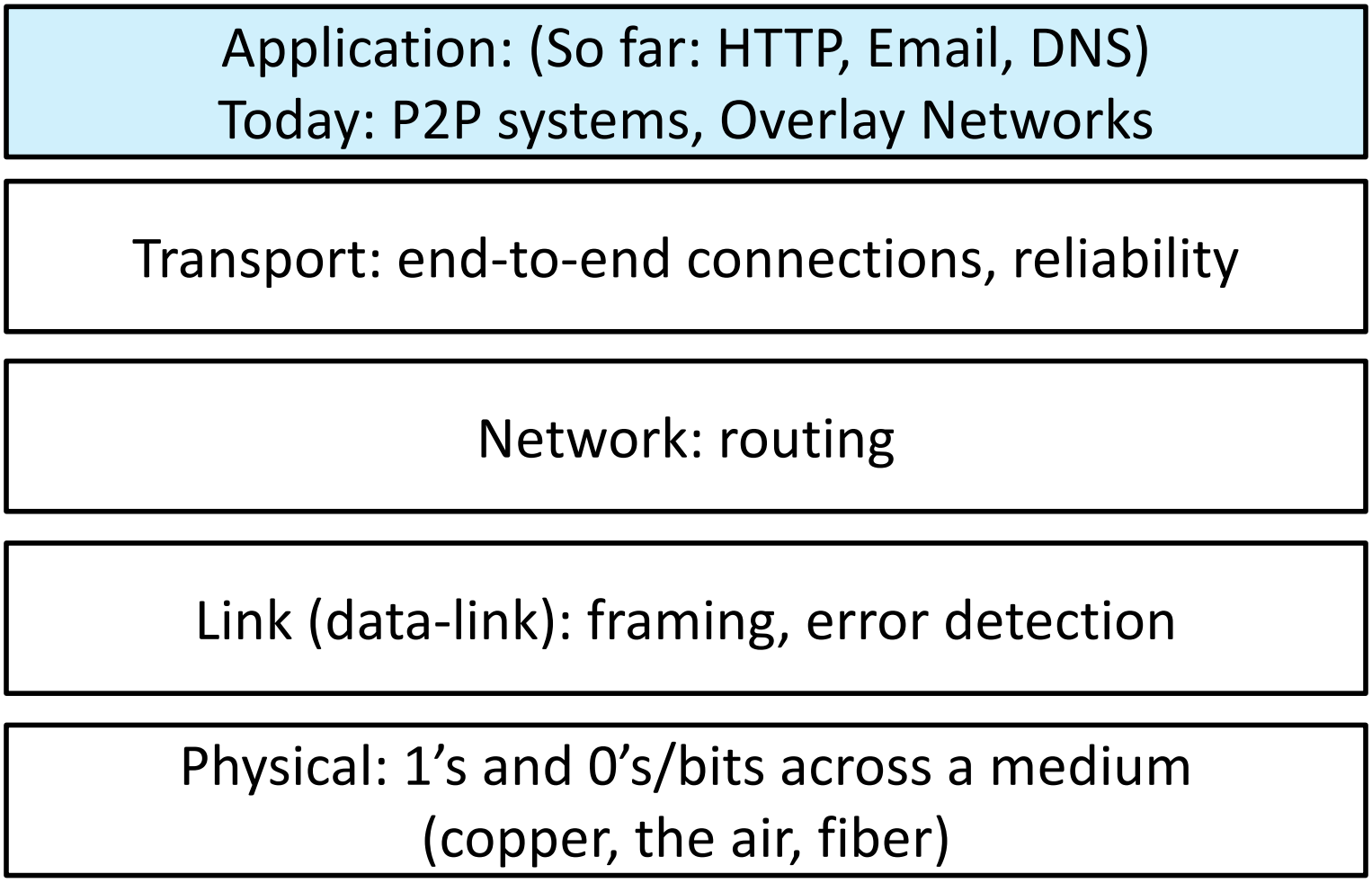
CS 43: Computer Networks

10: DHTs and CDNs

October 3, 2019

Where we are

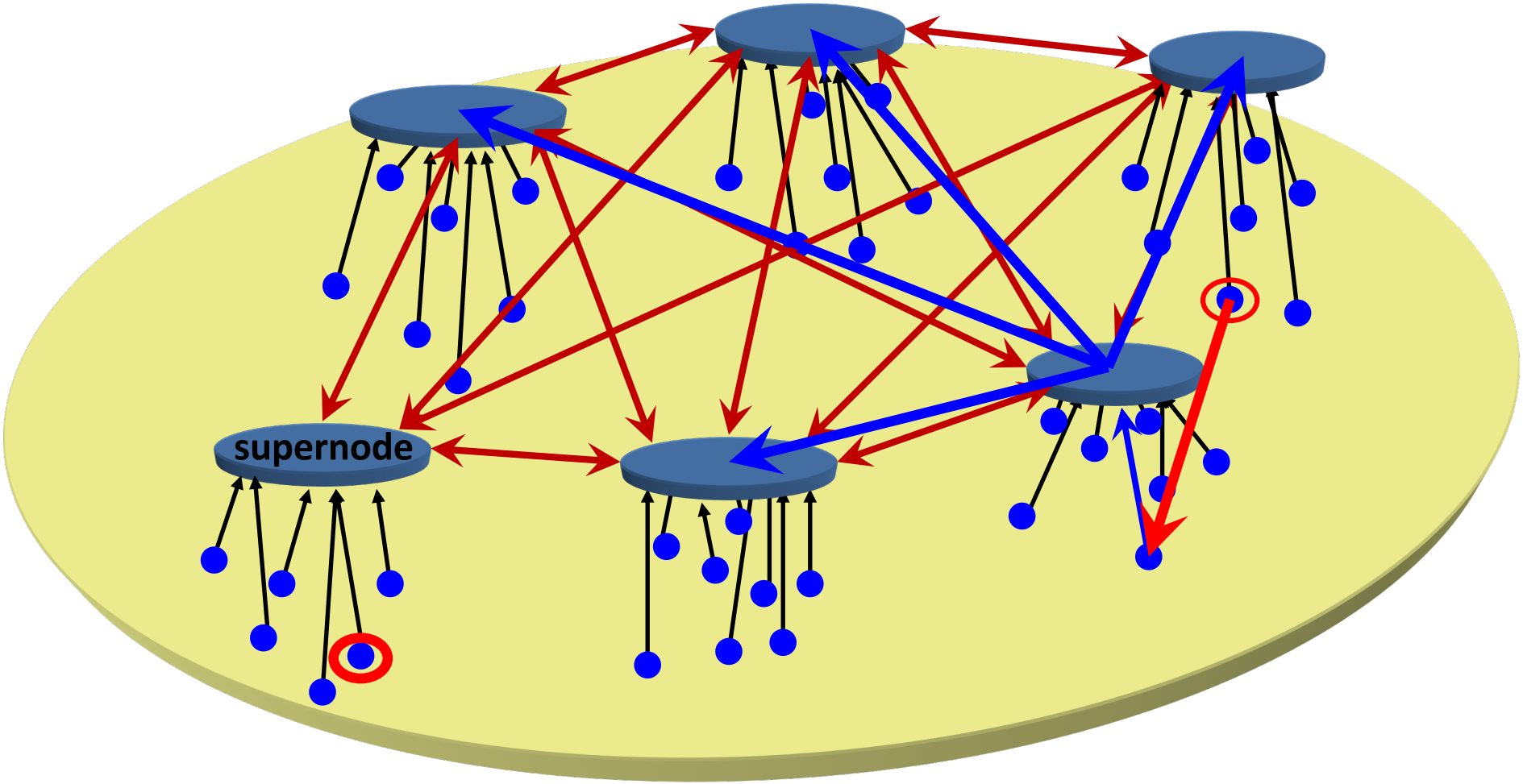

Application: (So far: HTTP, Email, DNS)
Today: P2P systems, Overlay Networks
Transport: end-to-end connections, reliability
Network: routing
Link (data-link): framing, error detection
Physical: 1's and 0's/bits across a medium
(copper, the air, fiber)

BitTyrant

- Piatek et al. 2007
  - Implements the “come in last strategy”
  - Essentially, an unfair unchoker
  - Faster than stock BitTorrent (For the Tyrant user)

Hierarchical P2P Networks

- FastTrack network (Kazaa, Grokster, Morpheus, Gnutella++)



Skype: P2P VoIP

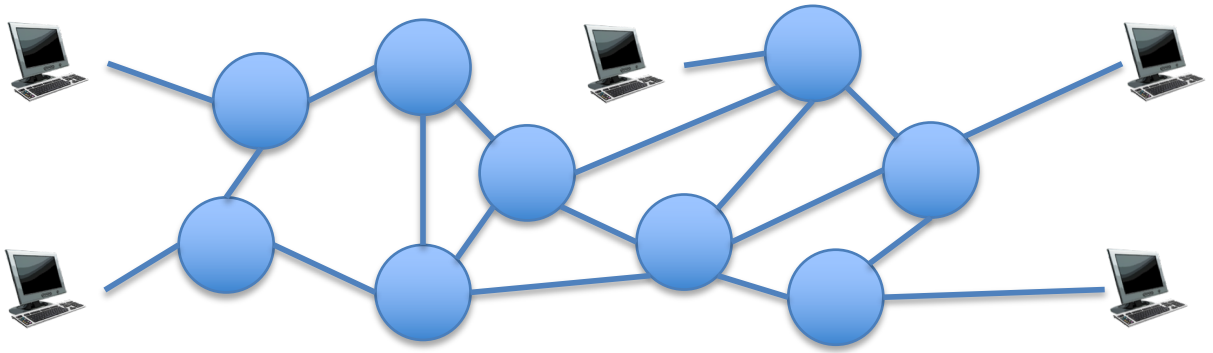
- P2P client supporting VoIP, video, and text based conversation, buddy lists, etc.
  - Overlay P2P network consisting of ordinary and Super Nodes (SN)
- Each user registers with a central server
  - User information propagated in a decentralized fashion

Do the benefits of hierarchical P2P networks out-weight the cons?

A. Pros: Scalability
B. Pros: Limits flooding
C. Cons: No guarantees of performance
D. Cons: Failure?

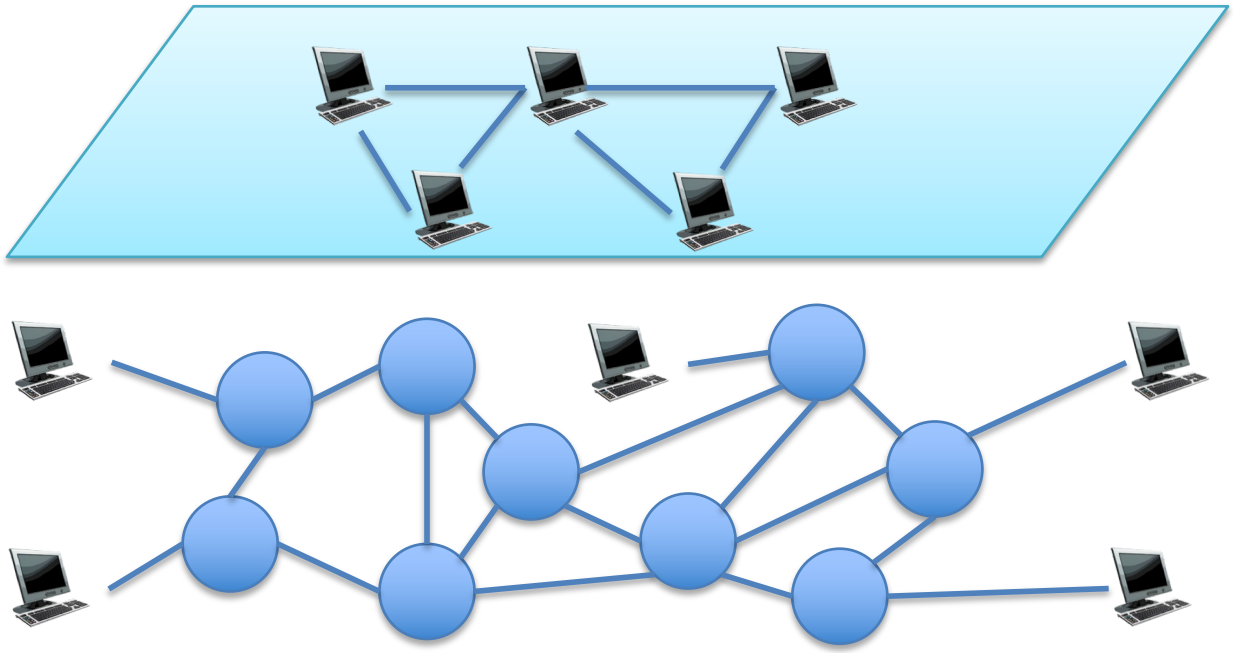
Overlay Network (P2P)

- A network made up of “virtual” or logical links
- Virtual links map to one or more physical links



Overlay Network (P2P)

- A network made up of “virtual” or logical links
- Virtual links map to one or more physical links



In our P2P examples with no central server, how would we maintain a mapping of content to nodes?

- Flooding each node and querying
- Maintaining an entire list at each node
- Some other system that scales

In our P2P examples with no central server, how would we maintain a mapping of content to nodes?

- Flooding each node and querying
- Maintaining an entire list at each node
- Some other system that scales (hint: where have we seen this before?)

Getting rid of that server…

- Distribute the tracker information using a Distributed Hash Table (DHT)

- A DHT is a lookup structure
  - Maps keys to an arbitrary value.
  - Works a lot like, well…a hash table.

Recall: Hash Function

- Mapping of any data to an integer
  - E.g., md5sum, sha1, etc.
  - md5: 04c3416cadd85971a129dd1de86cee49

- With a good (cryptographic) hash function:
  - Hash values very likely to be unique
  - Near-impossible to find collisions (hashes spread out)

Recall: Hash table

- N buckets
- Key-value pair is assigned bucket i
  - i = HASH(key)%N
- Easy to look up value based on key
- Multiple key-value pairs assigned to each bucket

Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*

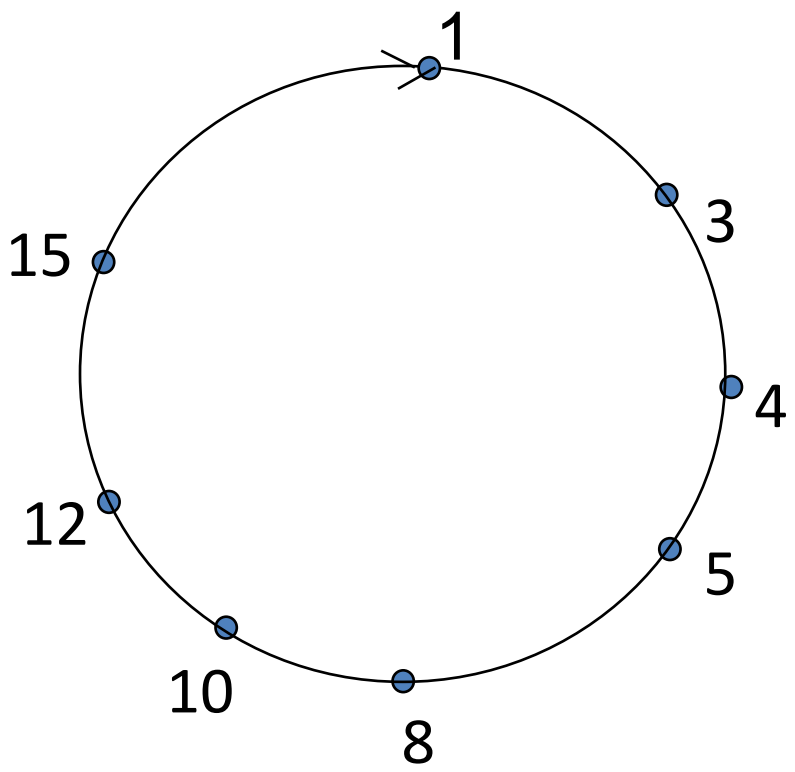
- Distribute the (k, v) pairs across the peers
  - key: ss number; value: human name
  - key: file name; value: BT tracker peer(s)

- Same interface as standard HT: (key, value) pairs
  - get(key) – send key to DHT, get back value
  - put(key, value) – modify stored value at the given key

Challenges

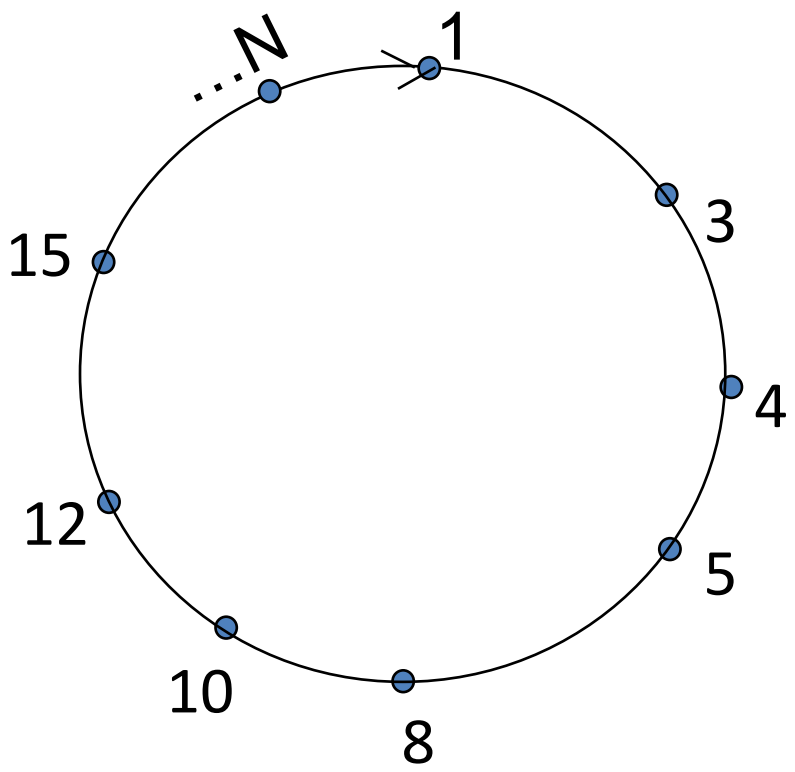
- How do we assign (key, value) pairs to nodes?
- How do we find them again quickly?
- What happens if nodes join/leave?
- Basic idea:
  - Convert each key to an integer via hash
  - Assign integer to each peer via hash
  - Store (key, value) pair at the peer closest to the key

Circular DHT Overlay



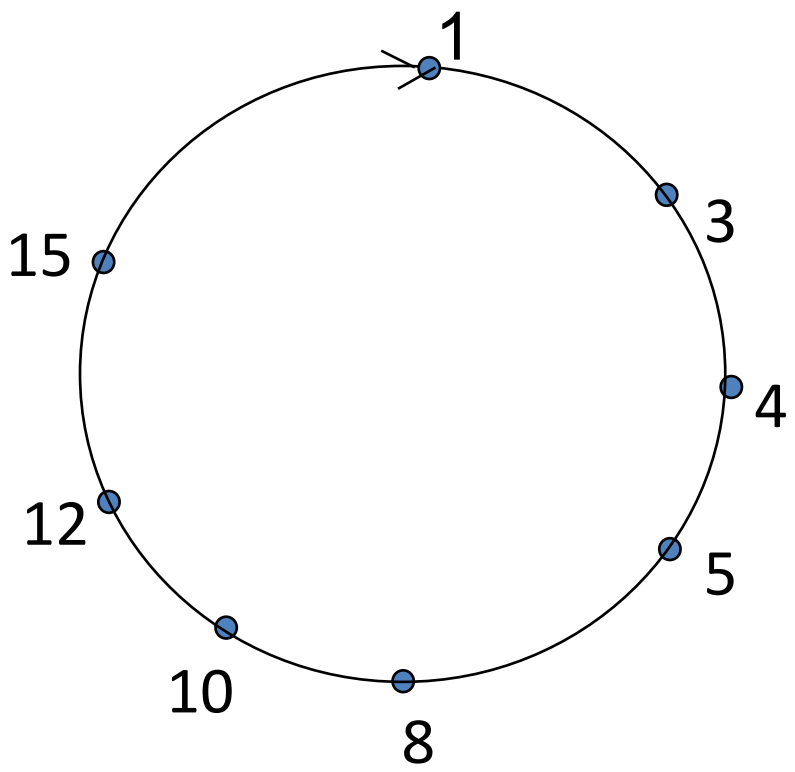
- Simplest form: each peer *only* aware of immediate successor and predecessor.

Circular DHT Overlay



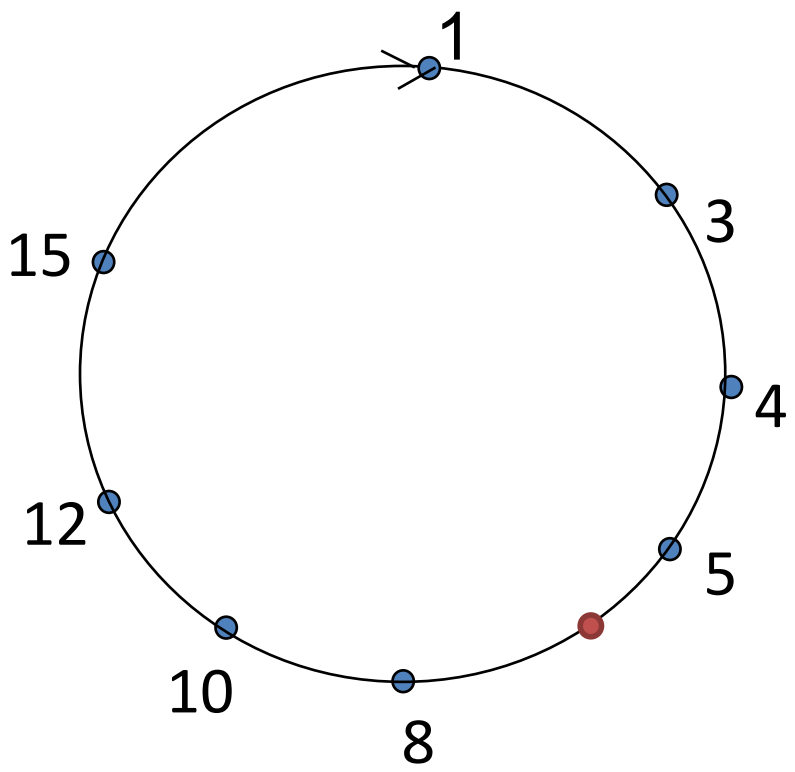
- Simplest form: each peer *only* aware of immediate successor and predecessor.

Circular DHT Overlay



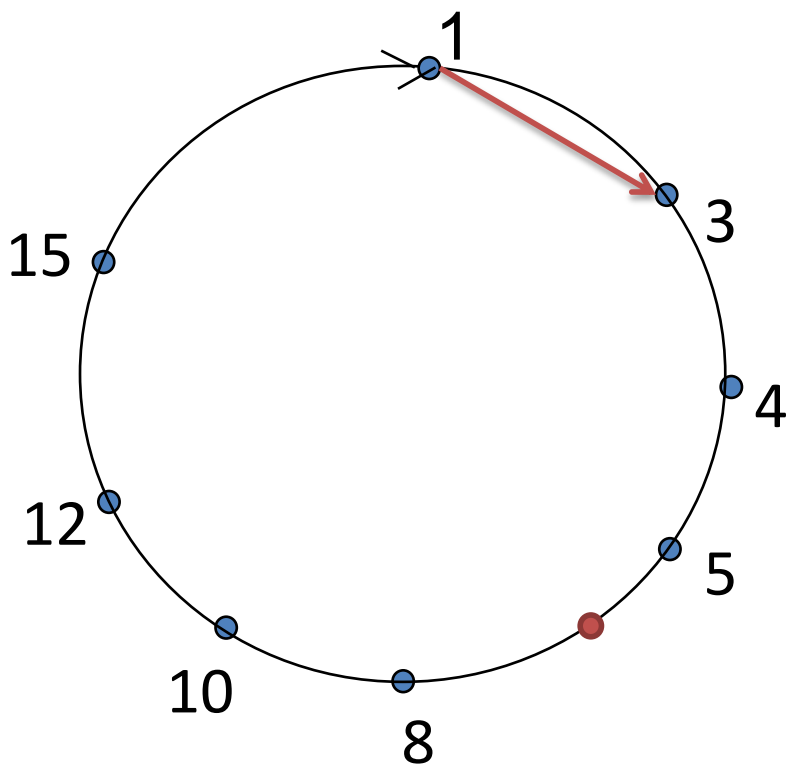
- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key

Circular DHT Overlay



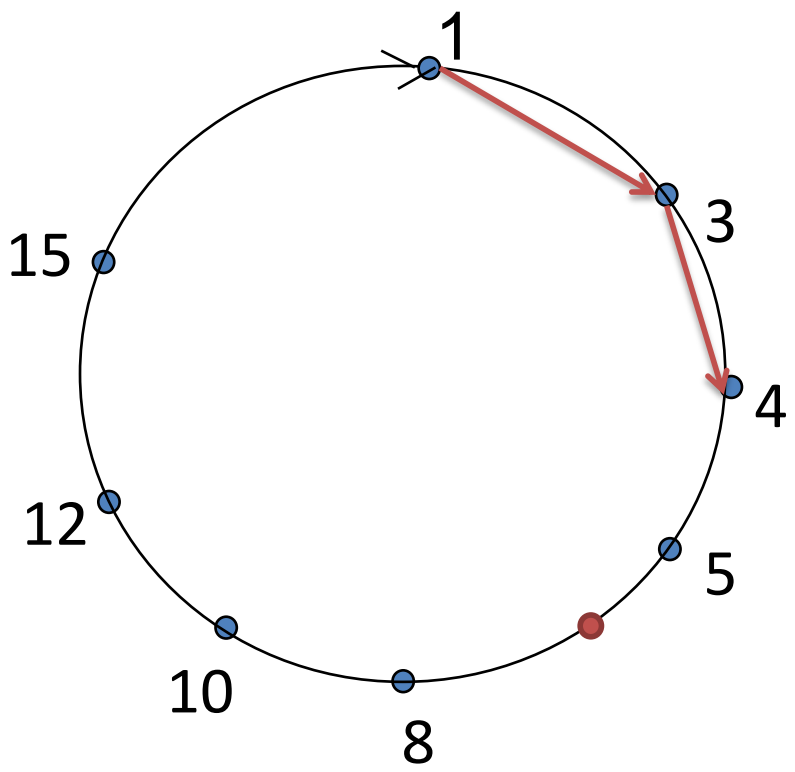
- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



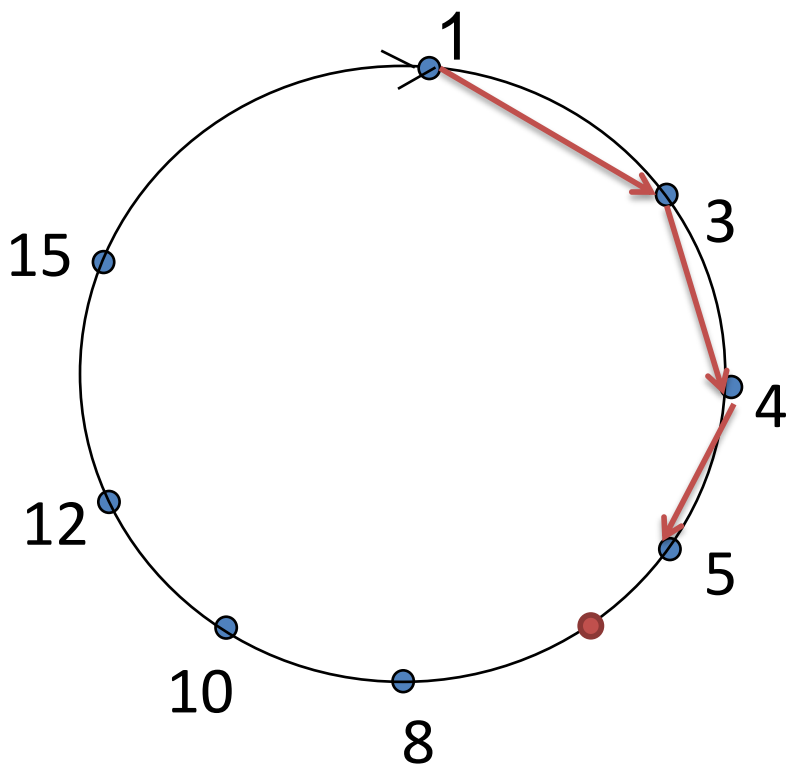
- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



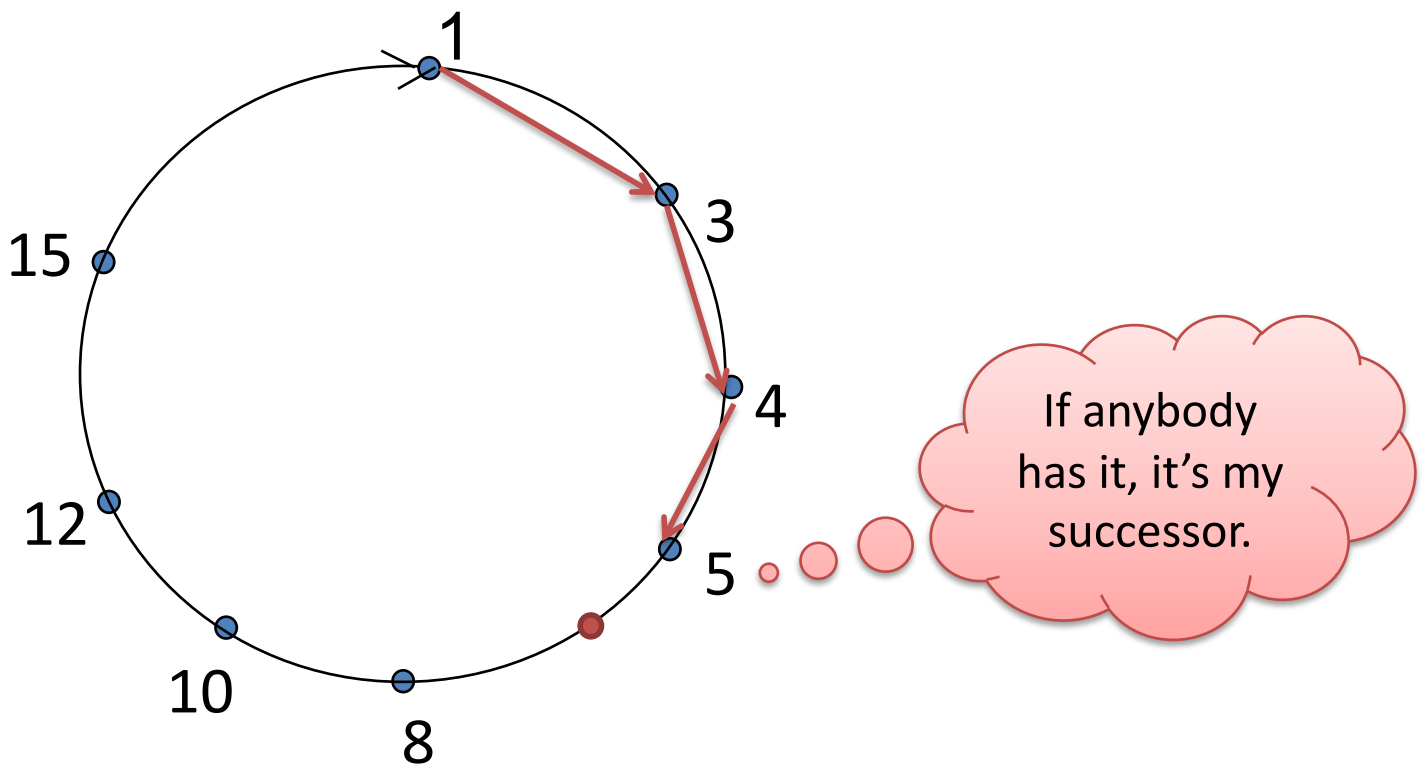
- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



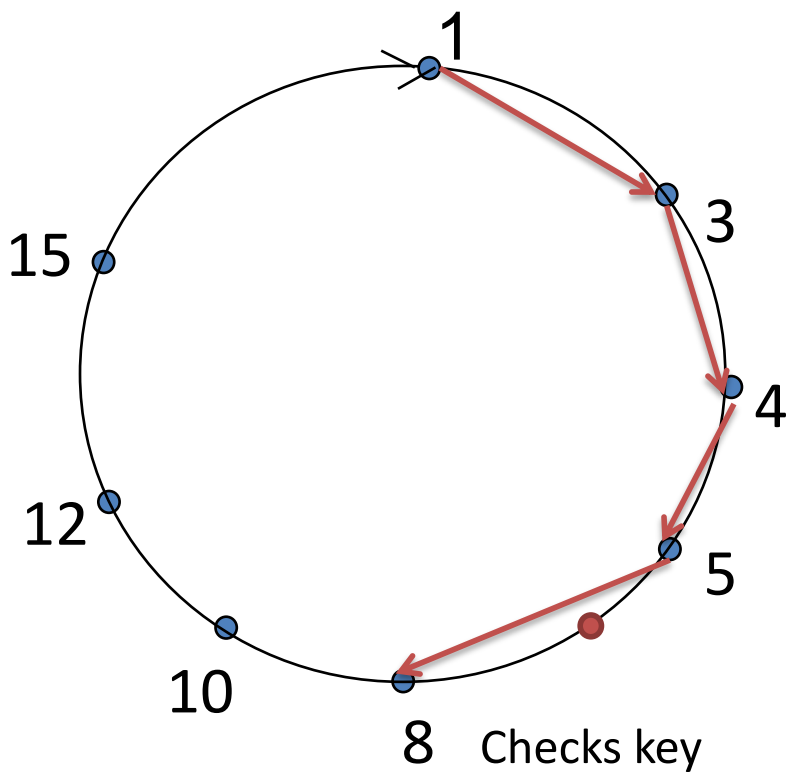
- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



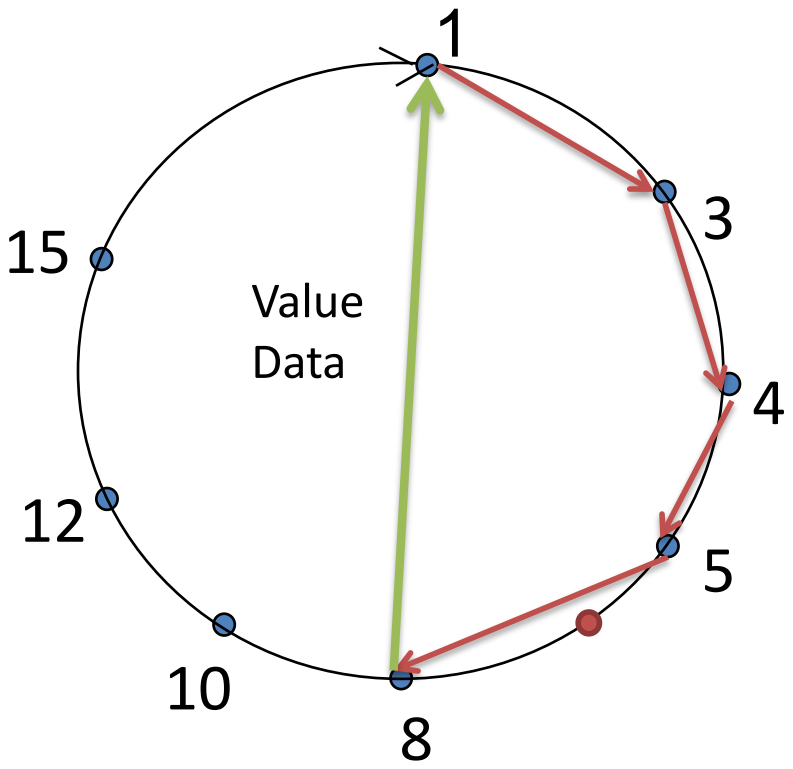
- Example: Node 1 wants key "Led Zeppelin IV"
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Circular DHT Overlay



- Example: Node 1 wants key “Led Zeppelin IV”
  - Hash the key (suppose it gives us 6)

Given N nodes, what is the complexity (number of messages) of finding a value when each peer knows its successor?

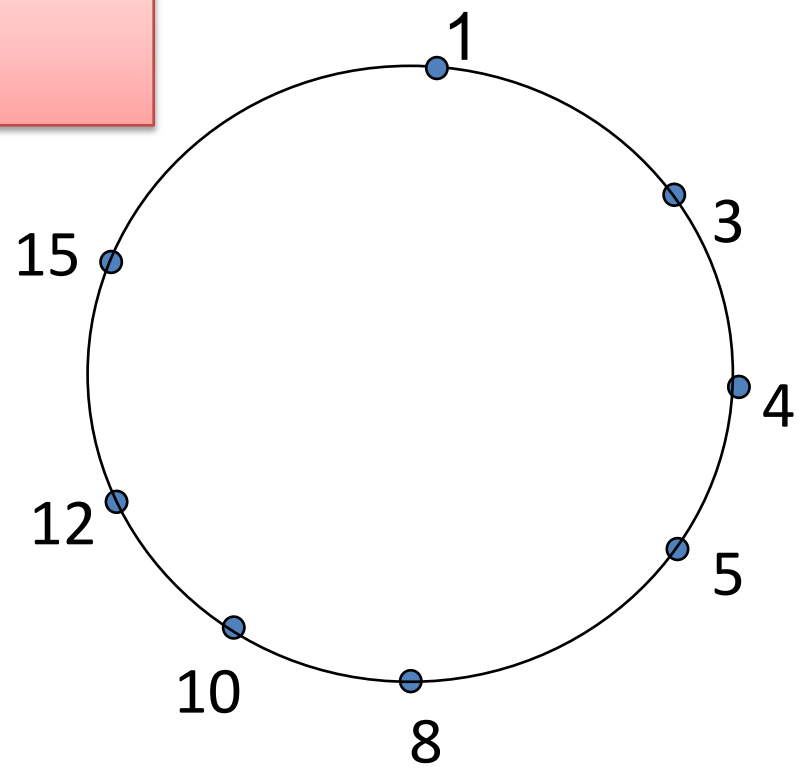
Can we do better? How?

A. O(log n)

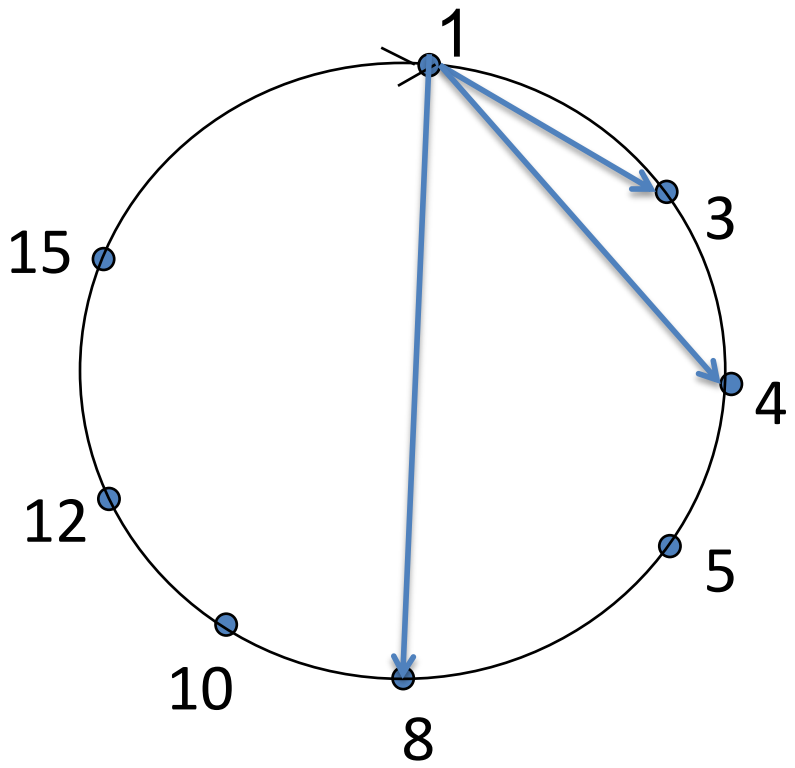
B. O(n)

C. $O(n^2)$

D. $O(2^n)$



Reducing Message Count



- Store successors that are 1, 2, 4, 8, …, N/2 away.
- Can jump up to half way across the ring at once.
- Cut the search space in half - lookups take O(log N) messages.

More DHT Info

- How do nodes join/leave?
- How does cryptographic hashing work?
- How much state does each node store?

More DHT Info

- How do nodes join/leave?
- How does cryptographic hashing work?
- How much state does each node store?

- Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications
- Dynamo: Amazon's Highly Available Key-value Store

High-Performance Content Distribution

- Problem:
  You have a service that supplies lots of data. You want good performance for all users!

(often “lots of data” means media files)

What is a CDN?

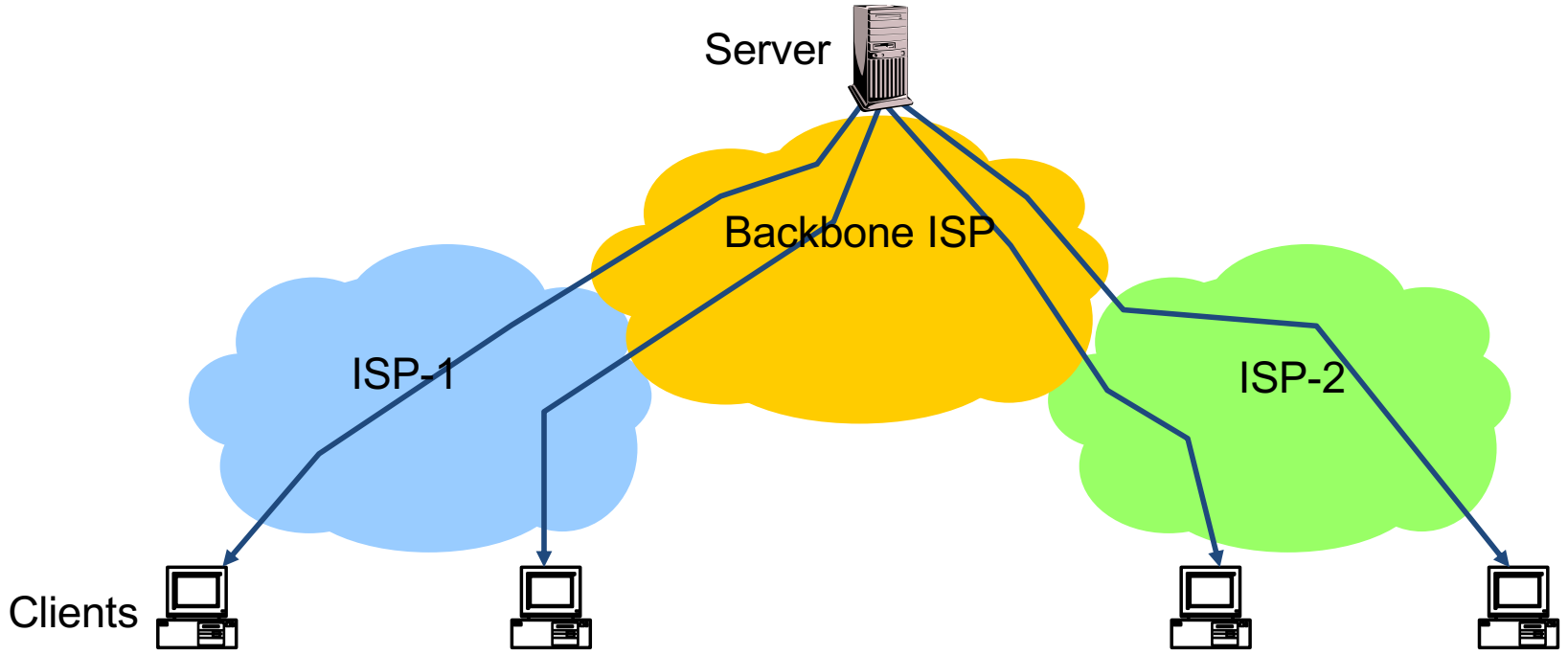
- Content Delivery/Distribution Network
  - At least 70% of the world's bits are delivered by a CDN!

What is a CDN?

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

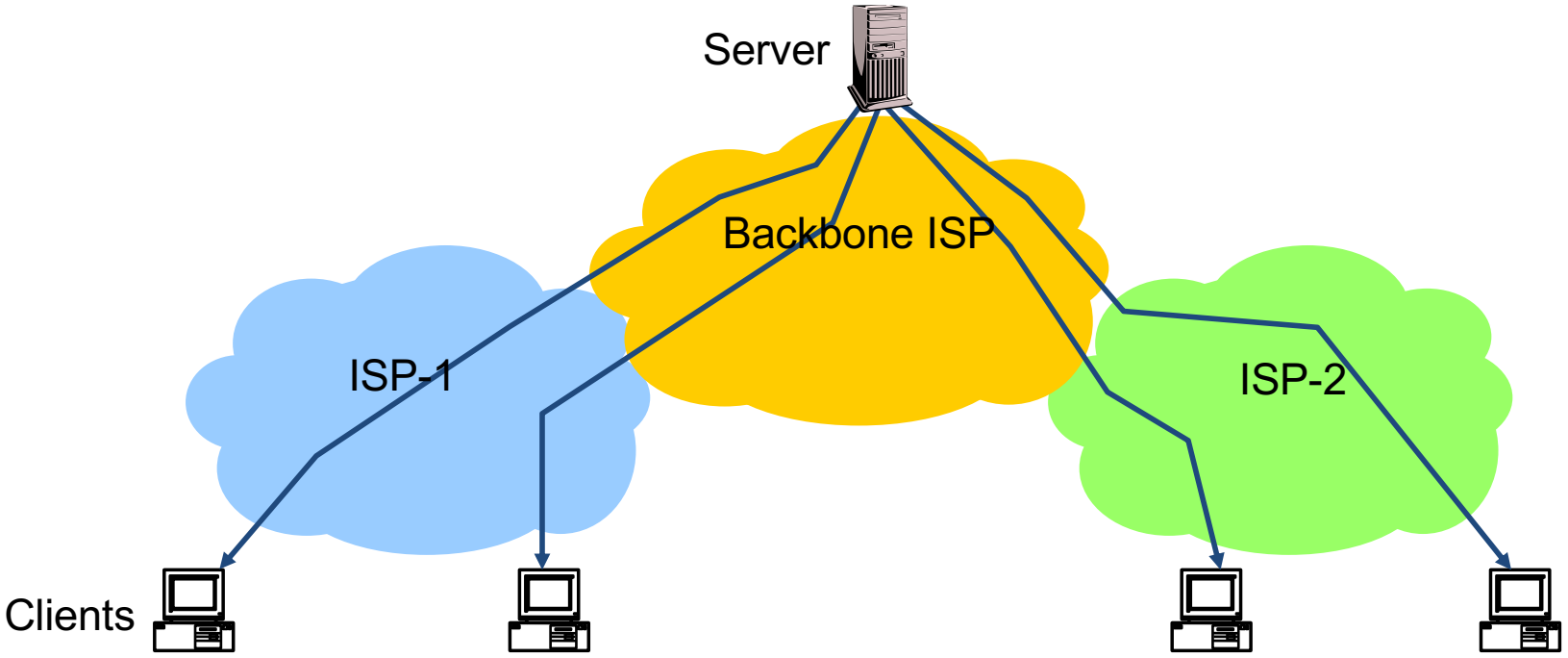
Where do we cache content in a CDN?

A. Client
B. Server
C. Internet Service Provider (ISP)



Caching

- Why caching works?
  - Locality of reference:
    - Users tend to request the same object in succession
    - Some objects are popular: requested by many users



High-Performance Content Distribution

- CDNs applied to all sorts of traffic.
  - You pay for service (e.g., Akamai), they'll host your content very "close" to many users.

CDN Challenges

- How do we direct the user to a nearby replica instead of the centralized source?
- How do we determine which replica is the best to send them to?

Key Components of a CDN

- Distributed servers
  - Usually located inside of other ISPs
  - Often located in IXPs (coming up next)
- High-speed network connecting them
- Clients (eyeballs)
  - Can be located anywhere in the world
  - They want fast Web performance
- Glue
  - Something that binds clients to “nearby” replica servers

Examples of CDNs

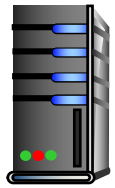
- Akamai
  - 147K+ servers, 1200+ networks, 650+ cities, 92 countries
- Limelight
  - Well provisioned delivery centers, interconnected via a private fiber-optic connected to 700+ access networks
- Edgecast
  - 30+ PoPs, 5 continents, 2000+ direct connections
- Others
  - Google, Facebook, AWS, AT&T, Level3, Brokers

Finding the CDN

- Three main options:
  - Application redirect (e.g., HTTP)
  - “Anycast” routing
  - DNS resolution (most popular in practice)

- Example: CNN + Akamai

CNN + Akamai

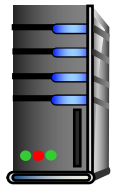


www.cnn.com

Request: cnn.com/article
Response: HTML with link to cache.cnn.com

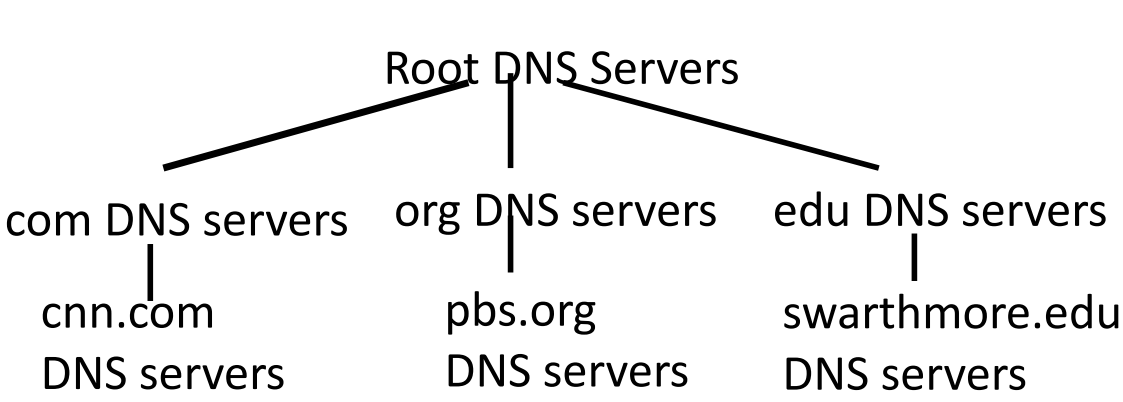
Content servers: serve content

CNN + Akamai



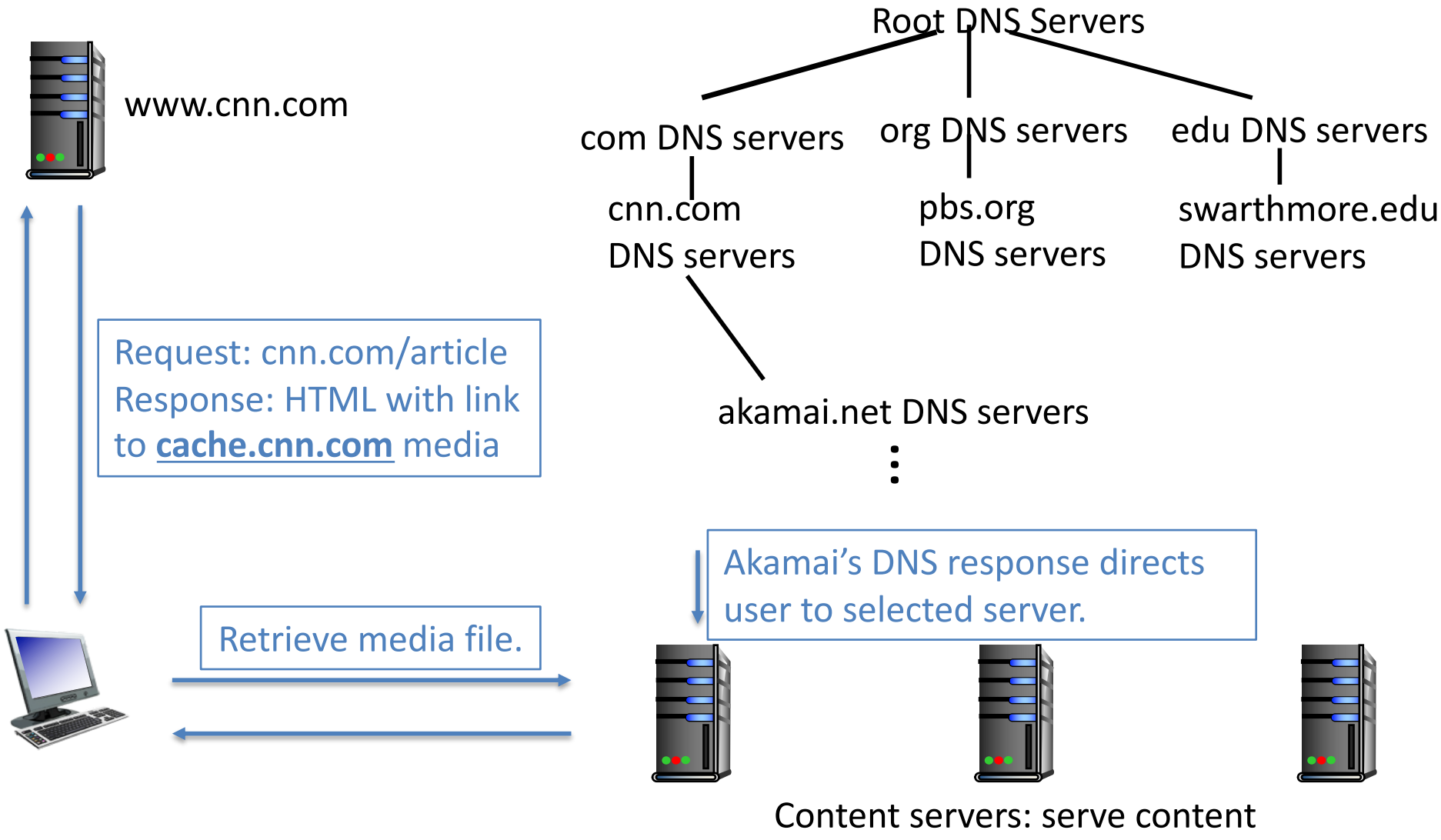
www.cnn.com

Request: cnn.com/article
Response: HTML with link to **cache.cnn.com** media

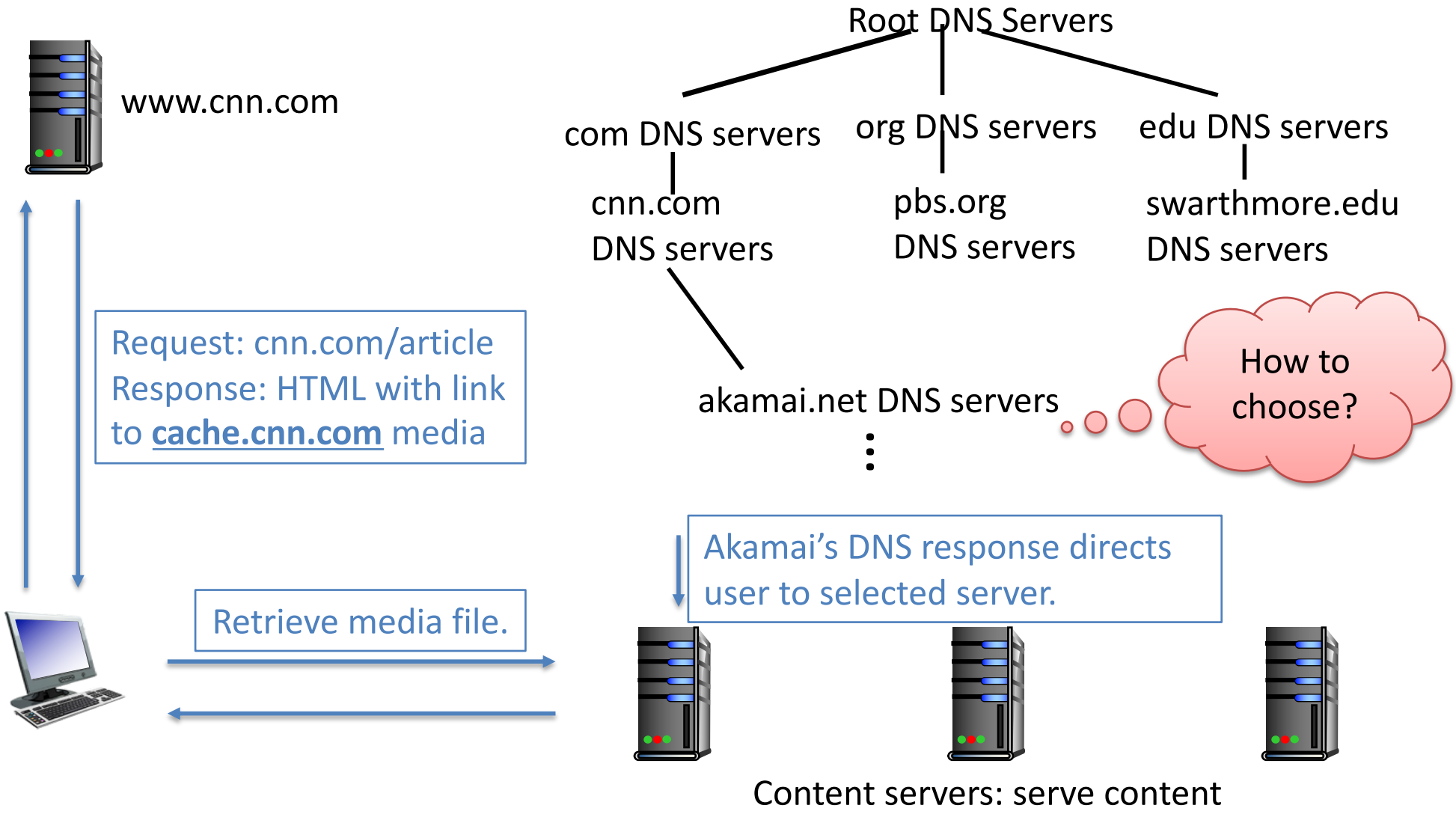


Content servers: serve content

CNN + Akamai


Root DNS Servers
com DNS servers
org DNS servers
edu DNS servers
cnn.com DNS servers
pbs.org DNS servers
swarthmore.edu DNS servers
akamai.net DNS servers
www.cnn.com
Request: cnn.com/article
Response: HTML with link to cache.cnn.com media
Retrieve media file.
Akamai's DNS response directs user to selected server.
Content servers: serve content

CNN + Akamai


Root DNS Servers
com DNS servers
org DNS servers
edu DNS servers
cnn.com DNS servers
pbs.org DNS servers
swarthmore.edu DNS servers
akamai.net DNS servers
How to choose?
www.cnn.com
Request: cnn.com/article
Response: HTML with link to cache.cnn.com media
Retrieve media file.
Akamai's DNS response directs user to selected server.
Content servers: serve content

Which metric is most important when choosing a server? (CDN or otherwise)

A. RTT latency

B. Data transfer rate / throughput

C. Hardware ownership

D. Geographic location

E. Some other metic(s) (such as?)

This is the CDN operator's secret sauce!

How well does caching work?

- Very well, up to a point
  - Large overlap in requested objects
  - Objects with one access place upper bound on hit ratio
  - Dynamic objects not cacheable*
- Example: Wikipedia
  - About 400 servers, 100 are HTTP Caches
  - 85% Hit ratio for text, 98% for media

Content in today's Internet

- Most flows are HTTP
  - Web is at least 52% of traffic
  - Median object size is 2.7K, average is 85K (as of 2007)

- Is the Internet designed for this common case?
  - Why?

Popping up: HTTP performance

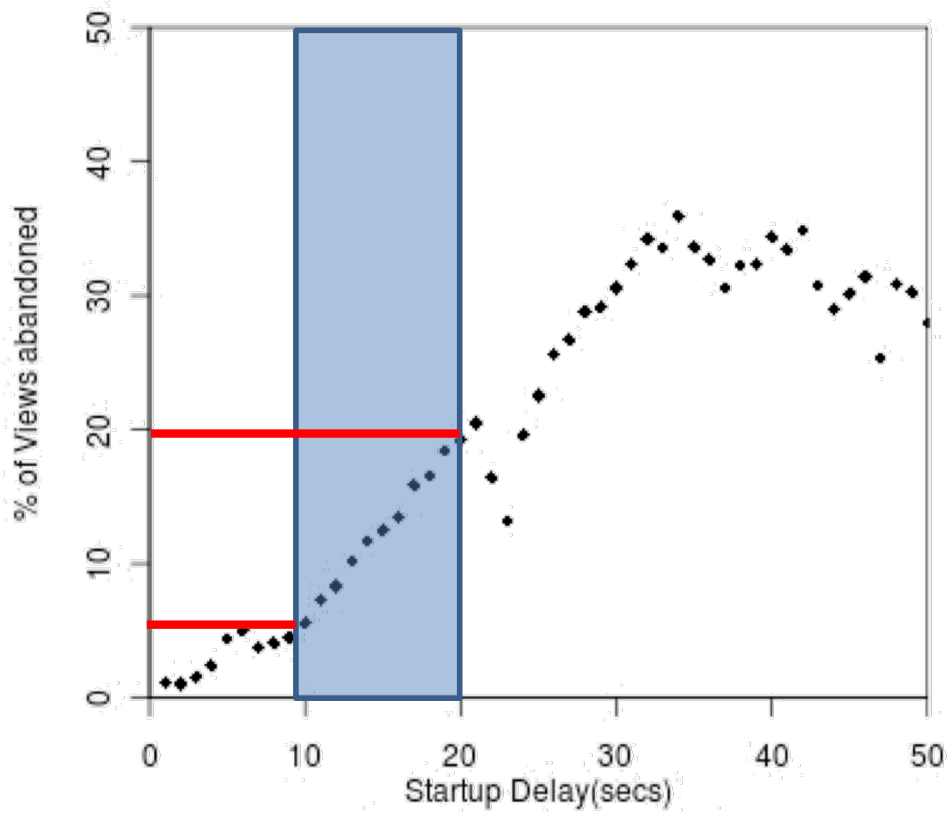
- For Web pages
  - RTT matters most
  - Where should the server go?
- For video
  - Available bandwidth matters most
  - Where should the server go?

- Is there one location that is best for everyone?

Why speed matters

- Impact on user experience
  - Users navigating away from pages
  - Video startup delay

- 4x increase in abandonment with 10s increase in delay



Streaming Media

- Straightforward approach: simple GET

- Challenges:
  - Dynamic network characteristics
  - Varying user device capabilities
  - User mobility

HTTP Performance

- What matters for performance?
- Depends on type of request
  - Lots of small requests (objects in a page)
  - Some big requests (large download or video)

Dynamic Adaptive Streaming over HTTP (DASH)

- Encode several versions of the same media file
  - low / medium / high / ultra quality
- Break each file into chunks
- Create a “manifest” to map file versions to chunks / video time offset

Dynamic Adaptive Streaming over HTTP (DASH)

- Client requests manifest file, chooses version
- Requests new chunks as it plays existing ones
- Can switch between versions at any time!

Summary

- Decentralized lookup: DHTs
- CDNs: locating “good” replica for content servers
- DASH: streaming despite dynamic conditions