Very Large, Distributed Data Volumes

Content delivery networks and DHTs

Theodoros Chondrogiannis

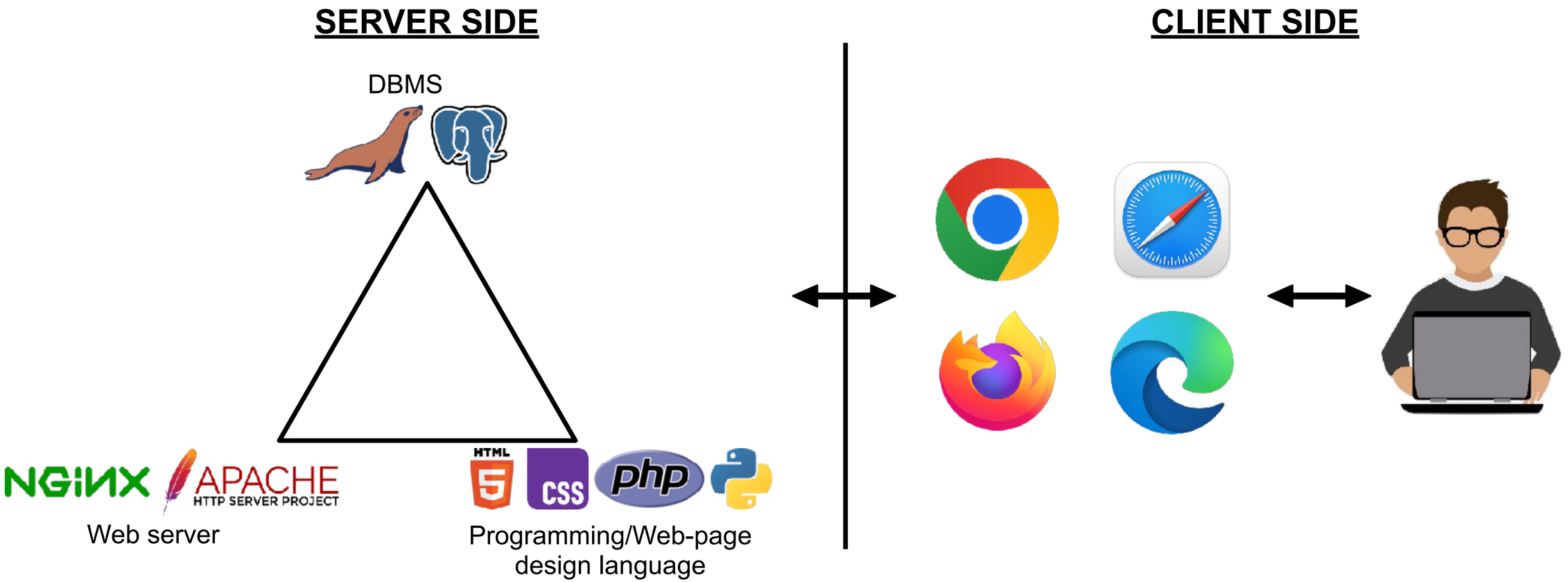
October 16, 2025

Outline

- Content delivery networks
- Overlay and P2P networks
- Distributed hash tables

Client-Server Model on the Internet

- Simple way to create a website on your own server



The Journey of a Request

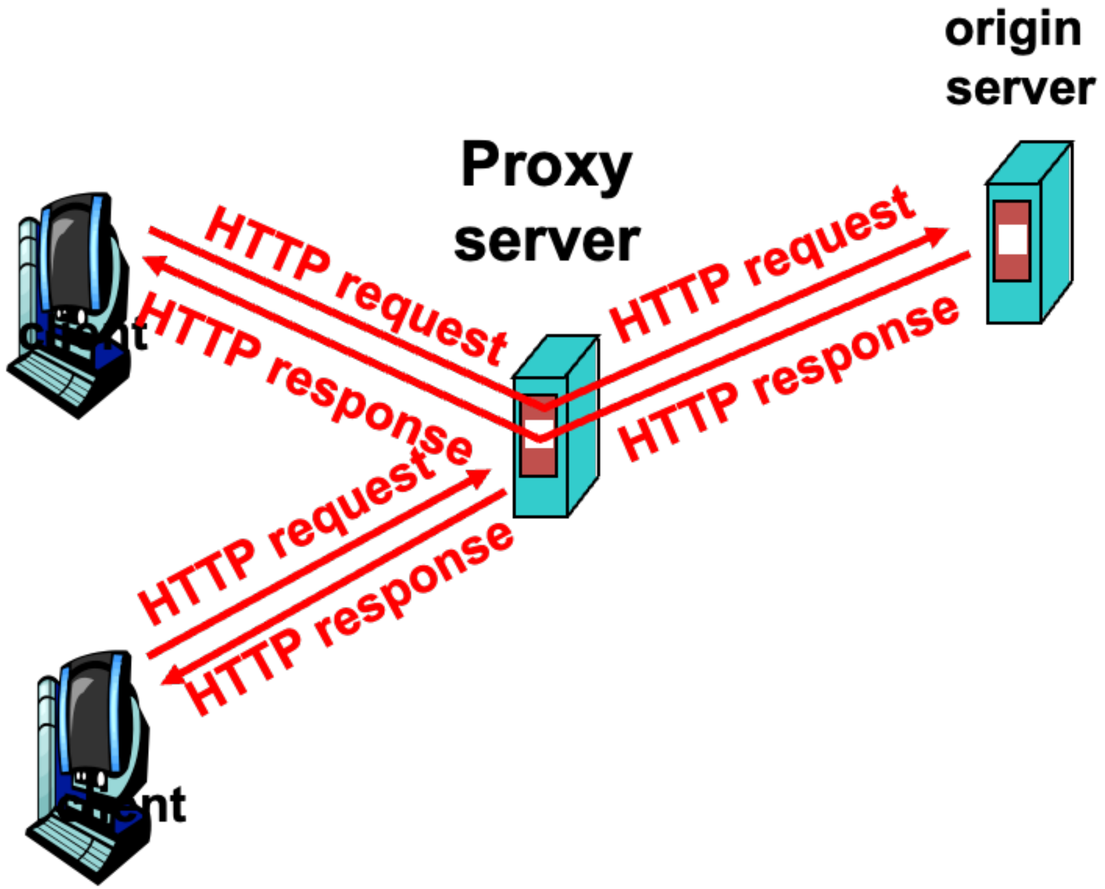
Single Server Architecture

- Single server
  - Easily overloaded
  - Single point of failure
  - Far from most clients (latency can be high)
- Popular content
  - Popular site
  - “Flash crowd” (aka “Slashdot effect”)
  - Denial of Service attack
- Solution: web caching

Outline

- Content delivery networks
- Overlay and P2P networks
- Distributed hash tables

Web Caching


origin
server
Proxy
server
HTTP request
HTTP response
HTTP request
HTTP response
HTTP request
HTTP response

Web Caching

- Cache “close” to the client
  - Under the administrative control of the client-side
- Explicit proxy
  - Requires configuring the browser
- Implicit proxy
  - Service provider deploys an “on path” proxy that intercepts and handles Web requests

Web Caching

- Cache “close” to the server
  - Either by proxy run by the server or in a third-party content distribution network (CDN)
- Directing clients to the proxy
  - Map the site name to the IP address of the proxy

Web Caching

- Proxy caches
  - Reactively replicates popular content
  - Reduces origin server costs
  - Reduces client ISP costs
  - Intelligent load balancing between origin servers
  - Offload form submissions (POSTs) and user auth
  - Content reassembly or transcoding on behalf of origin
  - Smaller round-trip times to clients
  - Maintain persistent connections to avoid TCP setup delay

Limitations of Web Caching

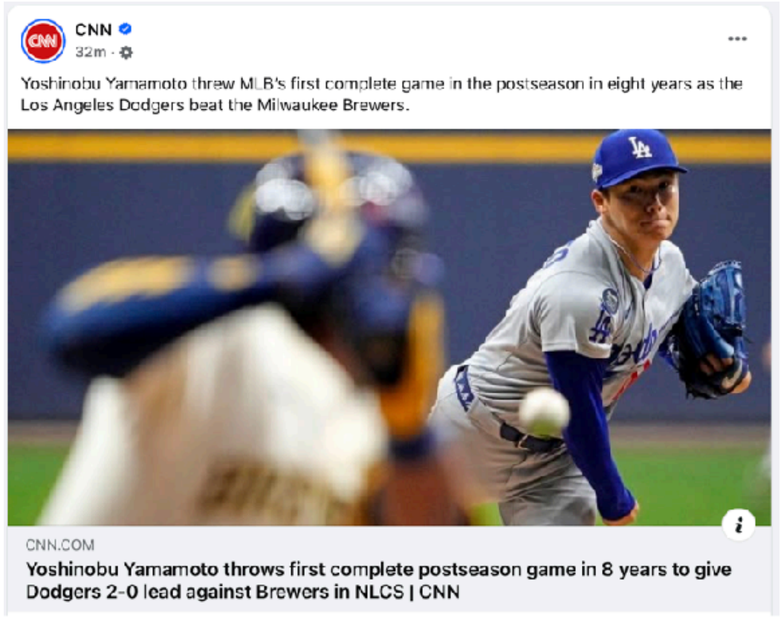
- Much content is not cacheable
  - Dynamic data: stock prices, scores, web cams
  - CGI scripts: results depend on parameters
  - Cookies: results may depend on past data
  - SSL: encrypted data is not cacheable
  - Analytics: the owner wants to measure hits
- Stale data
  - Or, the overhead of refreshing the cached data

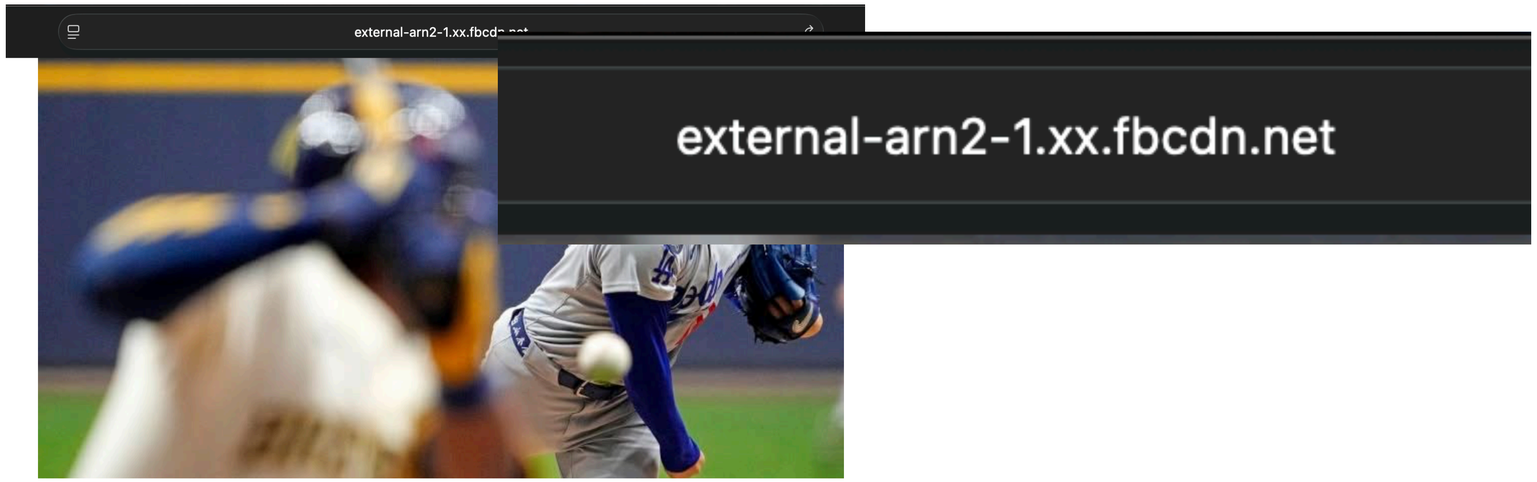
Content Delivery Networs

- The idea of a CDN is to geographically distribute a collection of server surrogates that cache pages normally maintained in some set of backend servers
  - Thus, rather than have millions of users wait forever to contact, e.g., *www.cnn.com* when a big news story breaks — such a situation is known as a flash crowd — it is possible to spread this load across many servers
  - Moreover, rather than having to traverse multiple ISPs to reach *www.cnn.com*, if these surrogate servers happen to be spread across all the backbone ISPs, then it should be possible to reach one without having to cross a peering point

Applications of CDNs

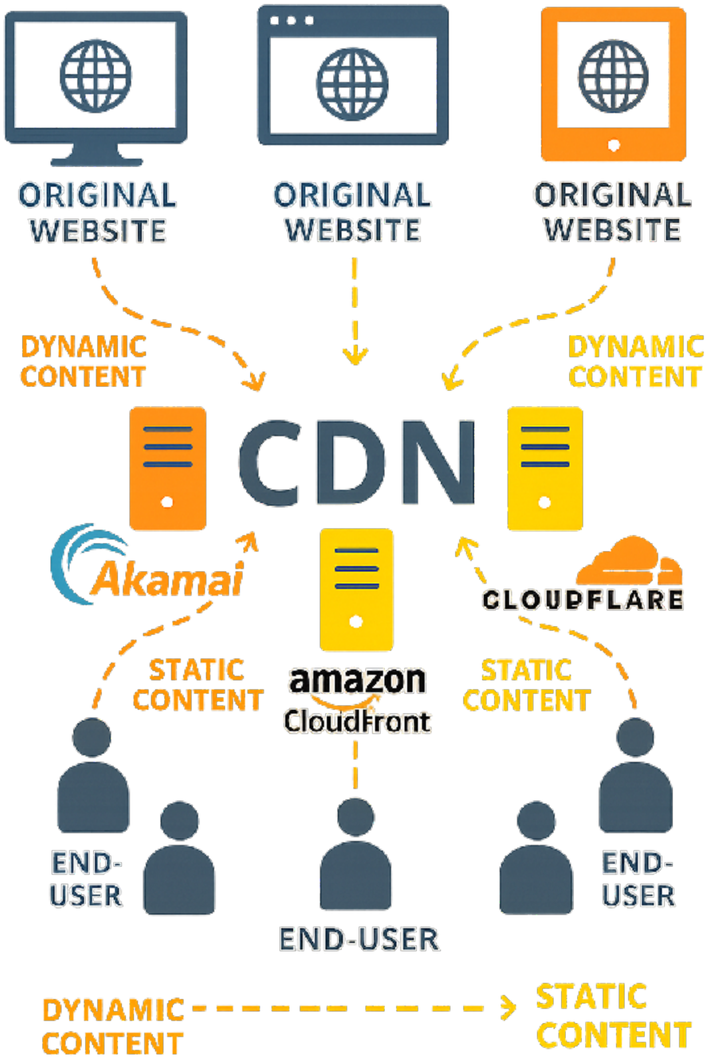
- While the initial idea is more than 25 years old and research in the area has stalled, CDNs are still widely used



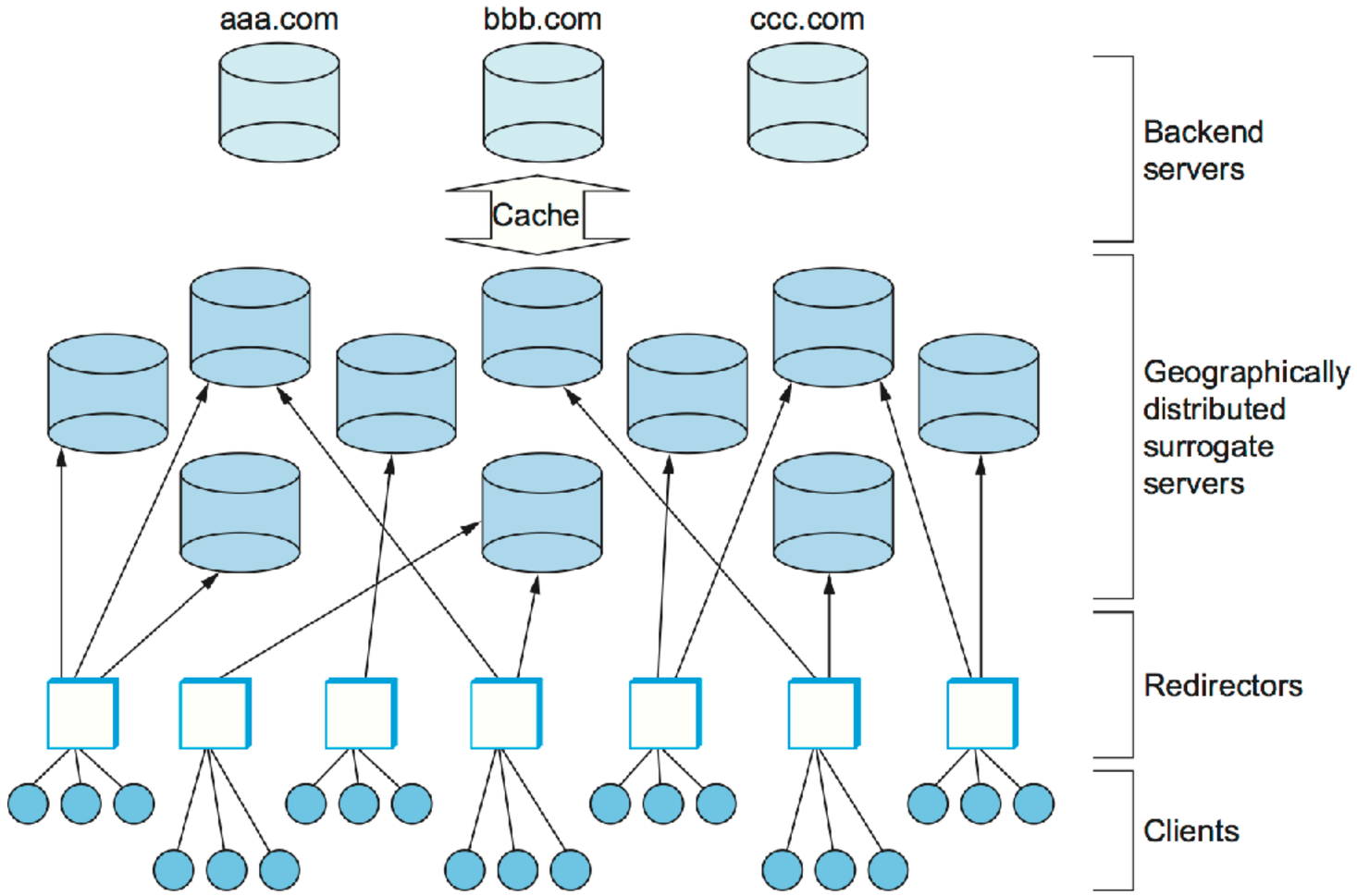


Content Delivery Networs

- Proactive content replication
  - Content provider (e.g., CNN) contracts with a CDN
- CDN replicates the content
  - On many servers spread throughout the Internet
- Updating the replicas
  - Updates are pushed to replicas when the content changes



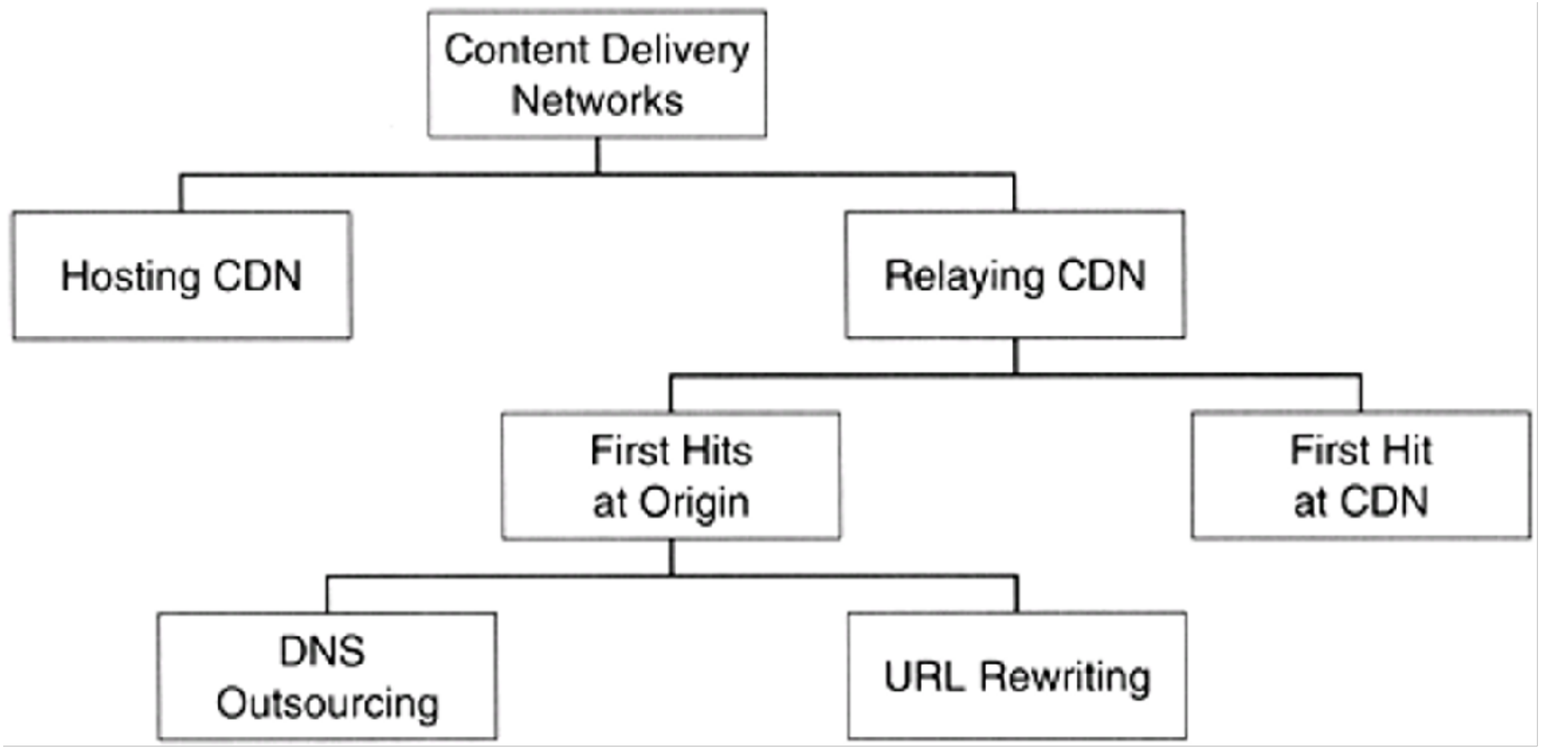
Content Delivery Networks



Surrogates as Server Replicas

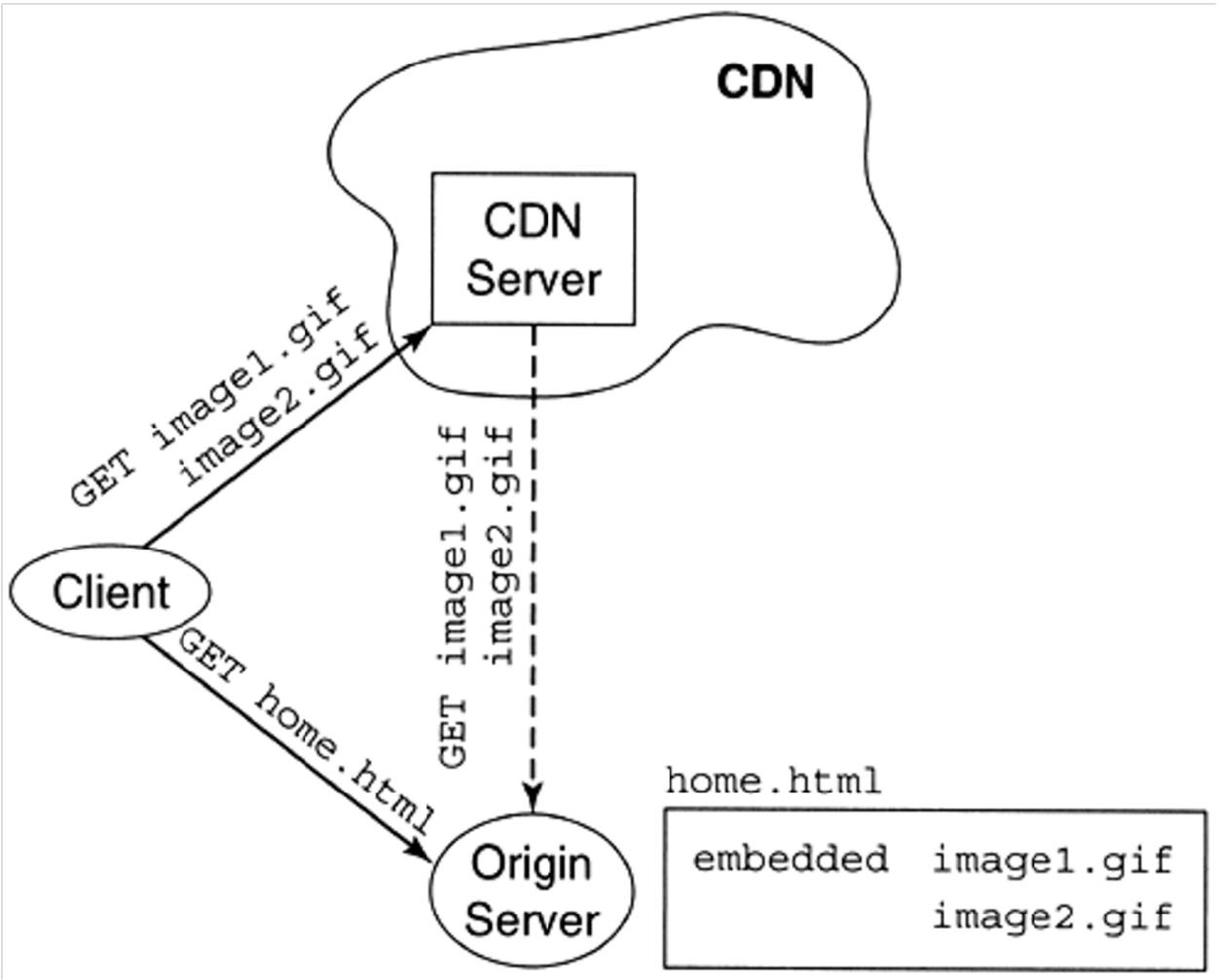
- Content requests are distributed among surrogates that are distinct from the origin servers
- A surrogate can fulfill the posed request
- Otherwise, the object is retrieved from the origin server
- DNS redirection is often used for distributing requests among the surrogates.

Types of CDNs



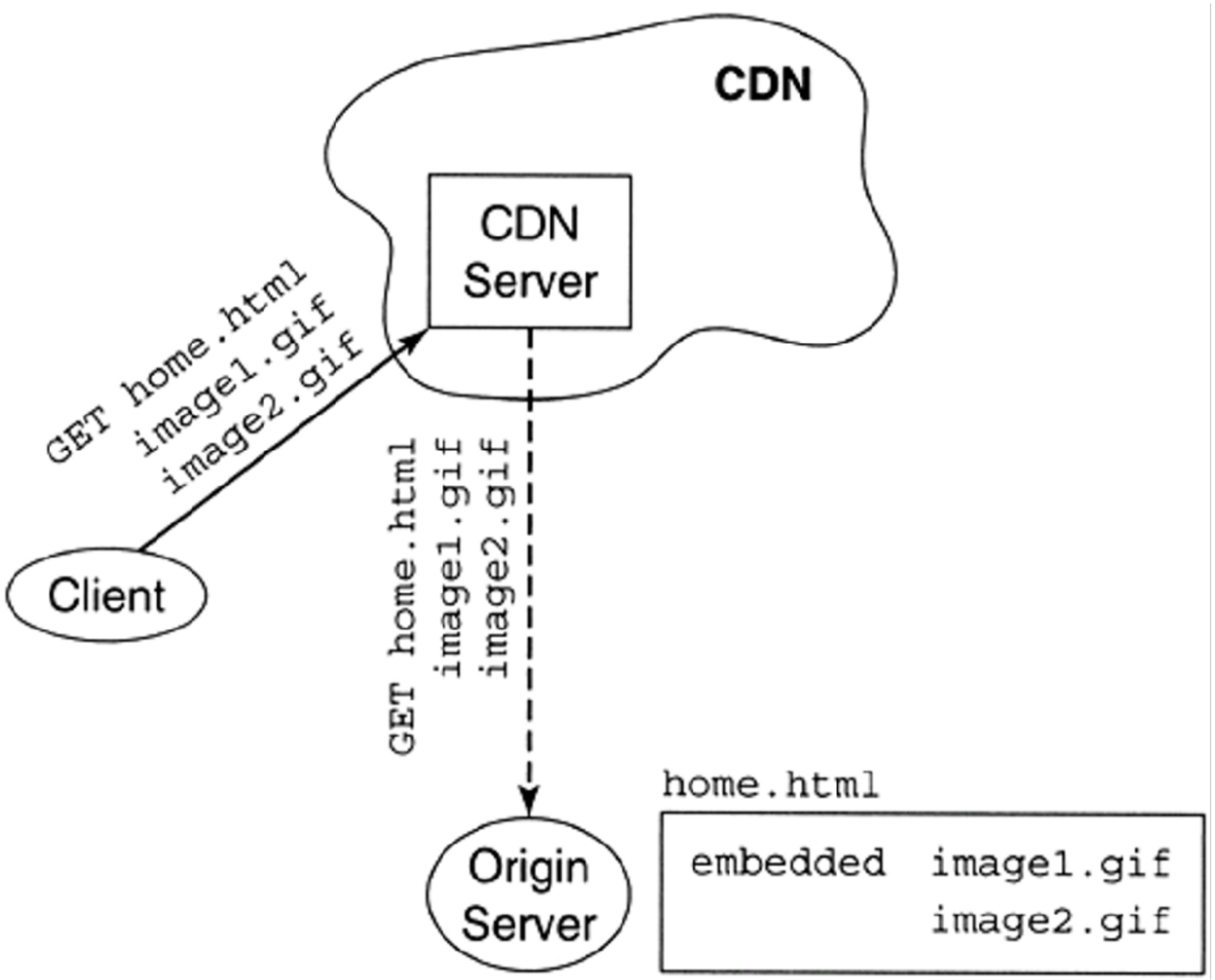
Relaying CDN

- 1st hit @ origin

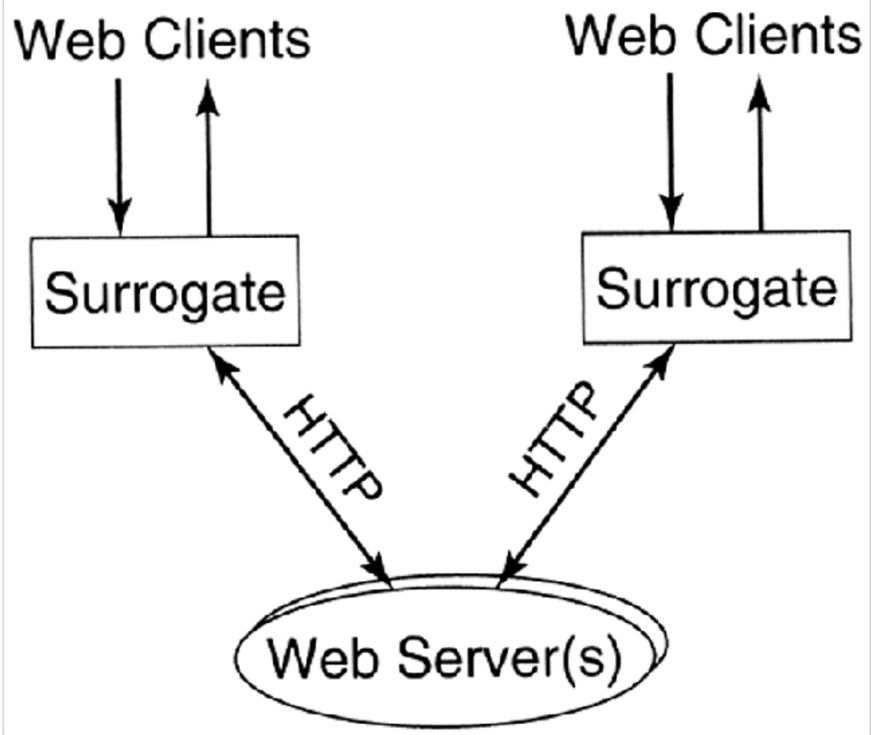


Relaying CDN

- 1st hit @ CDN

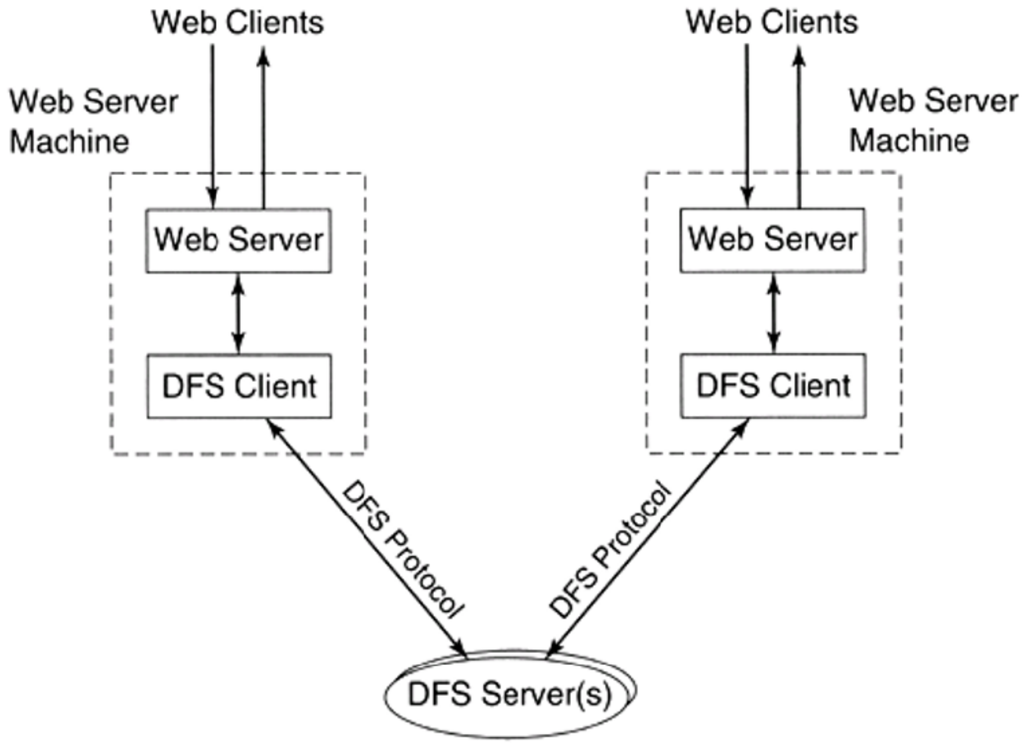


Partial Replication



Back-end File System

- Use a distributed file system to allow each replication server access to the same, shared file set.



Replication Issues

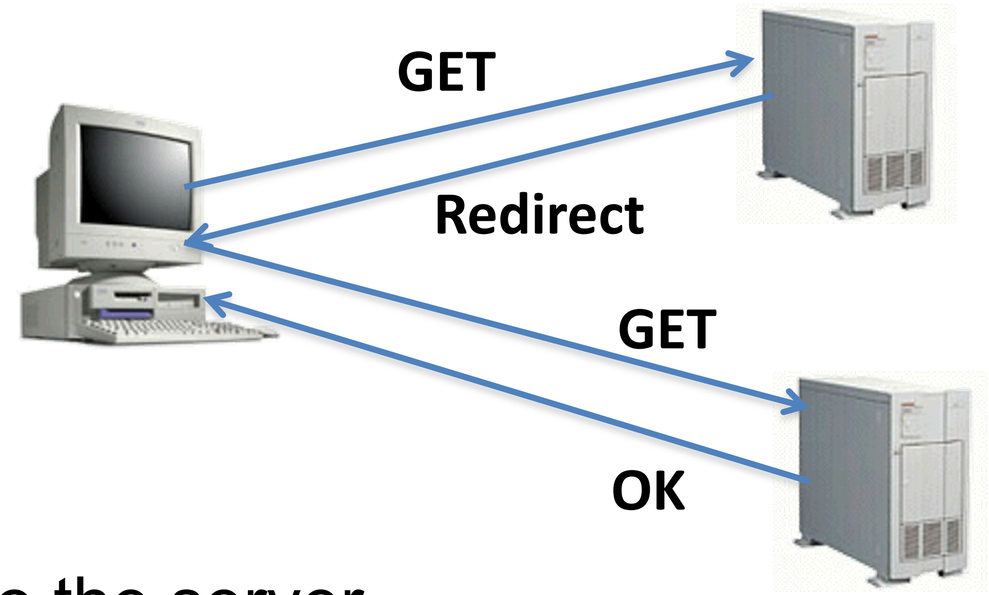
- Request distribution
  - How to transparently distribute requests for content among replication servers
- Server selection
  - How to select a server replica for a given request
- Content placement
  - How to decide how many replicas of given content to have and on which servers to place these replicas

Request Distribution

- Transparent replication
  - Techniques that require no user involvement or even awareness of the underlying replication scheme
- Clients use a single logical name when requesting content
- Basic issue
  - Re-direction of logically identical requests to distinct servers

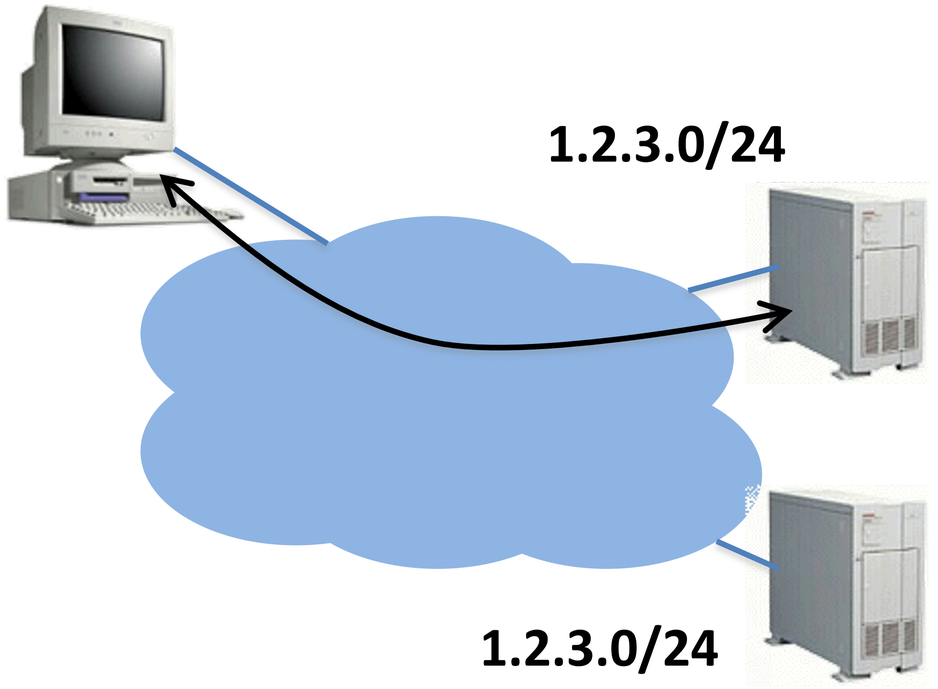
Server Selection Mechanism

- HTTP redirection
- Advantages
  - Fine-grain control
  - Selection based on client IP address
- Disadvantages
  - Extra round-trips for TCP connection to the server
  - Overhead on the server



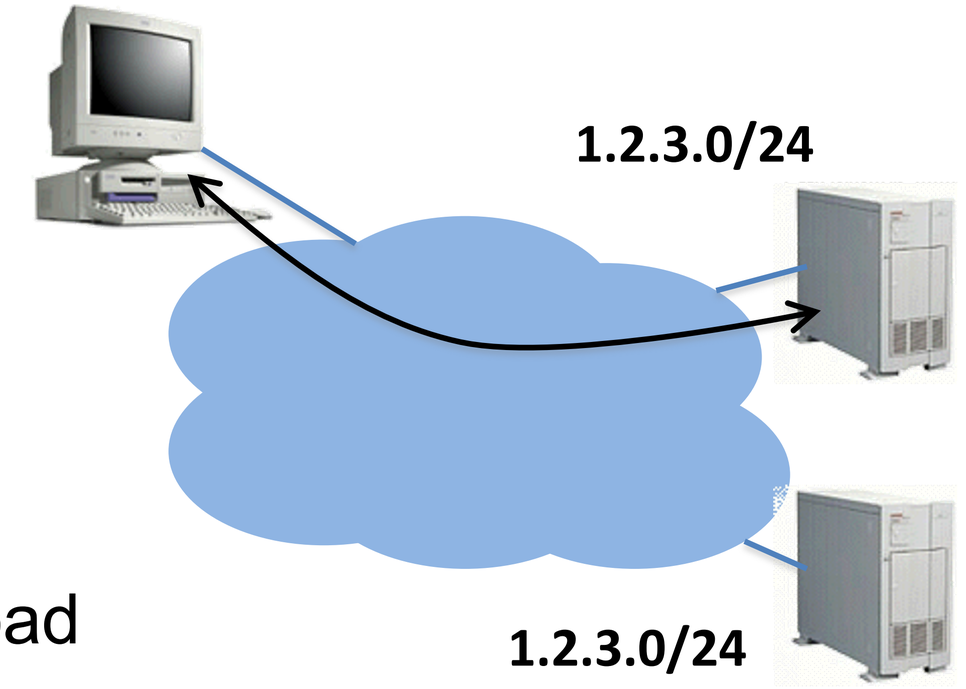
Server Selection Mechanism

- Anycast routing
  - Multiple physical servers use a single IP address called anycast address
  - Each server advertises both the anycast address and its regular address
  - Routers build paths that lead to the nearest anycast member-server



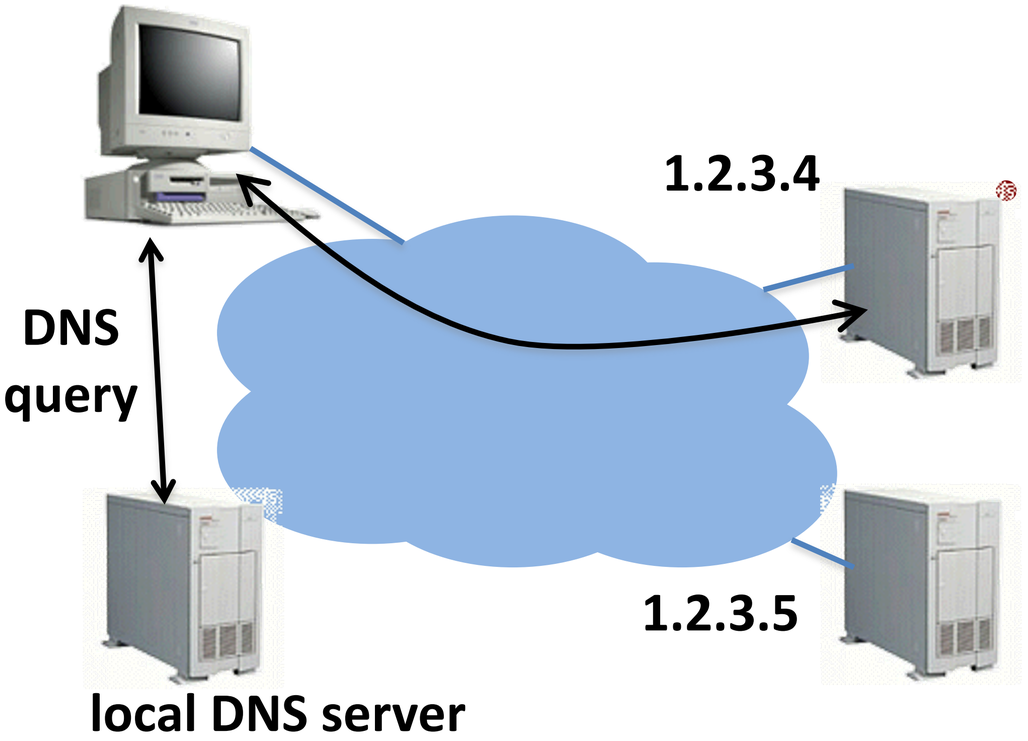
Server Selection Mechanism

- Anycast routing
- Advantages
  - No extra round-trips
  - Route to a nearby server
- Disadvantages
  - Does not consider network or server load
  - Different packets may go to different servers
  - Used only for simple request-response apps



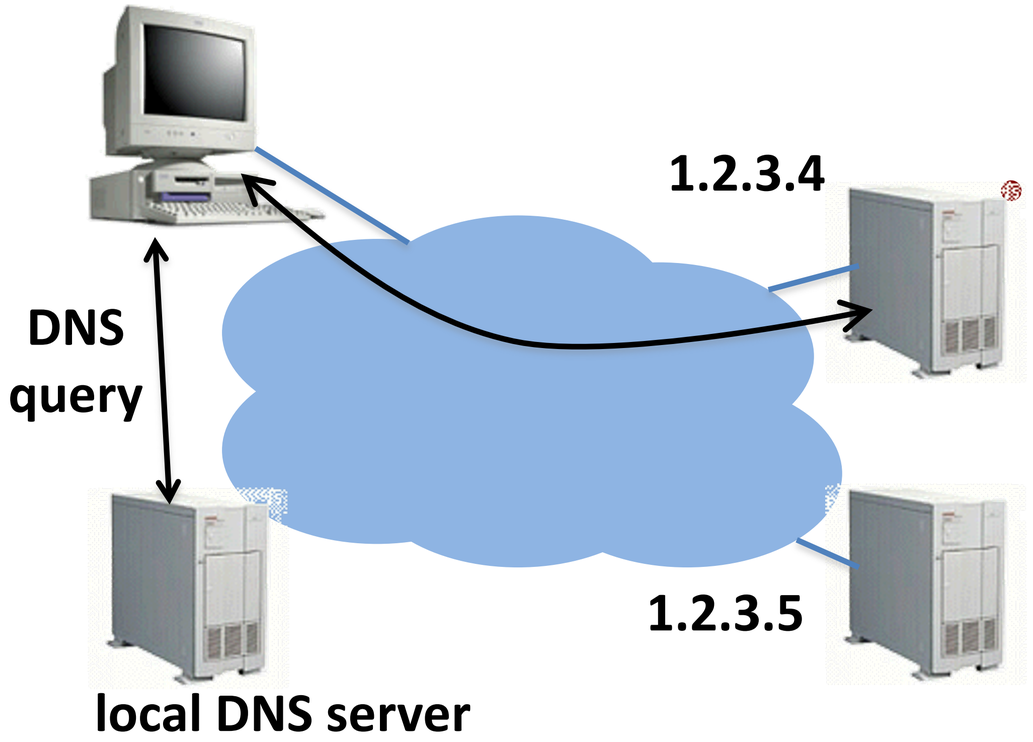
Server Selection Mechanism

- DNS-based selection
  - Many DNS implementations allow the Web site DNS to map a host domain name to a set of IP addresses and choose one of them for every query



Server Selection Mechanism

- DNS-based selection
- Advantages
  - Avoid TCP set-up delay
  - DNS caching reduces overhead
  - Relatively fine control
- Disadvantages
  - “Hidden load” effect
  - DNS TTL limits adaptation
  - Based on the IP address of the local DNS server



Server Selection Policy

- Live server
  - For availability
- Lowest load
  - To balance the load across the servers
- Closest
  - Nearest geographically, or in round-trip time
- Best performance
  - Throughput, latency, …
- Cheapest bandwidth, electricity, etc.

Outline

- Content delivery networks
- Overlay and P2P networks
- Distributed hash tables

Short Break - Let’s talk about music

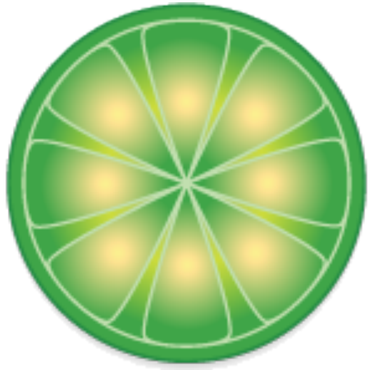
Short Break - Let's talk about music

NTNU

Short Break - Let’s talk about music

- P2P networks

That’s how people shared and could listen to music before Spotify, TIDAL, Apple Music, etc.



DISCLAIMER: The lecturer does not in any way condone the use of services that enable the illegal distribution of copyrighted content

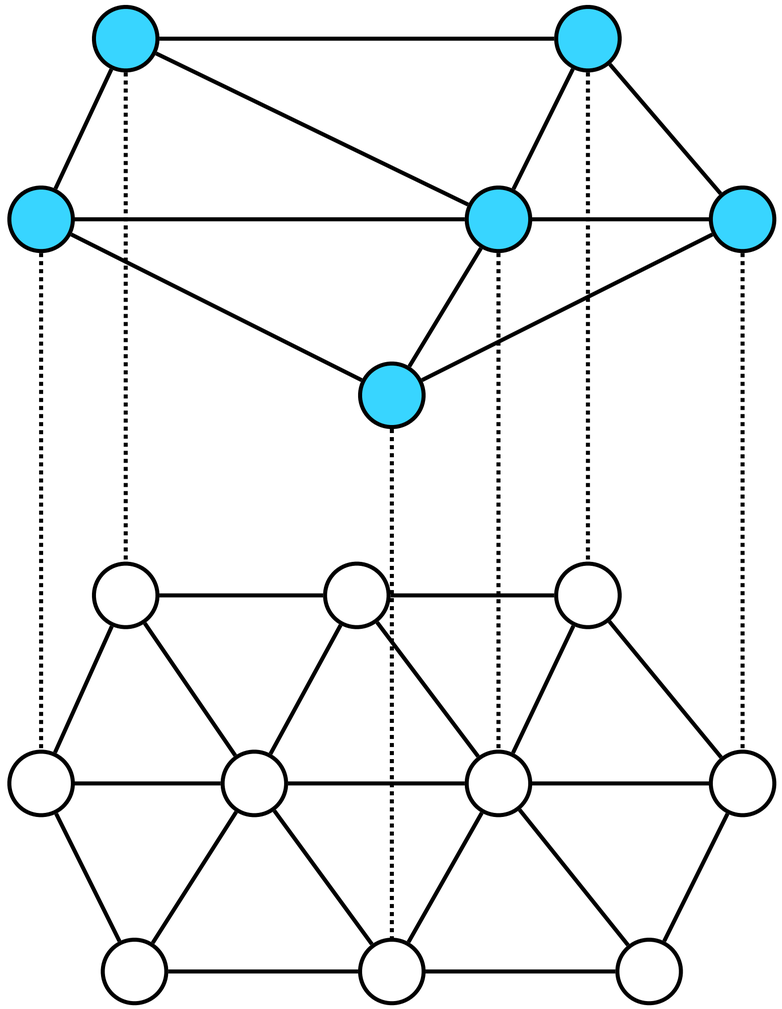
Overlay Networks

- You can think of an overlay as a logical network implemented on top of some underlying network
  - By this definition, the Internet started as an overlay network on top of the links provided by the old telephone network
- Each node in the overlay networks also exists in the underlying network
  - Nodes process and forward packets in an application-specific way
- The links that connect the overlay nodes are implemented as tunnels through the underlying network

Overlay Networks

Overlay network

Physical network



Overlay Networks

- The simplest kind of overlay is one that exists purely to support an alternative routing strategy
  - No additional application-level processing is performed at the nodes
- You can view a virtual private network as an example of a routing overlay
  - In this particular case, the overlay is said to use “IP tunnels”
  - The ability to utilize these VPNs is supported in many commercial routers

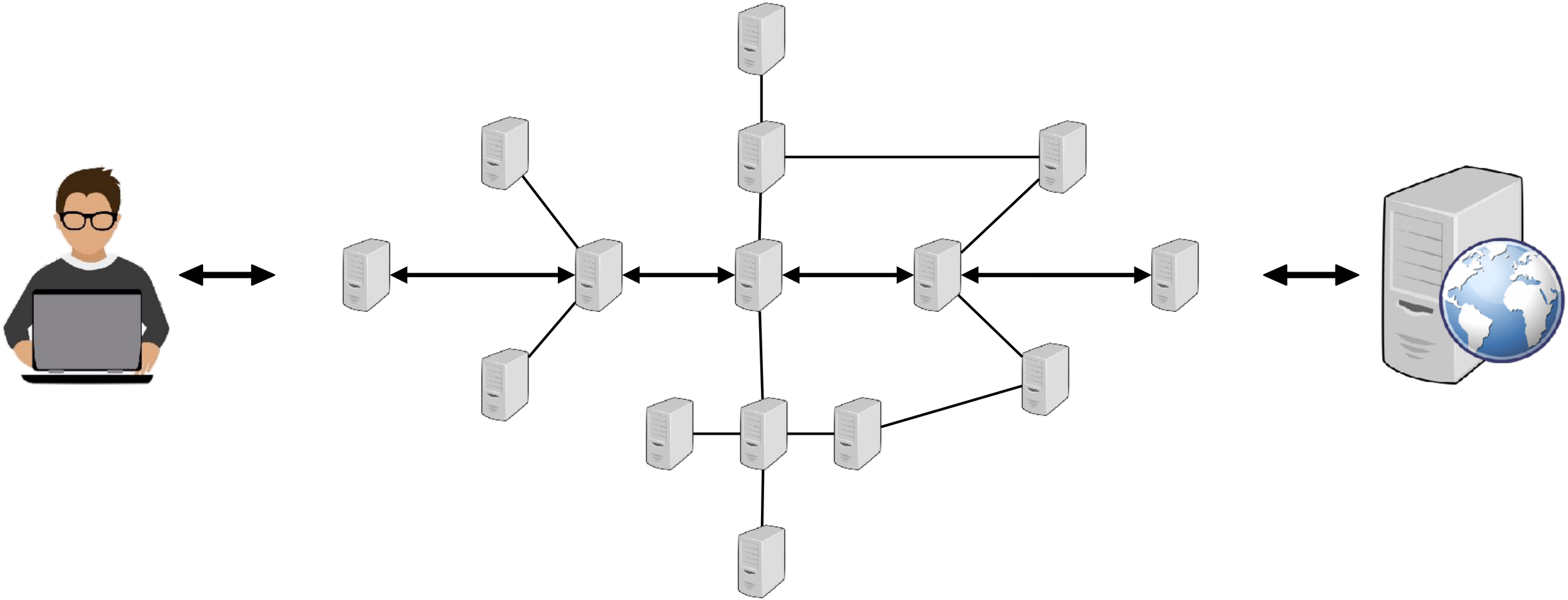
Overlay Networks

- Suppose, however, you wanted to use a routing algorithm that commercial router vendors were not willing to include in their products.
  - You could simply run your algorithm on a collection of end hosts, and tunnel through the Internet routers.
  - These hosts would behave like routers in the overlay network
    - As hosts they are probably connected to the Internet by only one physical link, but as a node in the overlay they would be connected to multiple neighbors via tunnels

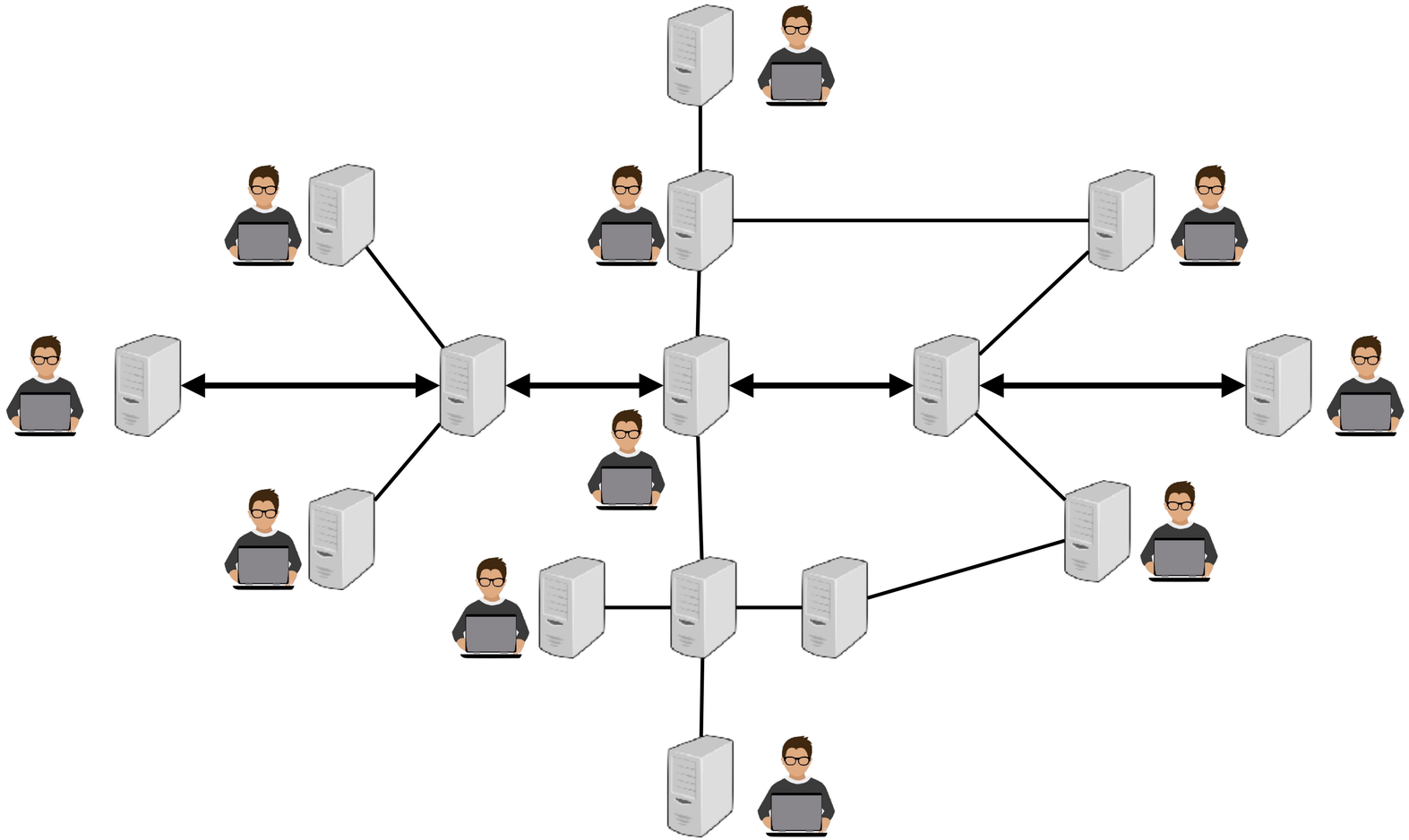
P2P Networks

- Fully decentralized system
  - There is no central authority to monitor the how the content is distributed to the users
- Self-organizing system
  - Individual nodes organize themselves into a network without any centralized coordination

The Centralized Way



The P2P Way



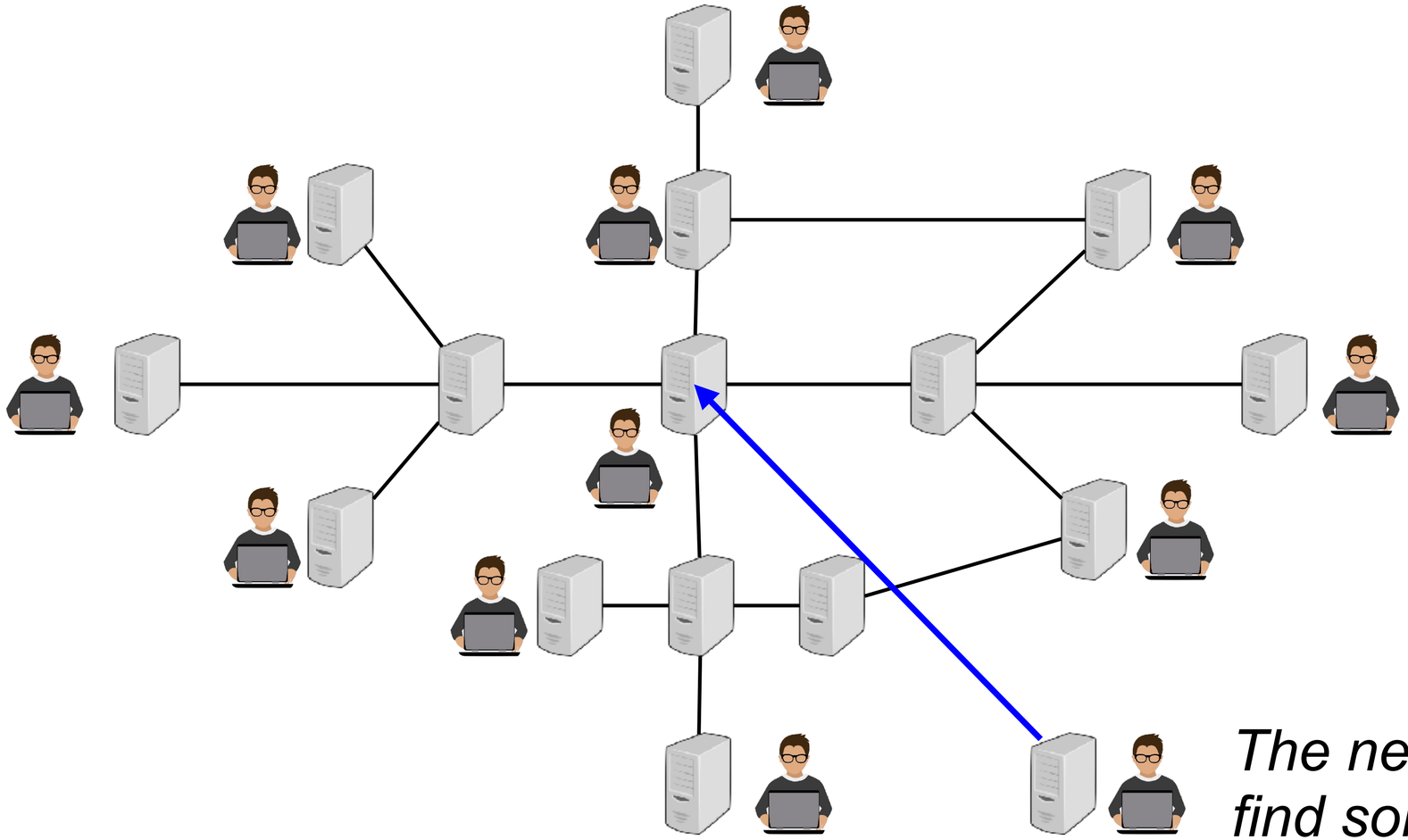
P2P Networks

- Every node acts as both client and server
  - both the process of locating an object of interest and the process of downloading that object onto your local machine happen without your having to contact a centralized authority
  - A peer-to-peer system that can accomplish these two tasks in a decentralized manner turns out to be an overlay network
  - the nodes are those hosts that are willing to share objects of interest
  - the links (tunnels) connecting these nodes represent the sequence of machines that you have to visit to track down the object you want

P2P Networks

- Gnutella
  - An early peer-to-peer network that attempted to distinguish between exchanging music and the general sharing of files
  - It was one of the first such systems to not depend on a centralized registry of objects
  - Gnutella participants arrange themselves into an overlay network
- On initial startup, the client software must bootstrap and find at least one other node

P2P Networks - Gnutella



The new node needs to find some existing node

P2P Networks

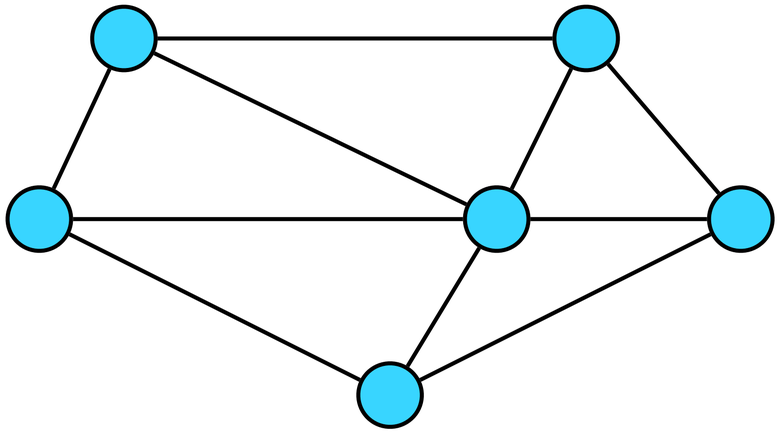
- Gnutella
  - An early peer-to-peer network that attempted to distinguish between exchanging music and the general sharing of files
  - It was one of the first such systems to not depend on a centralized registry of objects
  - Gnutella participants arrange themselves into an overlay network
- On initial startup, the client software must bootstrap and find at least one other node
  - There is no fixed rule on how the nodes are connects (unstructured overlay)

P2P Networks

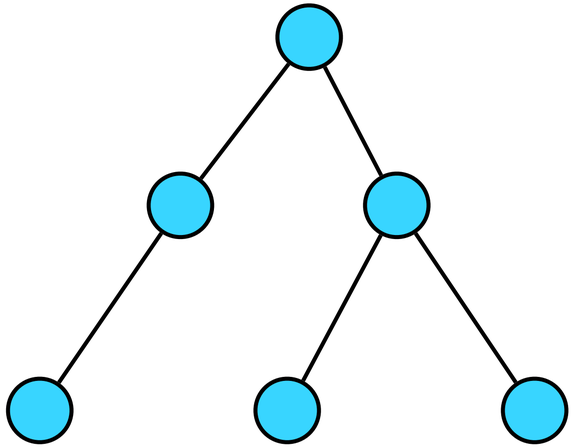
- Structured vs Unstructured
  - Unstructured overlay networks like Gnutella employ trivial overlay construction and maintenance algorithms, but the best they can offer is unreliable, random search
  - Structured overlay networks are designed to conform to a particular graph structure that allows reliable and efficient object location, in return for additional complexity during overlay construction and maintenance

P2P Networks - Example

- Structured vs Unstructured
  - How do we search for a file?



Unstructured



Structured

P2P Networks

- Structured vs Unstructured
  - Unstructured overlay networks like Gnutella employ trivial overlay construction and maintenance algorithms, but the best they can offer is unreliable, random search
  - Structured overlay networks are designed to conform to a particular graph structure that allows reliable and efficient object location, in return for additional complexity during overlay construction and maintenance
- How can we ensure that a fully decentralized system maintains some structure?

Outline

- Content delivery networks
- Overlay and P2P networks
- Distributed hash tables

Back to Basics

- What is a hash table?
- What is it good for?
- Key operations
  - Lookup (key) : value
  - Insert (key, value) pair
  - Delete (key)
- Question: Can we create a big distributed, Internet-scale Hash Table?

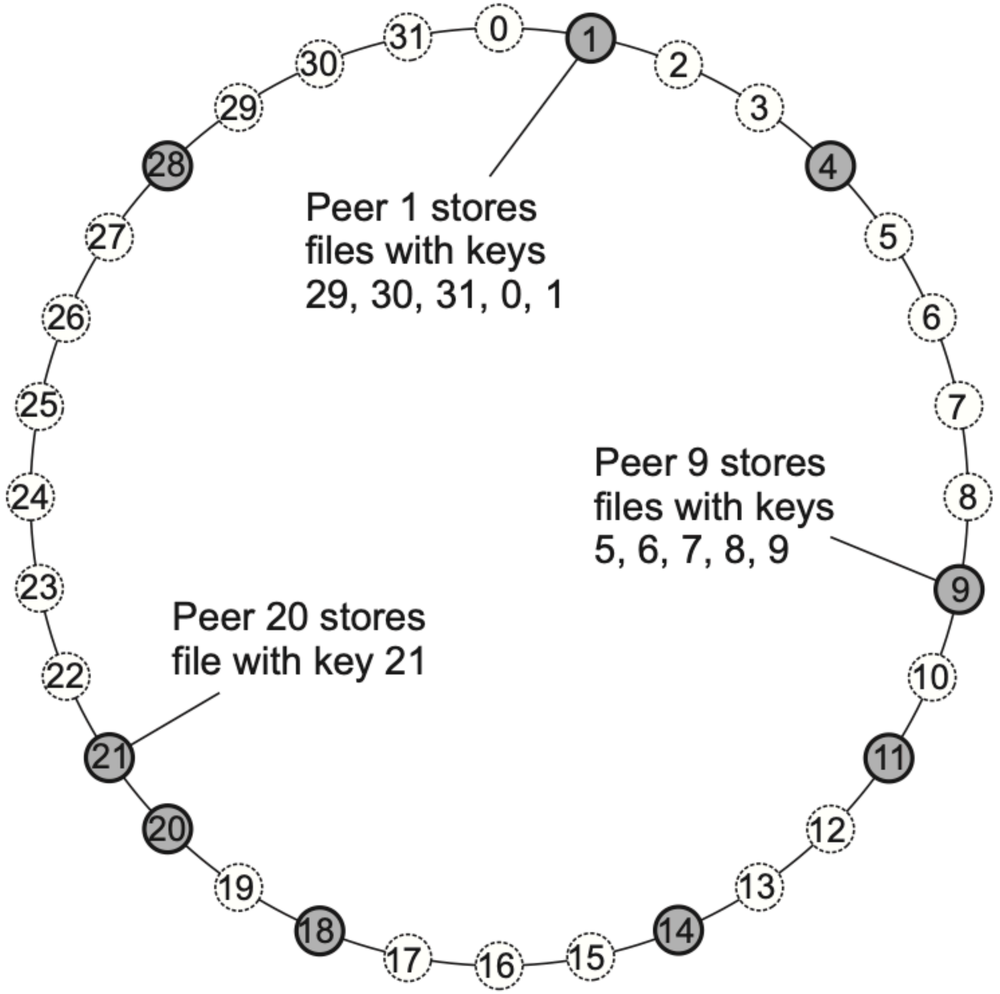
Chord - Main Ideas

- Given a key, Chord maps the key to a node
- Each node should maintain information for a few nodes
  - Chord achieves $O(logN)$
- Balance the load by distributing roughly evenly keys to nodes
- Involve little movement of keys when nodes join or leave the system
  - Chord achieves $O(log2N)$

Chord - Base Protocol

- Keys are ordered binary numbers of length m
- Nodes are also assigned an ID in the same number space
- Nodes are ordered in a ring according to their IDs
- For a given key k the responsible node n is the one with the smallest id larger than k, also called $successor(k)$

Chord - Base Protocol



Chord - Base Protocol

- Each node p maintains a finger table with at most m entries

$$FT_p[i] = succ(p + 2^{i-1})$$

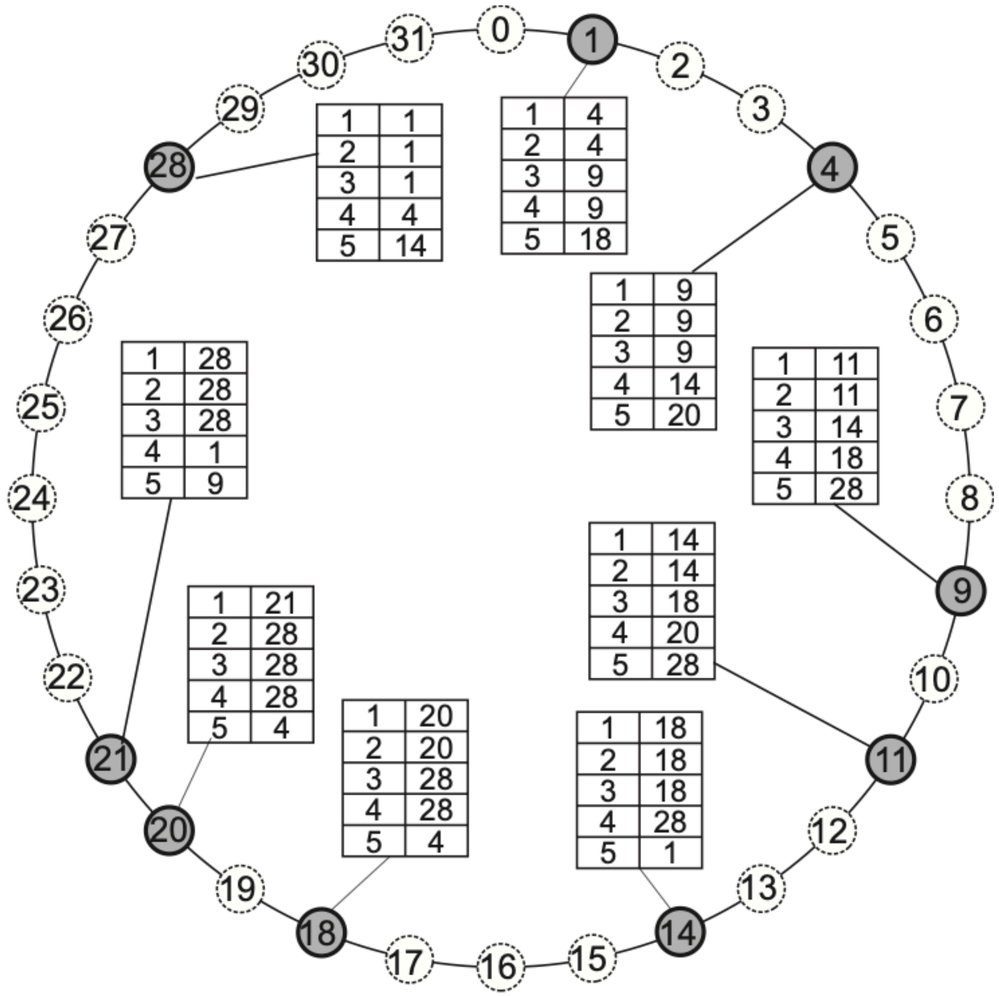
- To look up a key k, node p forwards the request to node with index j satisfying

$$q = FT_p[j] \leq k < FT_p[j+1]$$

- If $p < k < FT_p[1]$, the request is also forwarded to $FT_p[1]$

Example Lookup

- Key = 22
- @ node 4



$$FT_4[5] \leq 22 \Rightarrow 4 \rightarrow 20$$

$$FT_20[1] \leq 22 < FT_{20}[2] \Rightarrow 20 \rightarrow 21$$

$$21 < 22 < FT_{21}[1] \Rightarrow 21 \rightarrow 28$$

Chord - Node Insertion

- Initialize node n (the predecessor and the finger table)
  - The simplest approach is to execute find successor queries for all m entries, resulting in $O(m \cdot logN)$ initialization time
- Notify other nodes to update their predecessors and finger tables
- The new node takes over its responsible keys from its successor
  - To ensure correct lookups, a stabilization protocol is running periodically and updates finger tables and successor pointers

Chord - Node Deletion

- When a node leaves, it must transfer its content to its successor
- What if a node fails?
  - That’s what the stabilization protocol is for

Benefits of Chord

- Storage load balance
  - spread keys over nodes evenly
- Decentralization
  - fully distributed, no single point of failure
- Scalability
  - Chord lookup grows logarithmically in the number of nodes
- Availability
  - adjusts tables when nodes join/leave

END OF LECTURE