# Content Distribution Networks and Peer-to-Peer

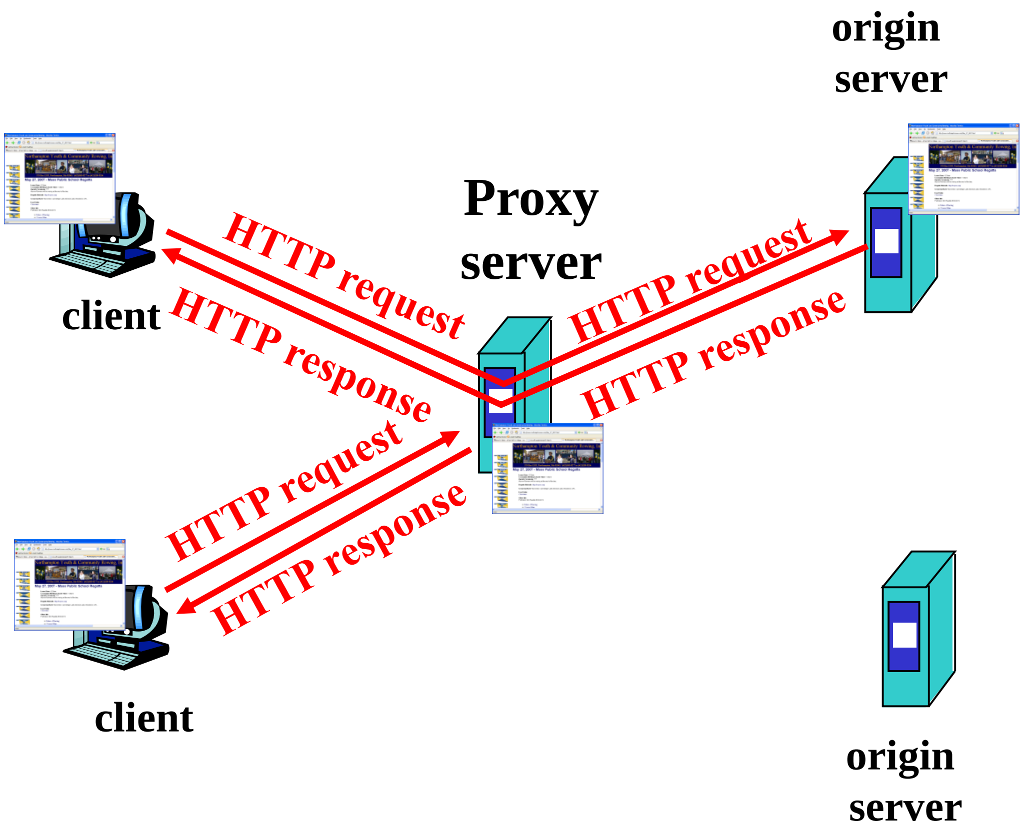
## Mestrado Integrado em Engenharia de Computadores e Telemática

## 2019/2020

# Still from FR: Web caches (proxy server)

**Goal:** satisfy client request without involving origin server

- user sets browser: Web accesses via proxy server
- browser sends all HTTP requests to proxy
  - object in cache: cache returns object
  - else proxy requests object from origin server, then returns object to client



# More about Web caching

- **Proxy server acts as both client and server**
- **typically proxy server is installed by ISP (university, company, residential ISP)**

**Why Web caching?**

- **reduce response time for client request**
- **reduce traffic on an institution's access link.**

# Caching example

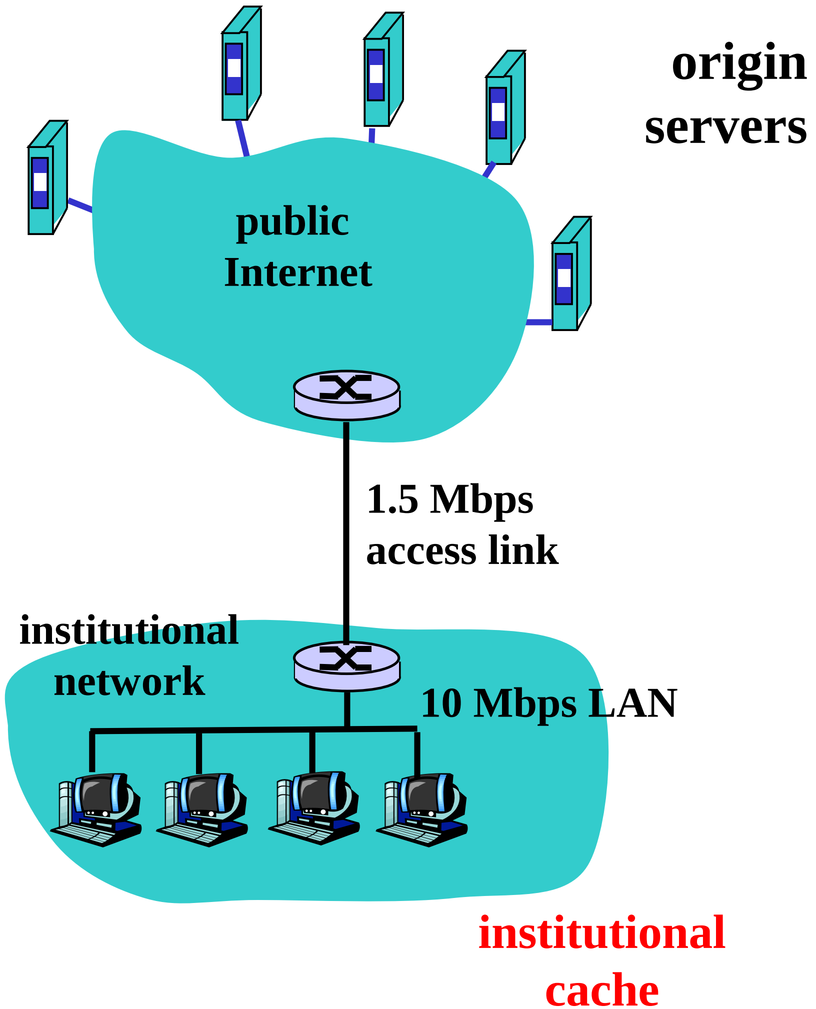
## Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec
- delay from institutional router to any origin server and back to router = 2 sec

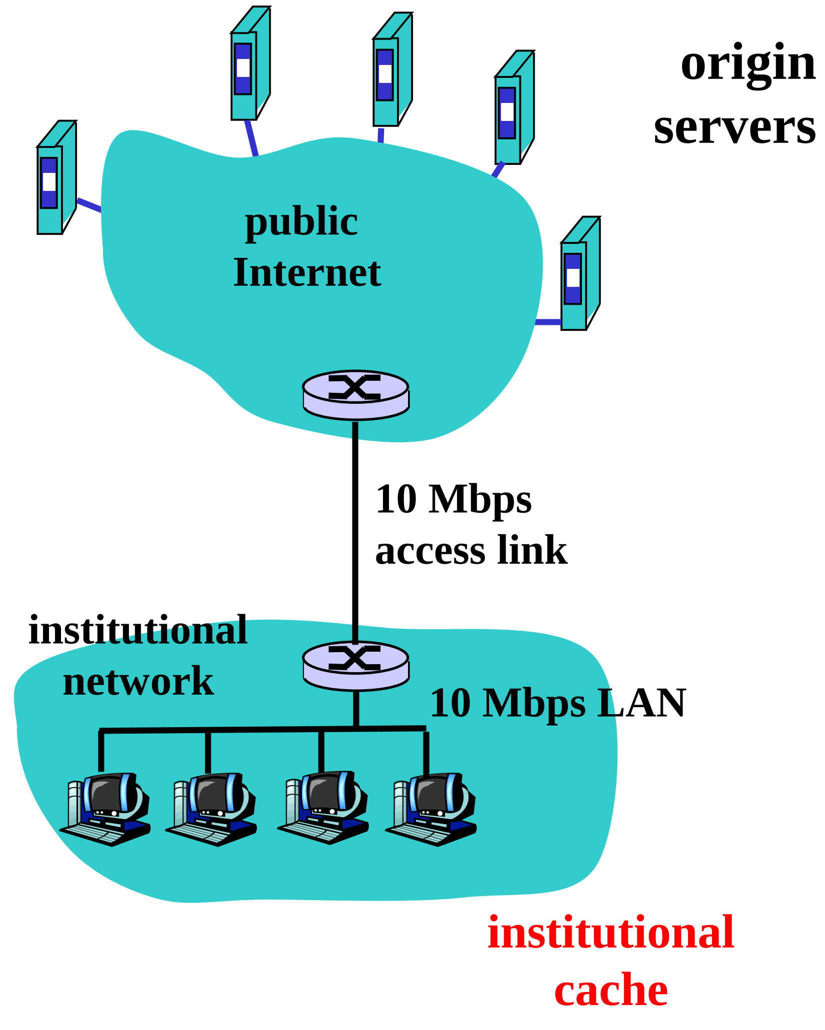
## Consequences

- utilization on LAN = 15%
- utilization on access link = 100%
- total delay = Internet delay + access delay + LAN delay

= 2 sec + minutes + milliseconds



# Caching example (cont)




## possible solution

- increase bandwidth of access link to, say, 10 Mbps

## consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay

  = 2 sec + msecs + msecs
- often a costly upgrade

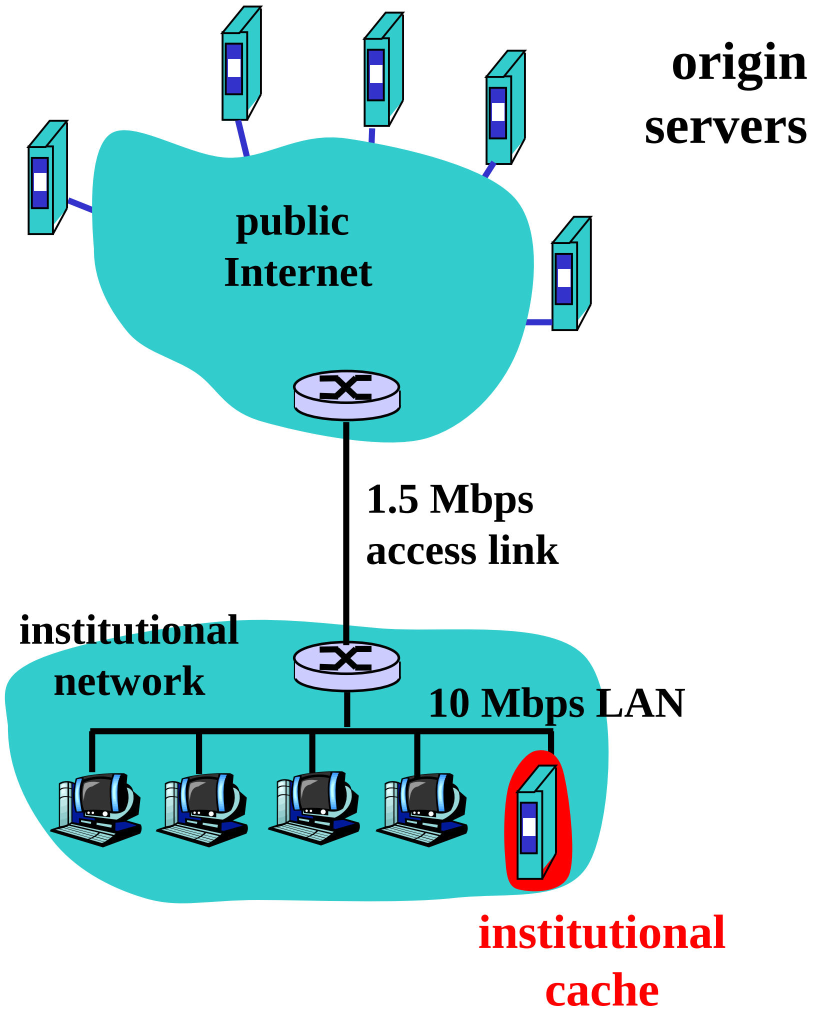
# Caching example (cont)

**possible solution: install cache**

- suppose hit rate is 0.4

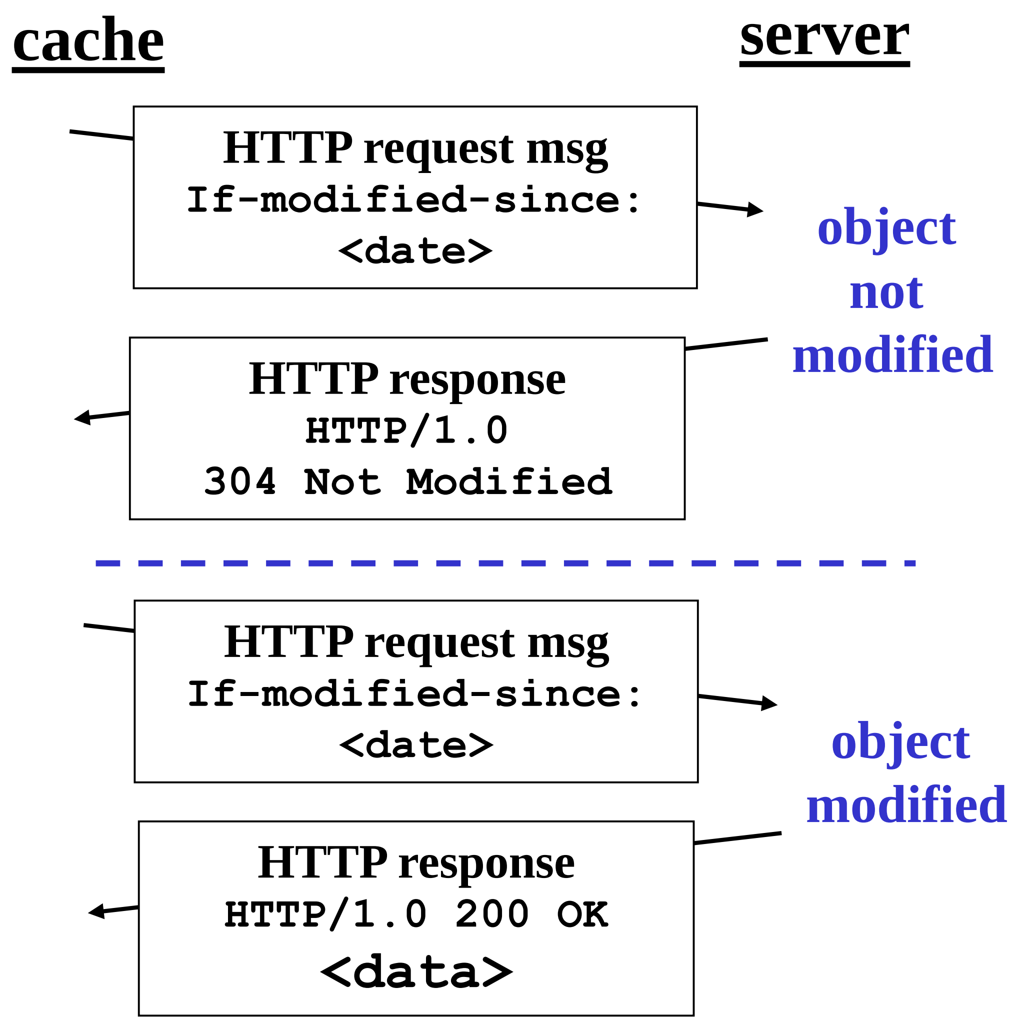
**consequence**

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)
- total avg delay = Internet delay + access delay + LAN delay = .6*(2.01) secs + .4*milliseconds < 1.4 secs



# Conditional GET

- **Goal:** don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
  `If-modified-since: <date>`
- server: response contains no object if cached copy is up-to-date:
  `HTTP/1.0 304 Not Modified`



# CDNs vs Web Caches

- Caches are used by ISPs to reduce bandwidth consumption
- CDNs are used by content providers to improve quality of service to end users

- Caches are reactive
- CDNs are proactive

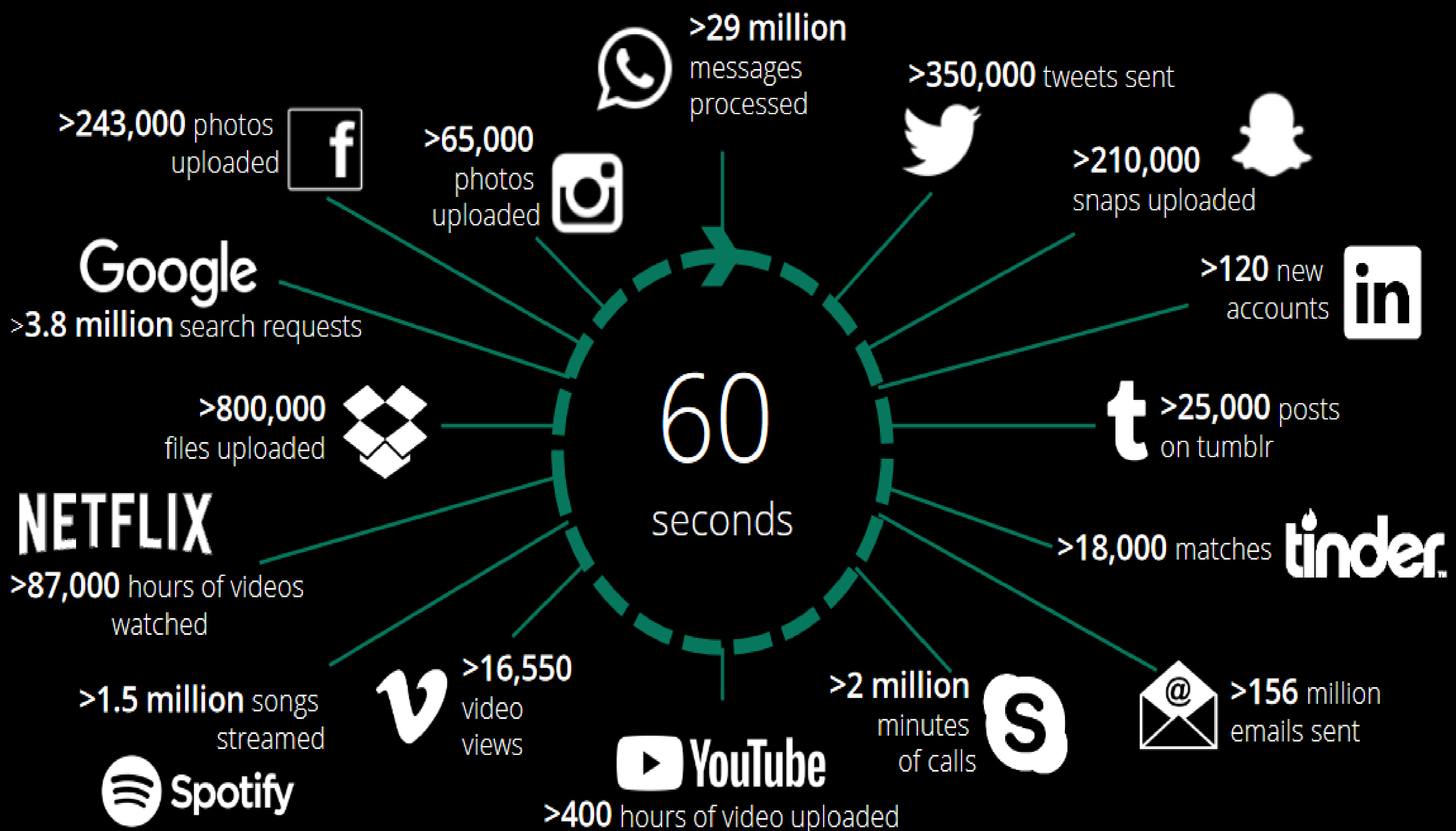
- Caching proxies cater to their users (web clients) and not to content providers (web servers)
- CDNs cater to the content providers (web servers) and clients

- CDNs give control over the content to the content providers
- Caching proxies do not

# So much happened in our digitalized world in 2017 – and we have the numbers behind it

Things that happened online in 2017 within 60 seconds

**Lots of multimedia content!**



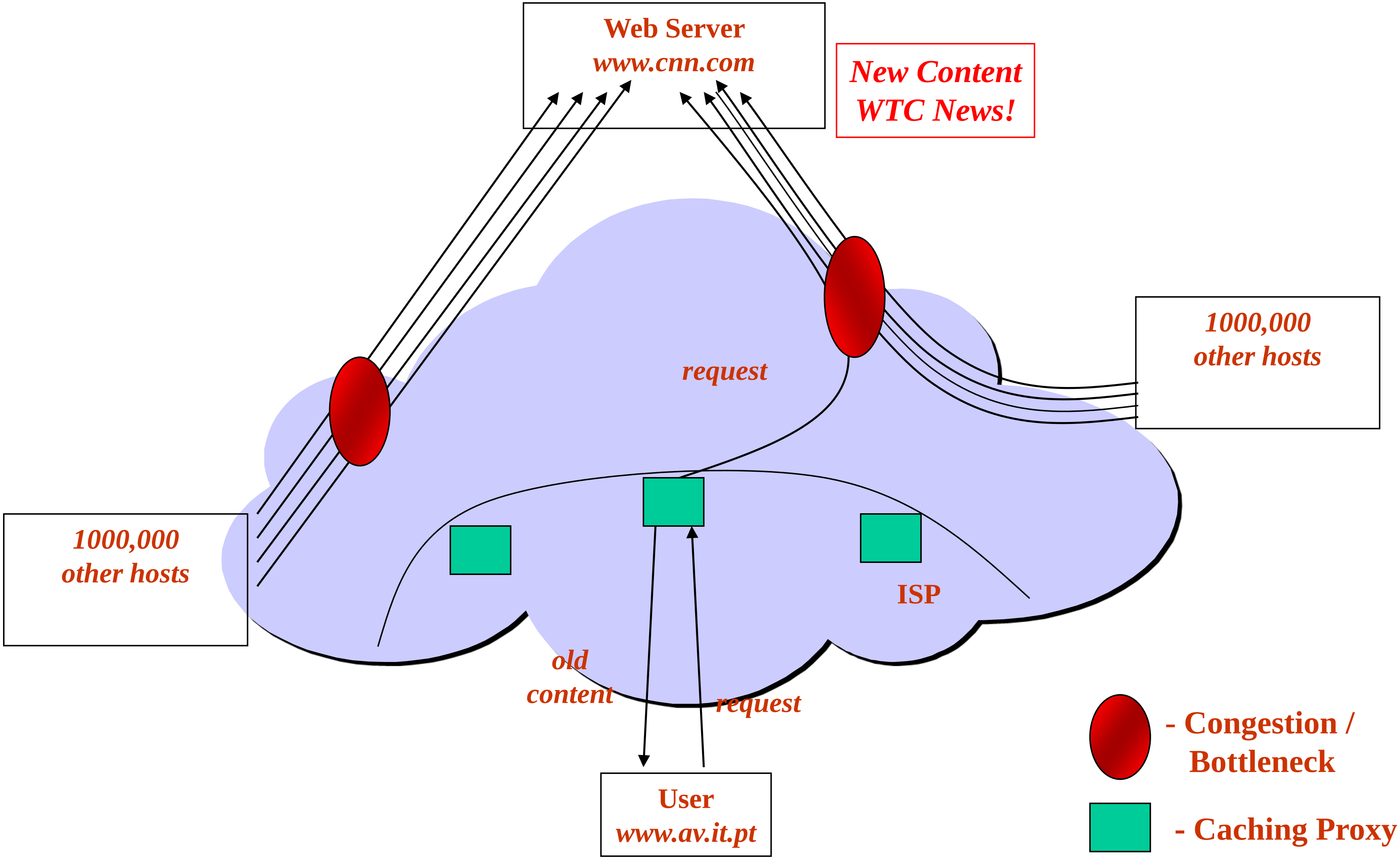
# Motivation (1)

- **IP based networks**
- **Web based applications have become the norm for corporate internal networks and many business-to-business interactions**
- **Large acceptance and explosive growth**
  - **Serious performance problems**
  - **Degraded user experience**

  **For a large set of applications, including VIDEO access**
- **Improving the performance of networked applications**
  - **Use many sites at different points within the network**
    - Stand alone servers
    - Routers

# Motivation (2): Flash Crowds:

## Consider, On September 11, 2001



# Content Distribution Network

## What is a CDN?

A network of servers delivering content on behalf of an origin site

## Large-file service with

- No custom client
- No custom server
- No prepositioning

# Content distribution networks

- Client attempts to access the main server site for an application
- It is redirected to one of the other sites
- Each site caches information
  - Avoid going to the main server to get the information/application
- Access a closelly located site
  - Avoid congestion on the path to the main server
- Set of sites used to improve the performance of web-based applications collectivelly
  - Content distribution network

# CDNs basics

- **A number of CDN companies well established now**
    - E.g. Akamai, Digital Island, Speedera, CDN77, Cloudfare, Stackpat
- **Many companies are exploring CDNs**
    - **Avoid congested portions of the Internet**
        - E.g., CNN, CNBC, …
- **Consist of**
    - **Edge servers deployed at several ISP (Internet Service Provider) access locations and network exchange points**
- **Improve the response time of an Internet site**
    - **Offloading the delivery of bandwidth-intensive objects, such as images and video clips**
- **Intelligent Internet infrastructure that improves the performance and scalability of distributed applications by moving the bulk of their *computation* to servers located at the edge of the network**

# CDN challenges (I)

- **Keep consistency among the enterprise data hosted by the offloaded applications**
- **Share session state among edge and origin application servers**
- **Distribution, configuration, and management**
- **Application security.**

# CDN challenges (II)

- Load-sharing content
  - Handle requests fairly amongst servers/sites
  - Easily add servers/sites to content service
  - Adjust connections based on server/site load

- Content Availability with multiple servers
  - Synchronize content amongst servers/sites
  - Avoid faulty servers/sites

- Handle applications which use ‘state’
  - Need to learn client ID to satisfy state requirement
  - Need to maintain state for period of time - variable

# What is a CDN?

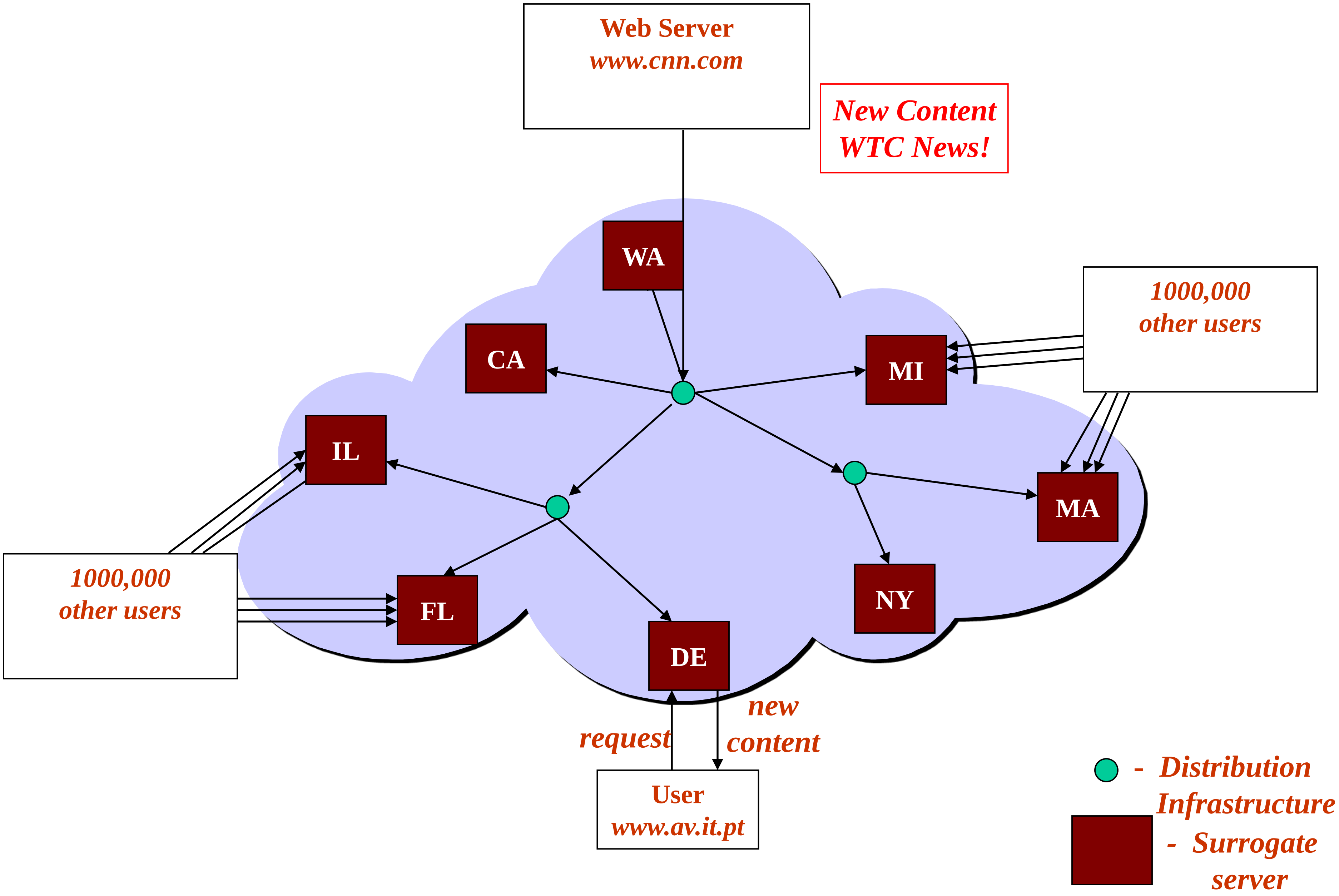
- **Content Delivery Network**
  - **Also sometimes called Content Distribution Network**
  - **At least half of the world's bits are delivered by a CDN**
    - Probably closer to 80/90%

- **Primary Goals**
  - **Create replicas of content throughout the Internet**
  - **Ensure that replicas are always available**
  - **Direct clients to replicas that will give good performance**

# Key Components of a CDN

- Distributed servers
  - Usually located inside of other ISPs
- High-speed network connecting them
- Clients
  - Can be located anywhere in the world
  - They want fast Web performance
- Binding clients and distributed servers
  - Something that binds clients to “nearby” replica servers

# CDN Architecture



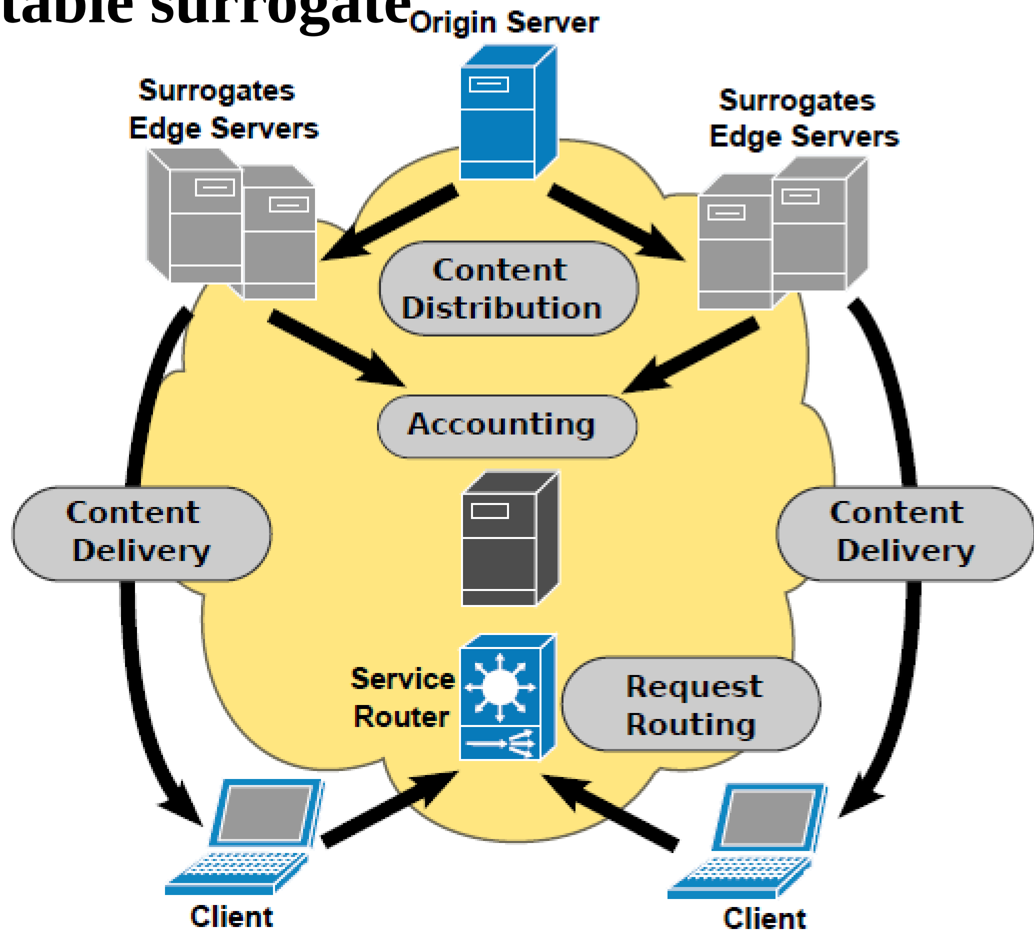
# CDN Components

- *Content Delivery Infrastructure:* Delivering content to clients from surrogates

- *Request Routing Infrastructure:* Steering or directing content request from a client to a suitable surrogate

- *Distribution Infrastructure:* Moving or replicating content from content source (origin server, content provider) to surrogates

- *Accounting Infrastructure:* Logging and reporting of distribution and delivery activities



# Inside a CDN

- **Servers are deployed in clusters for reliability**
  - **Some may be offline**
    - Could be due to failure
    - Also could be "suspended" (e.g., to save power or for upgrade)
- **Could be multiple clusters per location (e.g., in multiple racks)**
- **Server locations**
  - **Well-connected points of presence (PoPs)**
  - **Inside of ISPs**

# Mapping clients to servers

- CDNs need a way to send clients to the "best" server
  - The best server can change over time
  - And this depends on client location, network conditions, server load, …
  - What existing technology can we use for this?

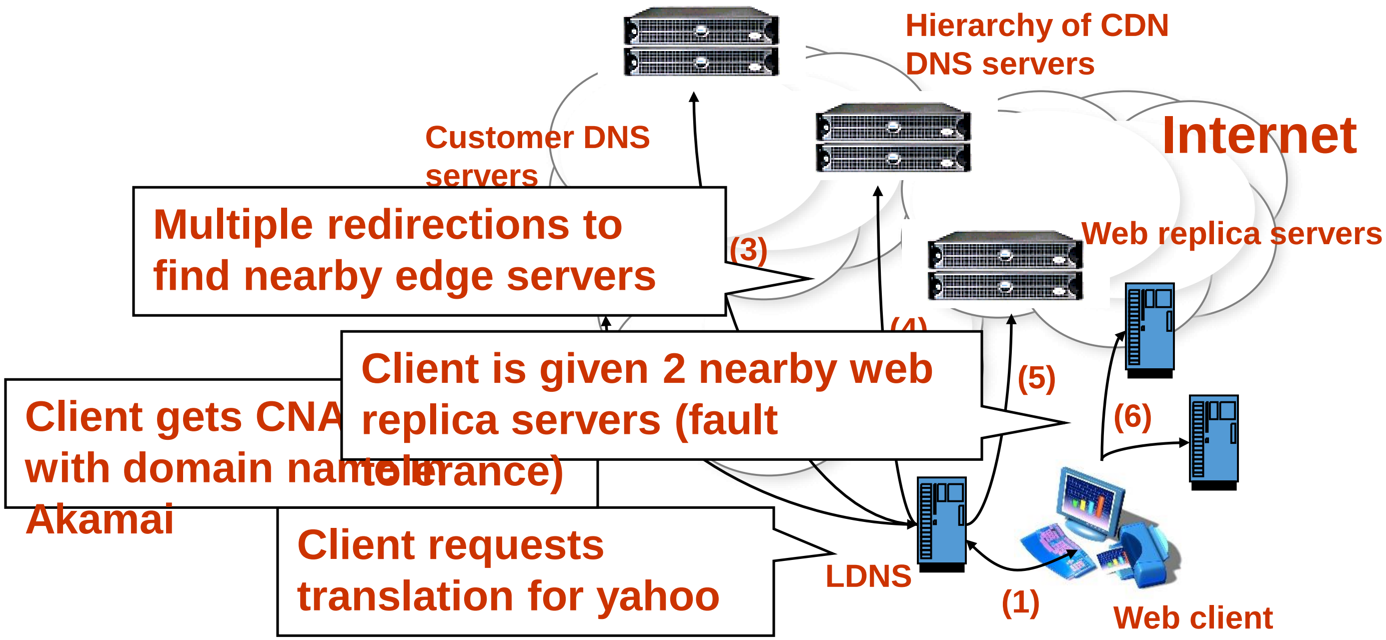
- DNS-based redirection
  - Clients request www.foo.com
  - DNS server directs client to one or more IPs based on request IP
  - Use short TTL to limit the effect of caching

# CDN redirection example

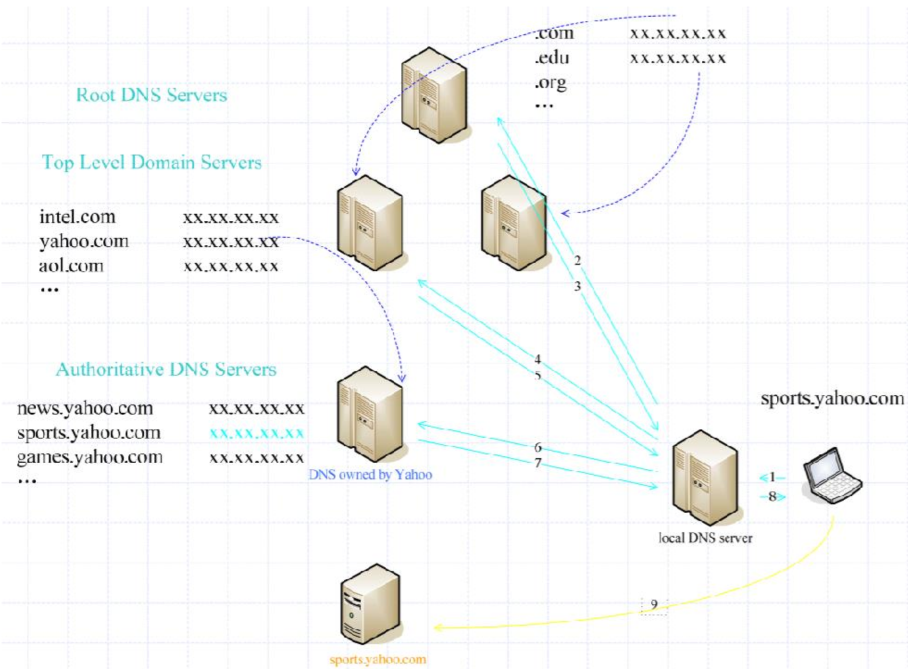
```
choffnes$ dig www.fox.com


;; ANSWER SECTION:
www.fox.com.                    510       IN        CNAME     www.fox-
rma.com.edgesuite.net.
www.fox-rma.com.edgesuite.net. 5139 IN  CNAME     a2047.w7.akamai.net.
a2047.w7.akamai.net.            4         IN        A         23.62.96.128
a2047.w7.akamai.net.            4         IN        A         23.62.96.144
a2047.w7.akamai.net.            4         IN        A         23.62.96.193
a2047.w7.akamai.net.            4         IN        A         23.62.96.162
a2047.w7.akamai.net.            4         IN        A         23.62.96.185
a2047.w7.akamai.net.            4         IN        A         23.62.96.154
a2047.w7.akamai.net.            4         IN        A         23.62.96.169
a2047.w7.akamai.net.            4         IN        A         23.62.96.152
a2047.w7.akamai.net.            4         IN        A         23.62.96.186
```

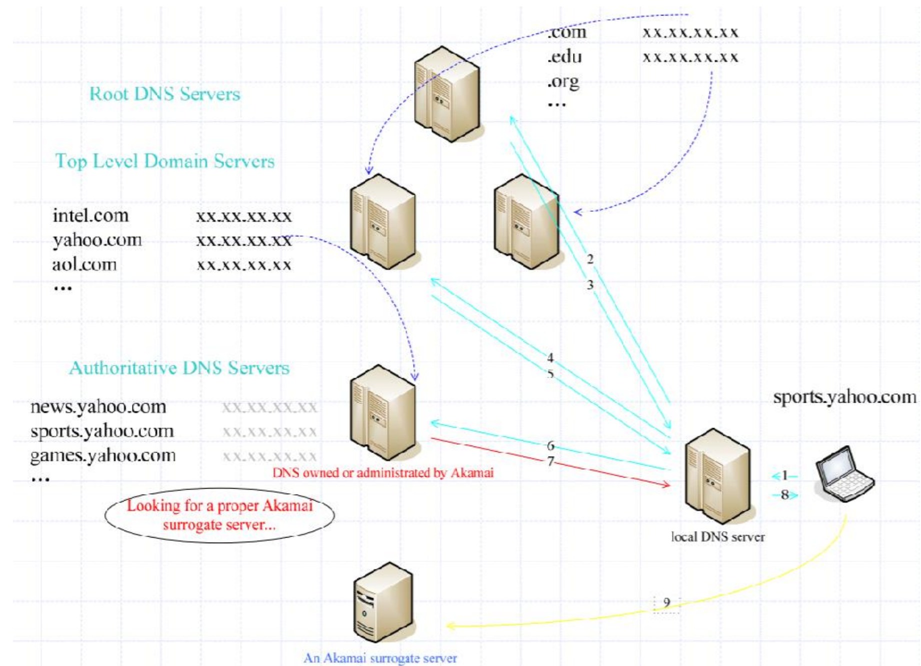
# DNS Redirection

- Web client's request redirected to 'close' by server
  - Client gets web site's DNS CNAME entry with domain name in CDN network
  - Hierarchy of CDN's DNS servers direct client to 2 nearby servers



# DNS Redirection



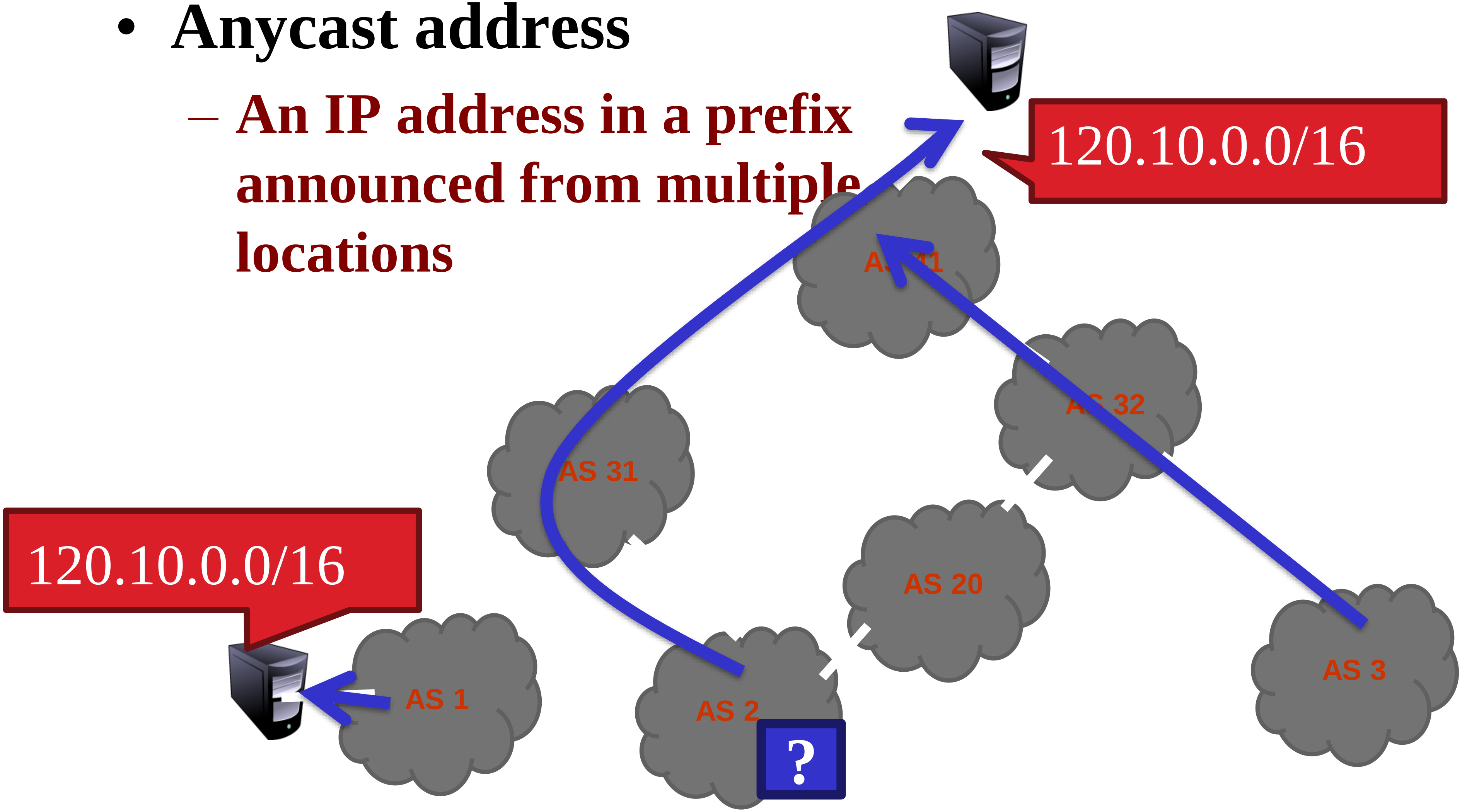
# DNS Redirection Considerations

- **Advantages**
  - **Uses existing, scalable DNS infrastructure**
  - **URLs can stay essentially the same**

- **Limitations**
  - **DNS servers see only the DNS server IP**
    - Assumes that client and DNS server are close. Is this accurate?
  - **Content owner must give up control**
  - **Unicast addresses can limit reliability**

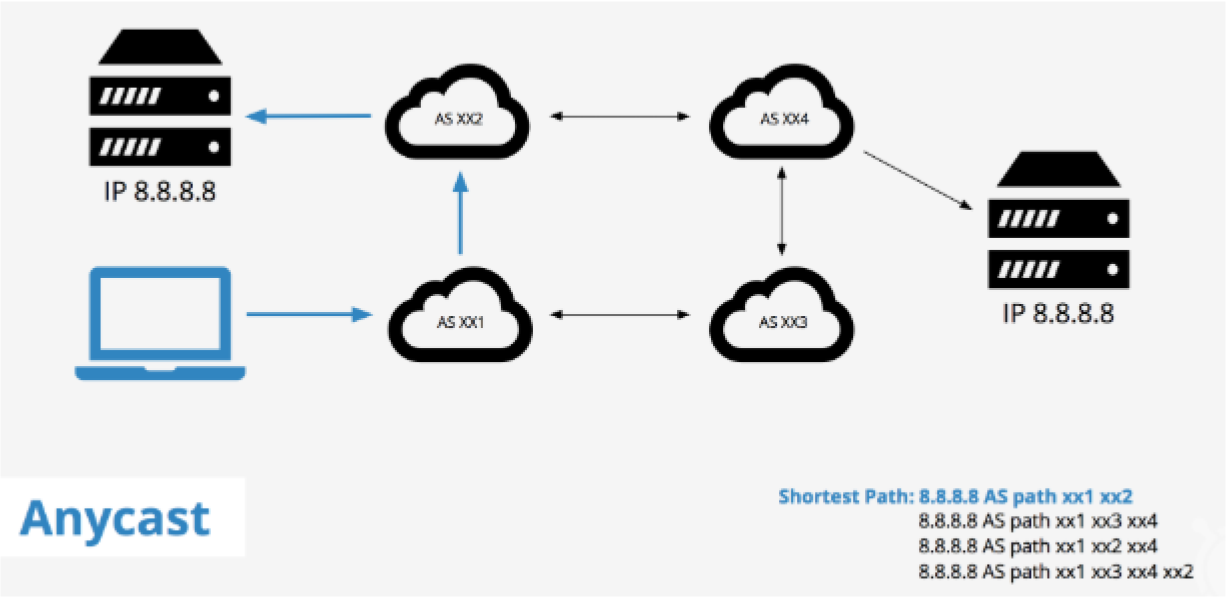
# CDN Using Anycast

- **Anycast address**
  - **An IP address in a prefix announced from multiple locations**



# CDN Using Anycast

- **Anycast IPv4 address /32**
- **Anycast IPv6 address /128**



# Anycasting Considerations

- **Why do anycast?**
  - **Simplifies network management**
    - Replica servers can be in the same network domain
  - **Uses best BGP path**

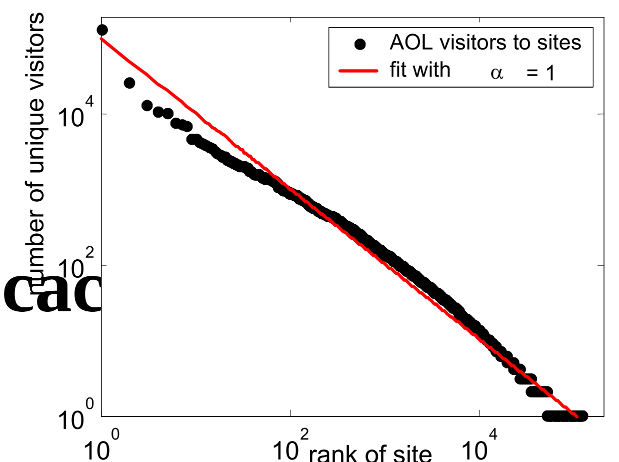
- **Disadvantages**
  - **BGP path may not be optimal**
  - **Stateful services can be complicated**

# CDN using URL Rewriting

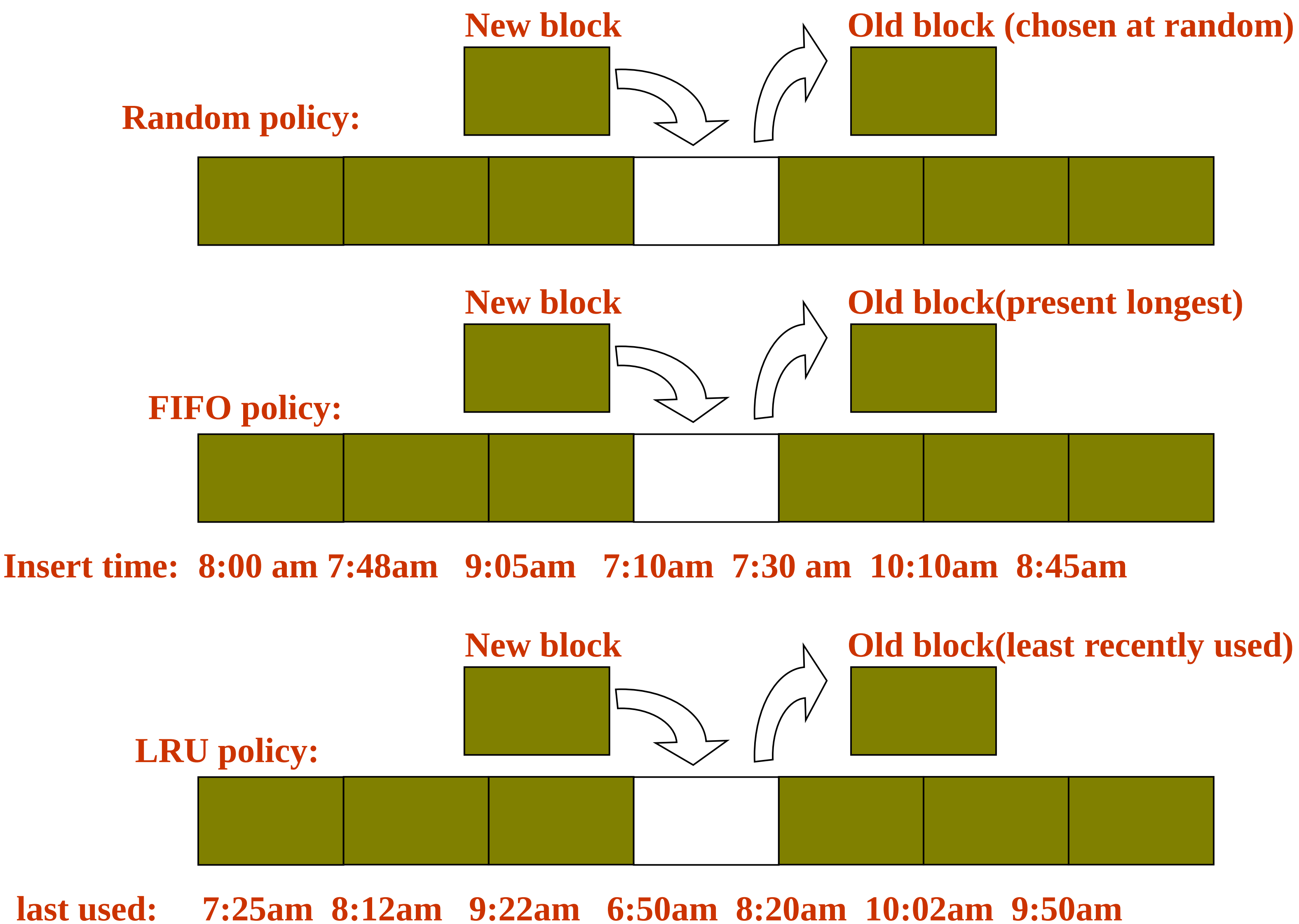
- **URL rewriting**
  - **Origin server redirects clients to different surrogate servers by rewriting the page's URL links**

# Optimizing performance

- **Where to cache content?**
  - **Popularity of Web objects is Zipf-like**
    - a few elements that score *very* high (the left tail in the diagrams)
    - a medium number of elements with middle-of-the-road scores (the middle part of the diagram)
    - a huge number of elements that score very low (the right tail in the diagram)
  - **Small number of sites cover large fraction of requests**
- **Given this observation, how should cache replacement work?**



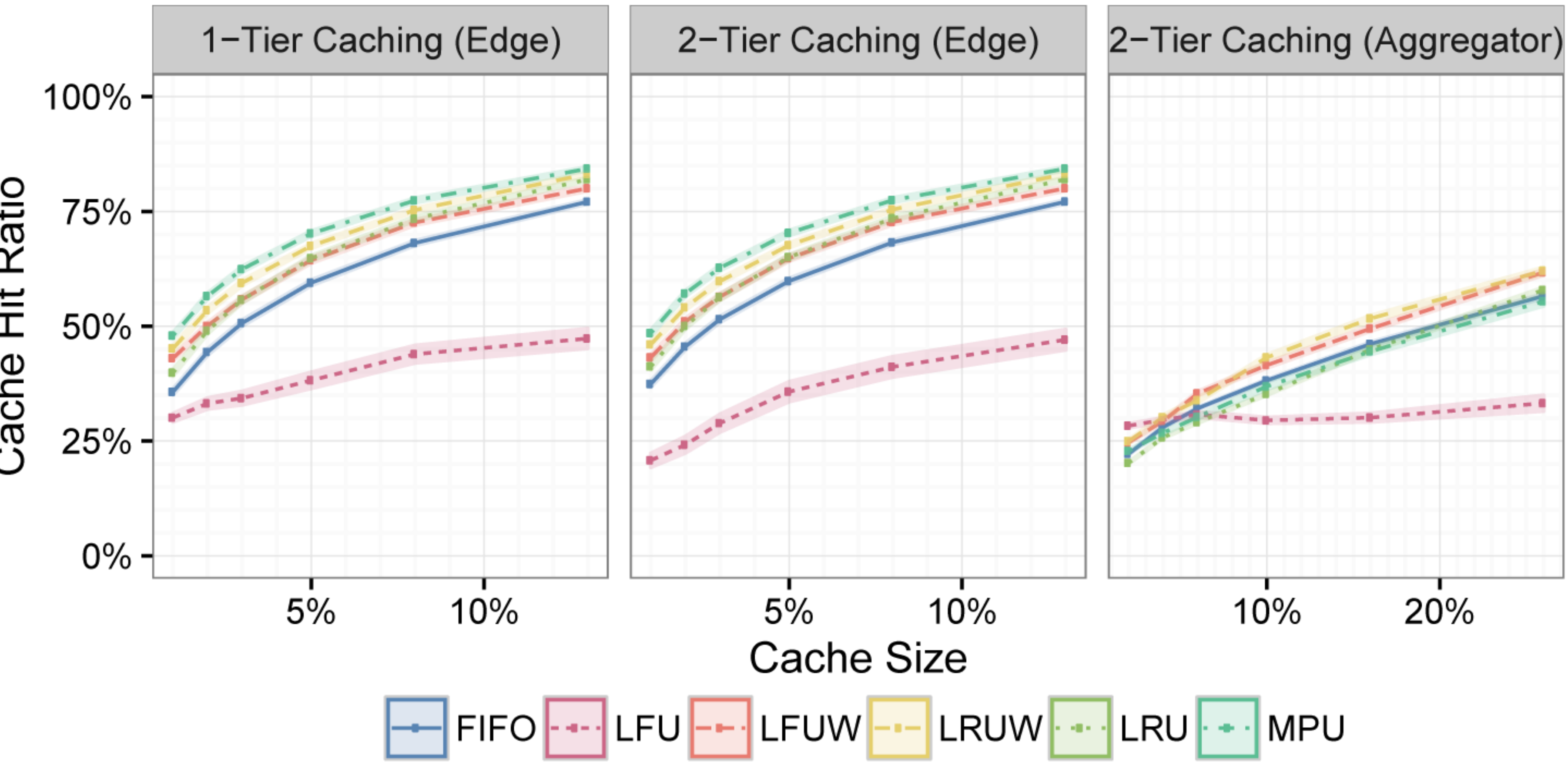
# Cache Replacement Policies (I)



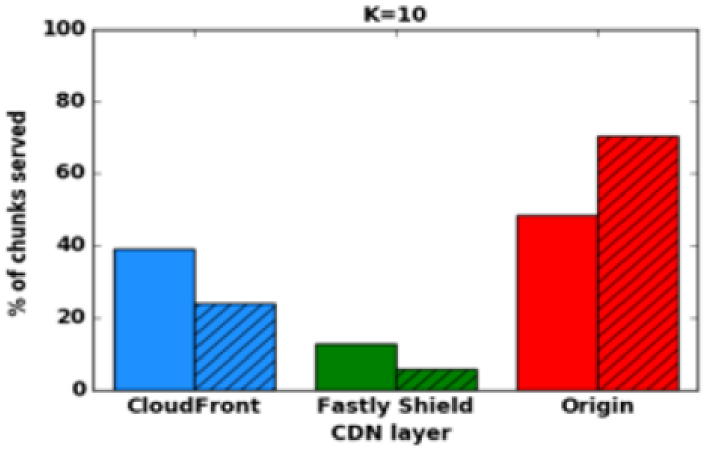
# Cache Replacement Policies (II)

- **LFU: Least Frequently Used**
- **MPU: Most Probably Used**
- **LFU and LRU weighted (give a weight to each page)**

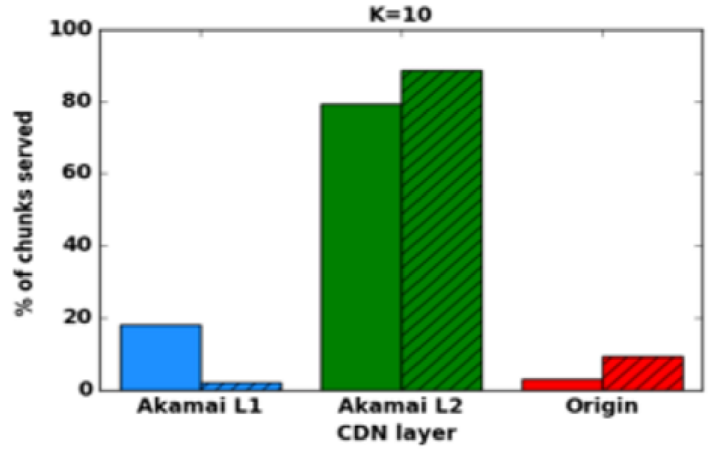
# Cache Hit Ratio vs Cache Size



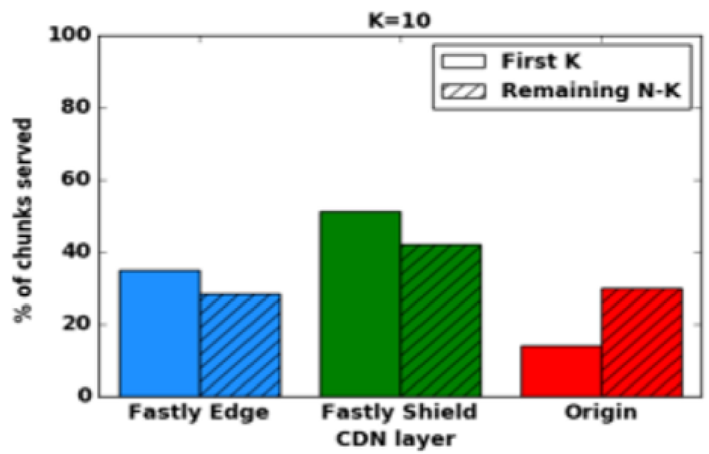
# Where are objects coming from (even from chunk to chunk)?




Twitch: CloudFront/Fastly.




Vimeo: Akamai.




Vimeo: Fastly.

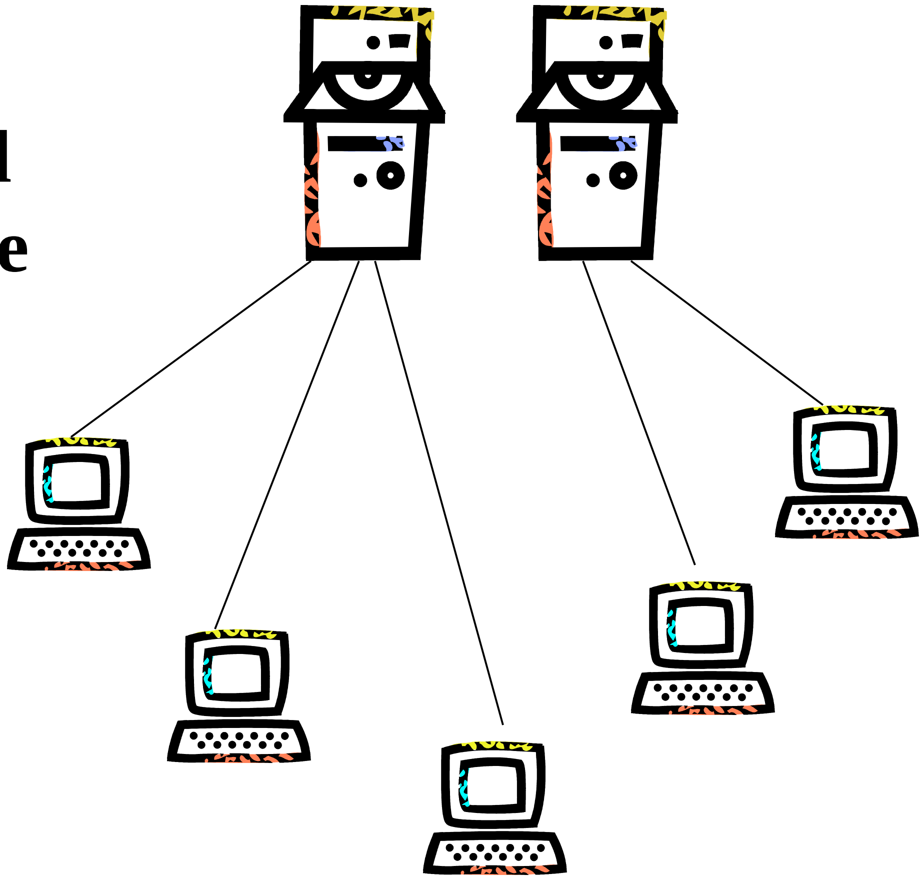
# Peer-to-peer networks

# Peer to peer networks

- **Exploits diverse connectivity between participants in a network**
- **Exploits the cumulative bandwidth of network participants**
- **Typically used for connecting nodes via large ad-hoc connections**
  - **Sharing content files containing audio, video, data**
  - **Even real-time data, such as telephony traffic, is also passed using P2P technology**
- **Pure peer-to-peer network**
  - **There is no notion of clients or servers**
  - **Equal peer nodes that simultaneously work as both "clients" and "servers" to the other nodes on the network**

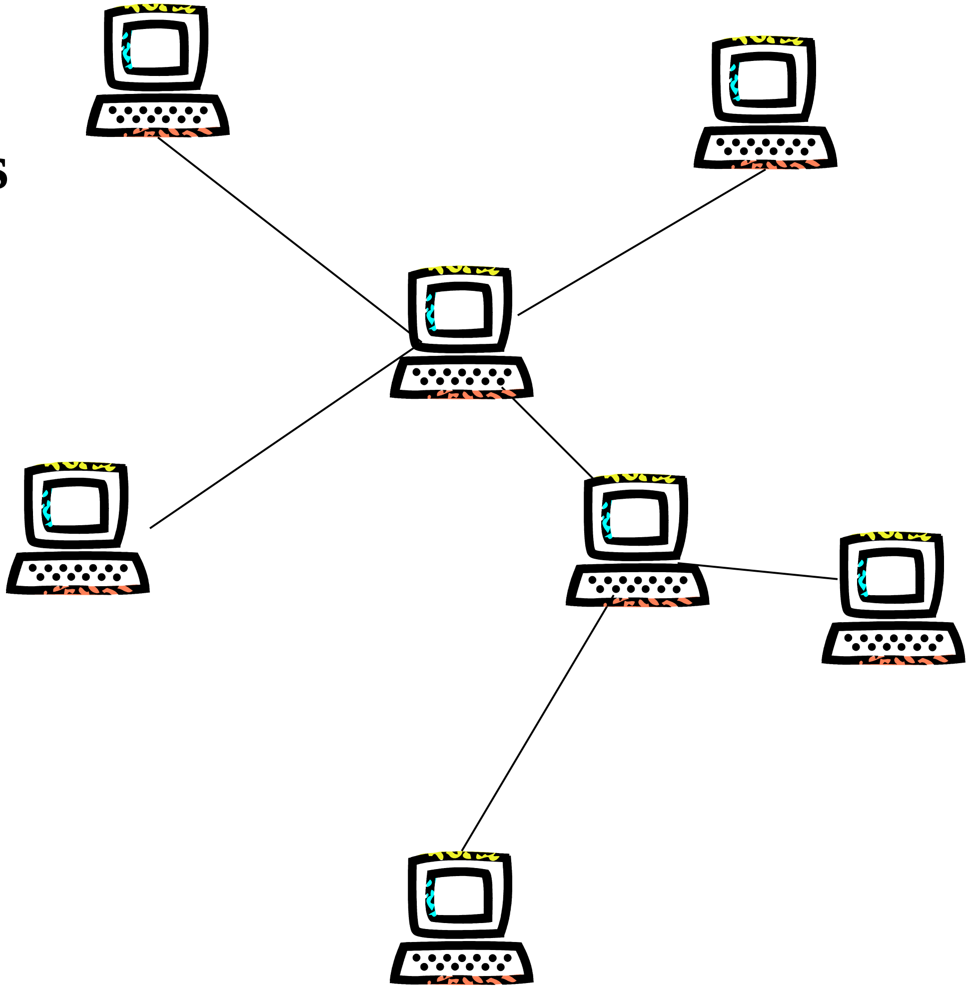
# The Web Model

- **Contact a server and download a web page**
- **Server has all the resources and capabilities**



# The P2P Model

- A peer's resources is similar to the resources of the other participants
- P2P – peers communicating directly with other peers and sharing resources
- P2P services
  - Distributed Computing
  - File Sharing
  - Collaboration



# Advantages

- **Clients provide resources, including bandwidth, storage space, and computing power**
- **As nodes arrive and demand on the system increases, the total capacity of the system also increases**
- **Distributed nature also increases robustness in case of failures by replicating data over multiple peers**
  - **Enable peers to find the data without relying on a centralized index server**

# P2P applications

- **File sharing**
  - **Using application layer protocols**
    - DirectConnect (centralized), Gnutella (flooding), BitTorrent (hybrid)
- **VoIP**
  - **Using application layer protocols**
    - SIP
- **Streaming media**
- **Instant messaging**
- **Software publication and distribution**
- **Media publication and distribution**
  - **radio, video**

# Challenges

- Peer discovery and group management
- Data location, searching and placement
  - Search and routing
- Reliable and efficient file delivery
- Security/privacy/anonymity/trust

# P2P types

- **Pure P2P** refers to an environment where all the participating nodes are peers
  - No central system controls, coordinates, or facilitates the exchanges among peers
- **Hybrid P2P** refers to an environment where there are servers which enable peers to interact with each other
  - The degree of central system involvement varies with the application
  - Different peers may have different functions (simple nodes, routers, rendezvous)

No method is better than the other

- each has its advantages and its drawbacks, each is the right choice for some applications

# Simple Peers

- Single end user, allowing that user to provide services from his device and consuming services provided by other peers on the network.
  - Will usually be located behind a firewall, separated from the network at large; peers outside the firewall will probably not be capable of directly communicating with the simple peer located inside the firewall.
  - Because of their limited network accessibility, simple peers have the least amount of responsibility in any P2P network.

- They are not responsible for handling communication on behalf of other peers or serving third-party information for consumption by other peers.

# Rendezvous Peers

- Gathering or meeting place
  - Provides peers with a network location to use to discover other peers and peer resources.
- Peers issue discovery queries to a rendezvous peer, and the rendezvous provides information on the peers it is aware of on the network.
- May cache information on peers for future use or by forwarding discovery requests to other rendezvous peers.
  - Improve responsiveness, reduce network traffic, and provide better service to simple peers.

- Usually outside a private internal network’s firewall. A rendezvous could exist behind the firewall, but it would need to be capable of traversing the firewall using either a protocol authorized by the firewall or a router peer outside the firewall.

# Router (Relay) Peers

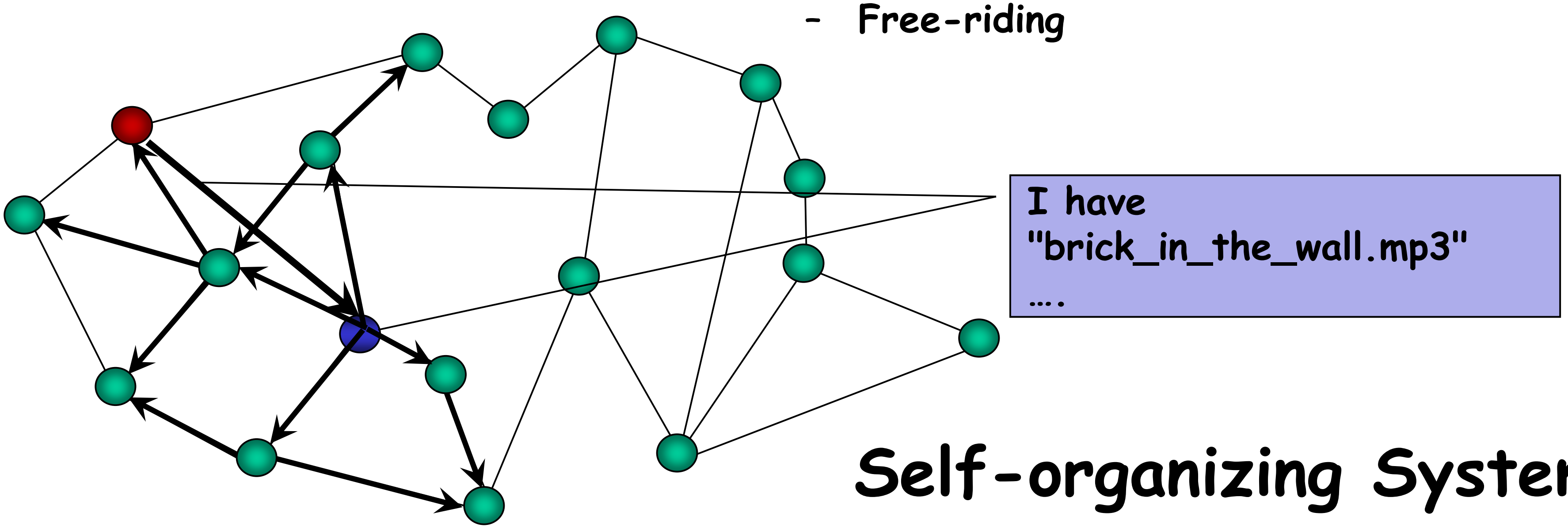
- A router peer provides a mechanism for peers to communicate with other peers separated from the network by firewall or Network Address Translation (NAT) equipment.
- Peers outside the firewall to communicate with a peer behind the firewall, and vice versa.
- A relay is not necessarily a rendez-vouz peer
  - Relay is on the data stream
  - Rendez-vous is always on the discovery path (and maybe in the data stream).

# Structured vs Unstructured

- Unstructured P2P networks
  - Formed when the overlay links are established arbitrarily.
  - If a peer wants to find a desired piece of data in the network, the query has to be flooded through the network to find as many peers as possible that share the data
    - The queries may not always be resolved
      - If a peer is looking for rare data shared by only a few other peers, then it is highly unlikely that search will be successful
    - Flooding causes a high amount of signalling traffic in the network
    - Gnutella and FastTrack/KaZaa, BitTorrent
- Structured P2P networks
  - Globally consistent protocol (logic) to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare
  - The most common type of structured P2P network is the Distributed Hash Table (DHT)
    - A variant of consistent hashing is used to assign ownership of each file to a particular peer
    - Chord, Pastry, Tapestry, CAN, Tulip, Kademlia, BitTorrent (trackerless)

# Fully Decentralized Information Systems

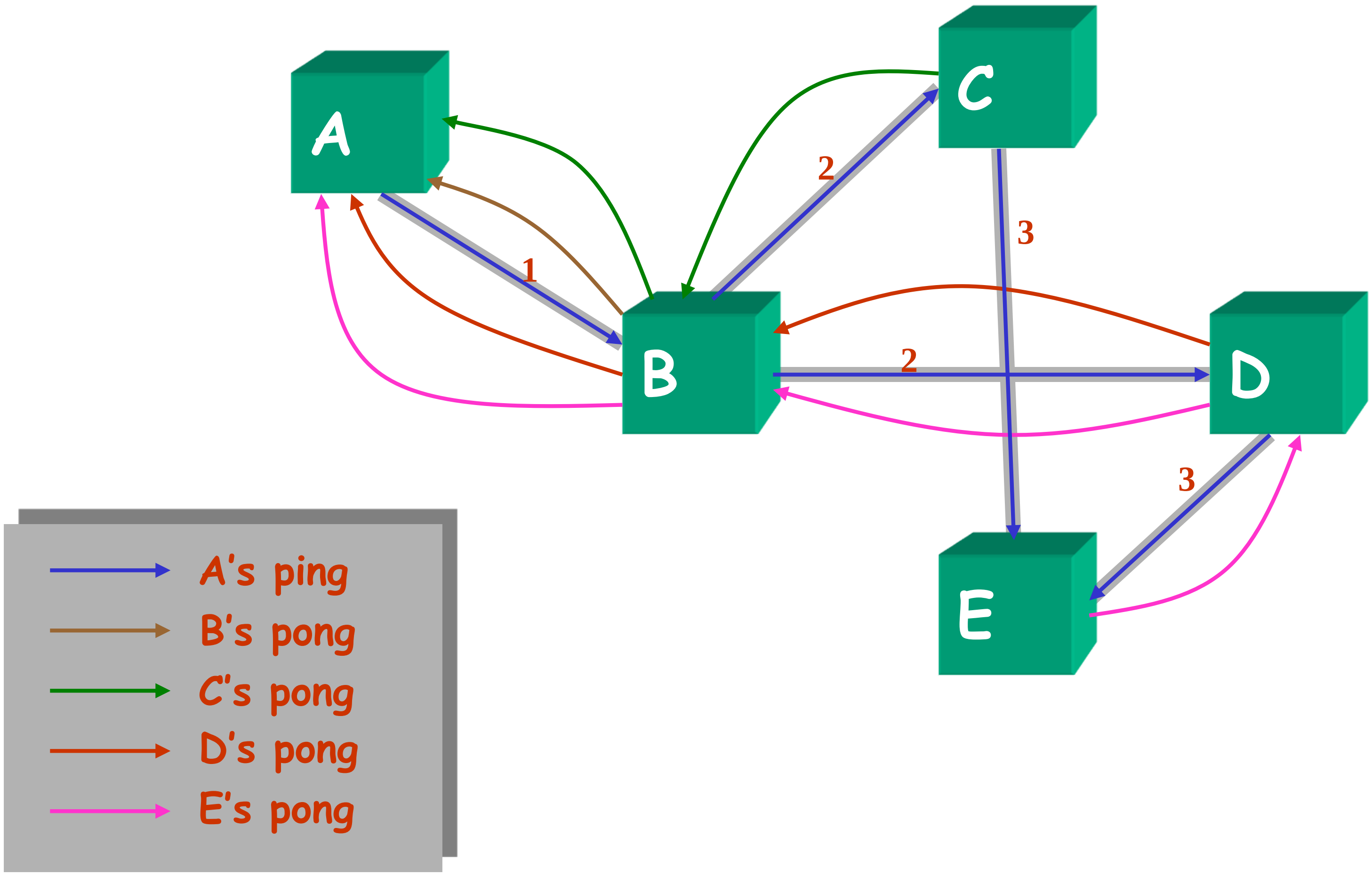
- P2P file sharing
  - Global scale application
- Example: Gnutella
  - 40.000 nodes, 3 Mio files (August 2000)
- Strengths
  - Good response time, scalable
  - No infrastructure, no administration
  - No single point of failure
- Weaknesses
  - High network traffic
  - No structured search
  - Free-riding




Self-organizing System

Gnutella: no servers

# Gnutella: Meeting Peers (Ping/Pong)

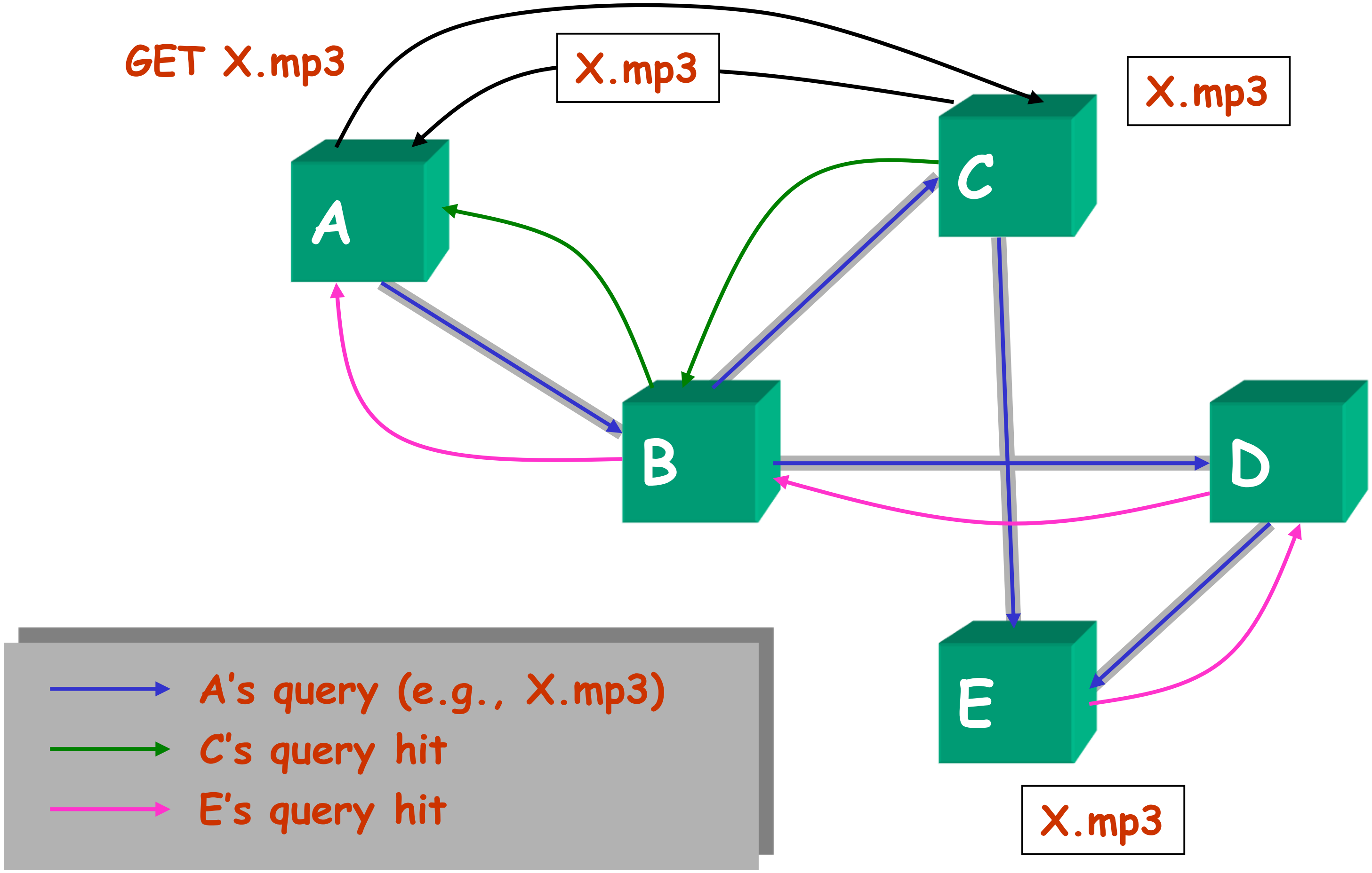


# Gnutella: Protocol Message Types

| Type | Description | Contained Information |
|---|---|---|
| Ping | Announce availability and probe for other servents | None |
| Pong | Response to a ping | IP address and port# of responding servent; number and total kb of files shared |
| Query | Search request | Minimum network bandwidth of responding servent; search criteria |
| QueryHit | Returned by servents that have the requested file | IP address, port# and network bandwidth of responding servent; number of results and result set |
| Push | File download requests for servents behind a firewall | Servent identifier; index of requested file; IP address and port to send file to |

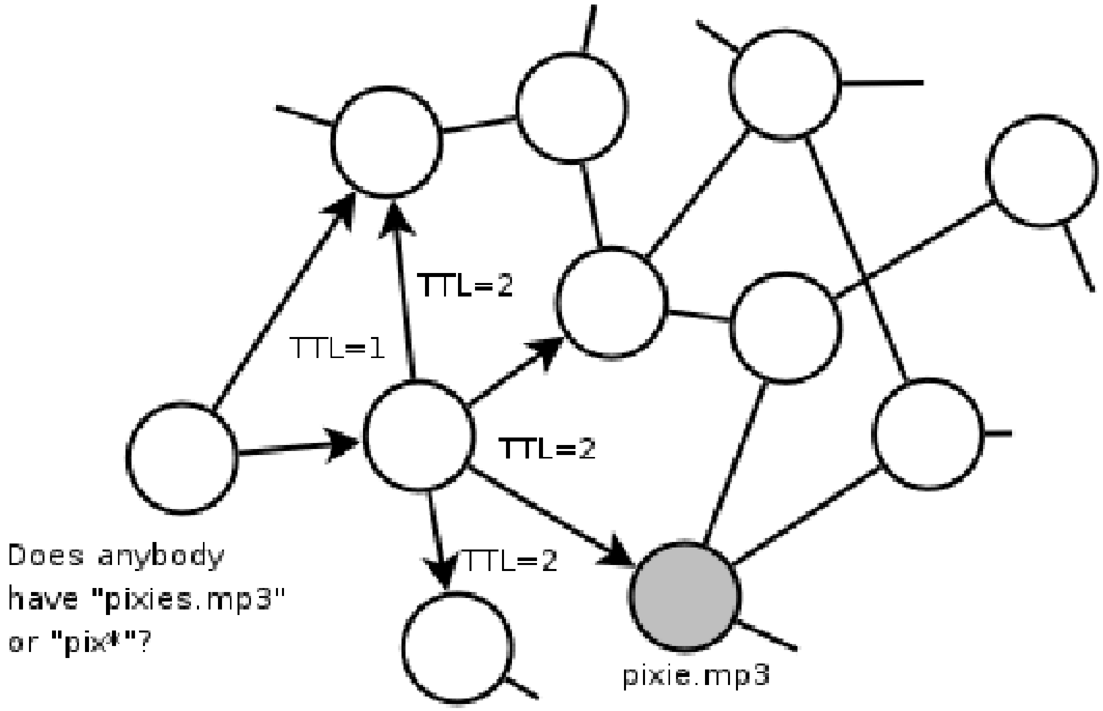
# Gnutella: Searching

## (Query/QueryHit/GET)



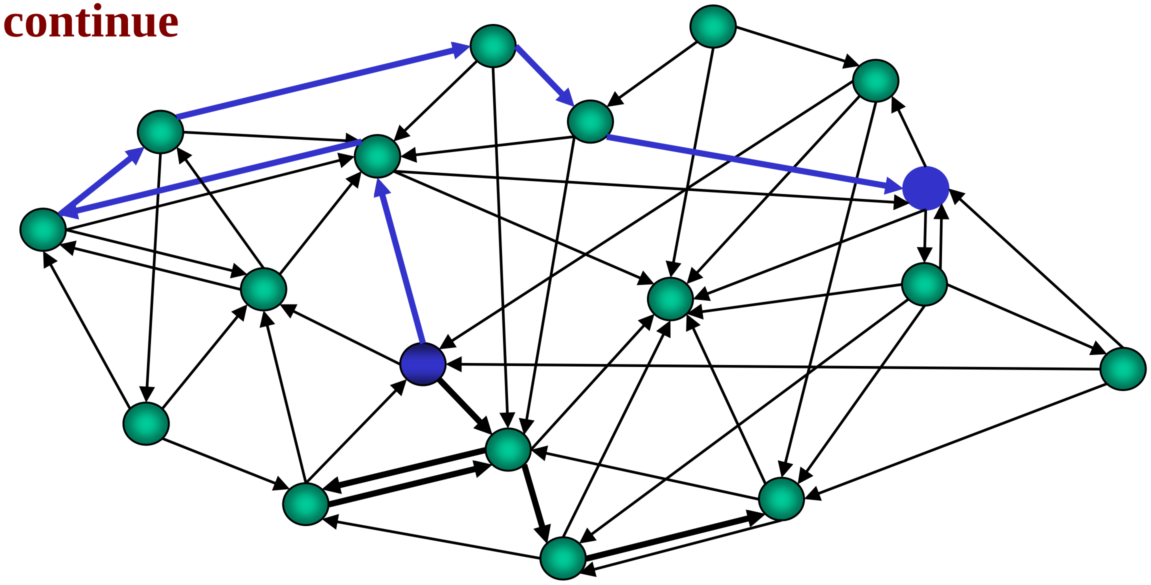
# Searching in Gnutella (structureless)

- **Queries are flooded to neighbours, have a TTL, and are forwarded only once**
- **Query may obtain several responses indicating which peers provides the requested file. Among those it selects one, and directly contacts it in order to download the file.**
  - **Can we search using fewer packets?**



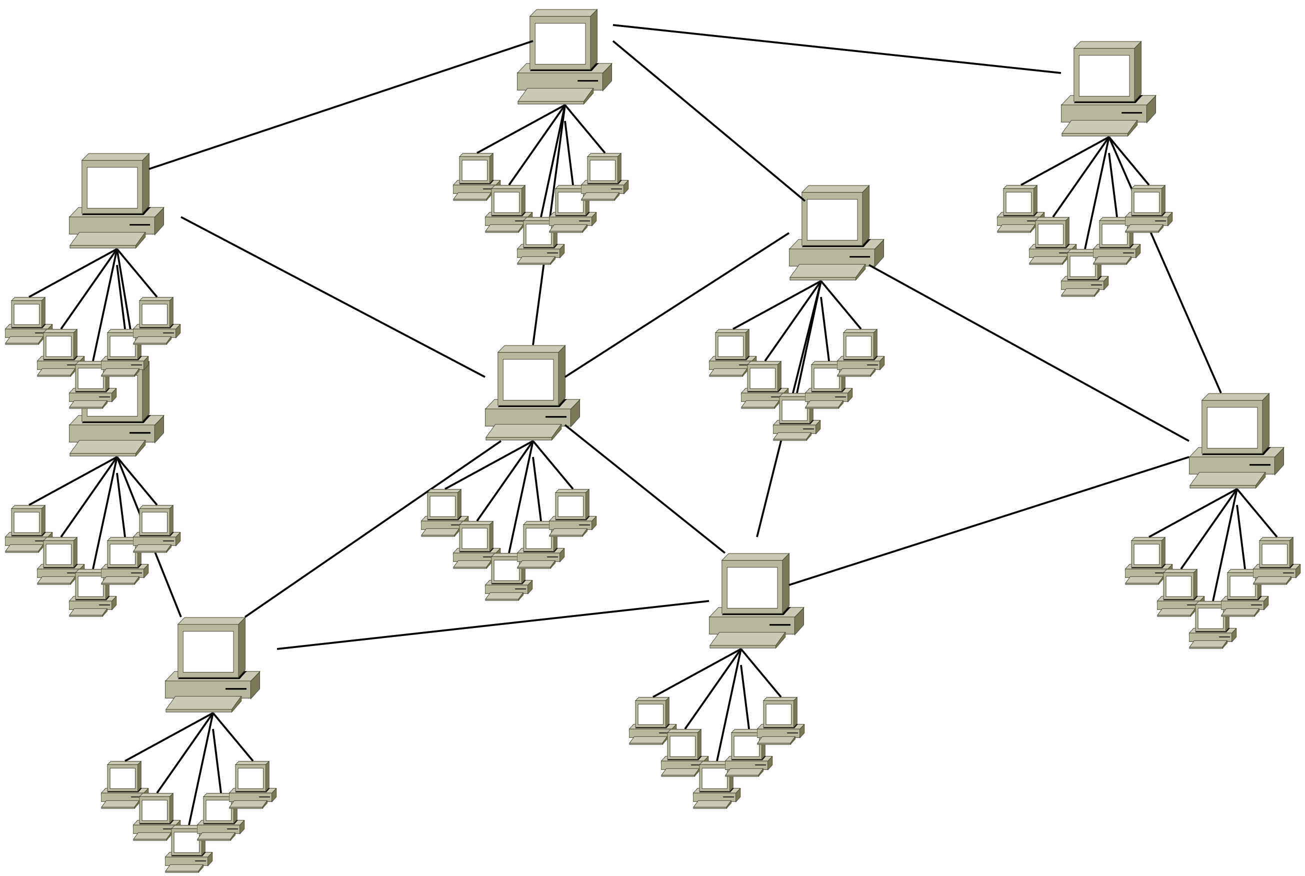
# Improvements of Message Flooding

- **Expanding Ring**
  - **start search with small TTL (e.g. TTL = 1)**
  - **if no success iteratively increase TTL (e.g. TTL = TTL +2)**
- **k-Random Walkers**
  - **forward query to one randomly chosen neighbor only, with large TTL**
  - **start k random walkers**
  - **random walker periodically checks with requester whether to continue**



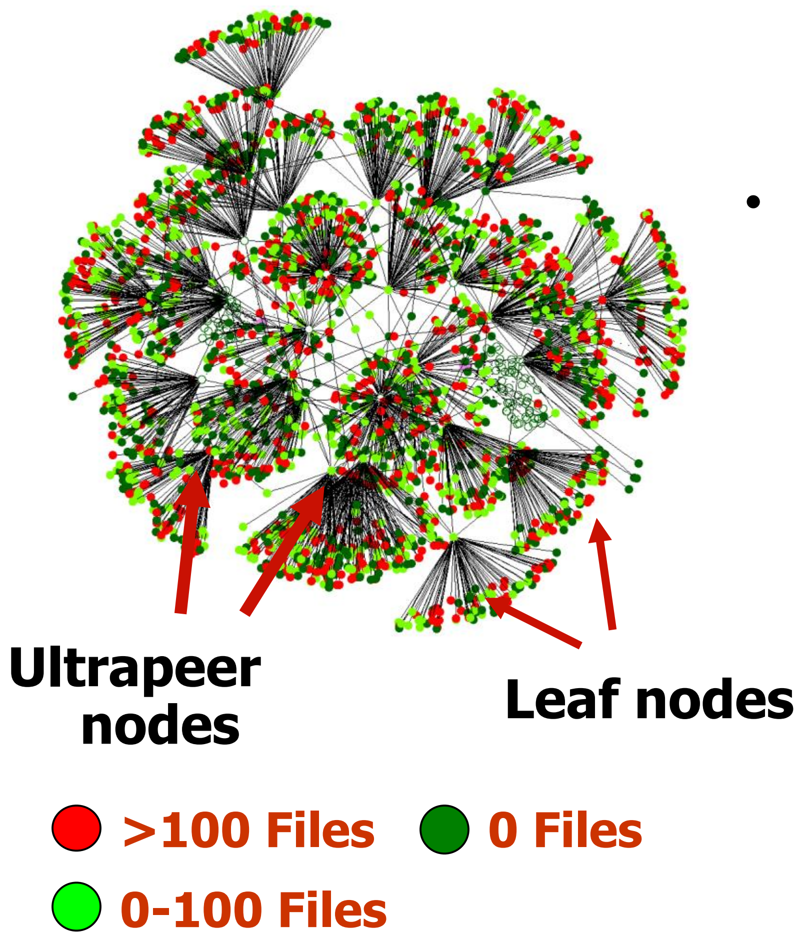
# Hybrid Gnutella: “Ultrapeers”

- Ultrapeers can be installed (KaZaA) or self-promoted (Gnutella v.2)



# Real Gnutella Network

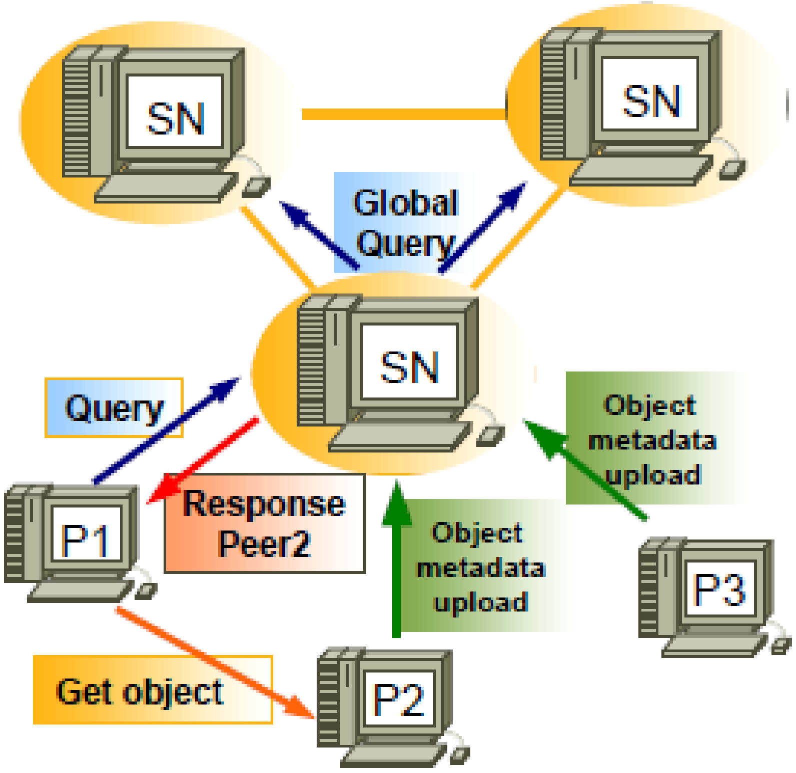
Oct 2003 Crawl of public gnutella (v.2)




- **Popular open-source file-sharing network**
  - ~450,000 users as of 2003
  - ~2,000,000 today
- **Ultrapeer-based Topology**
  - Queries flooded among ultrapeers
  - Leaf nodes shielded from query traffic
  - Based on multiple crawlers

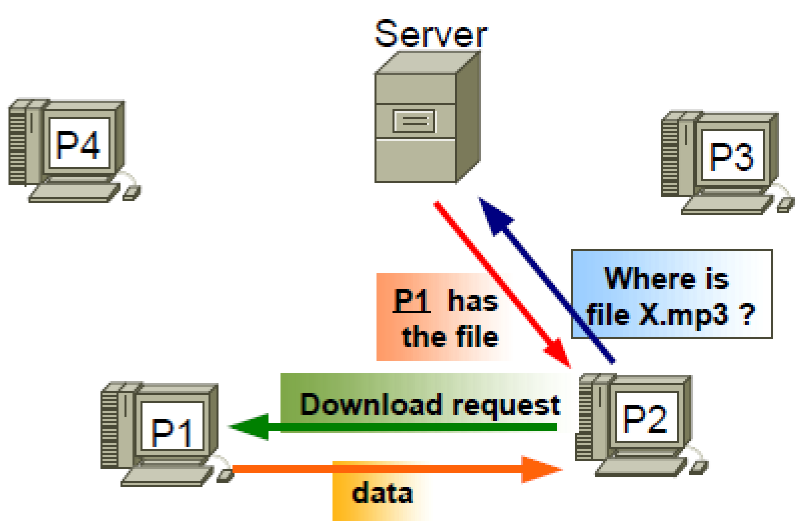
# FastTrack/KaZaA

- It is an extension of the Gnutella protocol which adds super-nodes to improve scalability (~gnutella v.2)
  - A peer application hosted by a powerful machine with a fast network connection become automatically a super-node, effectively acting as a temporary indexing server for other slower peers
  - Communicate between each others in order to satisfy search requests
- Network architecture: Hybrid Unstructured.
- Algorithm: Flooded Requests Model (FRM)



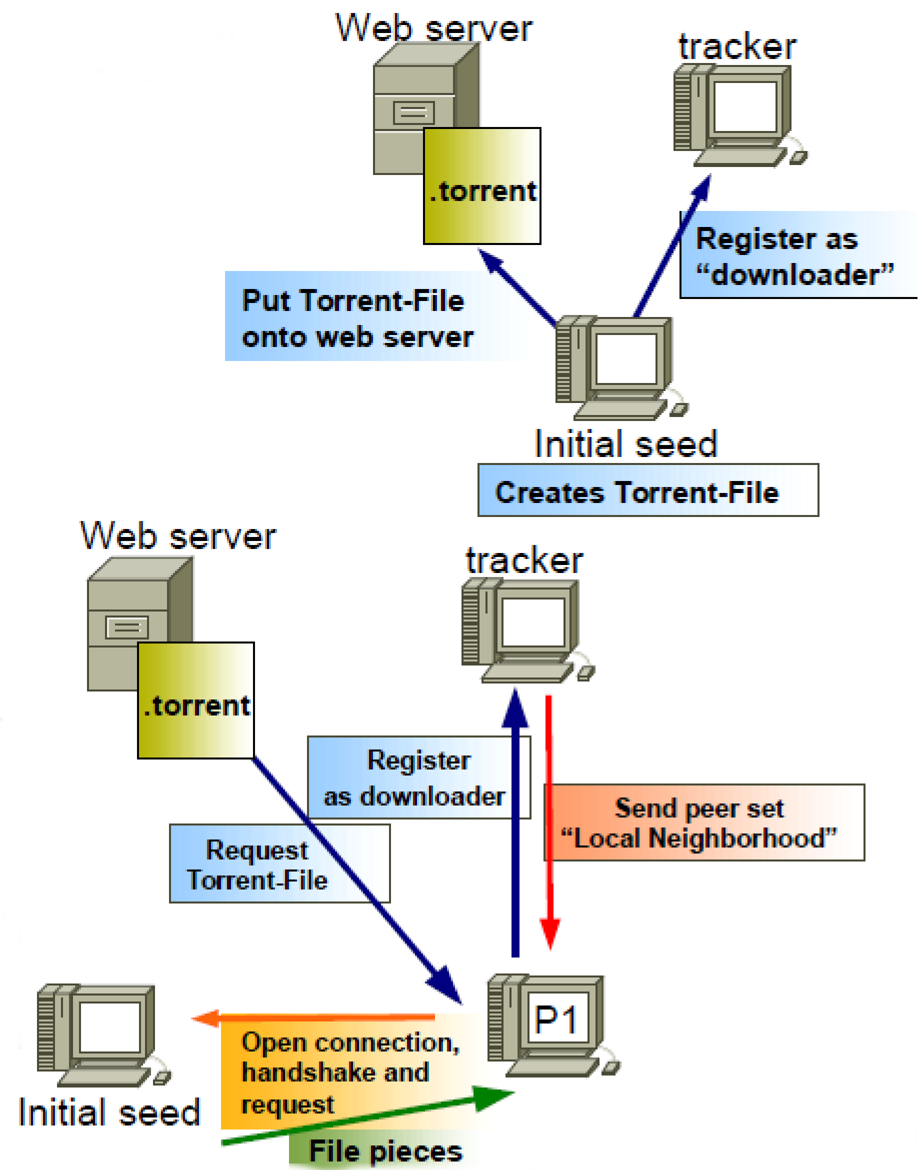
# OpenNAP/Napster

- Files (music) are on the client machine
- Servers provide search (rendezvous) and initiate direct transfers between clientes
- OpenNAP is an extension to other types and linking servers.
- Network architecture: Hybrid Unstructured.
- Algorithm: Centralized Directory Model (CDM)



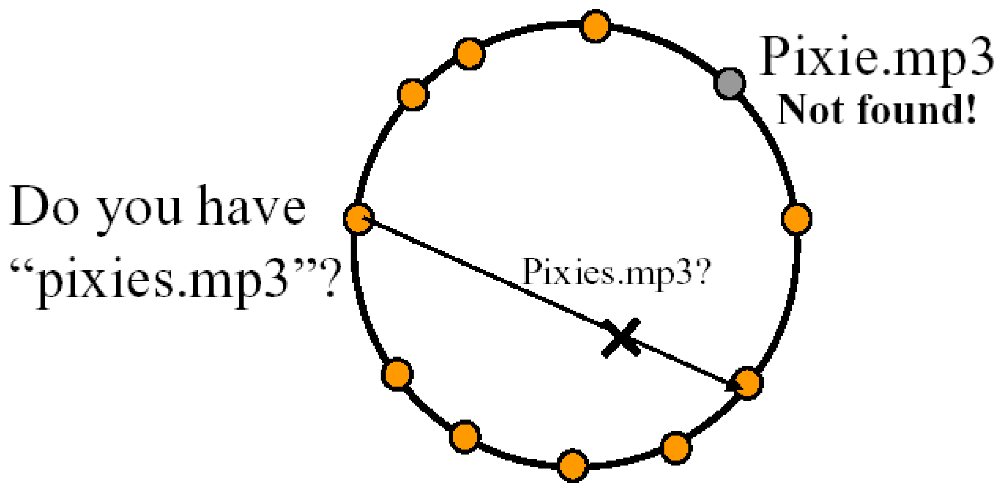
# BitTorrent

- Trackers: keeps track of the number of seeds/peers; responsible for helping downloaders find each other, using a simpl protocol on top of HTTP.
- Downloader sends status info to trackers, which reply with lists of contact information for peers which are downloading the same file.
- Web servers don’t have information about content location
  - Only store metadata files describing the objects (length, name, etc.) and associating to each of them the URL of a tracker
- Network architecture: Hybrid unstructured
- Algorithm: Centralized Directory Model (CDM)
- "trackerless" torrents through a system called the "distributed database”, through DHT (Distributed Hash Tables)

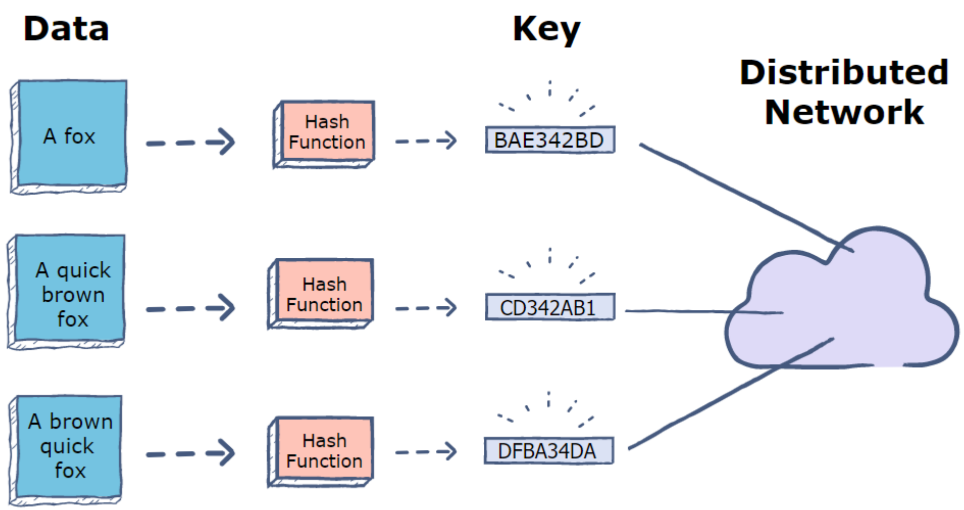


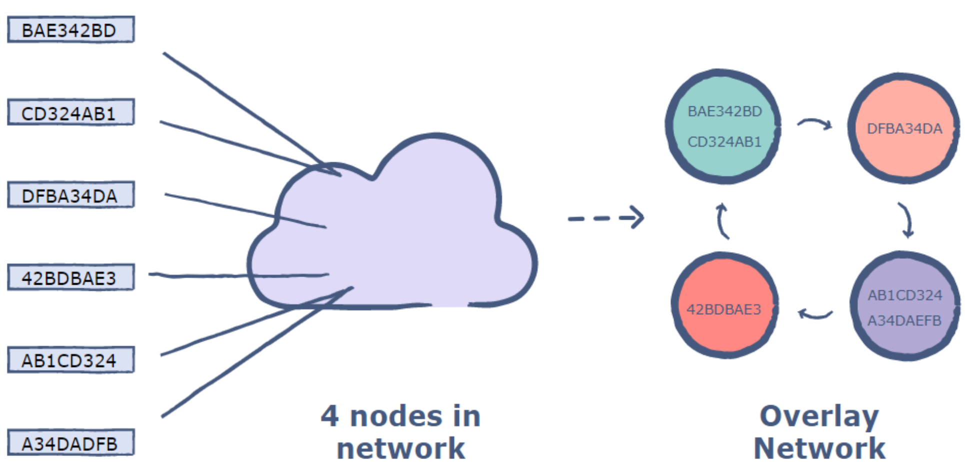
# Searching in DHTs (structured)

- **Need to know the exact filename**
  - **Keys (filenames) map to node-ids**
    - Change in file name → search at different nodes
    - No wildcard matching: cannot ask for file “pix*”



# DHT

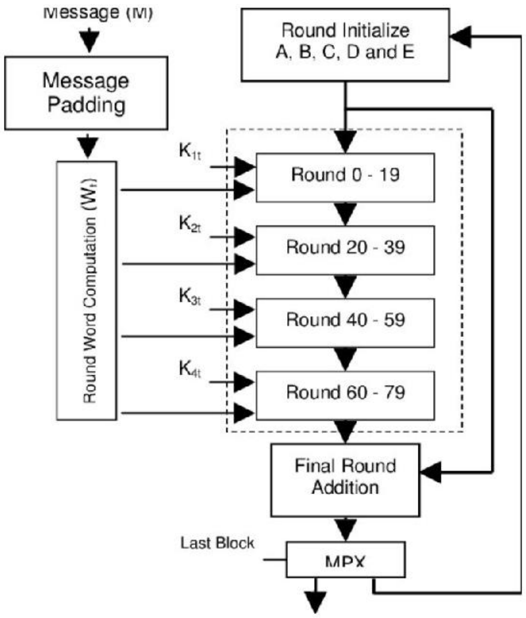

Data
Key
Distributed Network
A fox
A quick brown fox
A brown quick fox
Hash Function
Hash Function
Hash Function
BAE342BD
CD342AB1
DFBA34DA

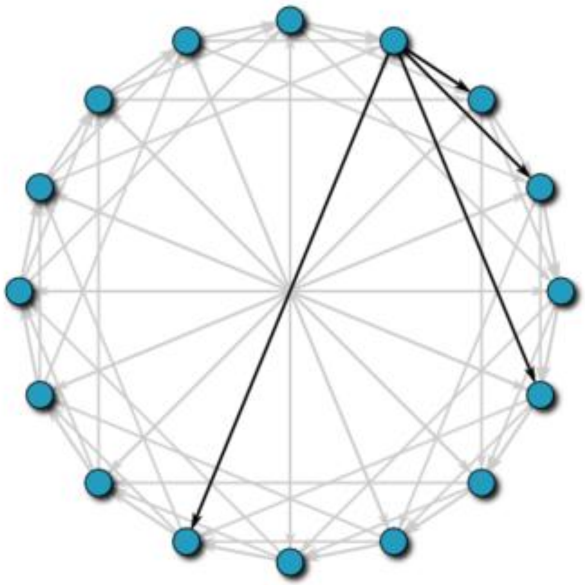

BAE342BD
CD324AB1
DFBA34DA
42BDBAE3
AB1CD324
A34DADFB
4 nodes in network
BAE342BD
CD324AB1
DFBA34DA
42BDBAE3
AB1CD324
A34DAEFB
Overlay Network

# CHORD: DHT Algorithm

- All files/data items in the network will have an identifier, which will be hashed to give a key for that particular resource
- If a node needs a file/data, it will hash its name and then send a request using this key.
- All *n* nodes also use the function to hash their IP address, and conceptually, the nodes will form a ring in ascending order of their hashed IP

**SHA-1**



# CHORD: DHT Algorithm

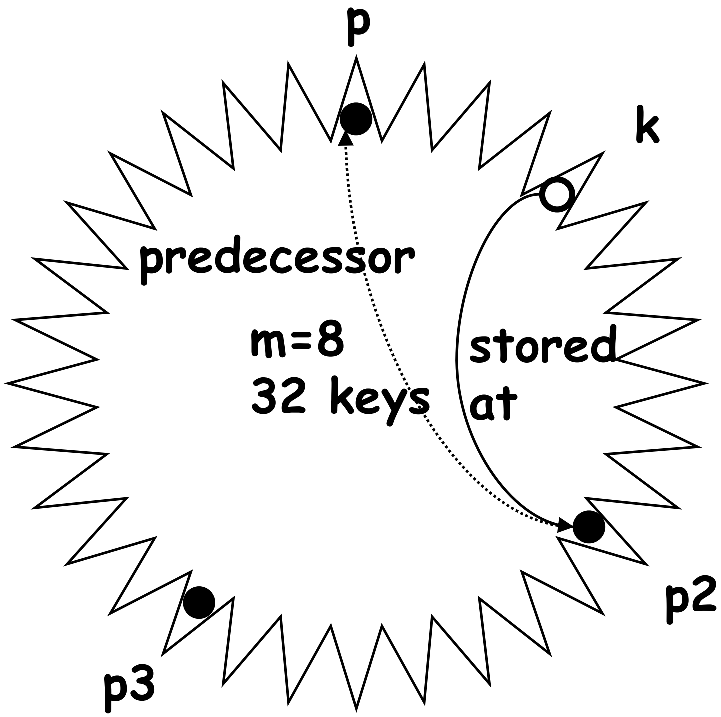
- The successor node of a key *k* is the first node whose ID equals to *k* or follows *k* in the identifier circle, denoted by successor(*k*)
- Every key is assigned to its successor node, so looking up a key *k* is to query successor(*k*).

- When a node wants to share a file or some data
  – Hashes the identifier to generate a key *k*, and sends its IP and the file identifier to successor(*k*)
  – These are then stored at this node
  – All resources are indexed in a large DHT across all participating nodes
  – If there are two or more nodes that hold a given file or resource, the keys will be stored at the same node in the DHT, giving the requesting node a choice

# DHT: Store Information

- Hashing of search keys AND peer addresses on binary keys of length m
  - e.g. m=8, key("jingle-bells.mp3")=17, key(196.178.0.1)=3

- Data keys are stored at next larger node key

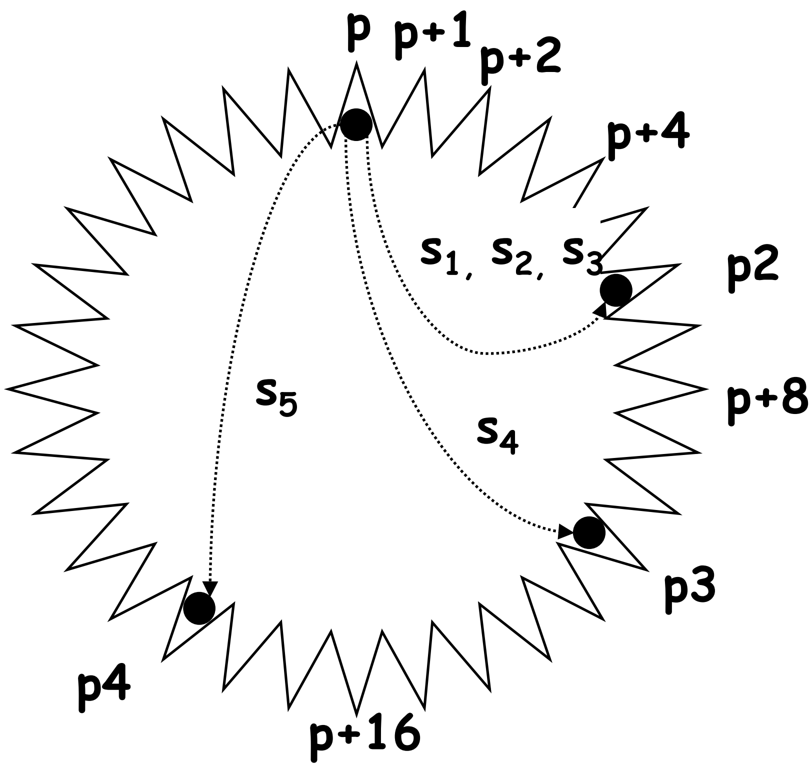



Search possibilities
1. every peer knows every other
   O(n) routing table size
2. peers know successor
   O(n) search cost

# DHT: Store Information

- **Every peer knows m peers with exponentially increasing distance**

Each peer p stores a routing table
First peer with hashed identifier $s_i$ such that
$s_i$ =successor($p+2^{i-1}$) for i=1,..,m

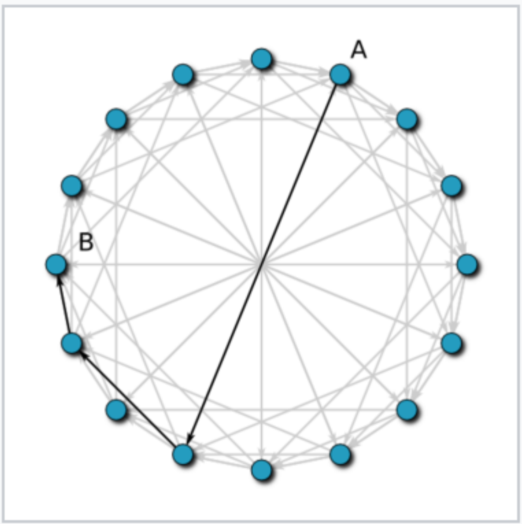



| i | $s_i$ |
|---|---|
| 1 | p2 |
| 2 | p2 |
| 3 | |
| 4 | p3 |
| 5 | p4 |

Search
O(log n) routing table size

# DHT: Search Information

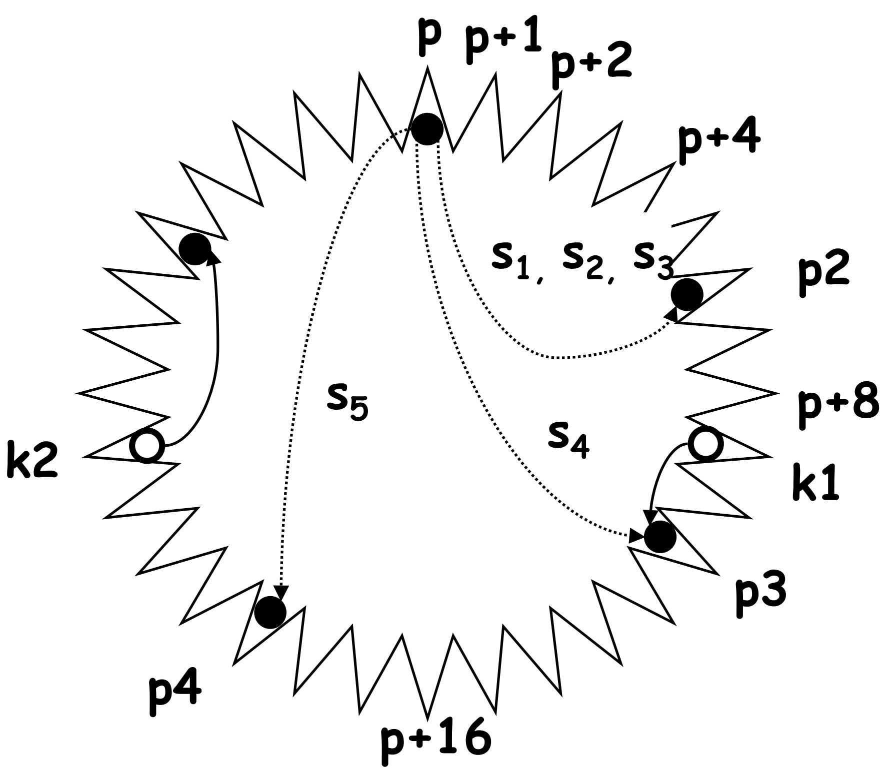
- When a node wants a content
  - Hashes its identifier and sends a request to successor(*k*)
  - Reply with the IP of the node that holds the actual data
  - How does a node request information from successor(*k*), when it doesn't know its IP, but only the key?
    - Every node holds what is known as a finger table
      - Contains a list of keys and their successor IP's
      - Each node holds the IP of an exponential sequence of nodes that follow it, i.e. entry *i* of node *k*'s finger table holds the IP of node *k* + 2^*i*




The routing path between nodes A and B. Each hop cuts the remaining distance in half (or better).

# Search

search(p, k)
find in routing table largest (i, p*) such that p* ∈ [p,k[
/* largest peer key smaller than the searched data key */
if such a p* exists then search(p*, k)
else return (successor(p)) // found




Search
O(log n) search cost

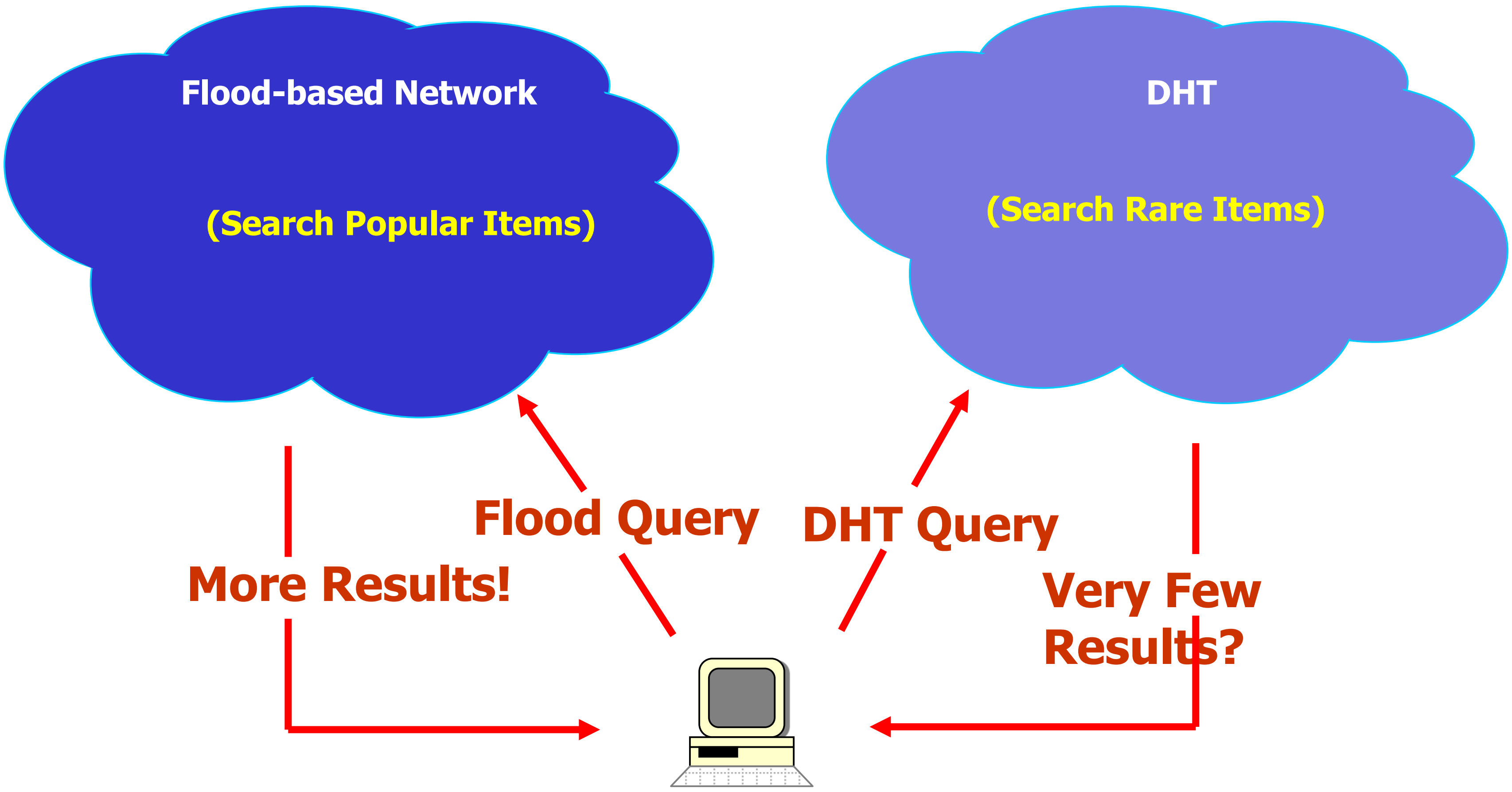
Routing Table with exp. increasing distance ⇒ O(log n) with high probability

# File Search: Flooding vs. DHTs

- **Recall**
  - **Flooding can miss files**
  - **DHTs should never**
- **Query complexity**
  - **Flooding can handle arbitrary single-site logic**
  - **DHTs can do equijoins, selections, aggregates, etc.**
    - But not so good at fancy selections like wildcards
- **Query Performance**
  - **Flooding can be slow to find things, uses lots of BW**
  - **DHTs: expensive to publish documents with lots of terms**
  - **DHTs: expensive to intersect really long term lists**
    - Even if output is really small!
- **Hybrid solution!**

# Hybrid Search

## Hybrid = “Best of both worlds”



# Security - attacks

- **Poisoning attacks**
  - **e.g. providing files whose contents are different from the description**
- **Polluting attacks**
  - **e.g. inserting "bad" chunks/packets into an otherwise valid file on the network**
- **Freeloaders**
  - **Users or software that make use of the network without contributing resources to it**
- **Insertion of viruses to carried data**
  - **e.g. downloaded or carried files may be infected with viruses or other malware**
- **Malware in the peer-to-peer network software itself**
  - **e.g. distributed software may contain spyware**
- **Denial of service attacks**
  - **Attacks that may make the network run very slowly or break completely**
- **Filtering**
  - **Network operators may attempt to prevent peer-to-peer network data from being carried**
- **Identity attacks**
  - **e.g. tracking down the users of the network and harassing or legally attacking them**
- **Spamming**
  - **e.g. sending unsolicited information across the network- not necessarily as a denial of service attack**

# Security

- **Most attacks can be defeated or controlled by careful design of the peer-to-peer network and through the use of encryption**
  - **However, almost any network will fail when the majority of the peers are trying to damage it**
- **Anonymity**
  - **Some peer-to-peer protocols (such as Freenet) attempt to hide the identity of network users by passing all traffic through intermediate nodes**
- **Encryption**
  - **Some peer-to-peer networks encrypt the traffic flows between peers**
    - Make it harder for an ISP to detect that peer-to-peer technology is being used (as some artificially limit bandwidth)
    - Hide the contents of the file from eavesdroppers
    - Impede efforts towards law enforcement or censorship of certain kinds of material
    - Authenticate users and prevent 'man in the middle' attacks on protocols
    - Aid in maintaining anonymity

# To Conclude:

## P2P and self-organization in Architectures

**P2P self-organization concepts can be found at every layer, reflecting some sort of self-organization in the communication structure.**

| Layer | Communication type | Information discovery | Information transport |
|---|---|---|---|
| *(at which Layer we are considering self-organization)* | *Communication concept explored* | *How to figure where the information is* | *How is the information encapsulated for transport* |
| Networking Layer | Regular Internet (IP) protocols | Routing, DNS | TCP |
| Data Access Layer | Overlay Networks, P2P | Resource Location (DHT, central server) | Gnutella, FreeNet |
| Service Layer | Application interface | Messaging, Distributed Processing | Napster, SETI, Groove |
| User Layer | User Communities, Google Circles | Collaboration | eBay, Google+, Facebook, |