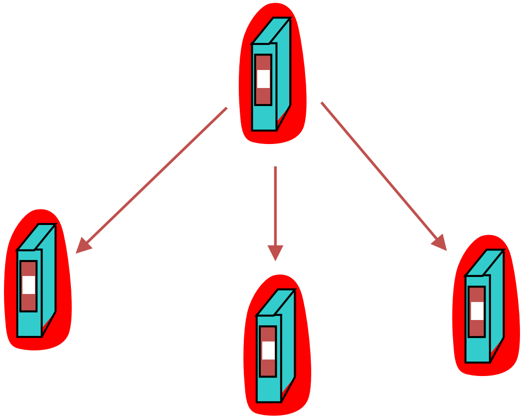
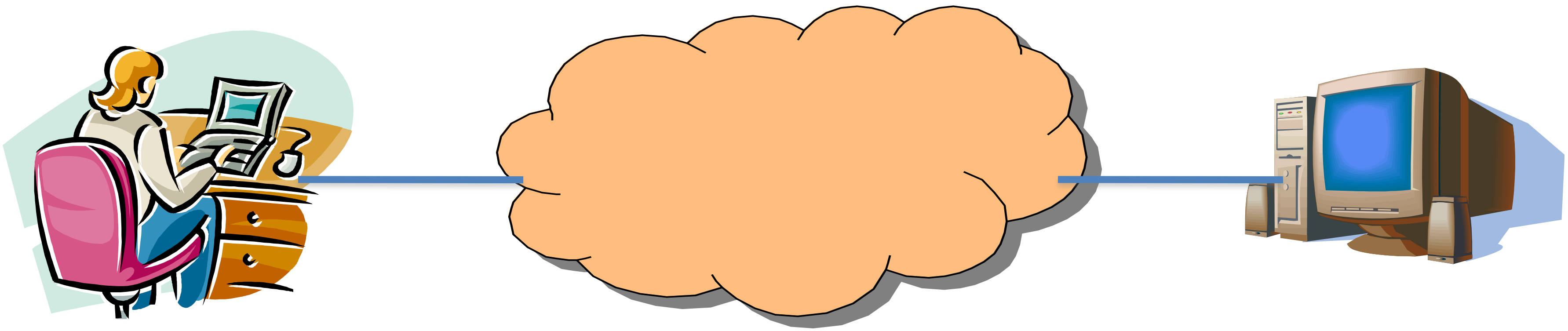


Content Distribution Networks (CDNs)

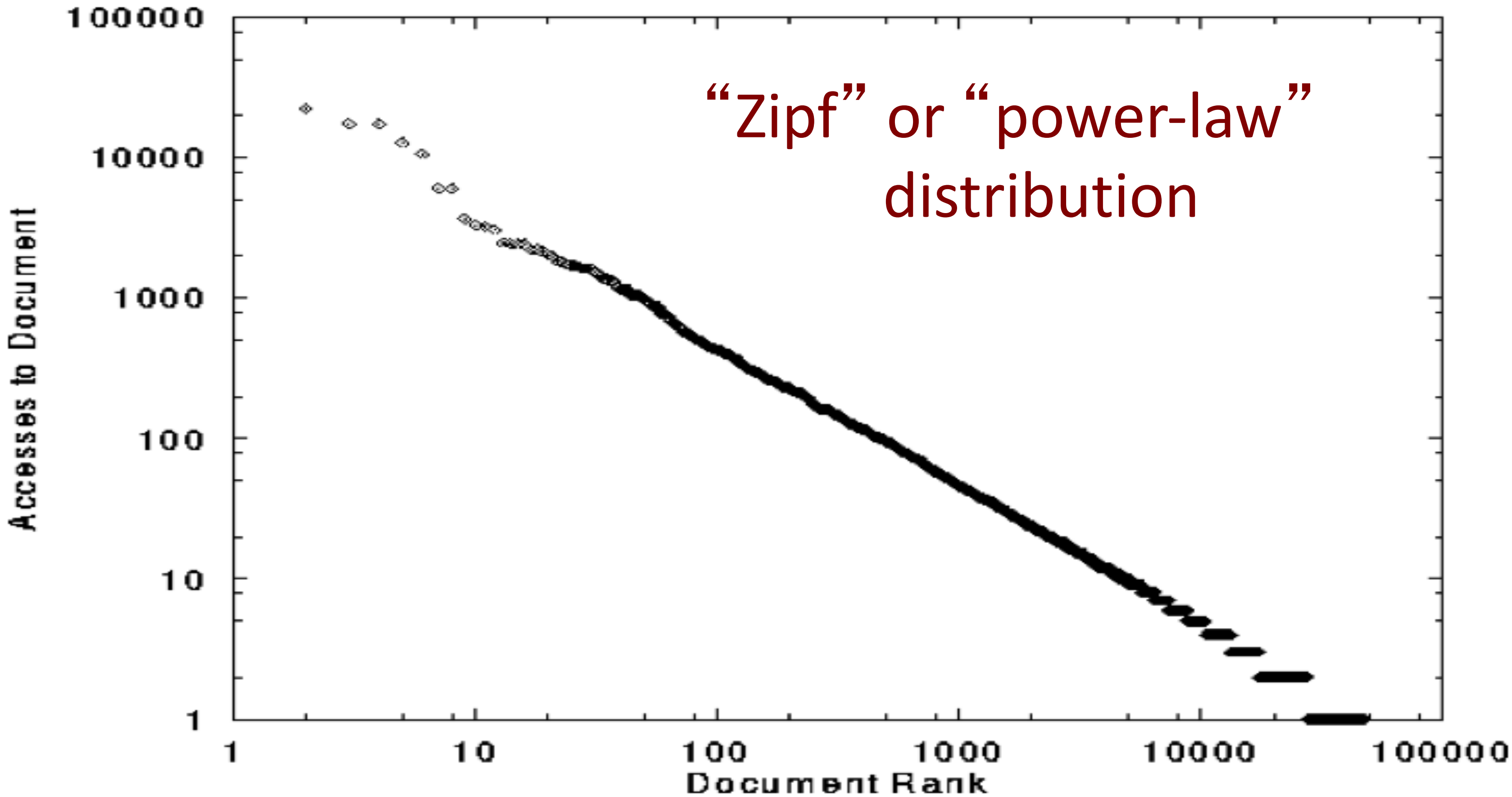
Note: The slides are adapted from the materials from Prof. Richard Han at CU Boulder and Profs. Jennifer Rexford and Mike Freedman at Princeton University, and the networking book (Computer Networking: A Top Down Approach) from Kurose and Ross.

Single Server, Poor Performance



- Single server
  - Single point of failure
  - Easily overloaded
  - Far from most clients
- Popular content
  - Popular site
  - “Flash crowd” (aka “Slashdot effect”)
  - Denial of Service attack

Skewed Popularity of Web Traffic

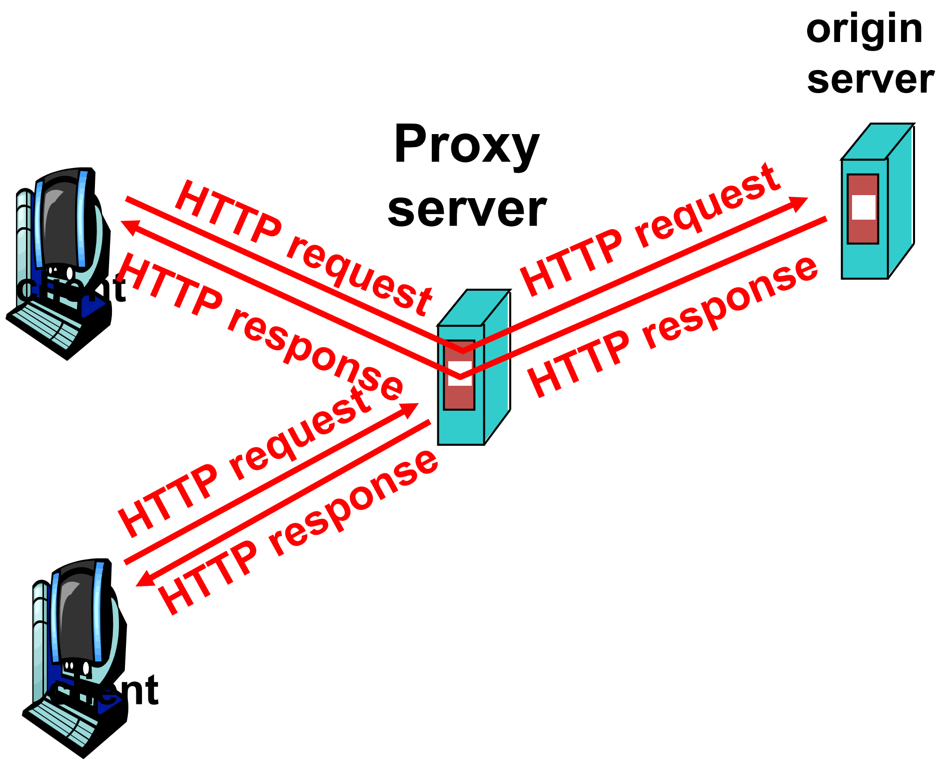


Characteristics of WWW Client-based Traces

Carlos R. Cunha, Azer Bestavros, Mark E. Crovella, BU-CS-95-01

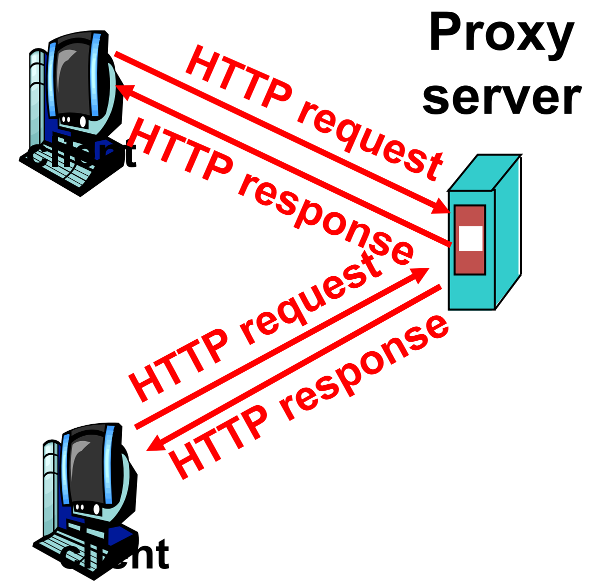
Web Caching

Proxy Caches



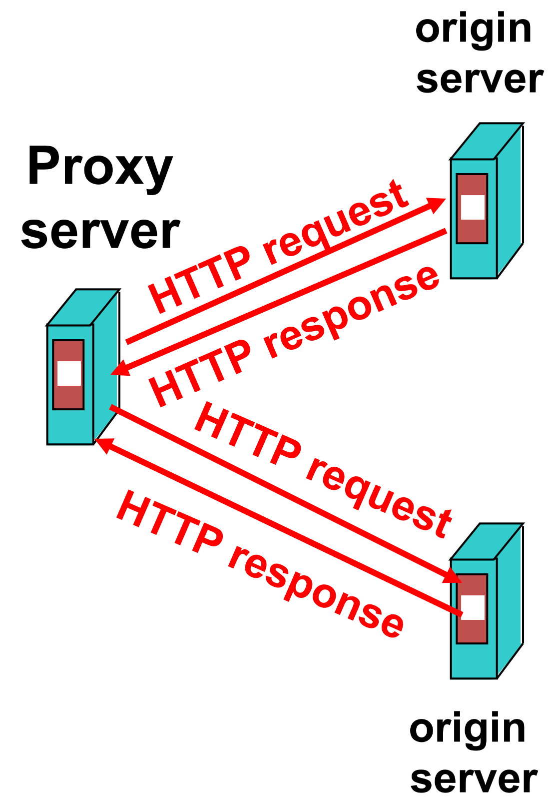
Forward Proxy

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

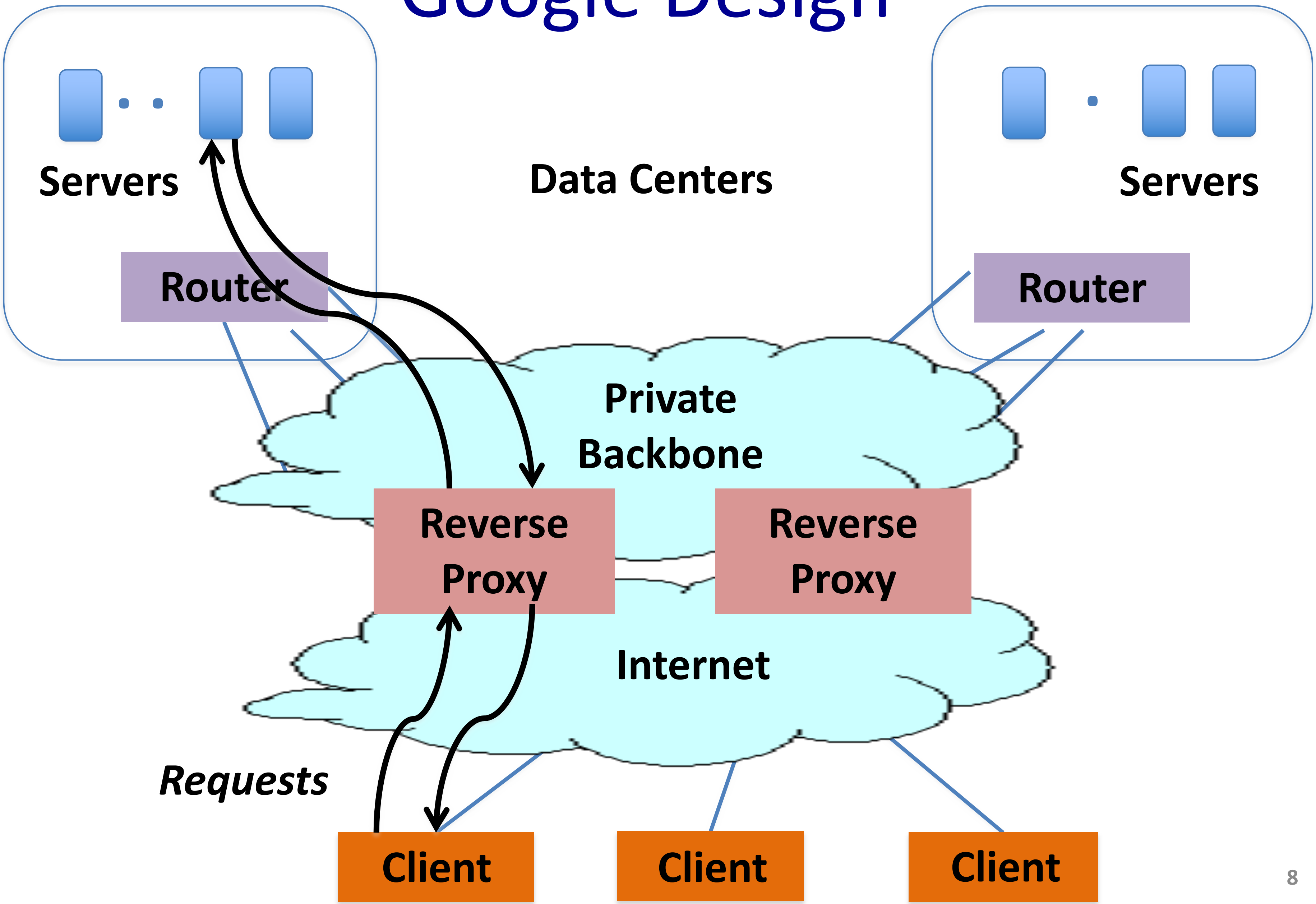


Reverse Proxy

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



Google Design



Proxy Caches

(A) Forward (B) Reverse (C) Both (D) Neither

- 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, transcoding on behalf of origin
- Smaller round-trip times to clients
- Maintain persistent connections to avoid TCP setup delay (handshake, slow start)

Proxy Caches

(A) Forward (B) Reverse (C) Both (D) Neither

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

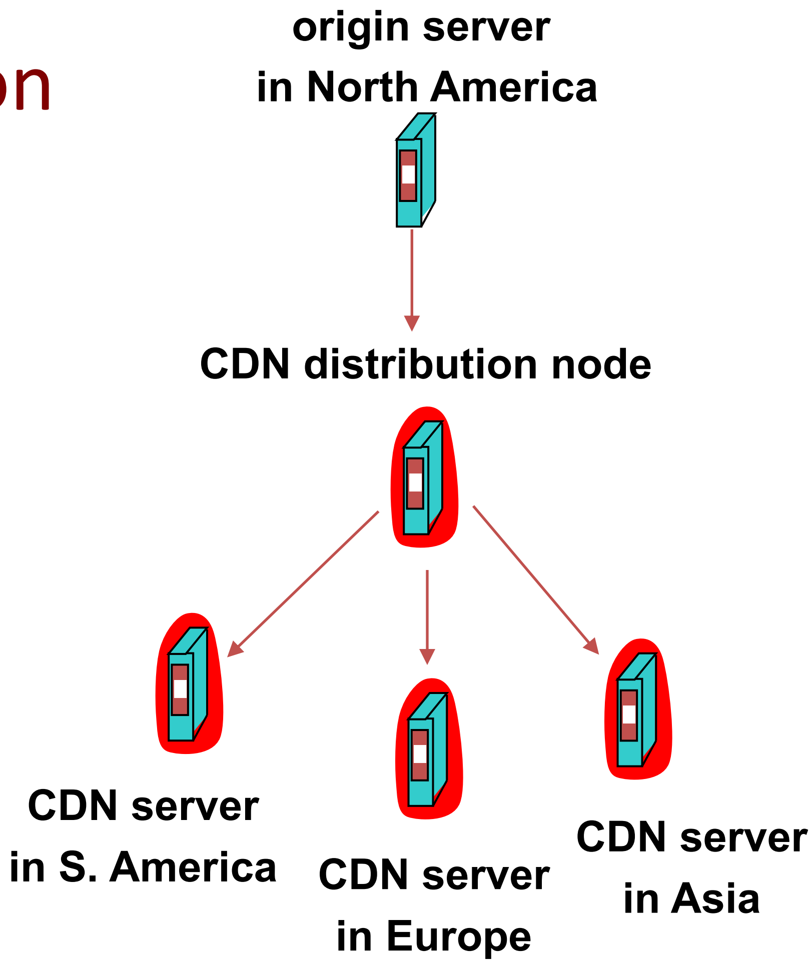
Limitations of Web Caching

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

Content Distribution Networks

Content Distribution Network

- 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 pushed to replicas when the content changes



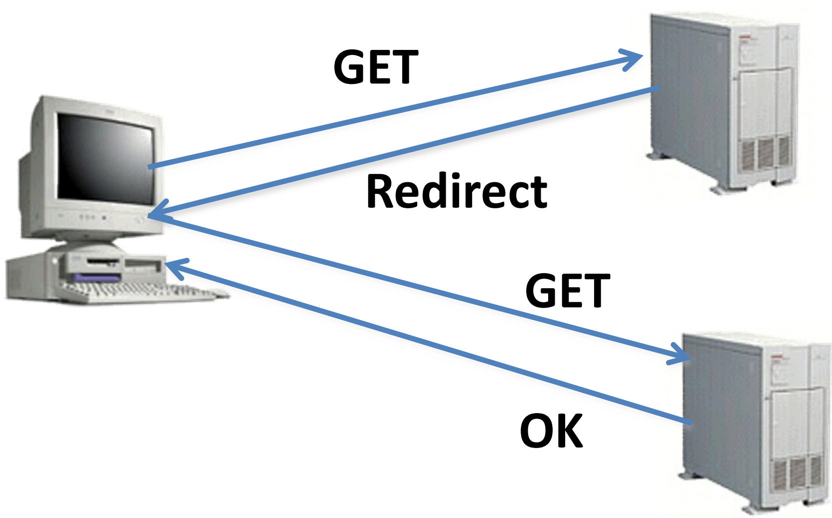
Server Selection Policy

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

Requires continuous monitoring of liveness, load, and performance

Server Selection Mechanism

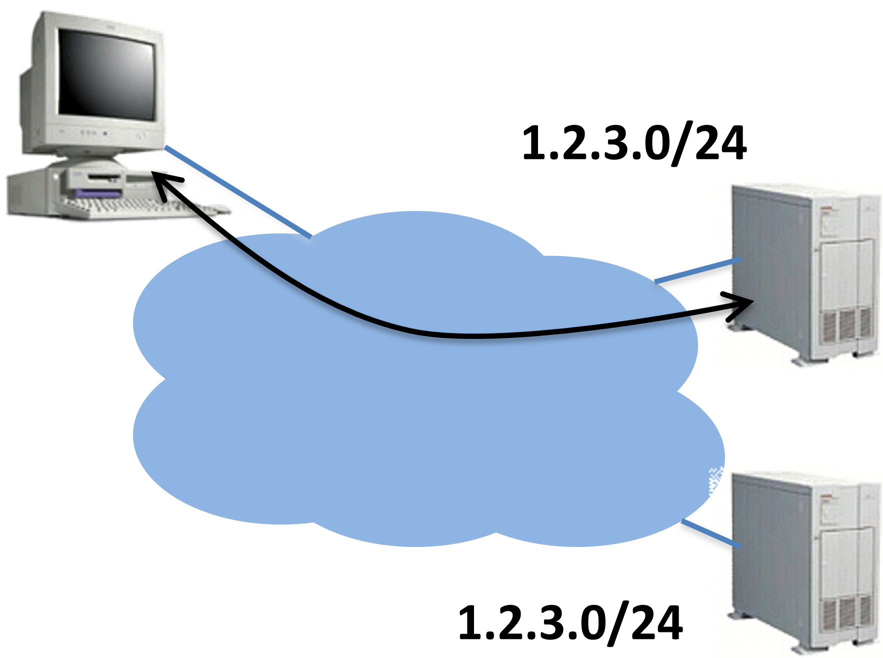
- Application
  - HTTP redirection



- 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

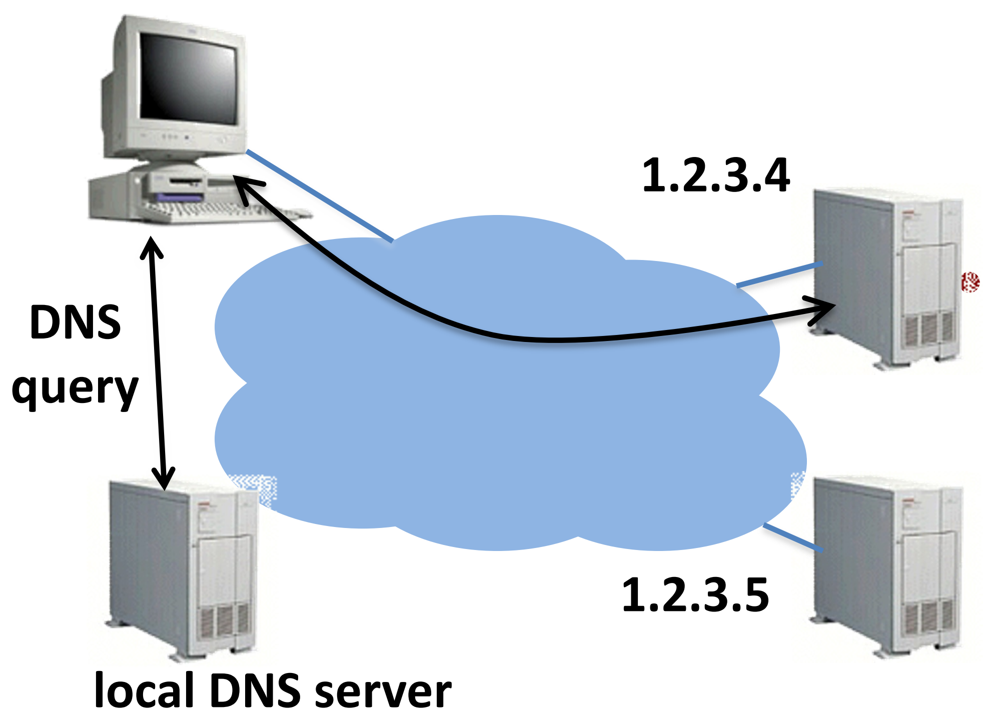
- Routing
  - Anycast routing



- 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



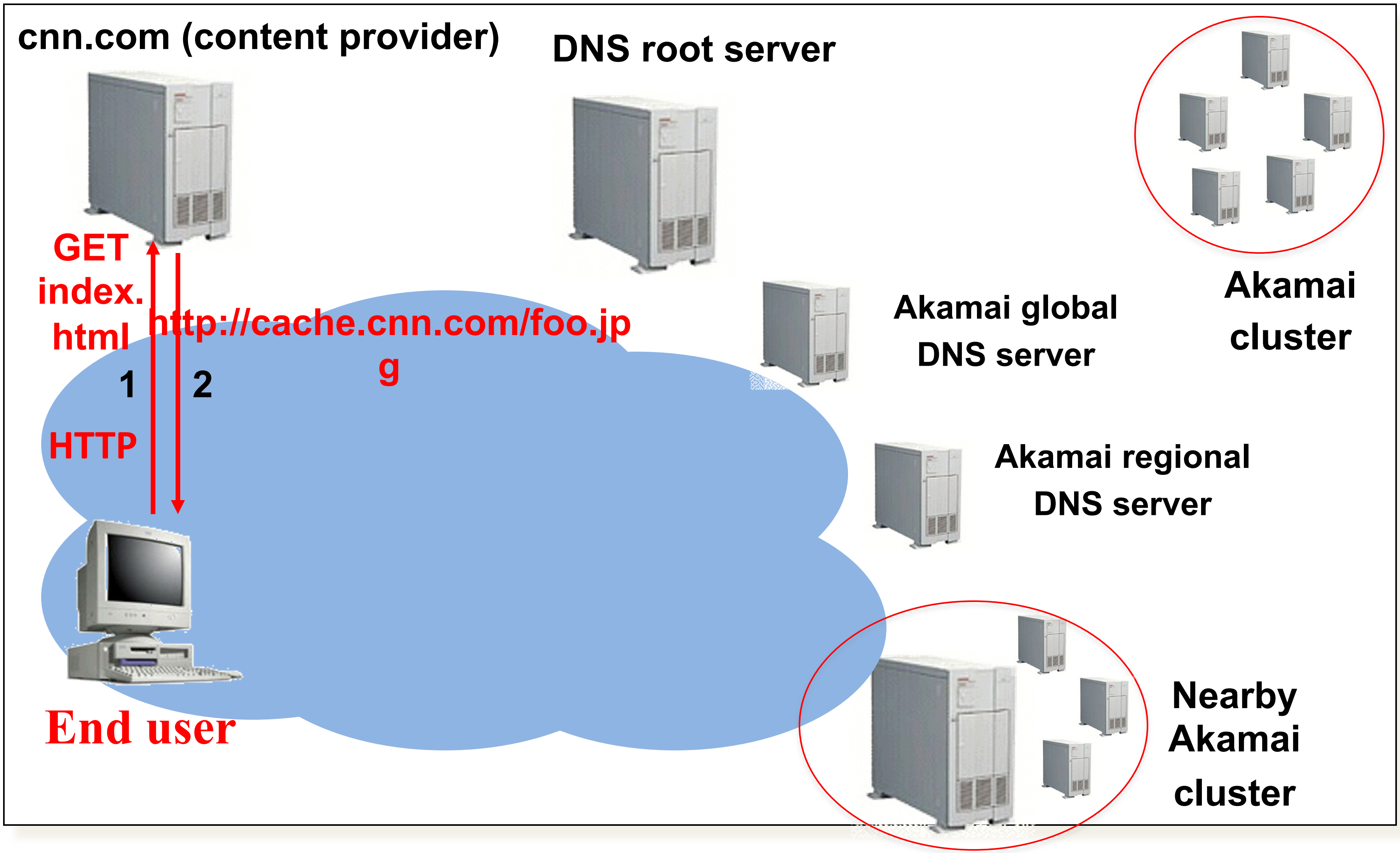
- Advantages
  - Avoid TCP set-up delay
  - DNS caching reduces overhead
  - Relatively fine control
- Disadvantage
  - Based on IP address of local DNS server
  - "Hidden load" effect
  - DNS TTL limits adaptation

How Akamai Works

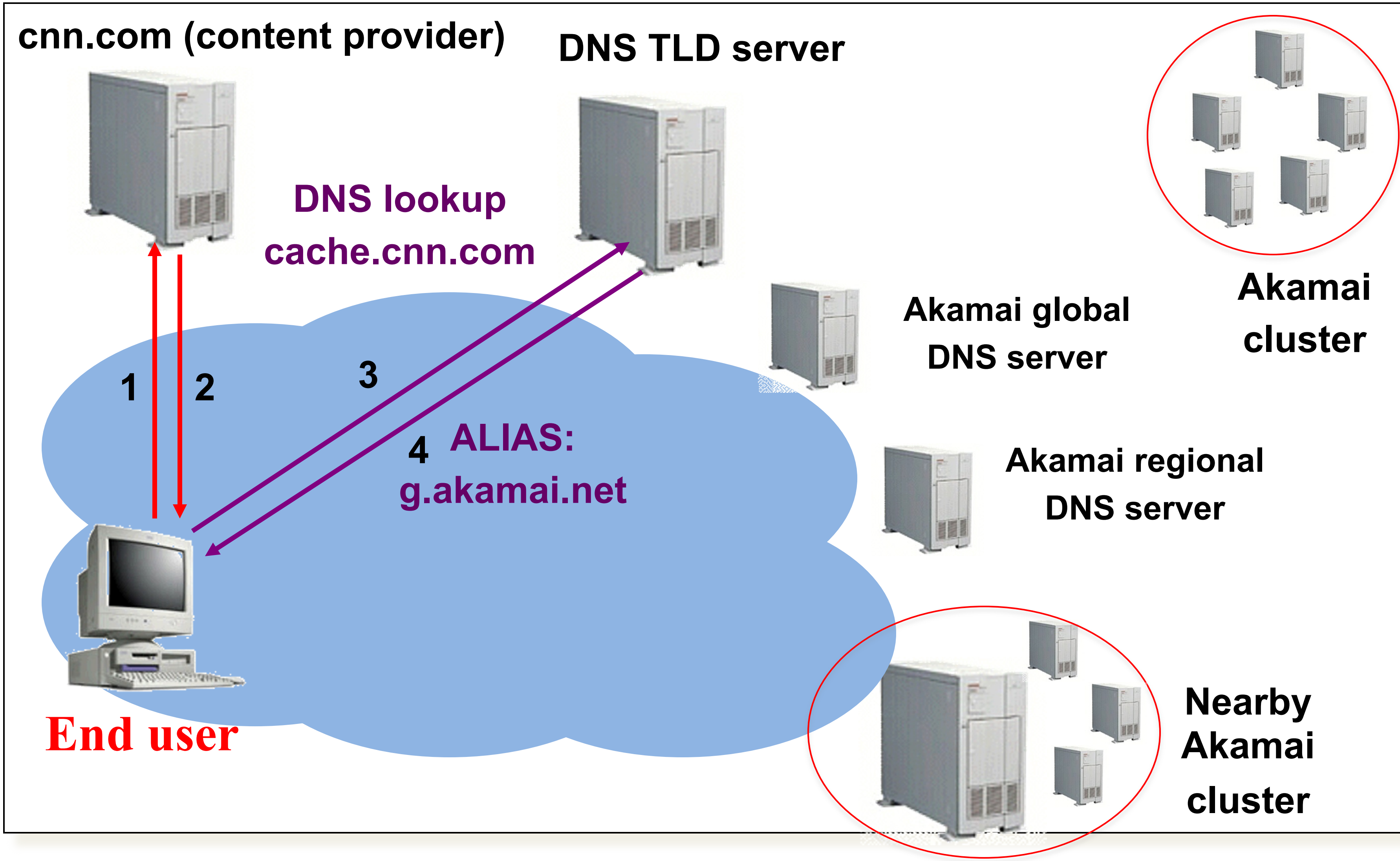
Akamai Statistics

- Distributed servers
  - Servers: ~100,000
  - Networks: ~1,000
  - Countries: ~70
- Many 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

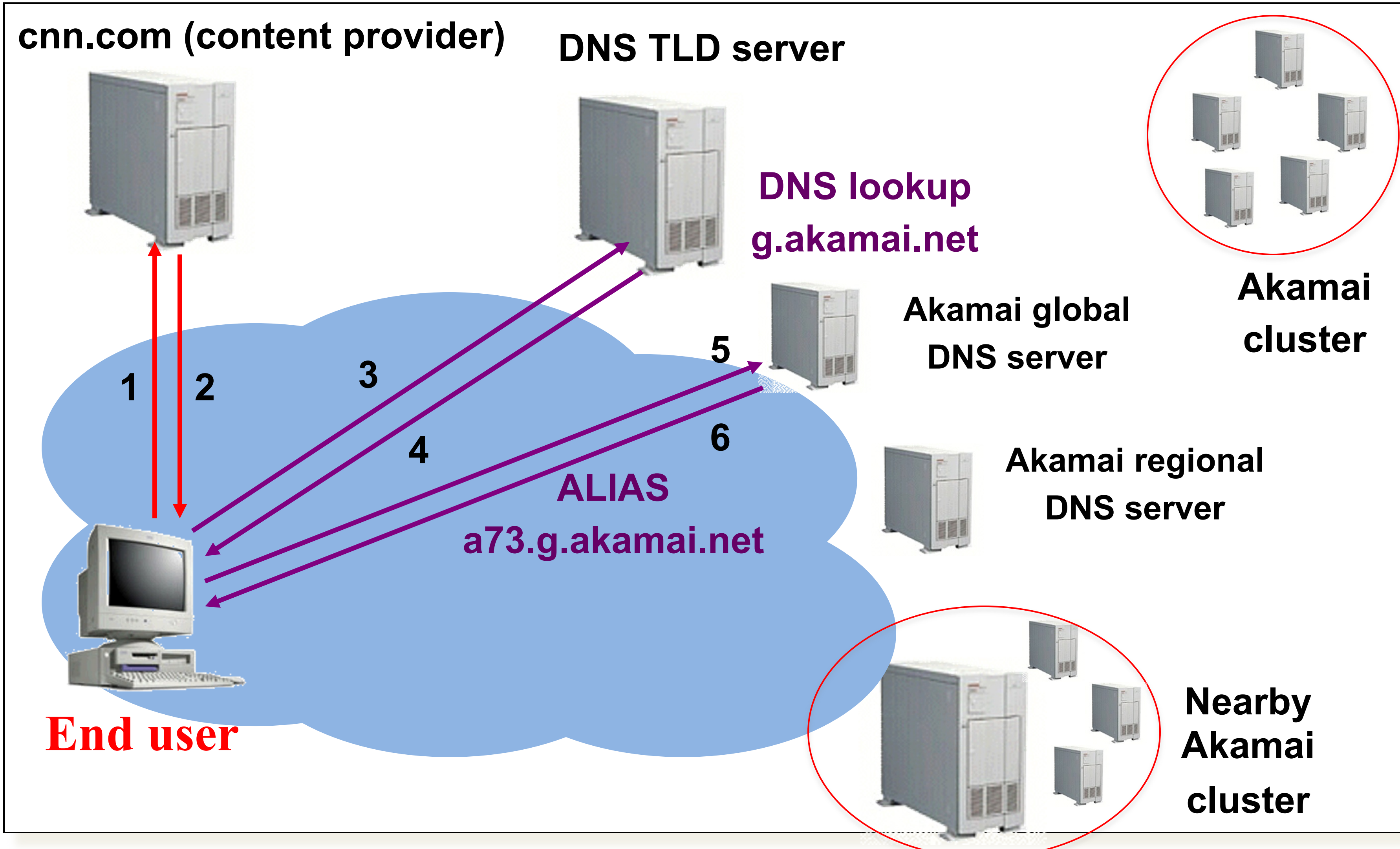
How Akamai Uses DNS



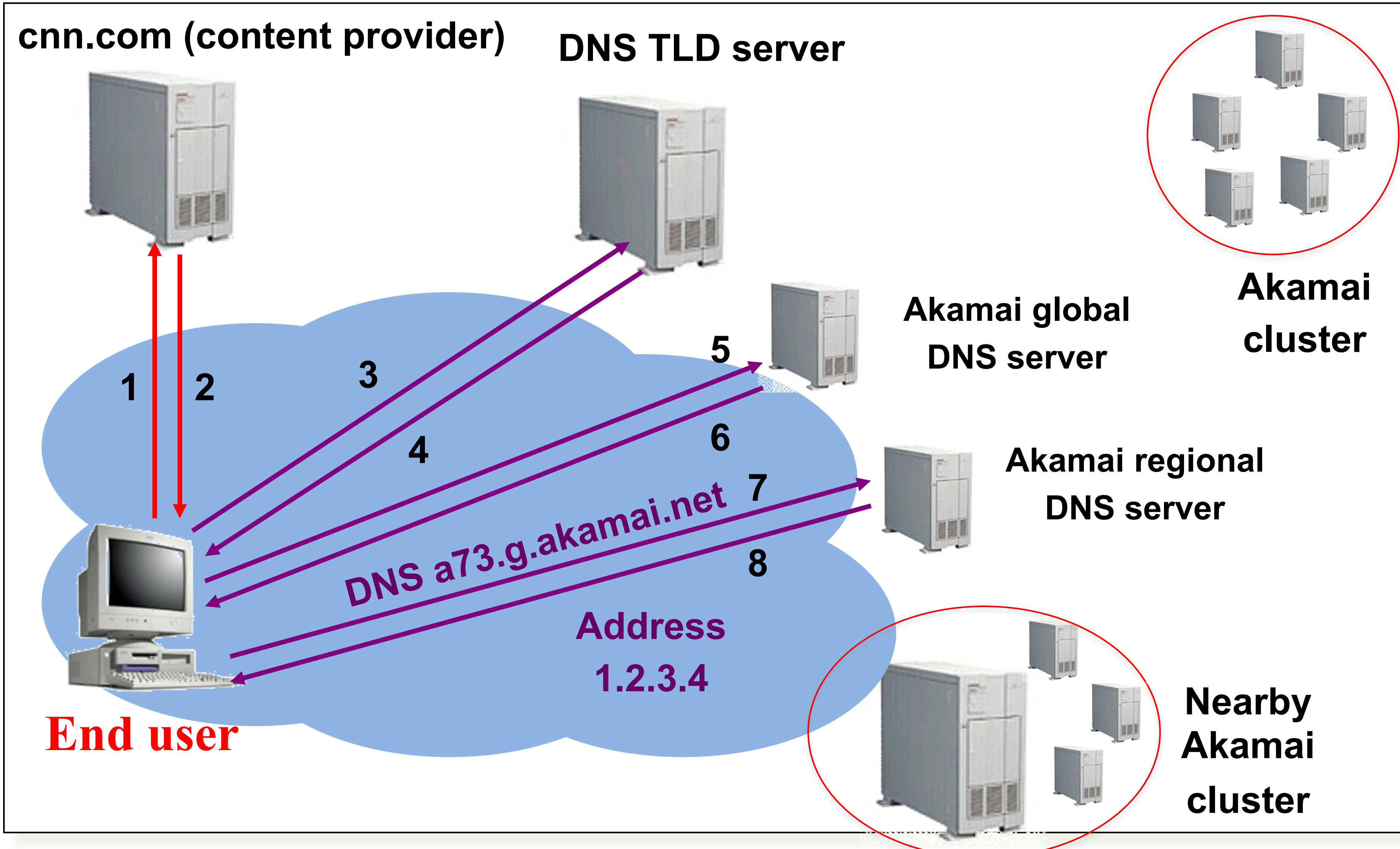
How Akamai Uses DNS



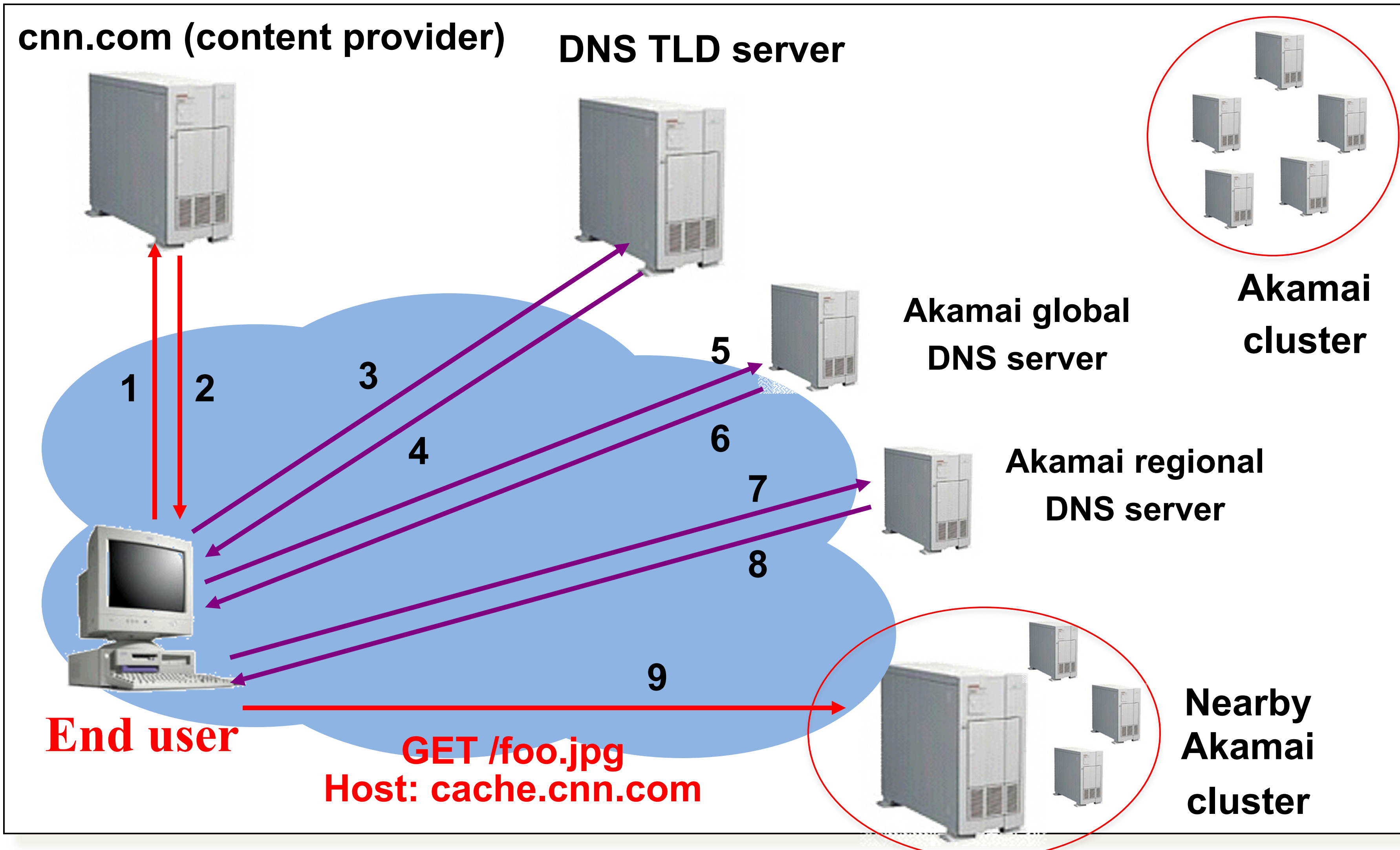
How Akamai Uses DNS



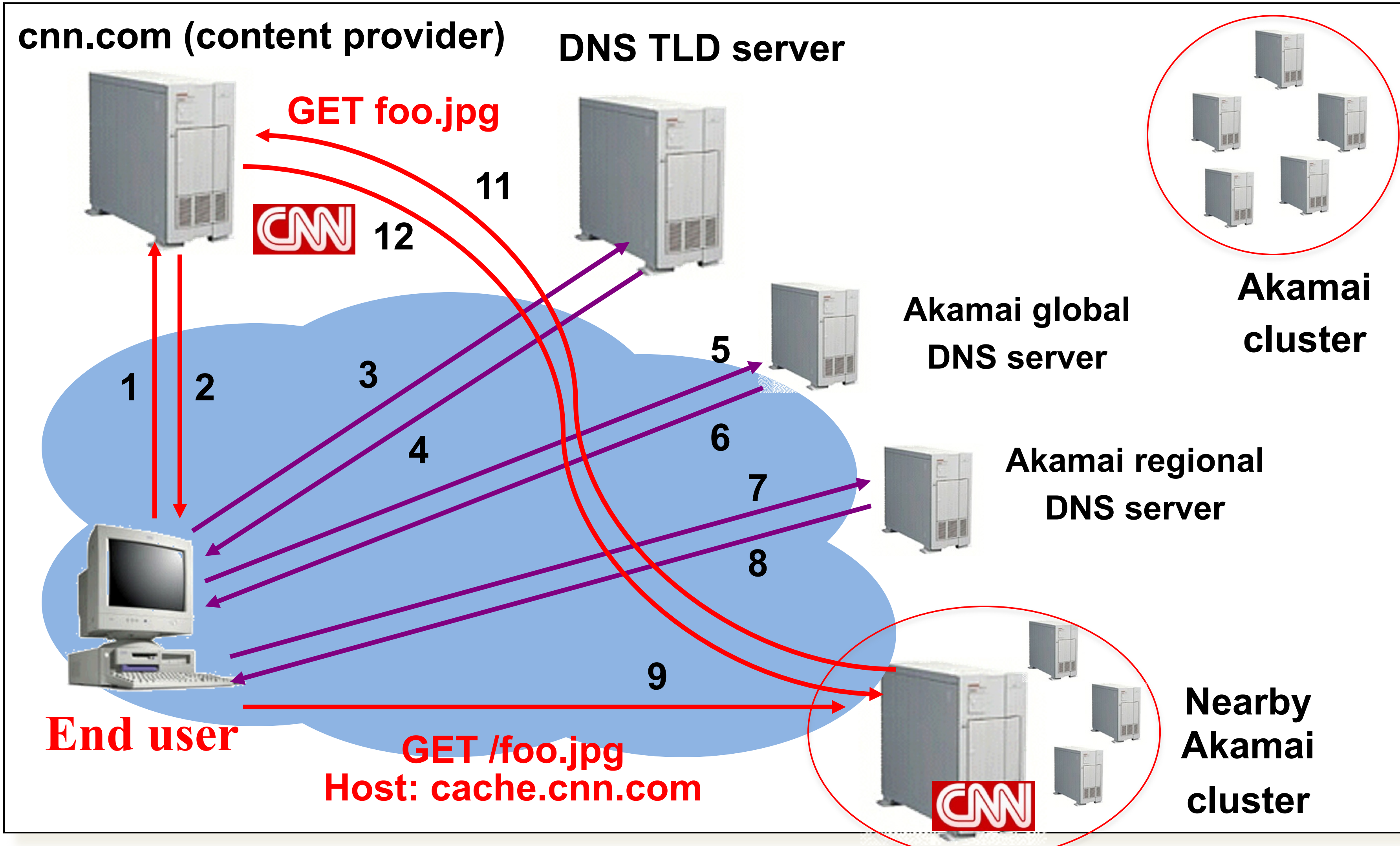
How Akamai Uses DNS



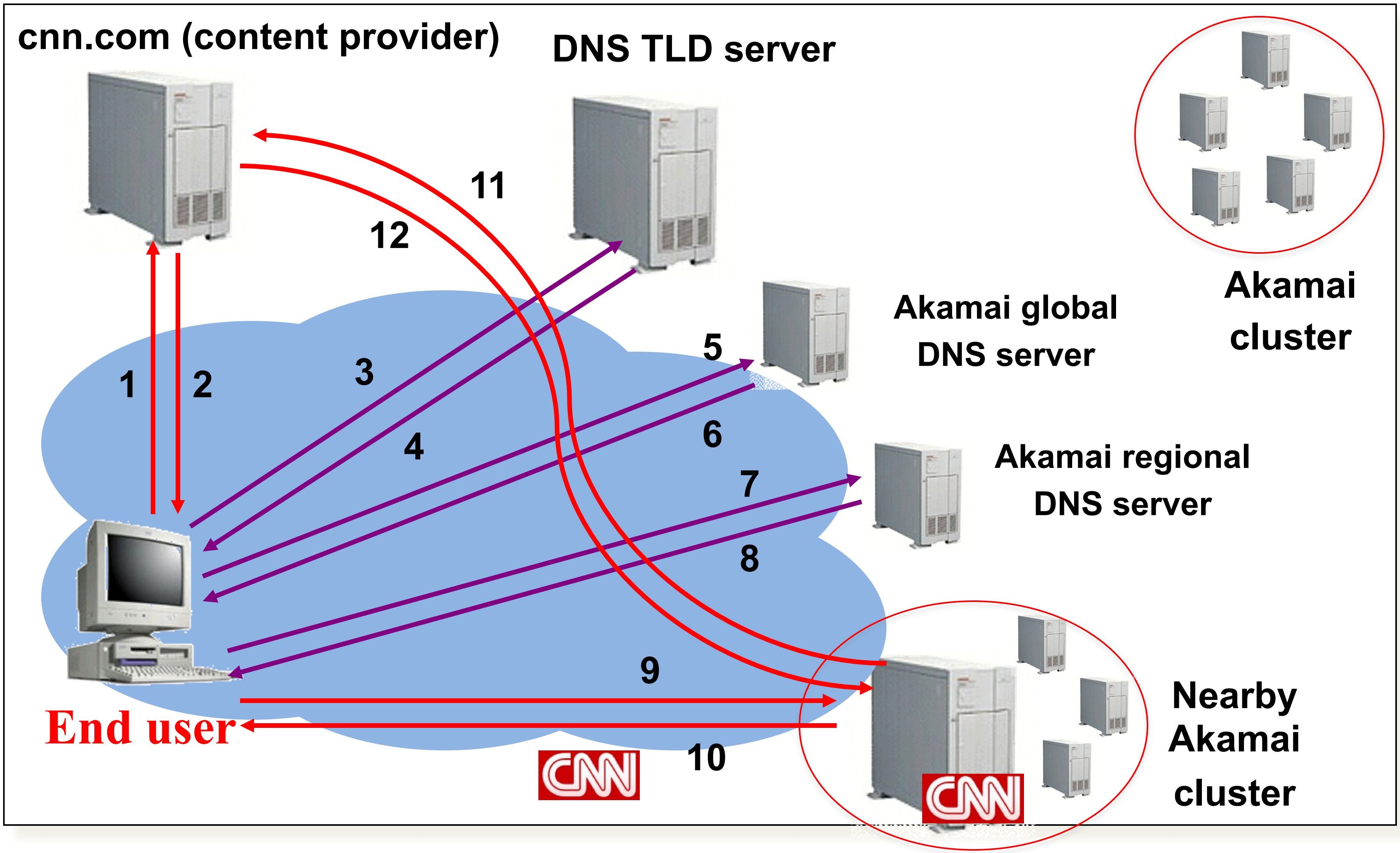
How Akamai Uses DNS



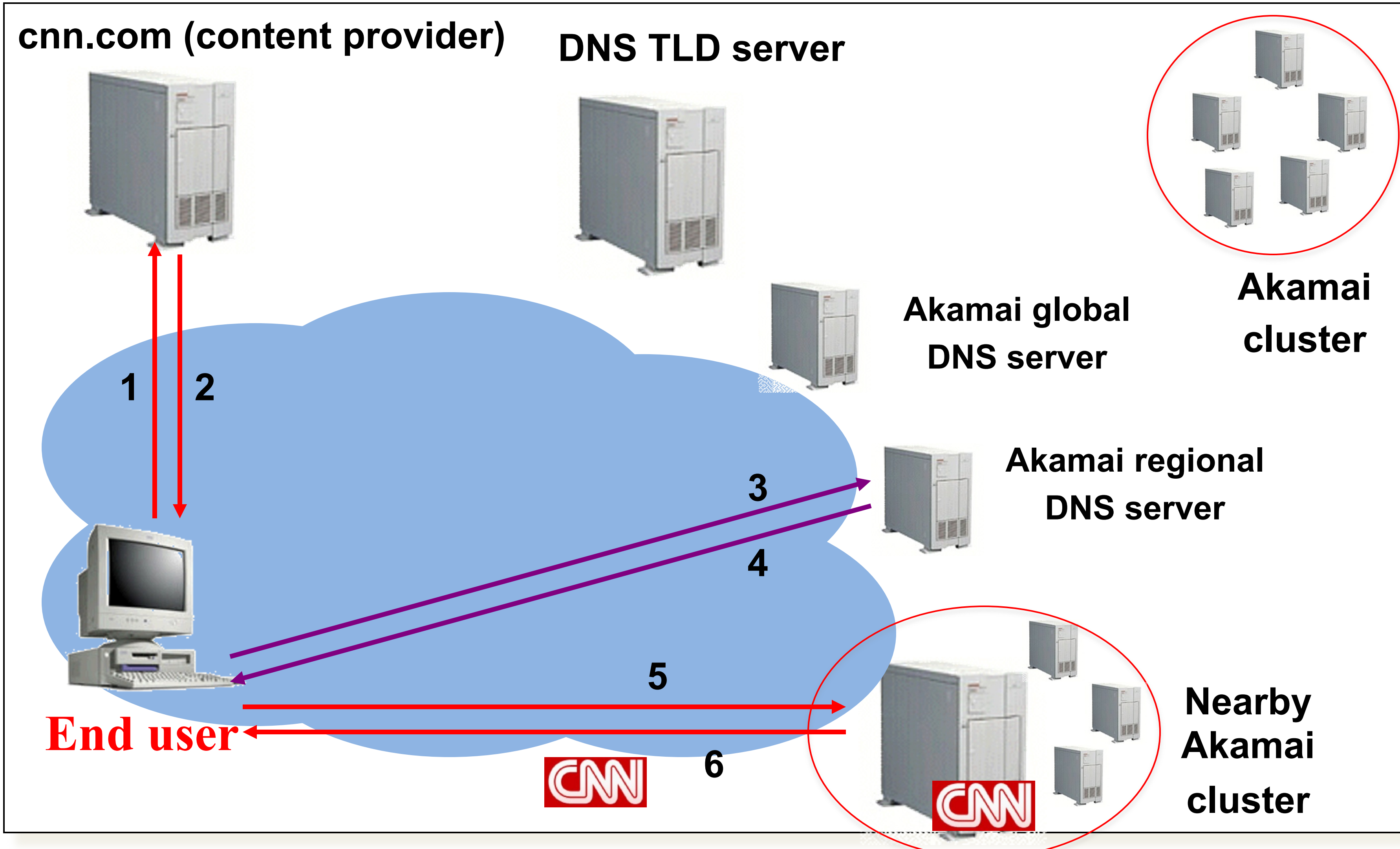
How Akamai Uses DNS



How Akamai Uses DNS



How Akamai Works: Cache Hit



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
    - E.g., over 100 TB of logs per days
  - Network latency, loss, and connectivity

Mapping System

- 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 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
- Failed cluster
  - High-level map updated quickly
- Failed path to customer's origin server
  - Route packets through an intermediate node

Akamai Transport Optimizations

- Bad Internet routes
  - Overlay routing through an intermediate server
- Packet loss
  - Sending redundant data over multiple paths
- TCP connection set-up/teardown
  - Pools of persistent connections
- TCP congestion window and round-trip time
  - Estimates based on network latency measurements

Akamai Application Optimizations

- Slow download of embedded objects
  - Prefetch when HTML page is requested
- Large objects
  - Content compression
- Slow applications
  - Moving applications to edge servers
  - E.g., content aggregation and transformation
  - E.g., static databases (e.g., product catalogs)

Modern HTTP Video-on-Demand

- Download "content manifest" from origin server
- List of video segments belonging to video
  - Each segment 1-2 seconds in length
  - Client can know time offset associated with each
  - Standard naming for different video resolutions and formats: e.g., 320dpi, 720dpi, 1040dpi, …
- Client downloads video segment (at certain resolution) using standard HTTP request.
  - HTTP request can be satisfied by cache: it's a static object
- Client observes download time vs. segment duration, increases/decreases resolution if appropriate

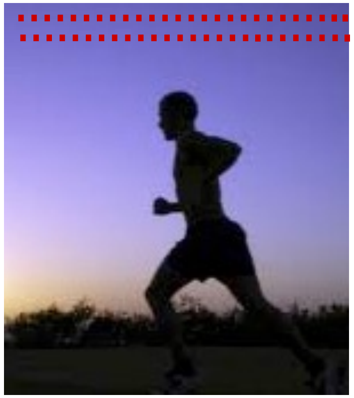
Video Streaming and CDNs: context

- **video traffic: major consumer of Internet bandwidth**
  - **Netflix, YouTube: 37%, 16% of downstream residential ISP traffic**
  - **~1B YouTube users, ~75M Netflix users**
- **challenge: scale - how to reach ~1B users?**
  - **single mega-video server won't work (why?)**
- **challenge: heterogeneity**
  - **different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)**
- ***solution:* distributed, application-level infrastructure**

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: **instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) and number of repeated values (N)**



frame *i*

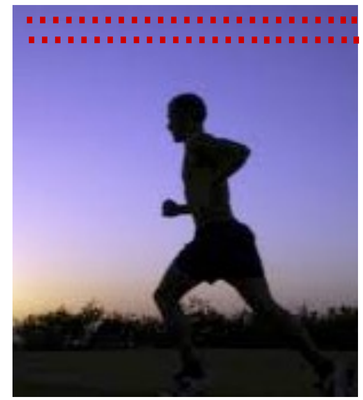
temporal coding example: **instead of sending complete frame at i+1, send only differences from frame i**

frame *i+1*

Multimedia: video

- **CBR: (constant bit rate):** video encoding rate fixed
- **VBR: (variable bit rate):** video encoding rate changes as amount of spatial, temporal coding changes
- **examples:**
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



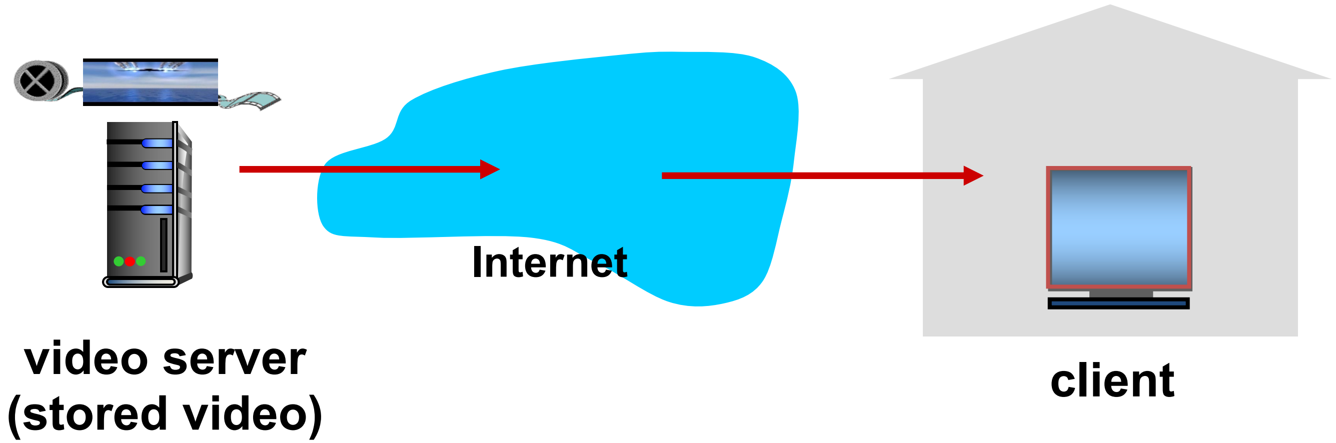
frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *DASH: Dynamic, Adaptive Streaming over HTTP*
- "*intelligence*" at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

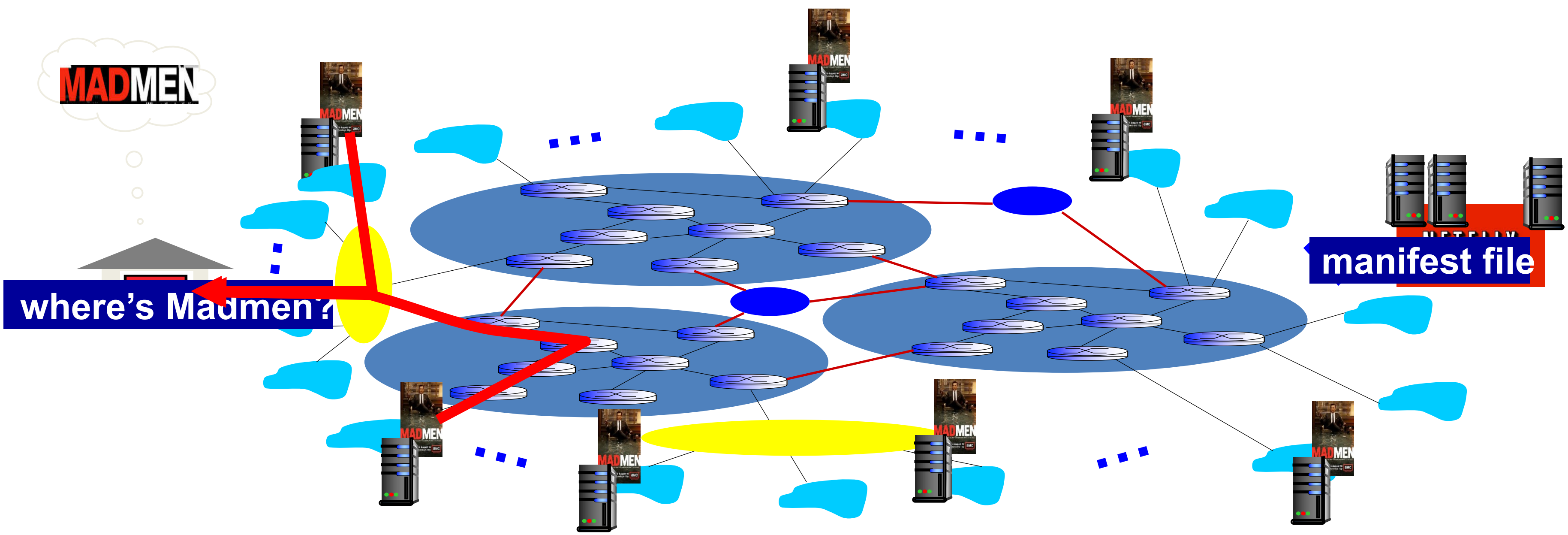
….quite simply: this solution *doesn't scale*

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?
- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

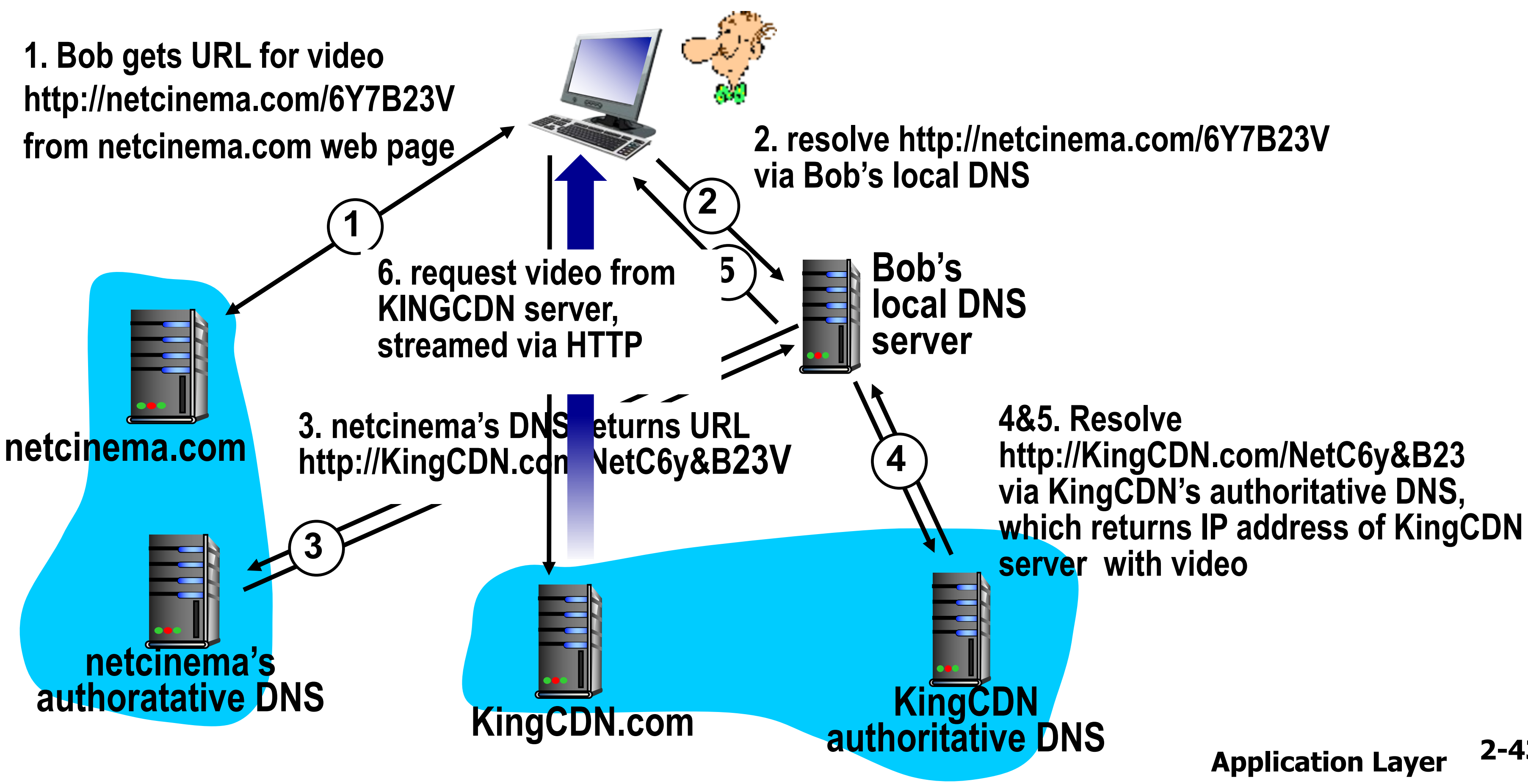
- **CDN: stores copies of content at CDN nodes**
  - **e.g. Netflix stores copies of MadMen**
- subscriber requests content from CDN
  - **directed to nearby copy, retrieves content**
  - **may choose different copy if network path congested**



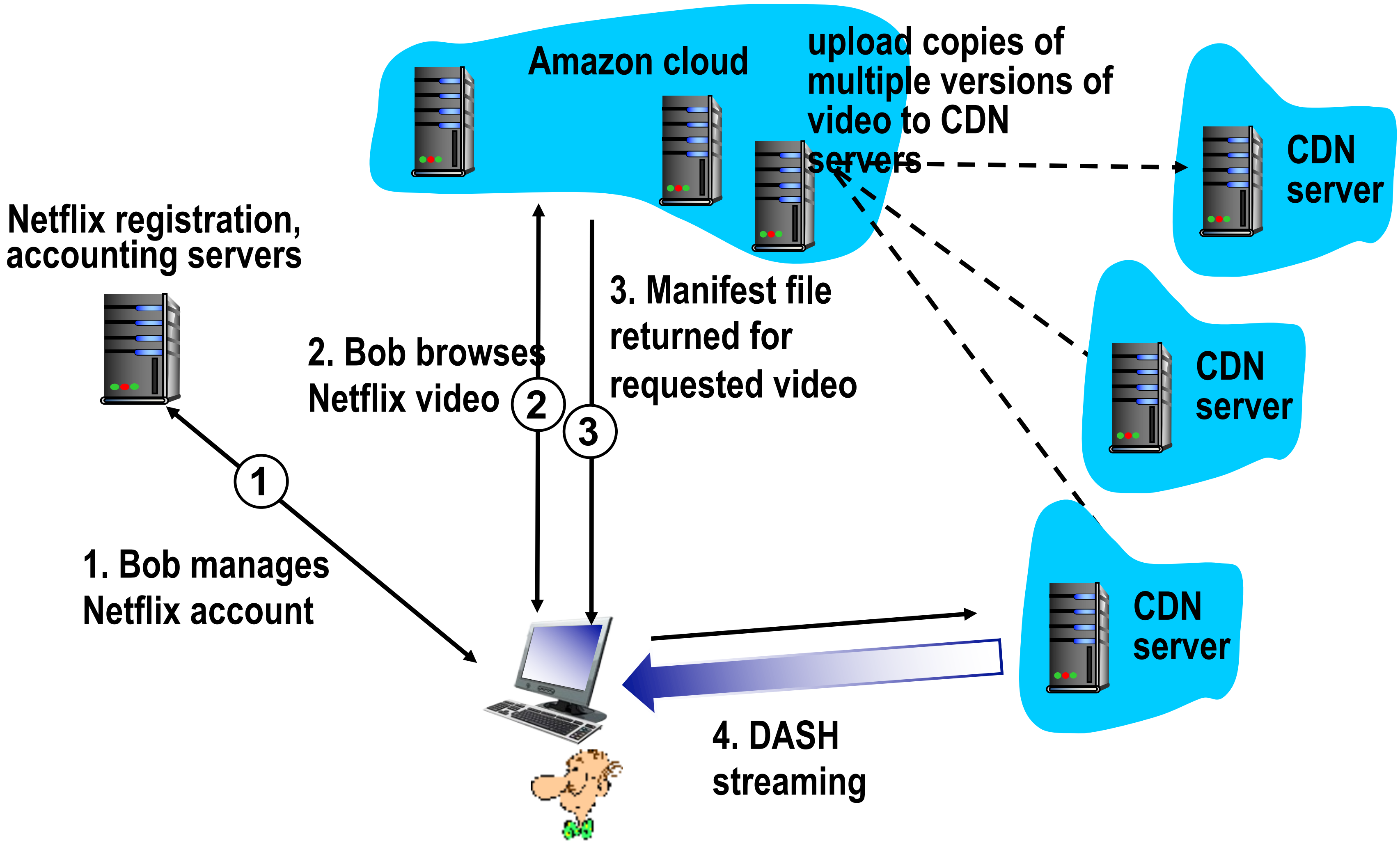
CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V

- **video stored in CDN at http://KingCDN.com/NetC6y&B23V**



Case study: Netflix



Conclusion

- Content distribution is hard
  - Many, diverse, changing objects
  - Clients distributed all over the world
  - Reducing latency is king
- Contribution distribution solutions
  - Reactive caching
  - Proactive content distribution networks