CDN and P2P

To do …

- CDNs
- P2P
- Hybrid CDN+P2P

Northwestern

Network trends and application need

- Some clear trends
  - Growing number of networks
  - Faster networks
  - Growing availability and demand for content
- For applications, higher demand on performance and reliability
  - Small degradation are expensive in lost revenue
    - $2.8m/hour in 2009
  - … damage reputation
  - … reduced productivity

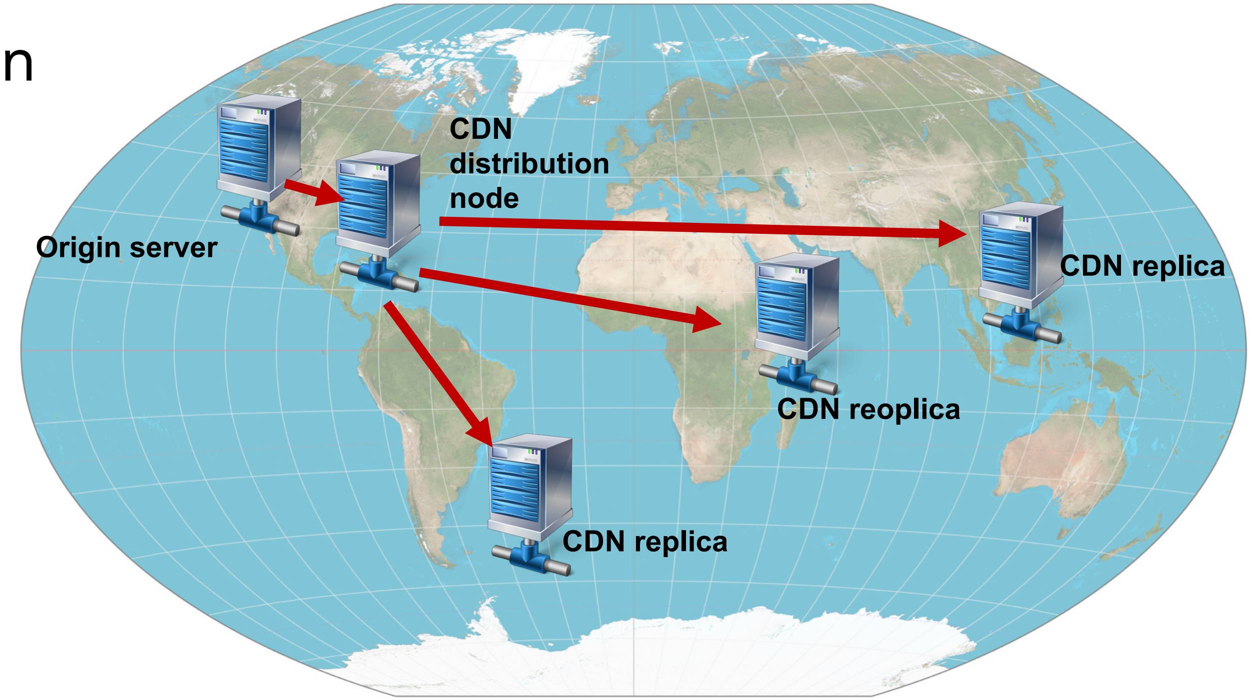
Content delivery

- The common answer
  - Replicate content around the world, closer to users
  - Bring users to nearby content, “nearby” in a network sense

- Challenges
  - How to replicate content
  - Where to replicate it
  - How to choose among known replicas
  - How to direct clients toward a replica

Content Distribution Network

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



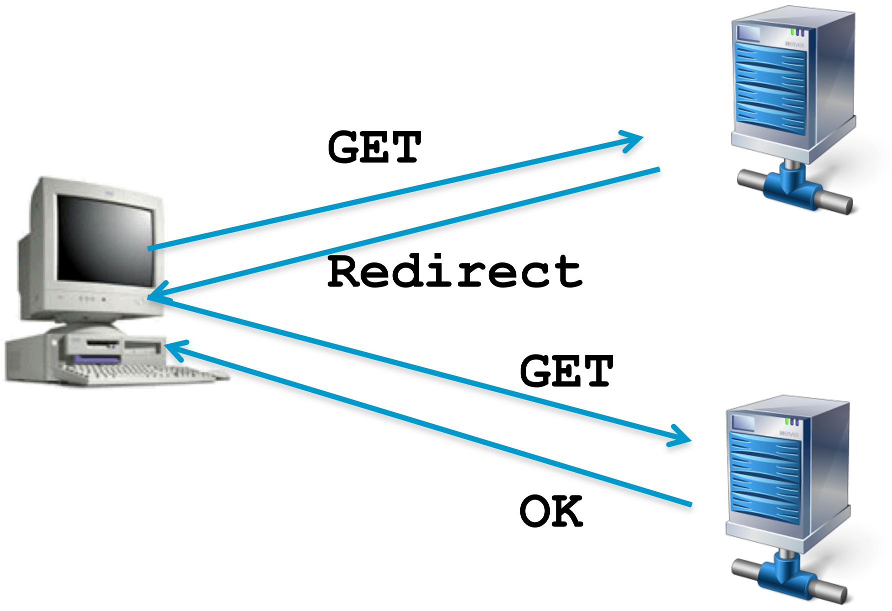
Server selection policy

- Live server
  - For availability
- Lowest load
  - Balancing load across servers
- Closest
  - Nearest geographically, or in round-trip time
- Best performance
  - Throughput, latency, …
- Cheapest bandwidth, electricity, pollution, …

Requires continuous monitoring of liveness, load, and performance

Server selection mechanism

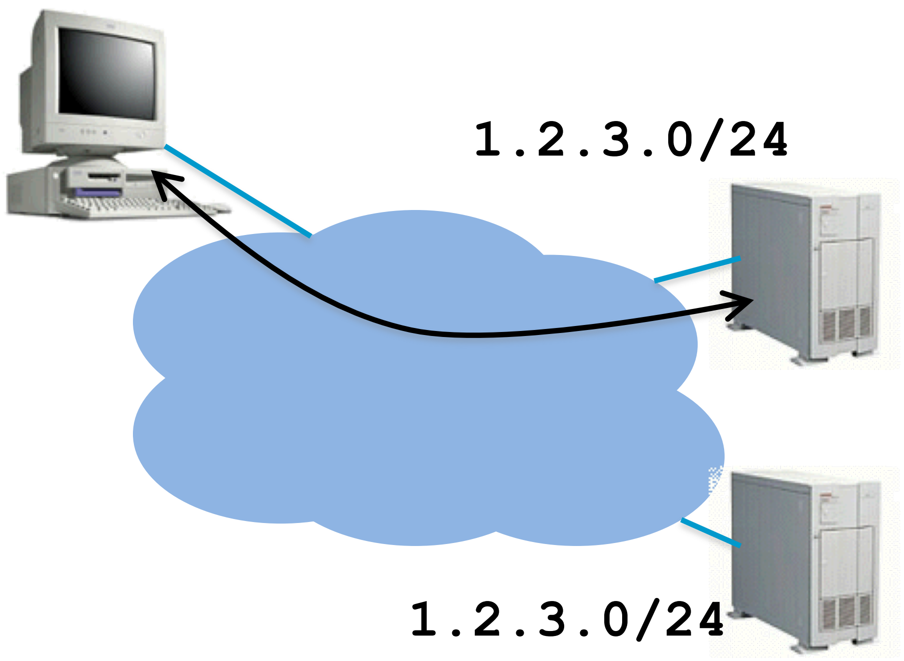
- Application
  - URL redirection (HTTP 3xx)



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

Server selection mechanism

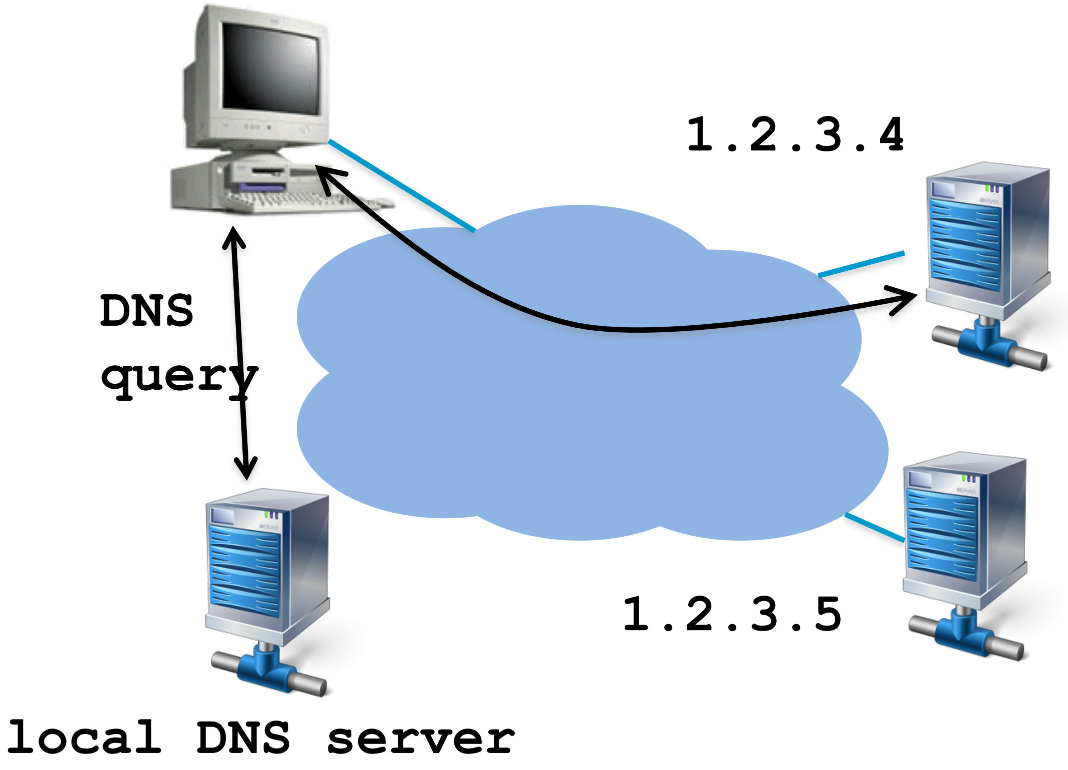
- Routing
  - Anycast routing



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

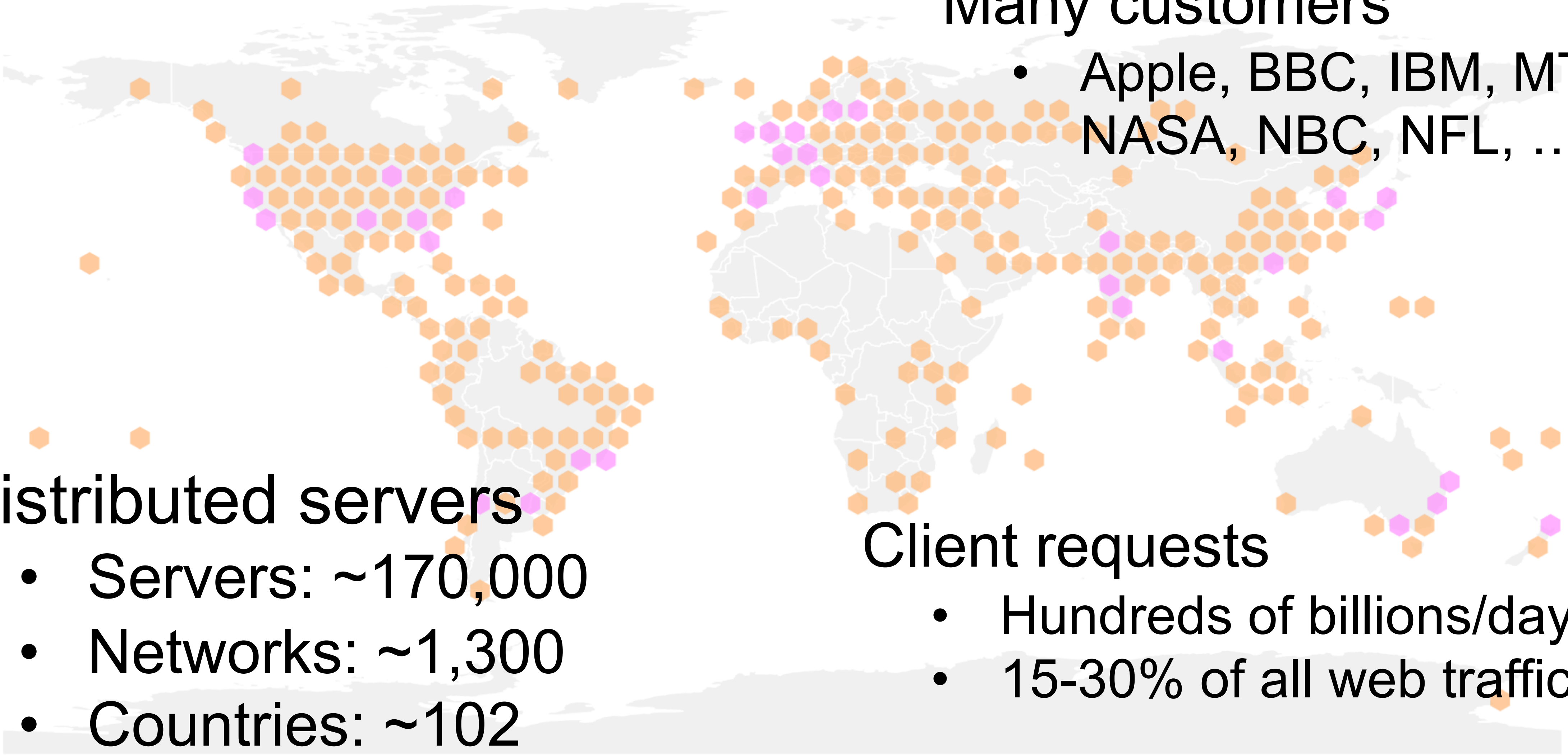
Server selection mechanism

- Naming
  - DNS-based server selection



- Advantages
  - Avoid TCP set-up delay
  - DNS caching reduces overhead
  - Relatively fine control
- Disadvantage
  - Based on IP address of local DNS server / recursive resolver
  - “Hidden load” effect
  - DNS TTL limits adaptation

Akamai as an example



Many customers

- Apple, BBC, IBM, MTV, NASA, NBC, NFL, …

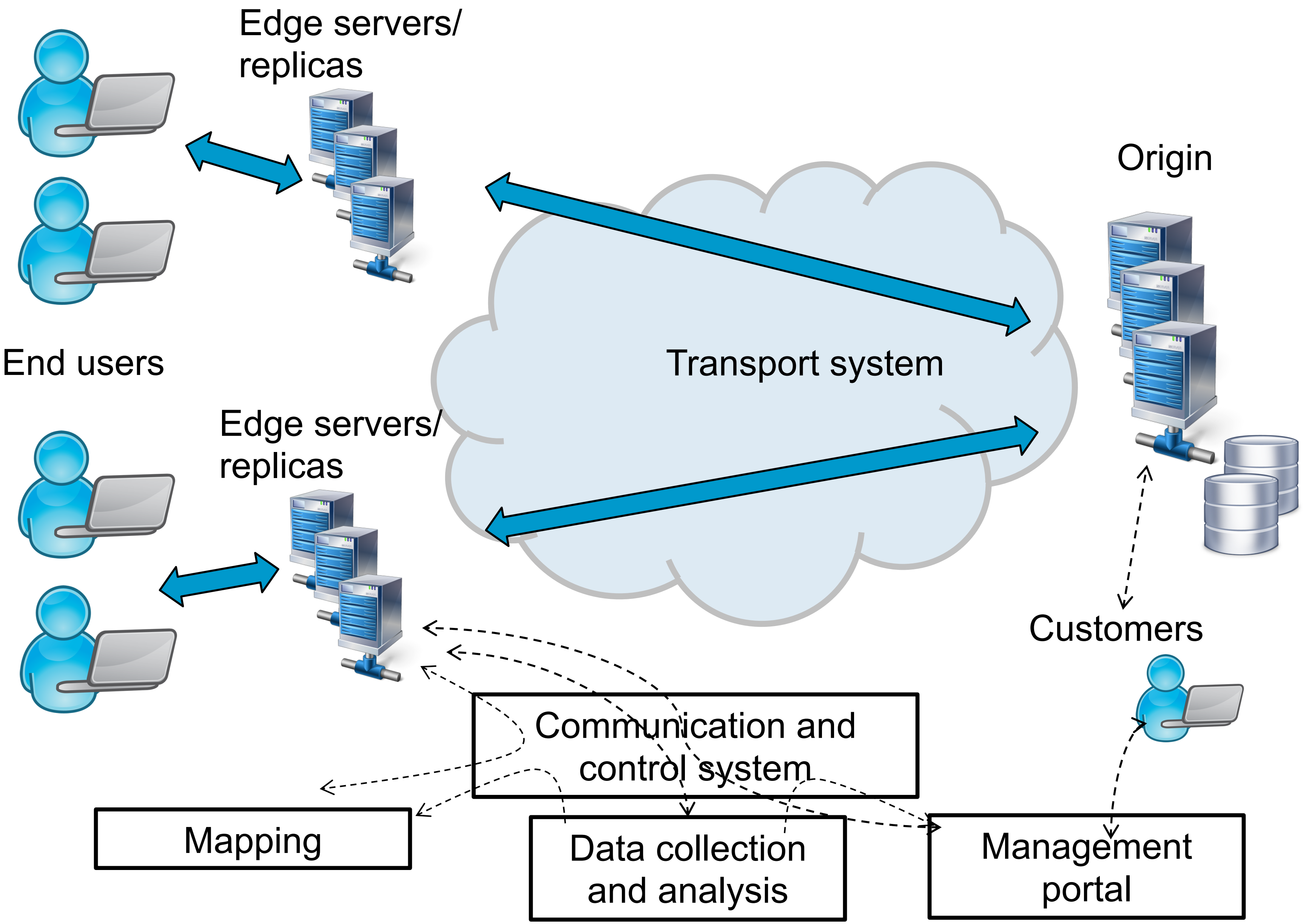
Distributed servers

- Servers: ~170,000
- Networks: ~1,300
- Countries: ~102

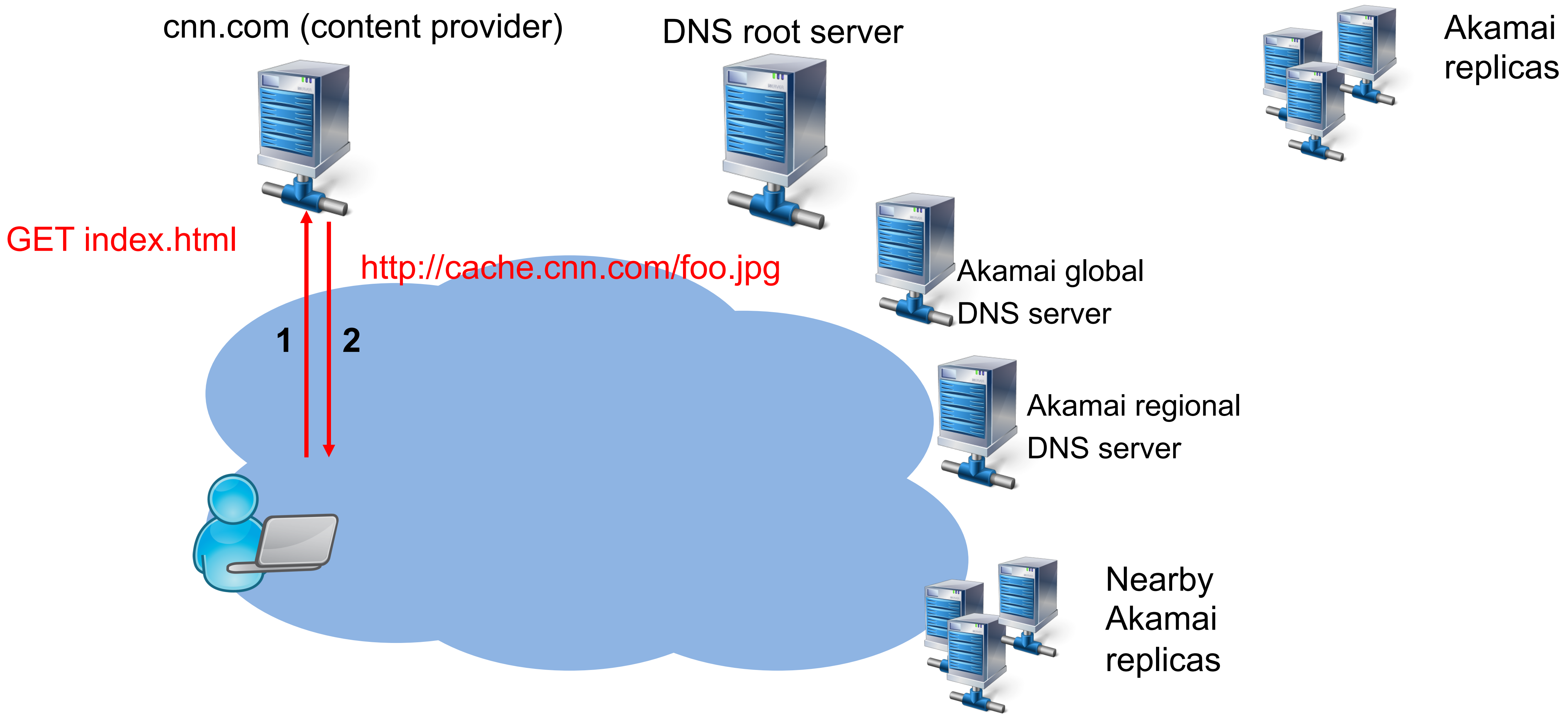
Client requests

- Hundreds of billions/day
- 15-30% of all web traffic

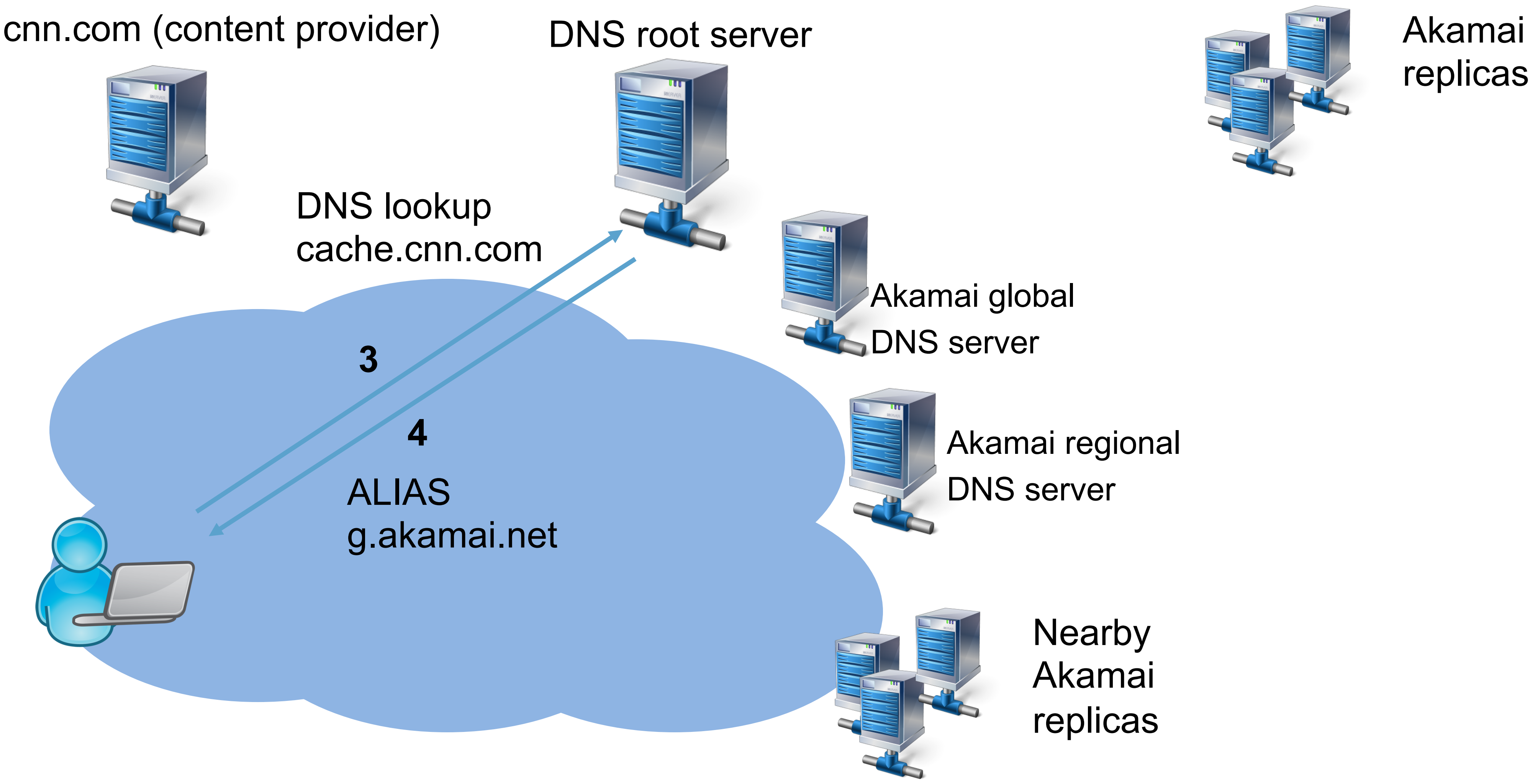
Components of a delivery network (Akamai)



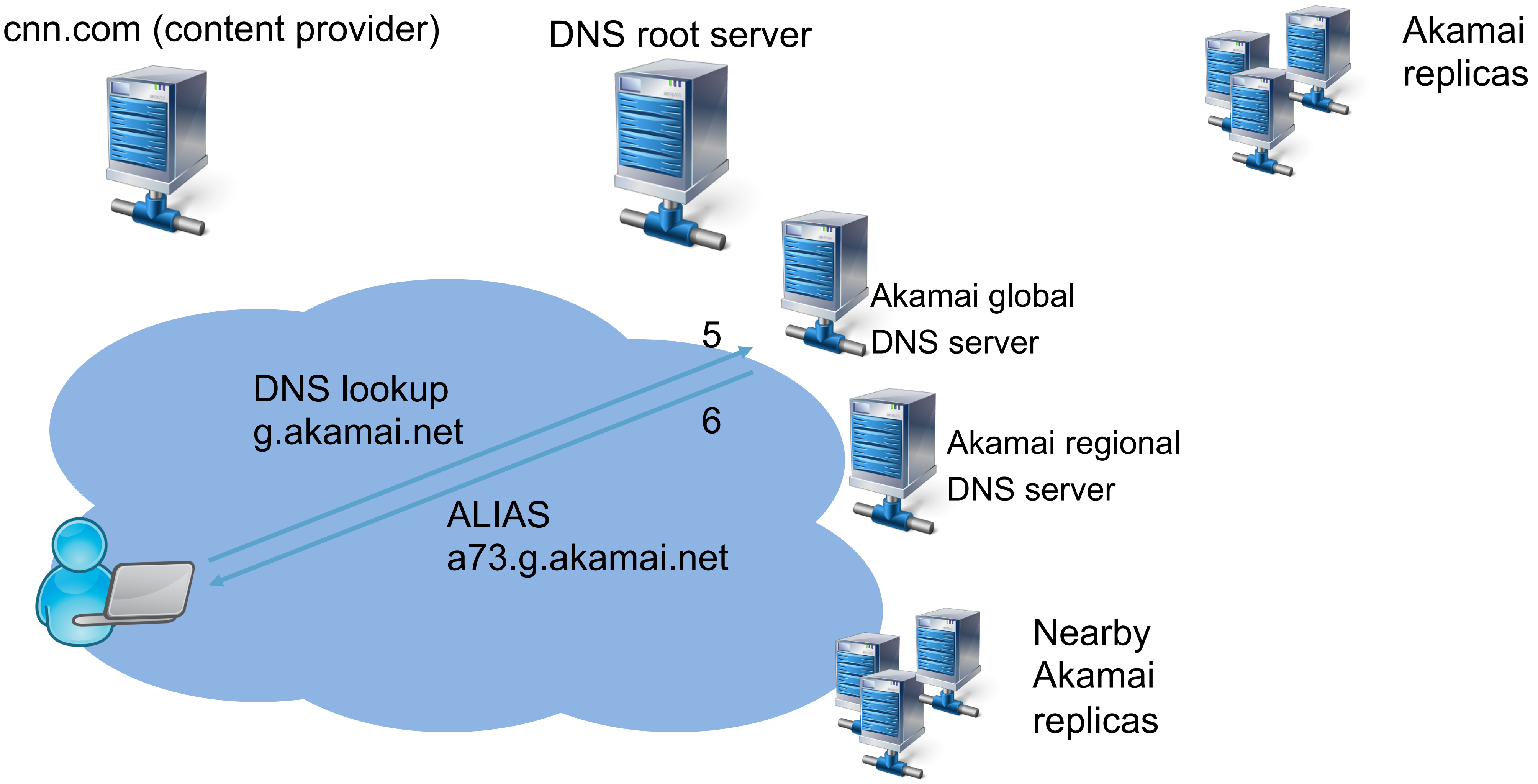
How Akamai uses DNS



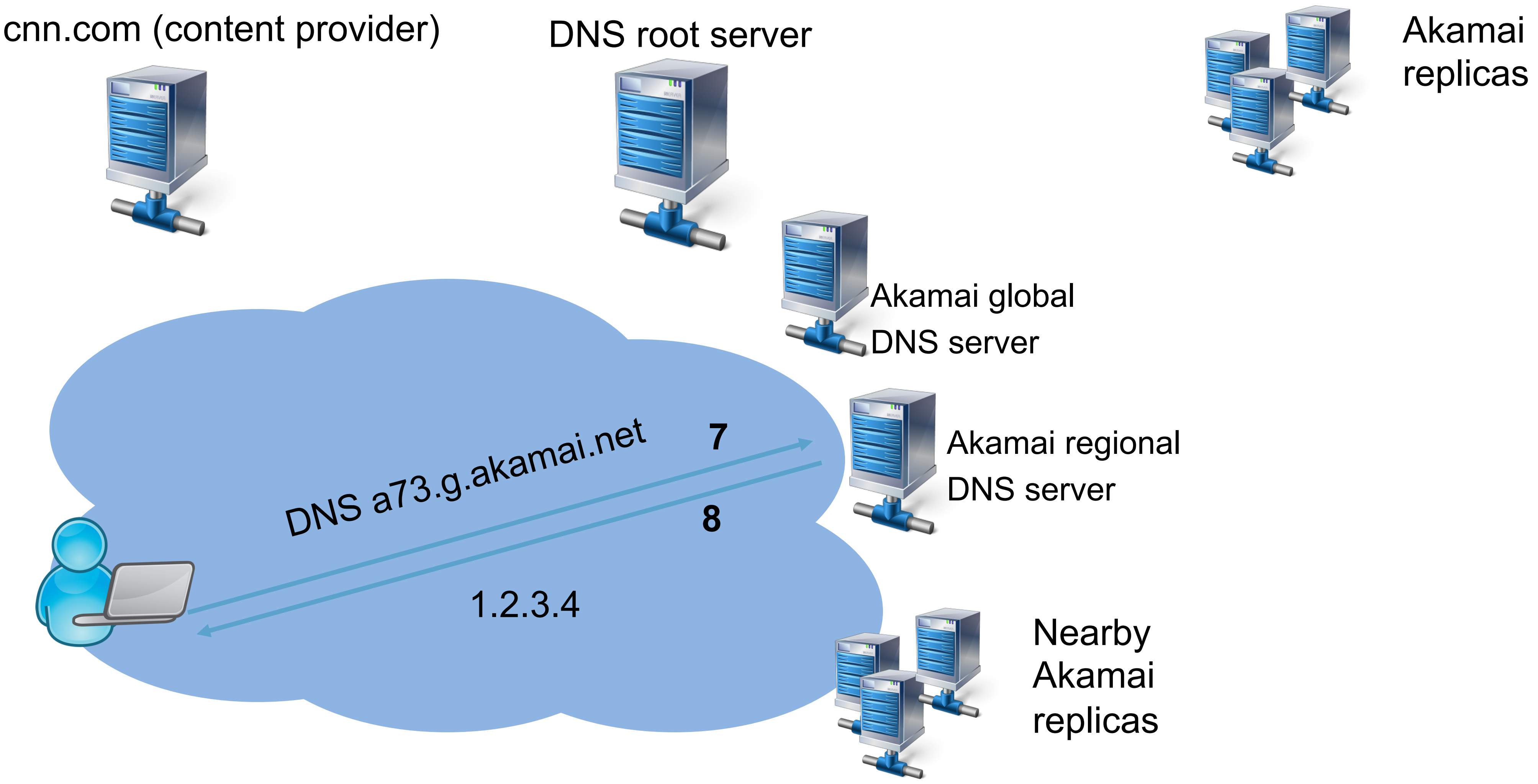
How Akamai uses DNS



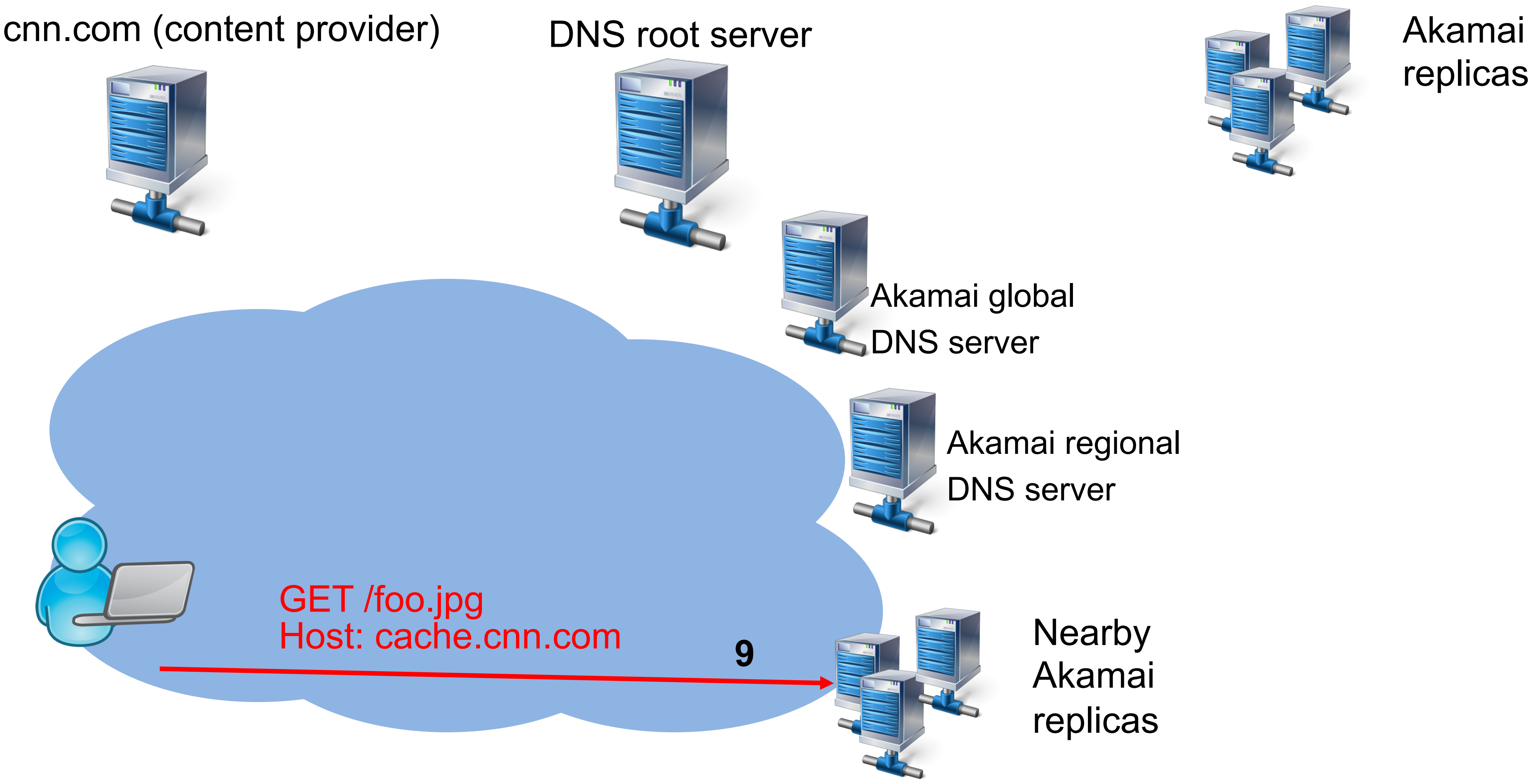
How Akamai uses DNS



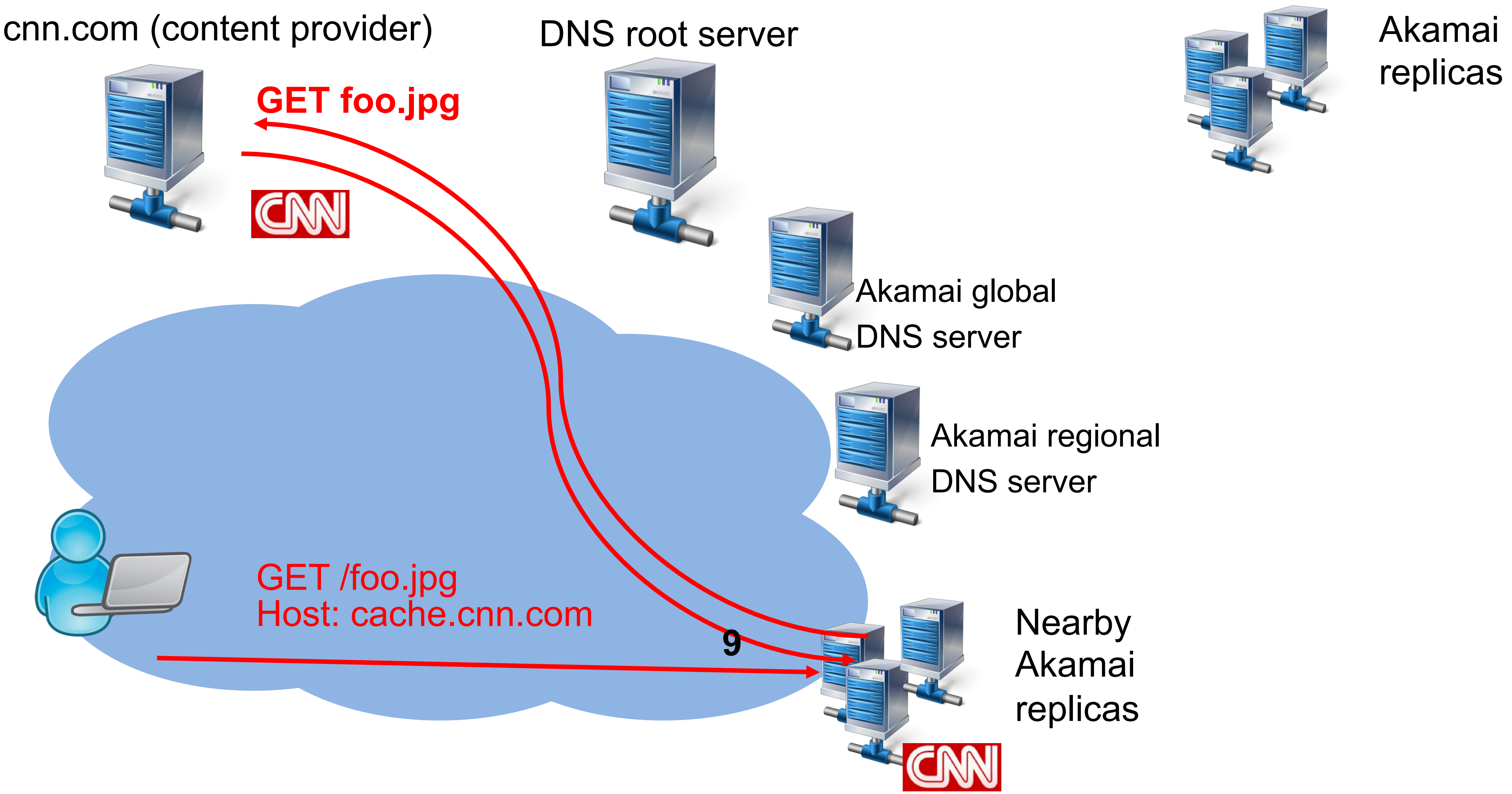
How Akamai uses DNS



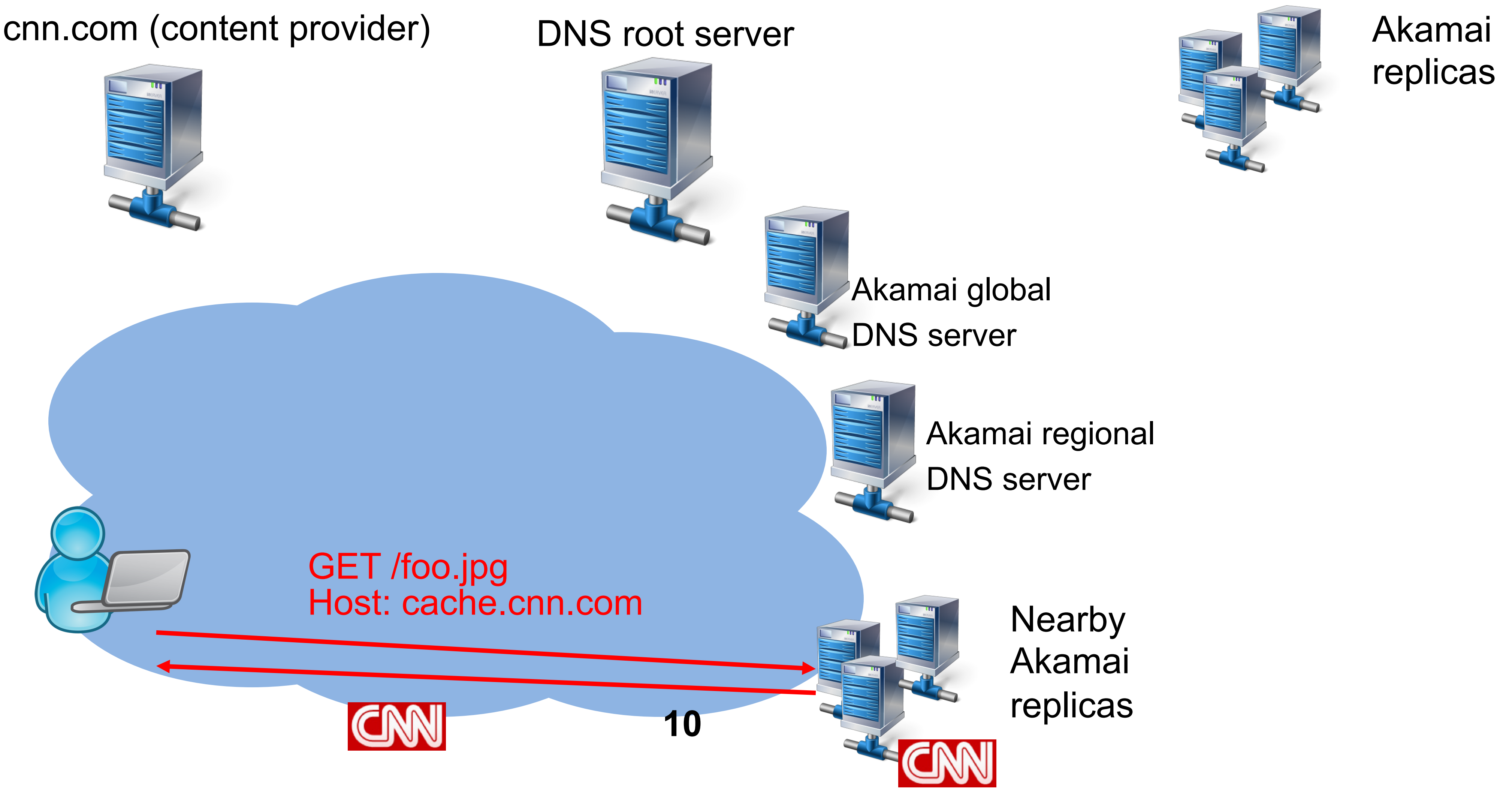
How Akamai uses DNS



How Akamai uses DNS



How Akamai uses DNS



Mapping System

- Equivalence classes of IP addresses
  - IP addresses experiencing similar performance
  - Quantify how well they connect to each other
- Collect and combine measurements
  - Ping, traceroute, BGP routes, server logs
  - Network latency, loss, and connectivity
- Map each IP class to a preferred server cluster
  - Based on performance, cluster health, etc.
  - Updated roughly every minute
- Map client request to a server in the cluster
  - Load balancer selects a specific server (e.g., to maximize cache hit rate)

TAKE 5

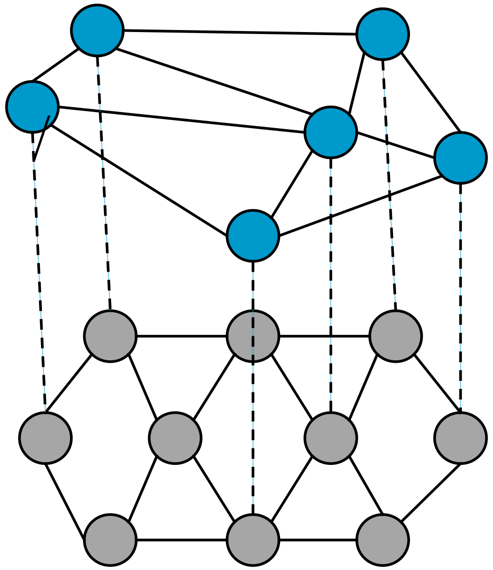
Overlay networks – virtual networks

- Different applications with a wide range of needs …

- Provide a service tailored to a class of applications
  - P2P file sharing, content distribution (CDNs)
- Support efficient operation in a given network environment
  - Wireless ad-hoc networks, delay tolerant networking
- Add extra features such as multicast or secure communication
  - IPv6, (overlay) multicast, resilience (RON), mobility, security (VPN)

Overlay networks

- A logical network built on top of a physical one
  - Overlay links are tunnels through the underlying network
- Nodes are often end hosts
  - Intermediate nodes contribute storage, CPU, just forward traffic for more reliable or faster communication
- Who controls the cooperating nodes?
  - The one who providing the service (e.g., Akamai)
  - A distributed collection of end users (e.g., P2P)
- The price to pay
  - Additional level of indirection
  - Opacity of the underlying network
  - Complexity of the network services



Peer-to-peer – A common overlays

- User computers talking directly (instead of via a central server)
  - Enabled by tech improvements in computing and networking
- A distributed architecture
  - No centralized control
  - Nodes are symmetric in function
- The promise
  - *Reliability* from many unreliable nodes – no central point of failure, multiple replicas, geographic distribution
  - High capacity through parallelism
  - Automatic configuration
  - Shifting control/power from organizations to users
  - …

Three generations of P2P

- (0) Many predecessors – DNS, Usenet, Grapevine, …
- (1) Unstructured and centralized
  - Napster – Sharing music; shutdown July 2001
- (2) Unstructured and decentralized
  - Gnutella, Kazaa, … - Peers are all equal and can connect to anyone
  - Super-peers to scale search and handle churn
- (3) Structured and decentralized
  - E.g. DHTs like Chord, Tapestry, Pastry, Kademlia and CAN

- Key common need – placing and finding resources on an overlay

Skype – an example overlay

- Peer-to-peer VoIP
  - Developed by Kazaa in 2003, acquired by Microsoft in 2011 (US$ 8.5B)
  - 40% of the International call market share (2014), 300M monthly users, 4.5M daily

- Notes on design*
  - Super-peer structure (super-peer selected based on availability, reachability, bandwidth, etc)
  - Users login through a well-known server, but connect to the network and others through super-peers
  - TCP for control, TCP or UDP for voice

*Baset & Schulzrinne's studies

Another classical example – BitTorrent

- A cooperative, popular service for content distribution
- Basic operation
  - User clicks on download link, gets torrent file with content hash and IP address of tracker
  - User’s BT app talks to the tracker, gets a list of other users with downloaded file
  - User’s BT app talks to one or more users with the file, and
  - tell tracker it has a copy too
  - User’s BT app servers the file to others for a while

The problem with trackers

- Hard to distribute files (need a tracker)
- Tracker may not be reliable
- Single point of failure
  - Easy target of copyright owners
  - Or people offended by content
- Could you use a distributed <key,value> store for this?
  - All apps cooperatively implementing it
  - Key is the torrent file content hash (“infohash”), value is the torrent IP
  - BT find other apps able to serve that content by asking around for <key,value> pairs
  - And adds itself as another one willing to help after it has it
  - … but how do you find the <key, value> pair(s) you want?

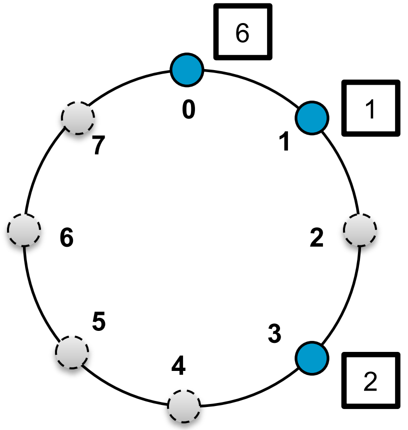
Distributed Hash Tables (DHTs)

- Goal – quick retrieval, storage of <key, value> pairs
- General approach
  - Map node IDs to a (large) circular space
  - Map keys to the same circular space
  - <key, value> pairs are stored in nodes with IDs that are close for some notion of closeness
- A simple interface
  - put(key, value) | get(key) ➔ value
- Weak consistency – likely that get(k) see put(k), no guarantees
- Two examples
  - Chord – one of the original DHTs [Stoica. 2001]
  - Kademlia – A popular second system [Maymounkov, Mazières, 2002]

Chord

- Basics
  - IDs space, m-bit long – 128-160 bits such as SHA-1
  - Identifiers are ordered in an identifier circle modulo 2^m (range $[0, 2^m-1]$)
  - Key k "belongs" to nearest node – node with the smallest id $\geq k$, the successor of k (closeness is "clockwise distance")

An id circle with $m = 3$
Three nodes: 0, 1 and 3



Key 2 is in node 3
pred(3) $< 2 \leq 3$

- To resolve k to address of succ(k)
  - Nodes keep track of their successor on the circle
  - Simplest way – go around the circle until we get k's successor

Chord

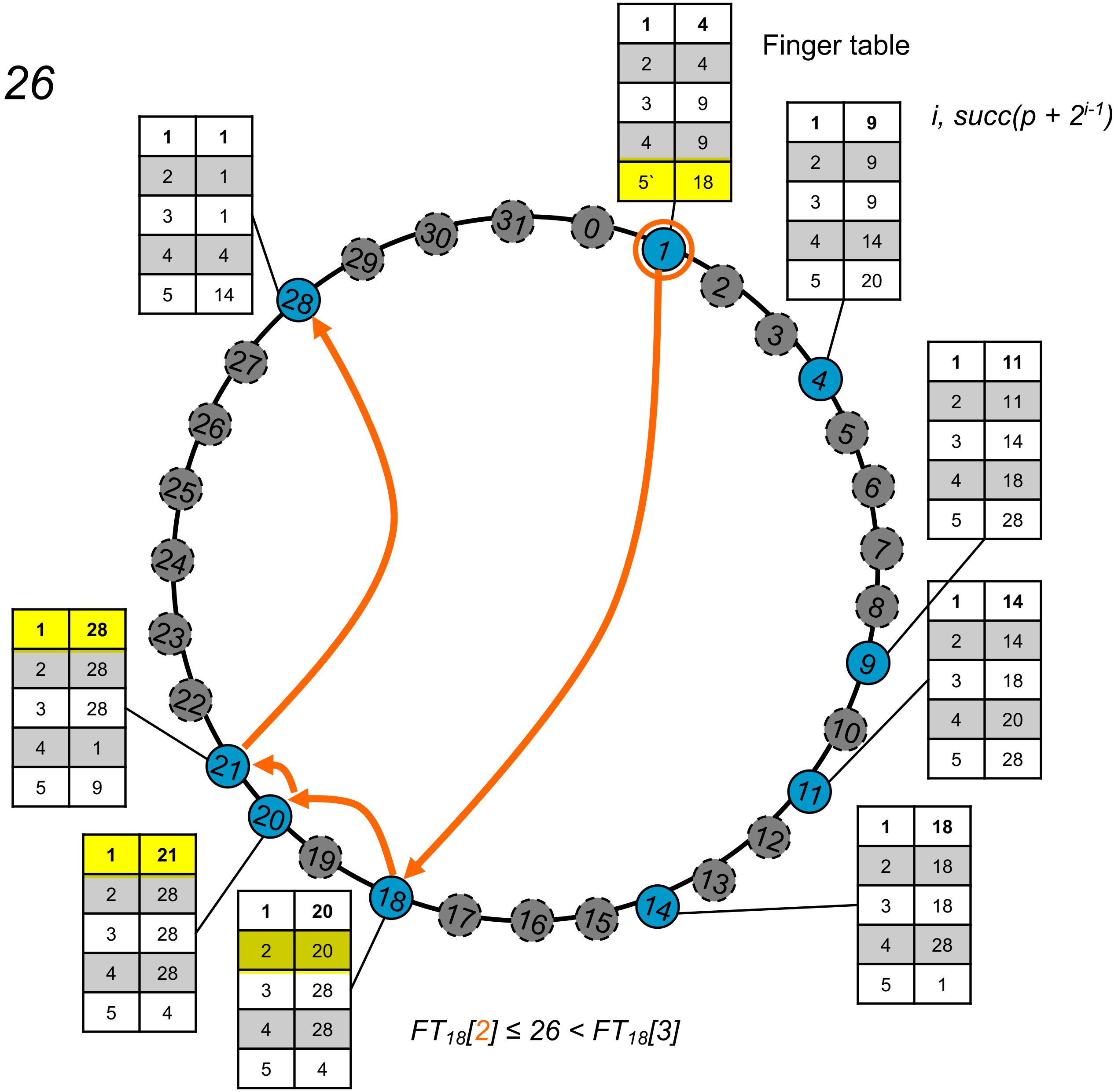
- Short-cuts to speeds things up (not for correctness)
- Nodes keep a finger table of at most *m* entries
  - If FT_p denotes the finger table of *p*, $FT_p[i] = succ(p+2^{i-1})$
    - i.e., the *i-th* entry points to the first node succeeding *p* by at least 2^{i-1} $(1 \leq i < m)$
  - FT entry contains Chord ID, IP and port
  - The first entry is *p* immediate *successor* on the circle
  - Shortcuts' distance increases exponentially with index
- To look up key *k*, node *p* will forward request to node *q* with index *j* in *p's* FT where

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

Resolving in Chord – example

Resolving $k = 26$ from node 1

Done, now return address of node 28 to node 1



Some details

- How much faster with FT?
  - Log(n) hops – one of the fingers takes you ~half-way to target
  - Is that good? 10 hops for 1m nodes
    - 50ms per hop, 0.5sec so, not bad
- How does a node gets correct tables?
  - Starting from scratch, add new nodes
  - Use DHT lookups to populate new nodes' finger tables
  - For a new node m
    - Send lookups for its own key, this yields m.successor
    - Gets successor's FT

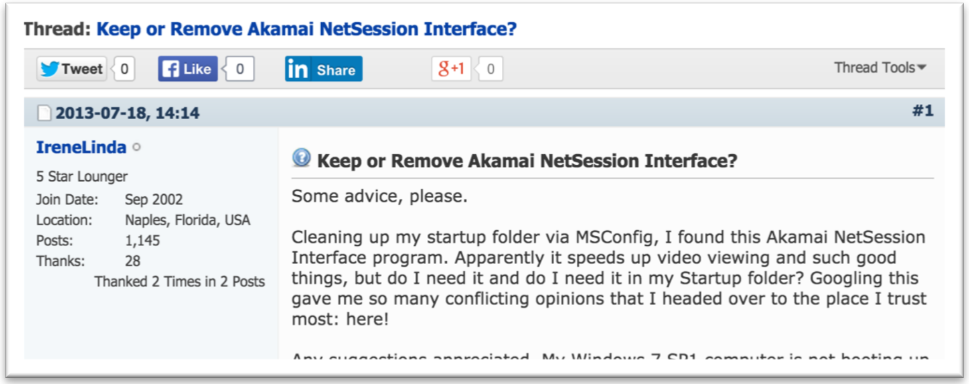
CDNs or P2P?

- P2P systems
  - Cheap, easy to scale
  - Security issues, potential low-quality, hard to find unpopular content, difficult accounting
- Infrastructure-based systems
  - Expensive to setup and scale
  - Akamai 137,000 servers in 87 countries (probably out of date)
  - Can provide predictable QoS and reliable accounting

CDNs or P2P? Both

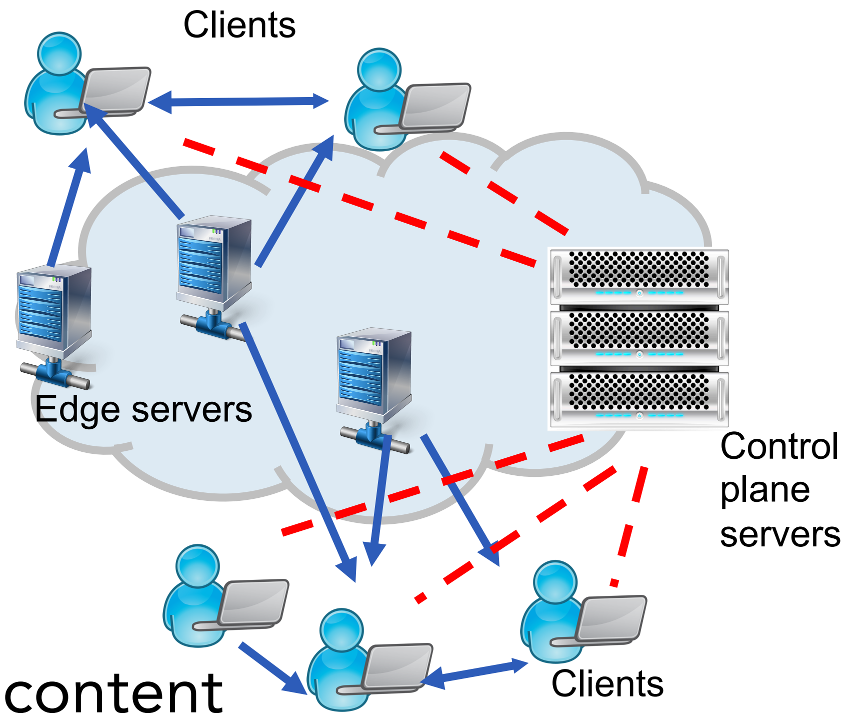
- Hybrid? Peer-assisted CDNs
  – Deliver content by peers, with operation coordinated (and backstopped) by dedicated infrastructure
  – Akamai's NetSession – Operating commercially since 2010
  – True global coverage – 239 countries in 2013

- Risks/Issues
  – Need for revenue, unlike P2P
  – No transparency – users are aware of them
  – Heterogeneity
  – NATs and firewalls
  – Impact to ISP – change of traffic patterns



NetSession's approach and some answers

- Download starts from edge servers
  - Standard HTTPS
- Ask control plane for nearby peers
- If anyone's around, download from them
  - ~Swarm – small pieces exchanged
  - No need for tit-for-tat
  - Edge servers generate unique secure IDs for content and hashes for validation
  - HTTPS connection is used for configuration and reporting



Recap

- New applications with new demands on the underlying network
- Architectural changes are, at best, difficult
- Overlays as a path to deployment and an experimental testbed
  - Deploying narrow fixes?
  - No demands on underlying network (to ensure deployment)
- From grassroots efforts and research labs to products
- Many open hard issues – security, churn, …