CS 5450

Content Distribution Networks

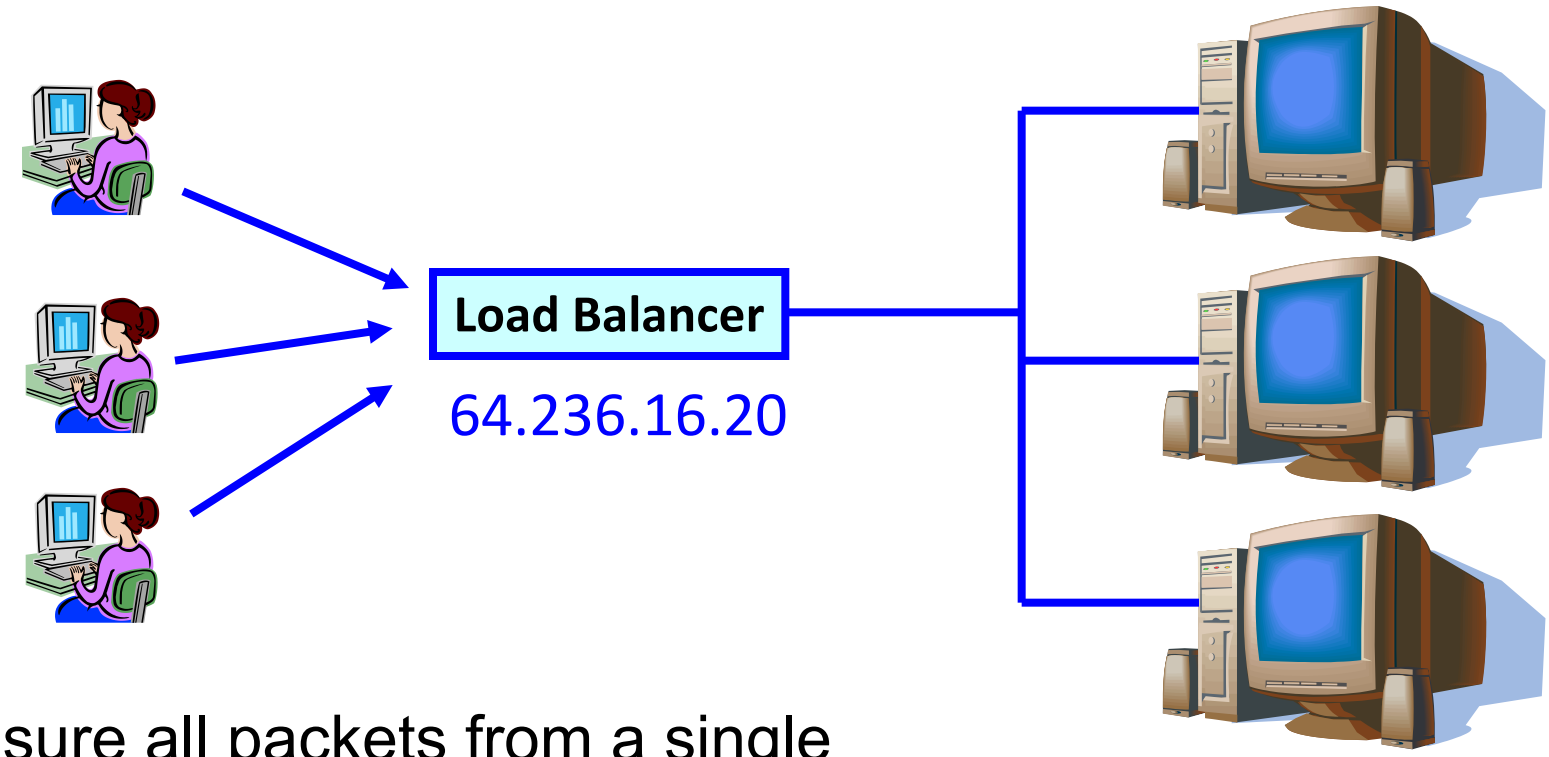
Vitaly Shmatikov

Problem: Overloaded Website

- Solution: replicate site across multiple machines
- Need to direct client to a particular replica
  - Goal: balance load across server replicas
- Solution #1: manual selection by client
  - Each replica has its own site name
  - Some webpage lists replicas (by name or location), asks clients to click link to pick

Load-Balancer Approach

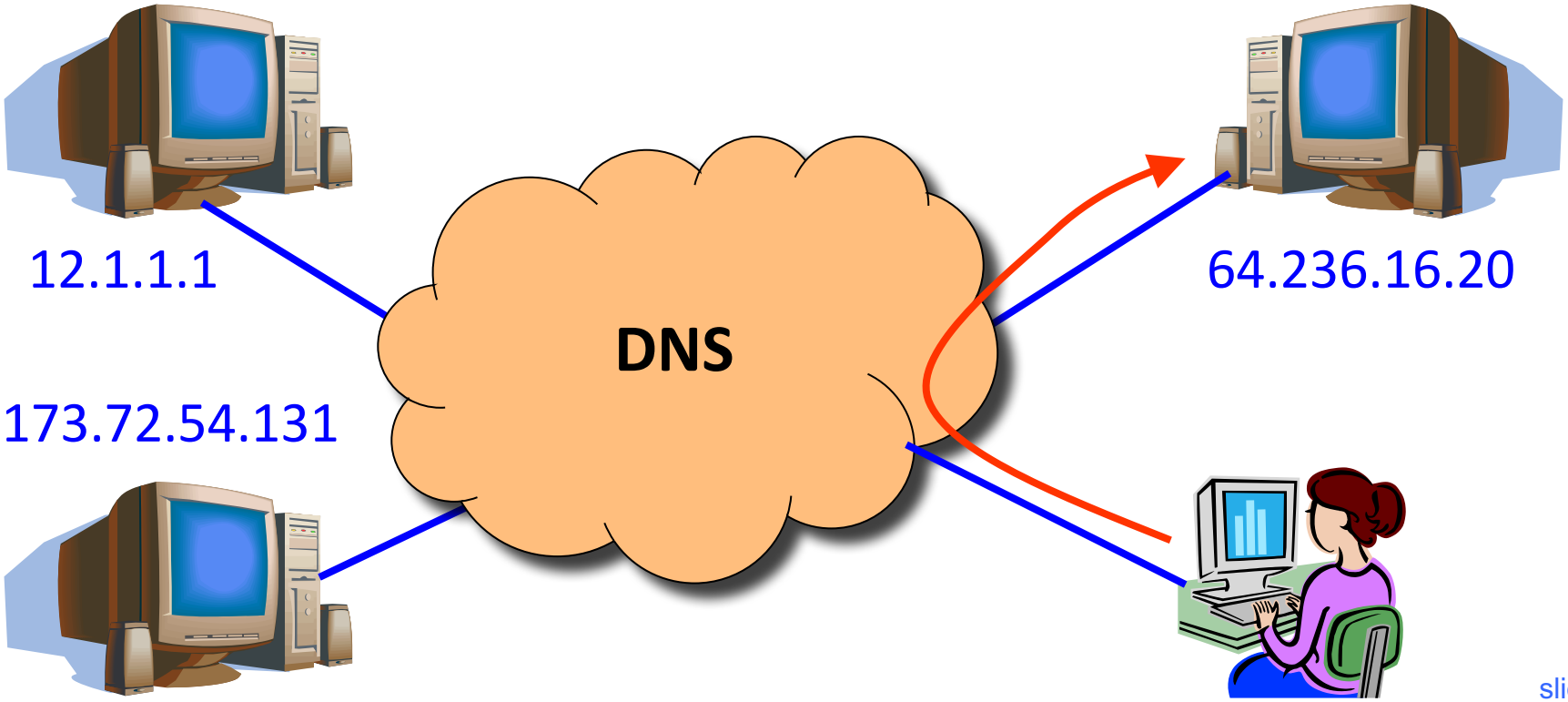
- Solution #2: single IP address, multiple machines



Ensure all packets from a single TCP connection go to the same replica

DNS Redirection Approach

- Solution #3: multiple IP addrs, multiple machines
  - Same DNS name, different IP for each replica
  - DNS server returns IP addresses "round robin"

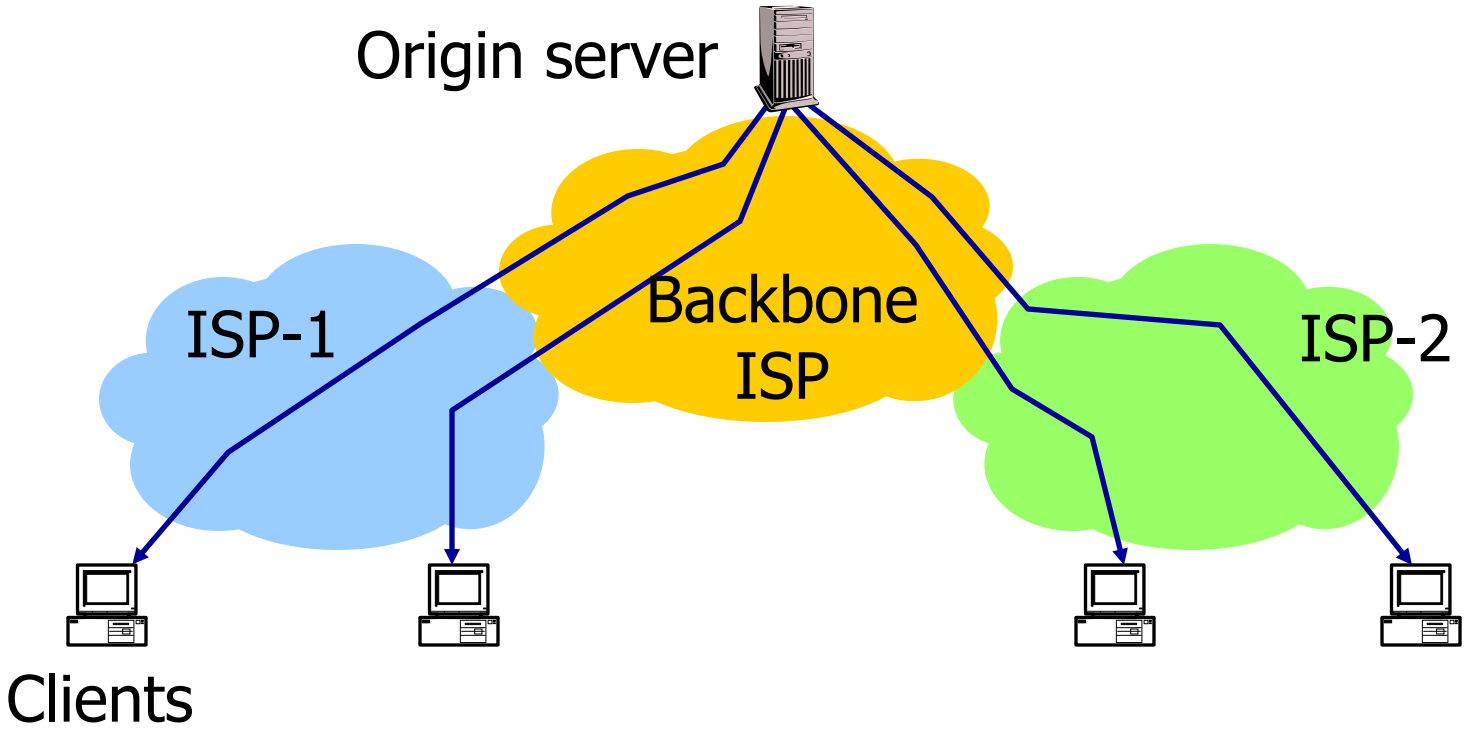


Distributing Client Requests

- Load-balancer approach
  - No geographical diversity ✘
  - TCP connection issue ✘
  - Does not reduce network traffic ✘
- DNS redirection approach
  - No TCP connection issues ✔
  - Simple round-robin server selection
    - May be less responsive ✘
  - Does not reduce network traffic ✘

Motivation for Web Caching

- Many clients request the same information
  - Generates redundant server and network load
  - Clients may experience high latency

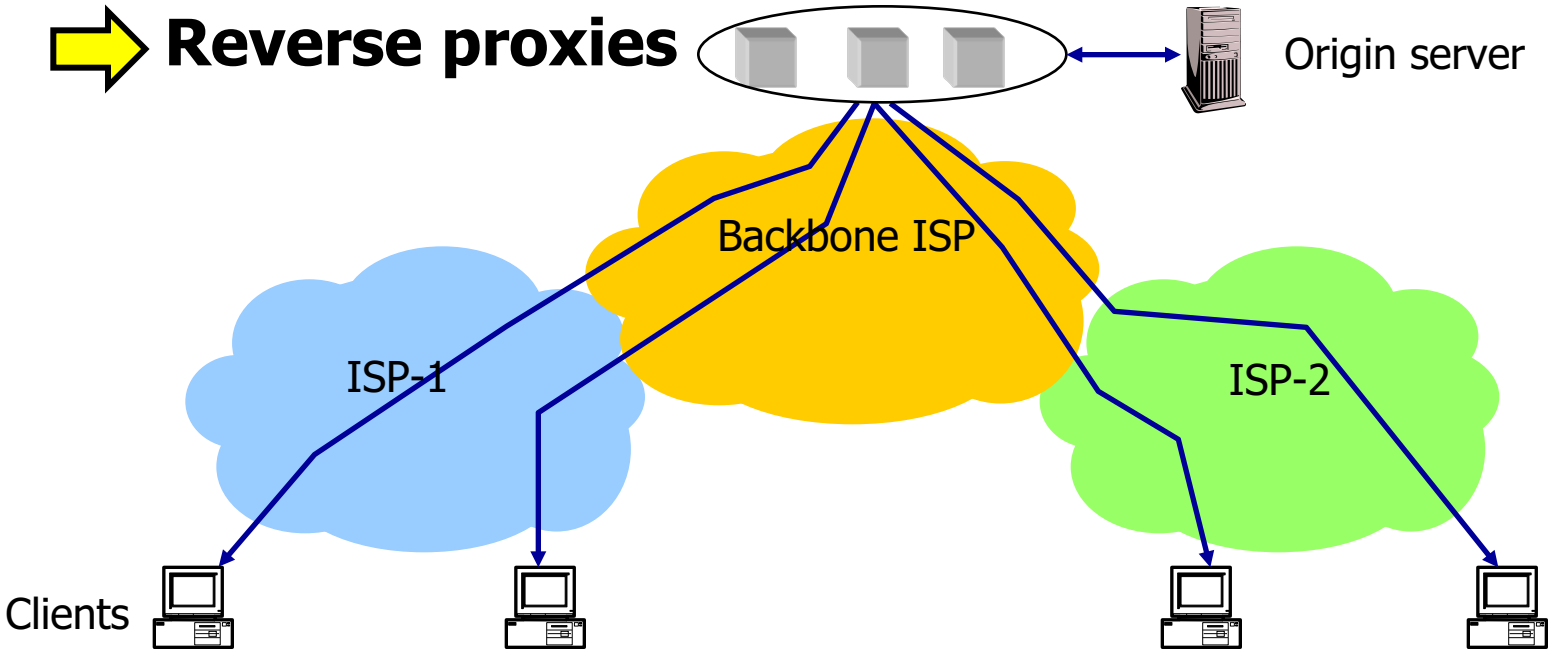


Web Caching

- Why place content closer to client
  - User gets better response time
  - Content providers get happier users
  - Network gets reduced load
- Why does caching work?
  - Exploits locality of reference
- How well does caching work?
  - Very well, up to a limit
  - Large overlap in content, but many unique requests

Caching with Reverse Proxies

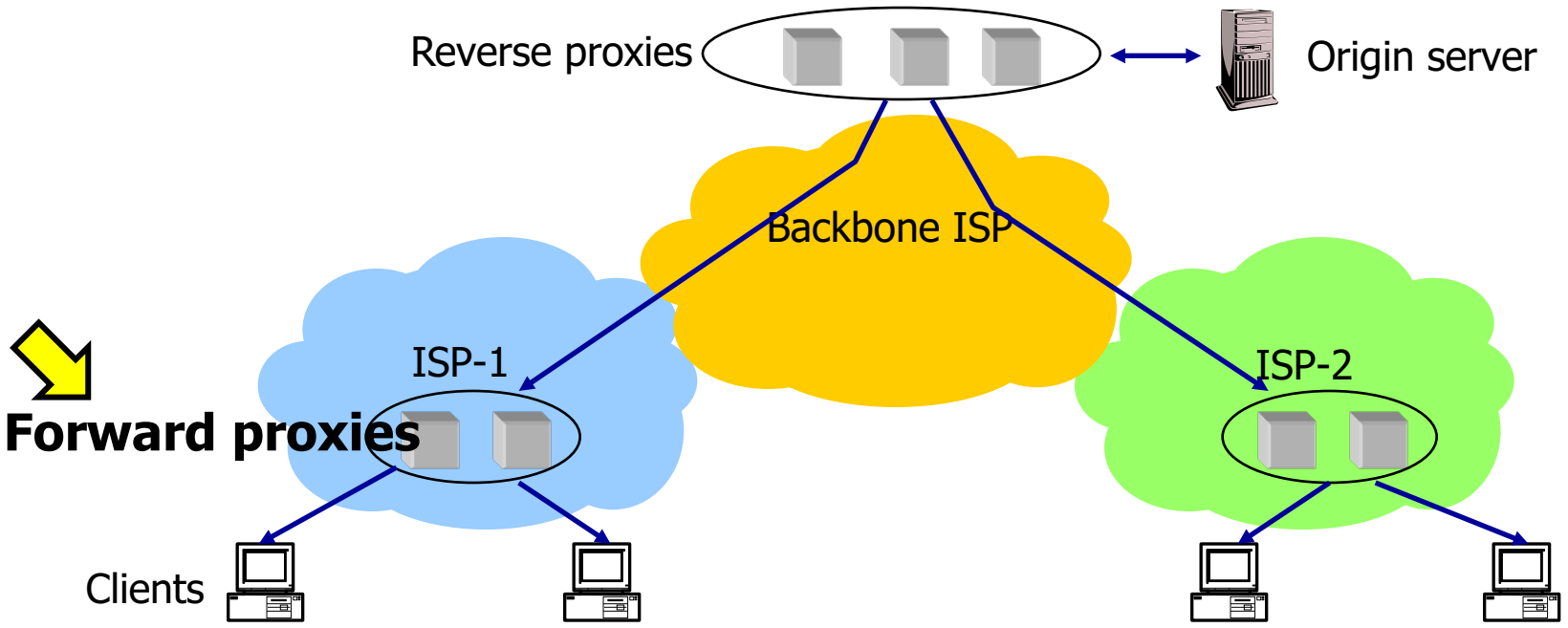
- Cache data close to origin content server
  - Typically done by content providers to reduce load
  - Client thinks it is talking to the origin server
- Does not work for dynamic content



Caching with Forward Proxies

- Cache close to clients → less network traffic, less latency
  - Typically done by ISPs or corporate LANs
  - Client configured to send HTTP requests to forward proxy
- Reduces traffic on client ISP's access link, origin server, and backbone ISP

Proxies


Reverse proxies
Origin server
Backbone ISP
ISP-1
ISP-2
Forward proxies
Clients

Challenges

- Problem ca. 2002: How to reliably deliver large amounts of content to users worldwide?
  - Popular event: "flash crowds" overwhelm (replicated) web server, access link, or back-end database infrastructure
  - More rich content: audio, video, photos
- Web caching: Diversity causes low cache hit rates (25–40%)

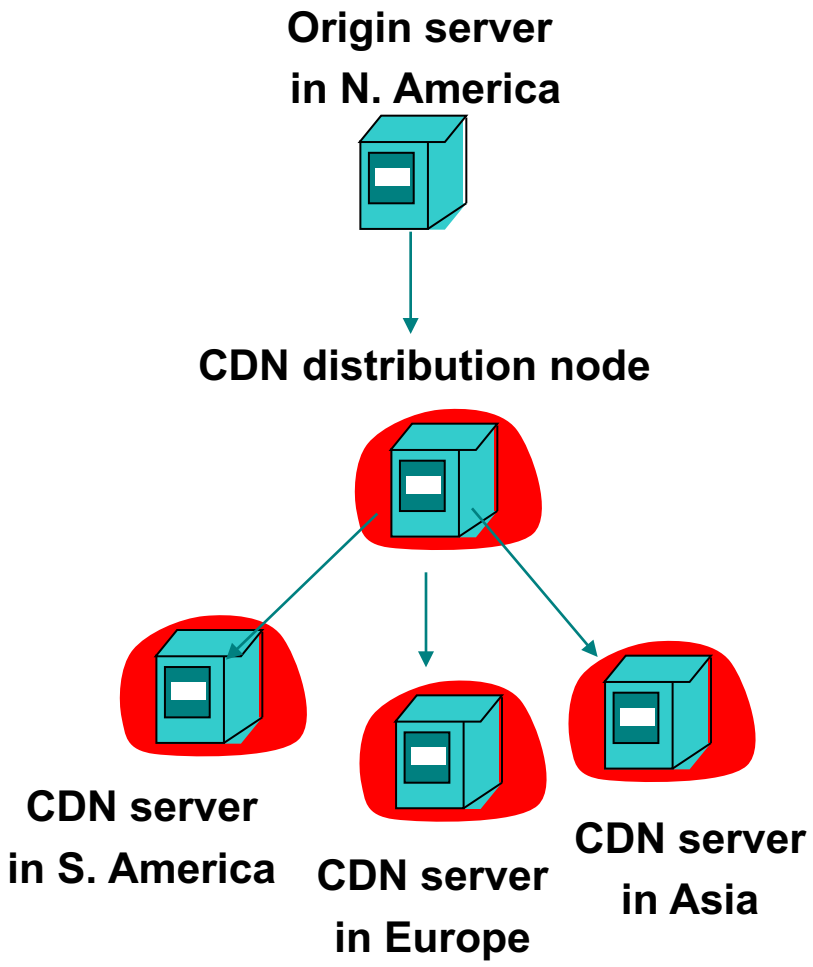
Typical Webpage Workload

- Multiple (typically small) objects per page
- File sizes are heavy-tailed
- Embedded references
- This plays havoc with performance. Why?

- Lots of small objects & TCP
- 3-way handshake
- Lots of slow starts
- Extra connection state

Content Distribution Network

- Proactive content replication
  - Content provider (e.g. CNN) pushes content out from its own origin server
- CDN replicates the content
  - On many servers spread throughout the Internet
- Updating the replicas
  - Updates pushed to replicas when the content changes



CDN Challenges

- How to replicate content
- Where to replicate content
- How to find replicated content
- How to choose among known replicas
- How to direct clients towards replica

Replica Selection

Requires continuous monitoring of liveness, load, and performance

- Which server?
  - Lowest load → to balance load on servers
  - Best performance → to improve client performance
    - Based on geography? RTT? Throughput? Load?
  - Any alive node → to provide fault tolerance
- How to direct clients to the chosen server?
  - As part of routing → anycast, cluster load balancing
  - As part of application → HTTP redirect
  - As part of naming → DNS

Application-Based Selection

- HTTP supports a simple way to indicate that Web page has moved
  - 30X responses
- Server receives GET request from client, decides which server is best suited for particular client and object, returns HTTP redirect to that server
- Additional overhead
  - Multiple connection setup, name lookups, etc.
- HTTP redirect has some design flaws – especially with current browsers

Naming-Based Selection

- Client does name lookup for service
- Name server chooses appropriate server address
  - "A" record returned is the "best" one for the client
- How does the name server choose?
  - Server load and location (info must be collected!)
  - Information in the name lookup request
    - … typically from the local name server for client

How Akamai Works

- Akamai creates new domain names for each customer
  - Ex: a73.g.akamaitech.net
  - Akamai's DNS servers are authoritative for these names
- Clients fetch HTML document from origin server
  - Ex: fetch index.html from cnn.com
- URLs for replicated content are replaced in HTML
  - Ex:<img src="http://cnn.com/af/x.gif"> becomes <img src="http://a73.g.akamaitech.net/7/23/cnn.com/af/x.gif">
- Client's browser issues GET to CDN instead of origin server

Content Replication in Akamai

- Akamai only replicates static content
  - Akamai also lets sites write code that can run on Akamai's server, but that's different
- Modified name contains original file name
- Akamai server is asked for content
  - First checks local cache
  - If not in cache, requests file from primary server and caches file

DNS-Based Redirection

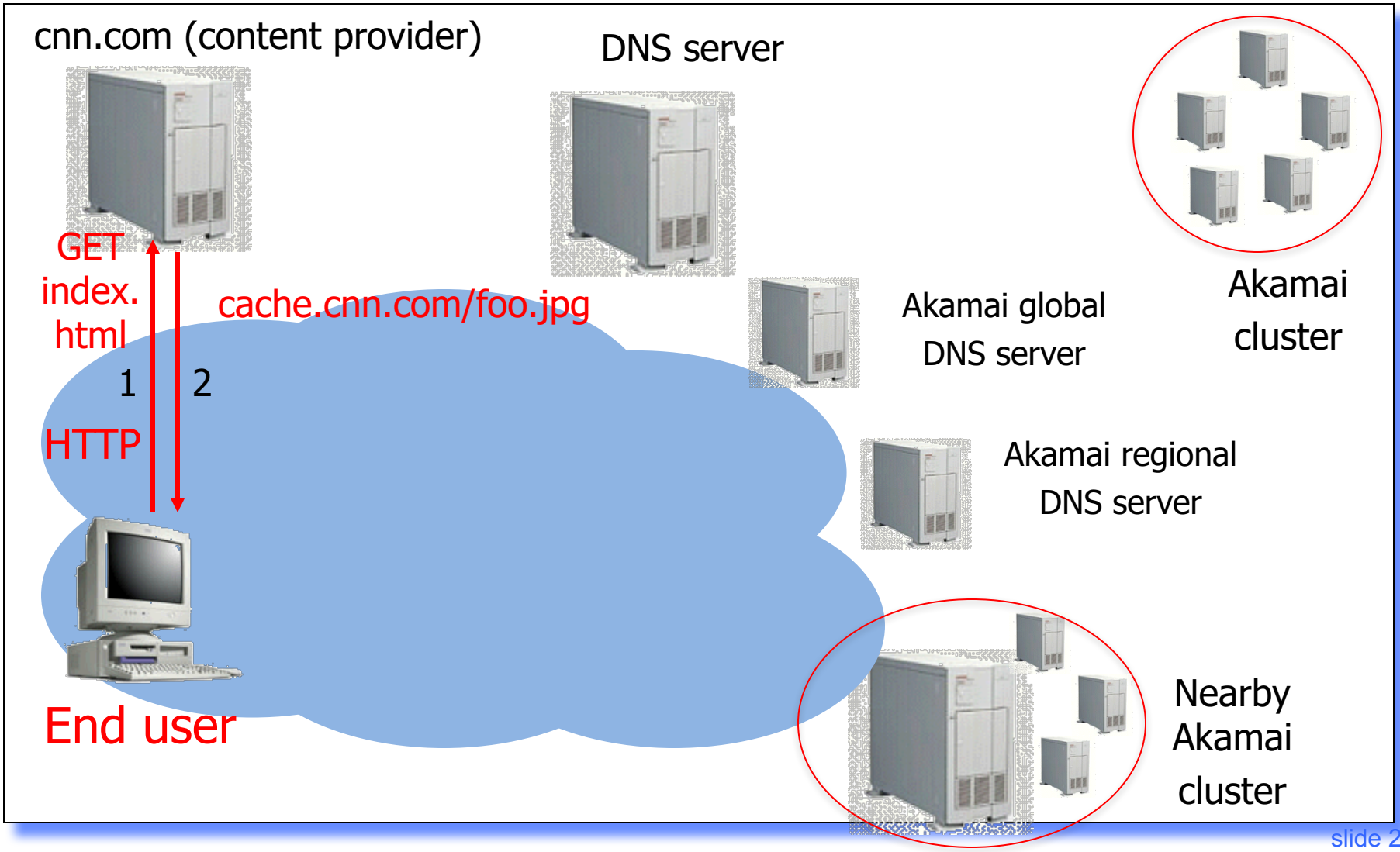
Two levels of DNS indirection

- Akamai top-level name servers (TLNSs)
  - 4 in the U.S., 4 in Europe, 1 in Asia
  - TLNSs return eight LLNSs in three different regions
    - Chosen to be close to the requesting client
    - Handles complete failure of any two regions
- Akamai low-level name servers (LLNSs)
  - Point to Akamai edge servers, which serve content
  - Do most of the load-balancing

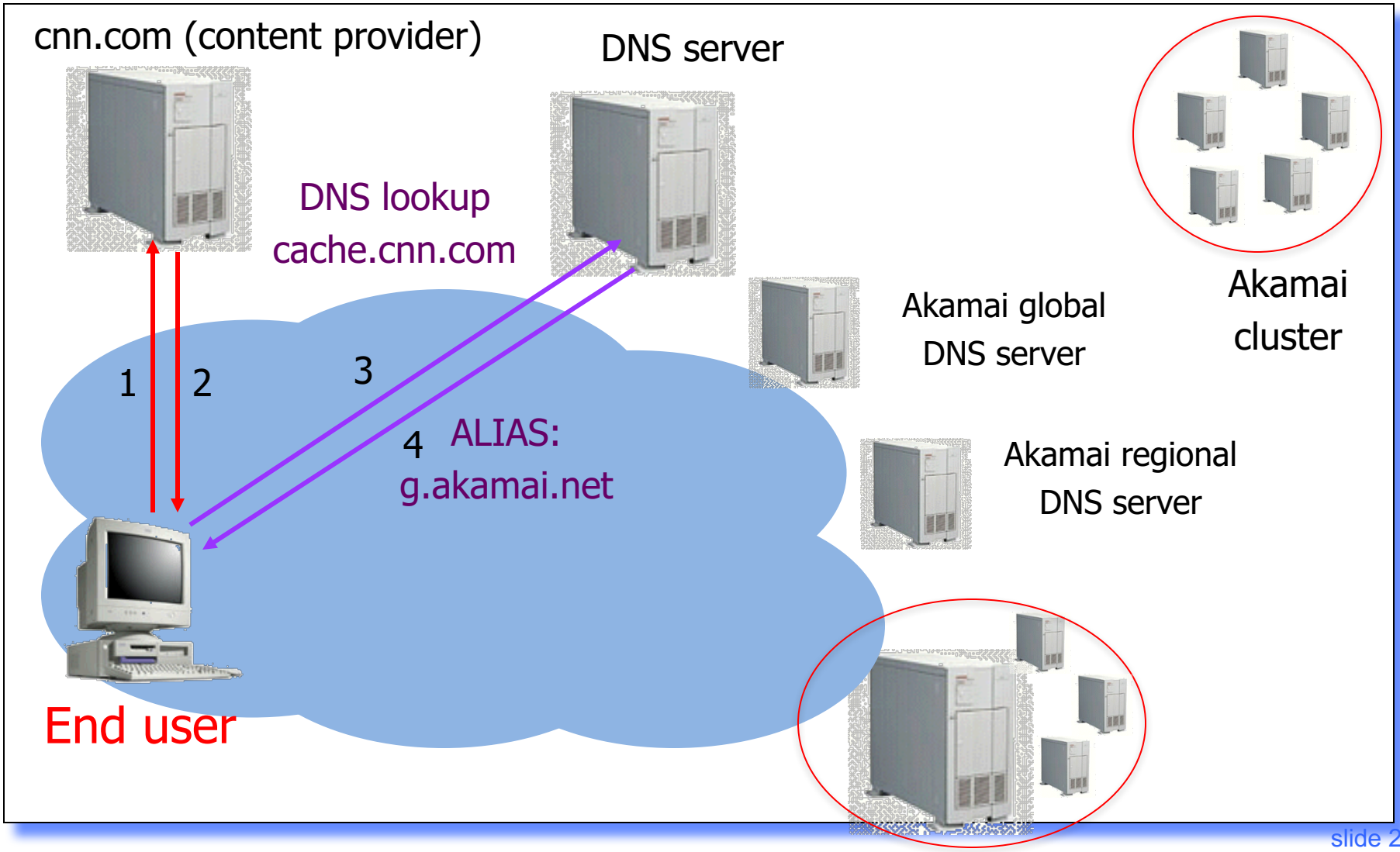
Using DNS in Akamai

- Root server gives NS record for akamai.net
- akamai.net name server returns NS record for g.akamaitech.net
  - Name server chosen in the region of client's name server
  - TTL is large
- g.akamaitech.net name server chooses server in the client's region
  - Should try to choose server that has file in cache (how?)
  - Uses aXYZ name and hash
  - TTL is small (why?)

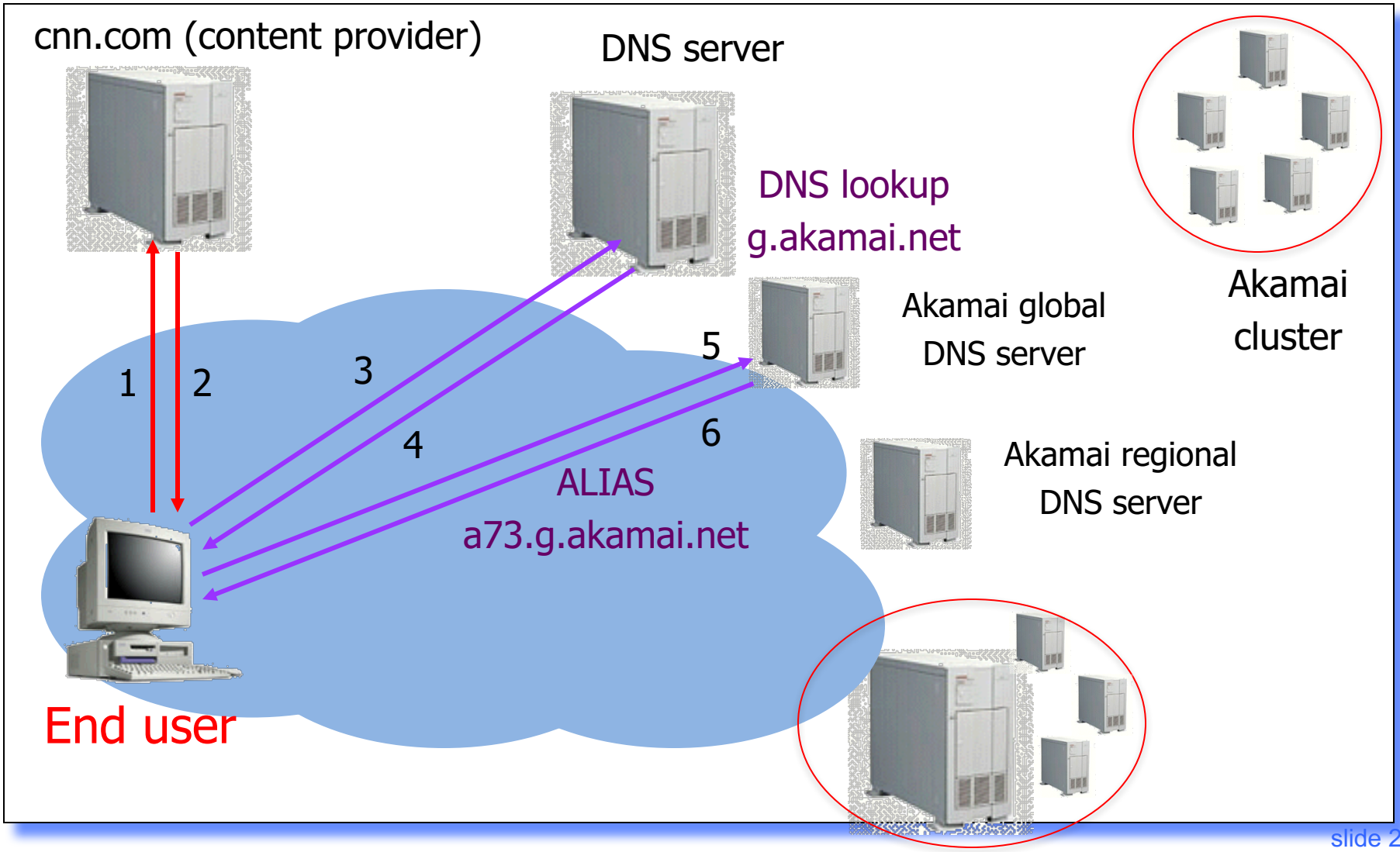
How Akamai Uses DNS



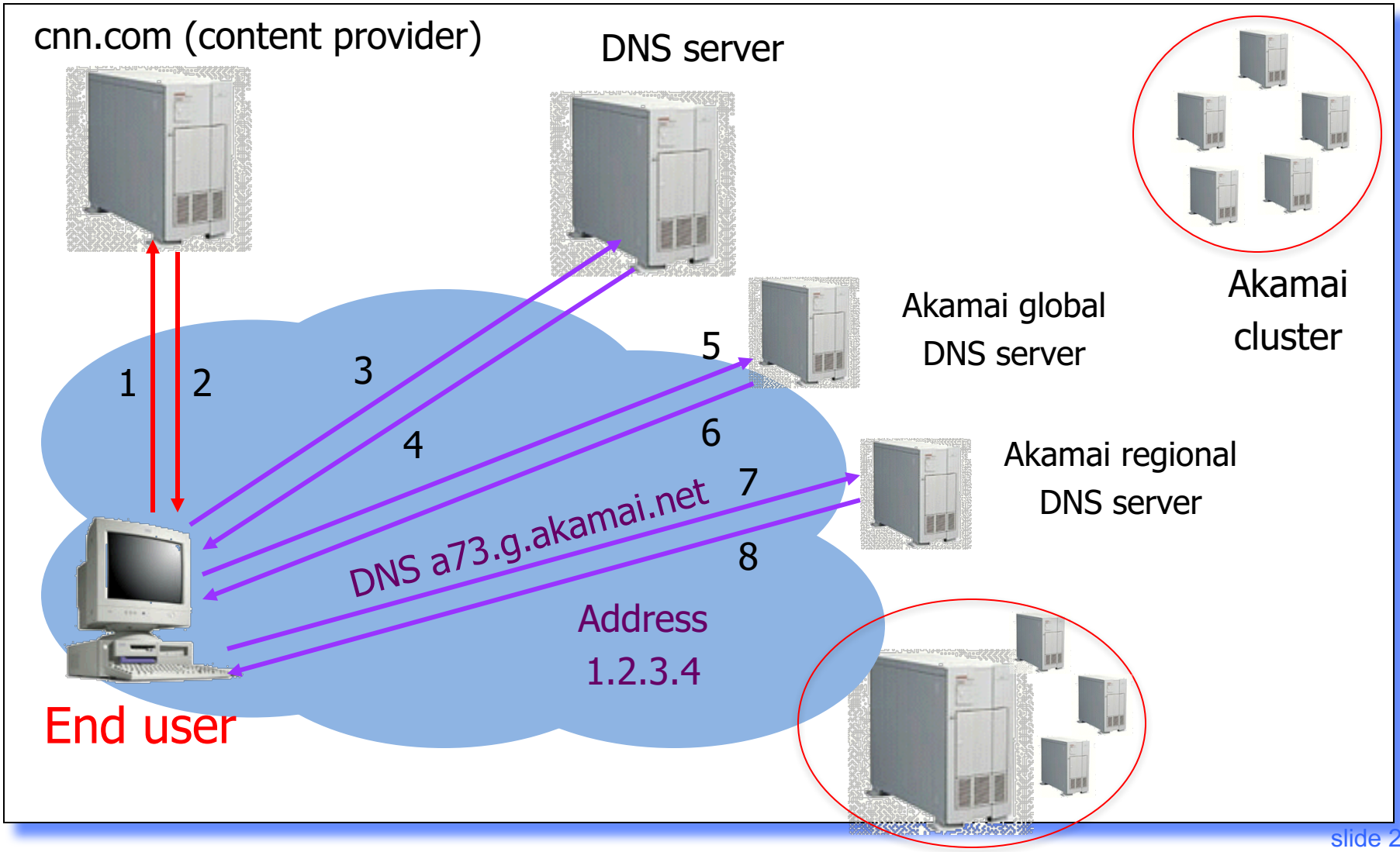
How Akamai Uses DNS



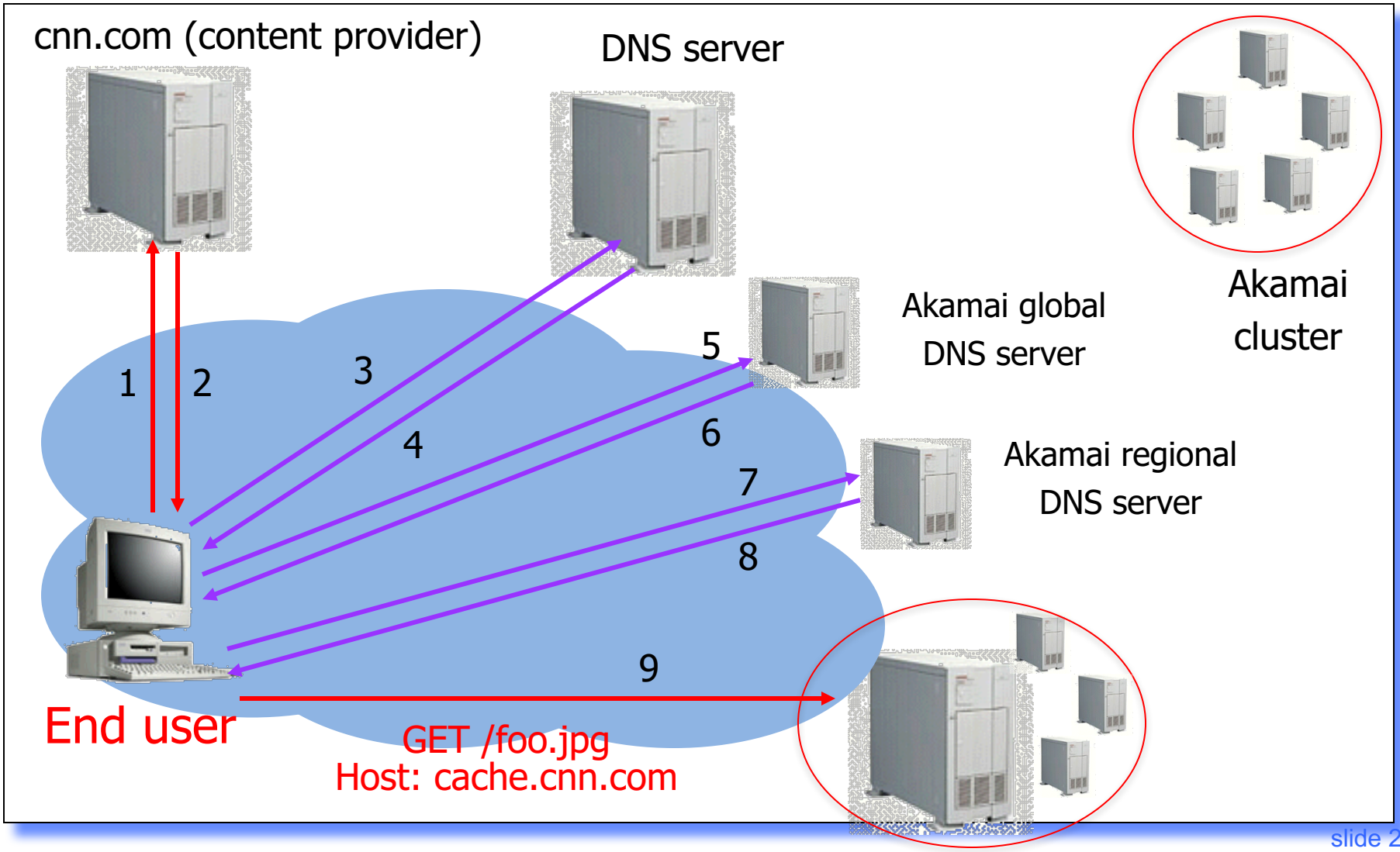
How Akamai Uses DNS



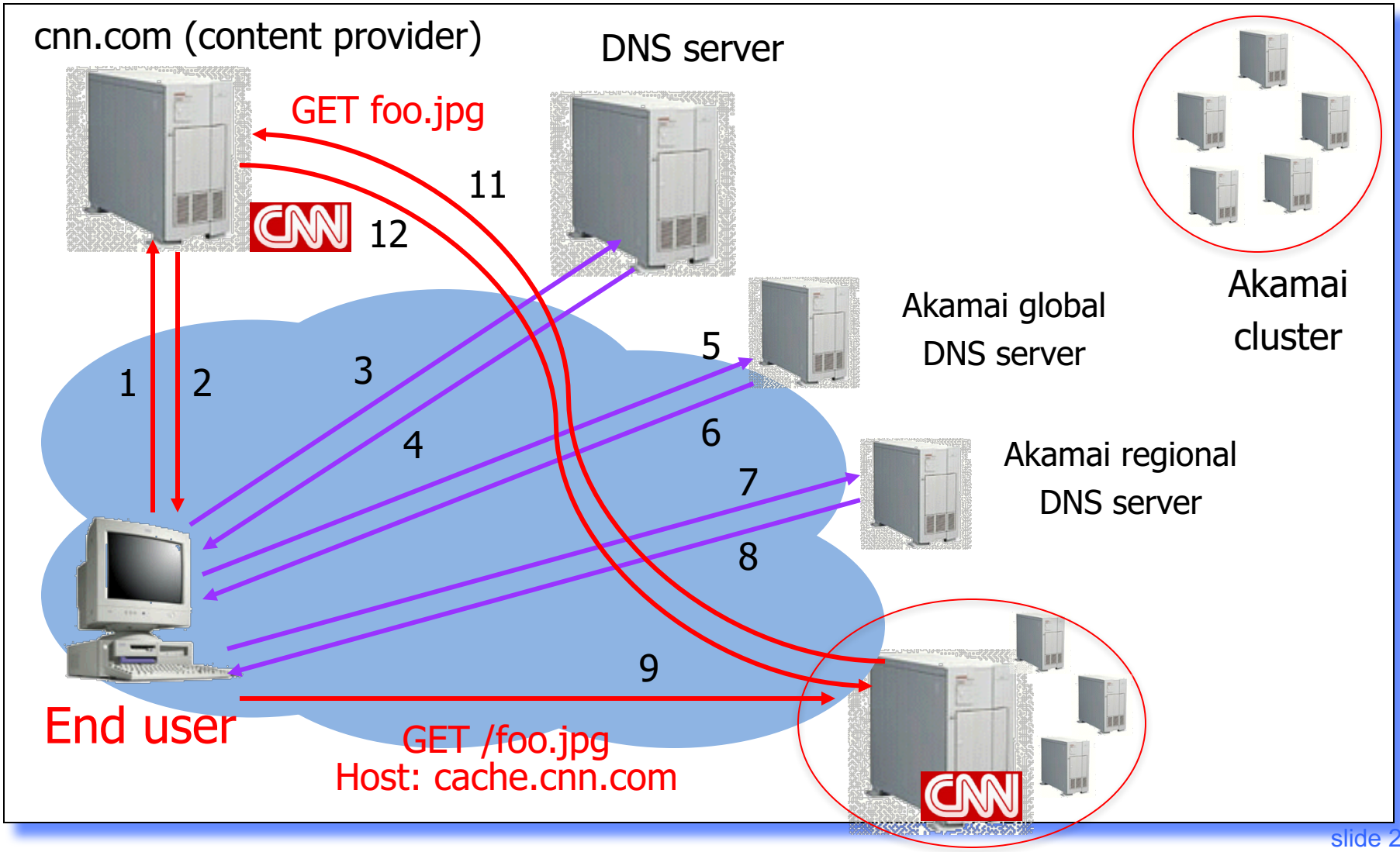
How Akamai Uses DNS



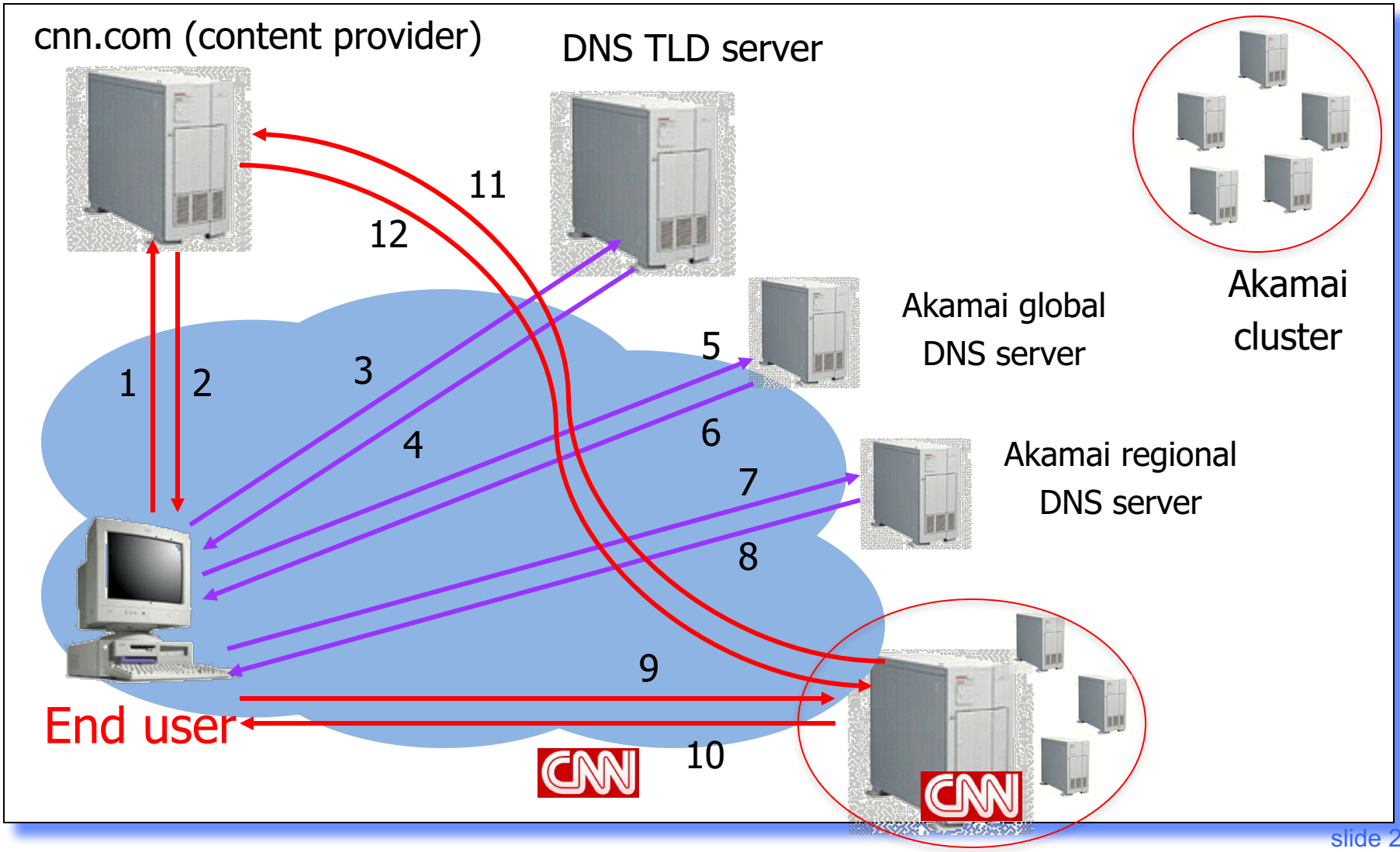
How Akamai Uses DNS



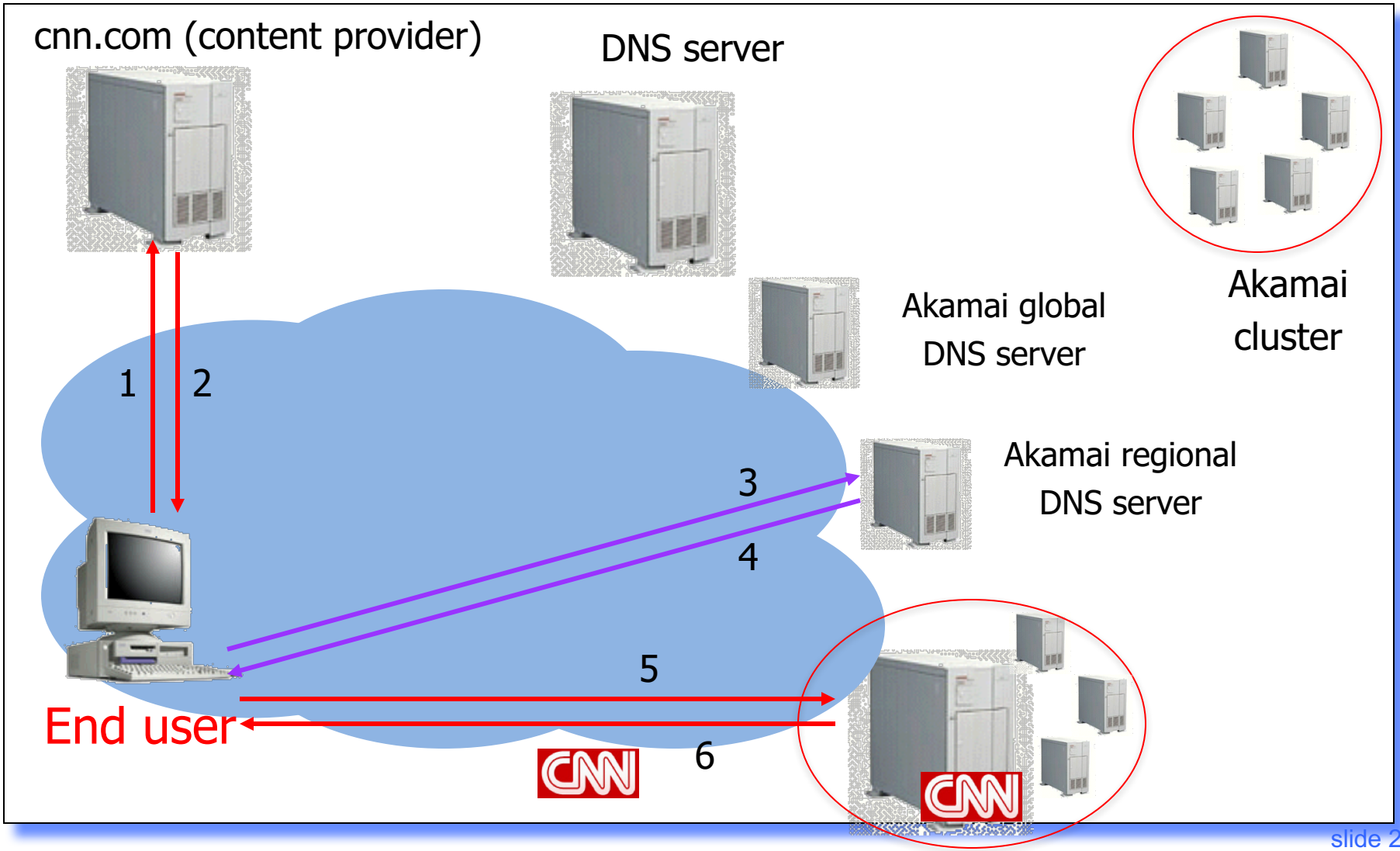
How Akamai Uses DNS



How Akamai Uses DNS



How Akamai Uses DNS



Akamai Statistics (Old)

- Distributed servers
  - Servers: ~100,000
  - Networks: ~1,000
  - Countries: ~70
- Customers
  - Apple, BBC, FOX, GM IBM, MTV, NASA, NBC, NFL, NPR, Puma, Red Bull, Rutgers, SAP, …
- Client requests
  - 20+M per second
  - Half in the top 45 networks
  - 20% of all Web traffic worldwide

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
    - Over 100 TB of logs per days
  - Network latency, loss, throughput, and connectivity

Routing Client Requests

- Create map of the Internet
  - BGP peering sessions with Internet border routers → coarse-grained AS map of the Internet

    + live traceroute, loss measurements betw. CDN servers
- Map each IP class to a preferred server cluster
  - Based on performance, cluster health, network status
  - Updated roughly every minute
    - Short, 60-sec DNS TTLs in Akamai regional DNS accomplish this
- Map client request to a server in the cluster
  - Load balancer selects a specific server
    - For example, to maximize the cache hit rate

Adapting to Failures

- Failing hard drive on a server
  - Suspends after finishing "in progress" requests
- Failed server
  - Another server takes over for the IP address
  - Low-level map updated quickly (load balancer)
- Failed cluster, or network path
  - High-level map updated quickly (ping/traceroute)

Take-Away Points on CDN

- Content distribution is hard
  - Many diverse, changing objects
  - Clients distributed all over the world
- Moving content to the client is key
  - Reduces latency, improves throughput, reliability
- Content distribution solutions evolved from load balancing and reactive caching to proactive content distribution networks