where you are
  - Need bootstrap super-peers
- Those Superpeers organize index of users
- Making a call
  - Many nodes don't allow incoming connections
  - Uses regular nodes, outside of NATs, as decentralized relays

Skype