HTTP and the Web

Overview

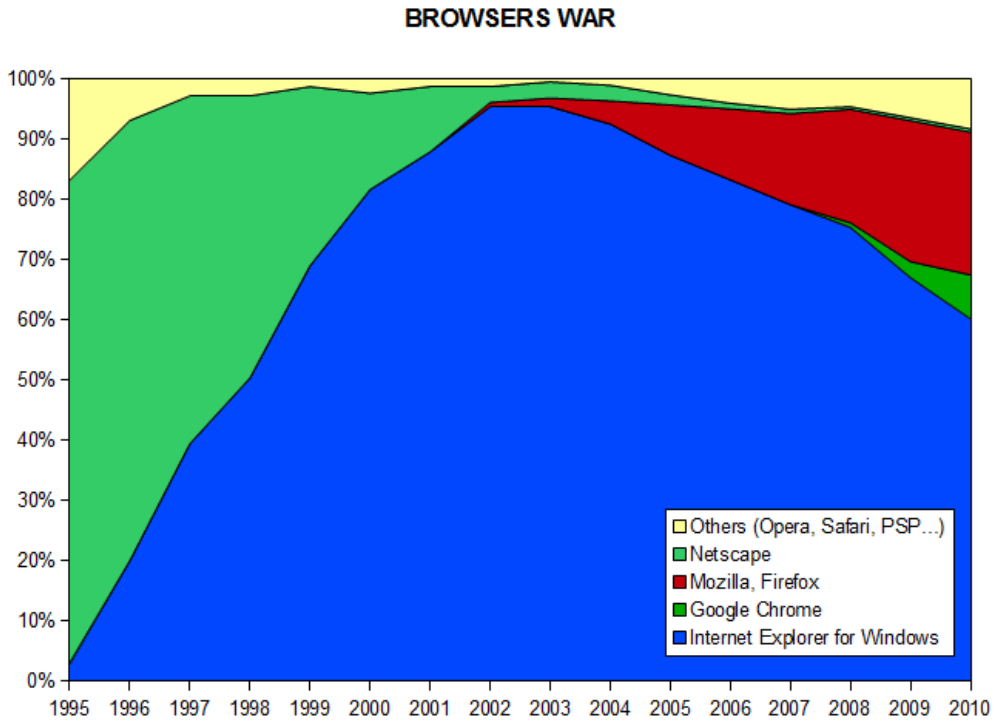
- The birth of the web
- Main components of the web
  - URLs
  - HTML
  - HTTP
- Caching
- Context Distribution Networks

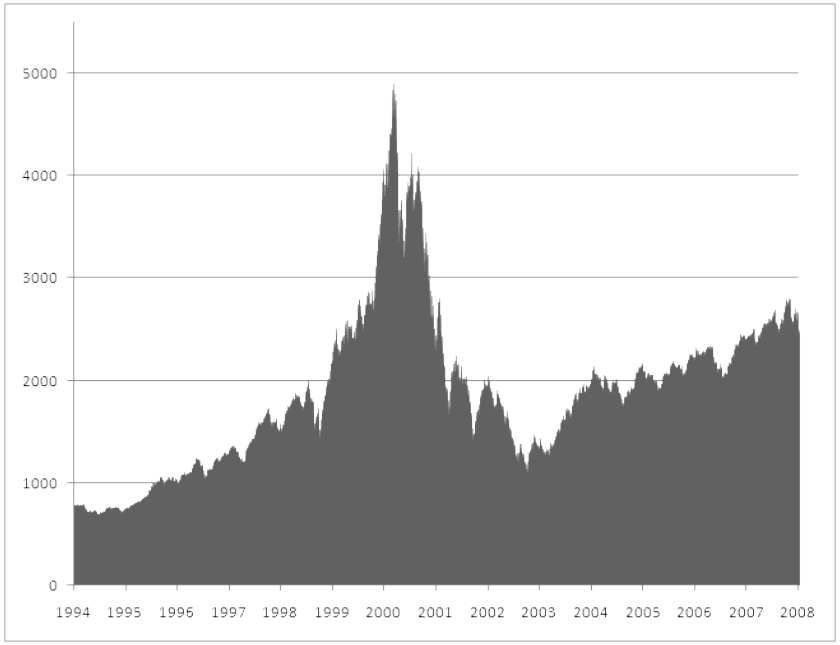
A short history of the web

- 1989 Tim Berners-Lee at CERN
- 1990 HTTP/0.9, HTML, URLs, first text-based browser
- 1993 Marc Andreesen releases NCSA Mosaic, graphical browser
- 1993 CERN agrees to release protocol royalty-free
- 1994 Andreesen forms Netscape
- 1994 W3C formed, standardizing protocols, encouraging interoperability

A short history of the web

- 1994+ Browser wars between Netscape and IE
- 1990s-2000 Dot com era





In the Web's first generation, Tim Berners-Lee launched the Uniform Resource Locator (URL), Hypertext Transfer Protocol (HTTP), and HTML standards with prototype Unix-based servers and browsers.

A few people noticed that the
Web might be better than Gopher.

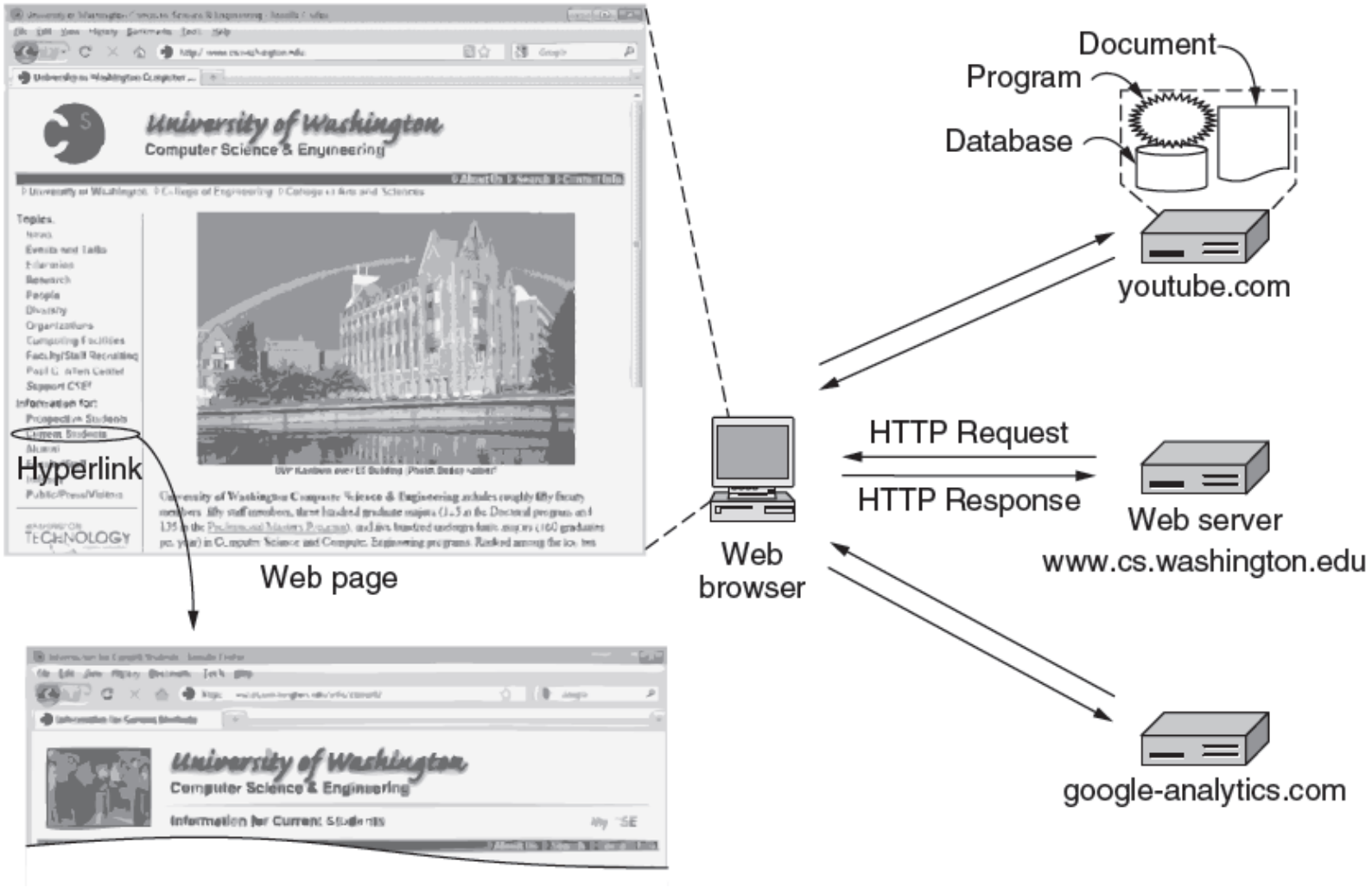
In the second generation, Marc Andreessen and Eric Bina developed NCSA Mosaic at the University of Illinois.

Several million then suddenly noticed that the
Web might be better than sex.

In the third generation, Andreessen and Bina left NCSA to found Netscape...

Microsoft and Netscape open some new fronts in escalating Web Wars
By Bob Metcalfe, InfoWorld, August 21, 1995, Vol. 17, Issue 34.

Architecture of the web



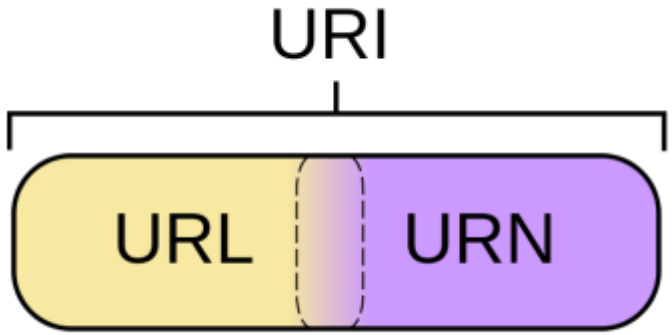
Web components: finding stuff

- Uniform Resource Locator (URL)
  - A page's worldwide name
  - Three parts:
    - Protocol (scheme)
    - DNS name of machine
    - Hierarchical name that models a file directory structure

Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

Web components: finding stuff

- URL points to one specific host
- Uniform Resource Identifier (URI)
  - Say what you want, but not necessarily where from
  - Uniform Resource Locators (URL)
    - http://www.amazon.com/Last-Unicorn-Peter-S-Beagle/dp/0451450523
  - Uniform Resource Name (URN)
    - urn:isbn:0451450523



Web components: HTML

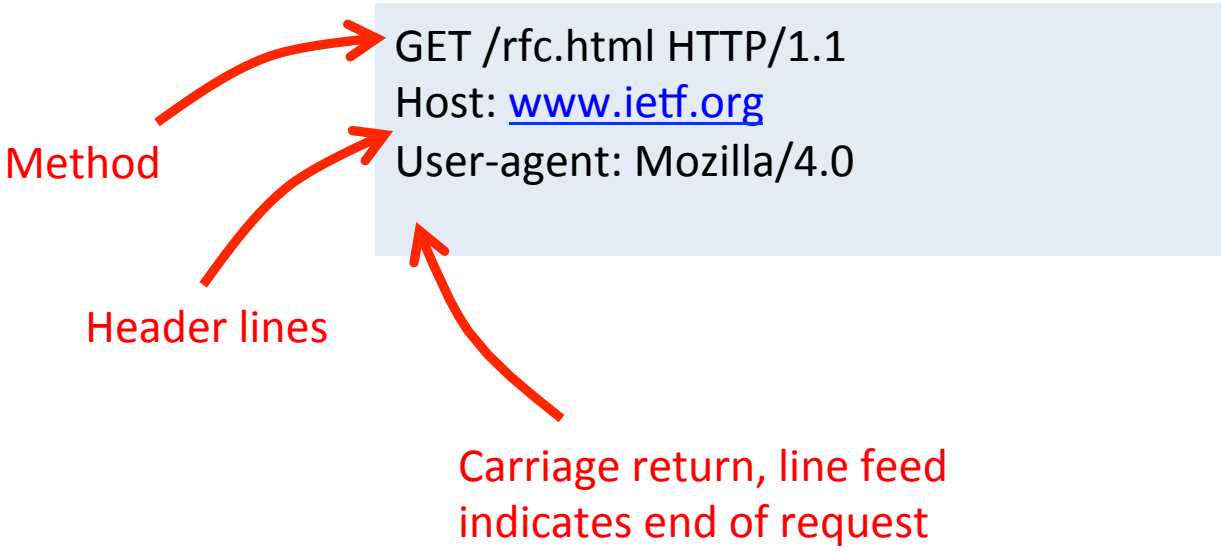
- HyperText Markup Language (HTML)
  - Represents hypertext documents in ASCII form
  - Format text, add images, embed hyperlinks
  - Web browser renders
- Simple and easy to learn
  - Hack up in any text editor
  - Or use a fancy authoring program
- Web page
  - Base HTML file references objects
  - Each object has its own URL

HTML versions

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

Web components: HTTP

- HyperText Transfer Protocol (HTTP)
  - Simple request-response protocol
  - Runs over TCP
  - ASCII format request and response headers



Request methods

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

```
GET /rfc.html HTTP/1.1
Host: www.ietf.org
User-agent: Mozilla/4.0
```

```
POST /login.html HTTP/1.1
Host: www.store.com
User-agent: Mozilla/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded

userid=joe&password=guessme
```

Message headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
If-Modified-Since	Request	Time and date to check freshness
If-None-Match	Request	Previously sent tags to check freshness
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Referer	Request	The previous URL from which the request came
Cookie	Request	Previously set cookie sent back to the server
Set-Cookie	Response	Cookie for the client to store
Server	Response	Information about the server

Message headers

Content-Encoding	Response	How the content is encoded (e.g., *gzip*)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Content-Range	Response	Identifies a portion of the page's content
Last-Modified	Response	Time and date the page was last changed
Expires	Response	Time and date when the page stops being valid
Location	Response	Tells the client where to send its request
Accept-Ranges	Response	Indicates the server will accept byte range requests
Date	Both	Date and time the message was sent
Range	Both	Identifies a portion of a page
Cache-Control	Both	Directives for how to treat caches
ETag	Both	Tag for the contents of the page
Upgrade	Both	The protocol the sender wants to switch to

HTTP response

- Response from server
  - Status line: protocol version, status code, status phrase
  - Response headers: extra info
  - Body: optional data

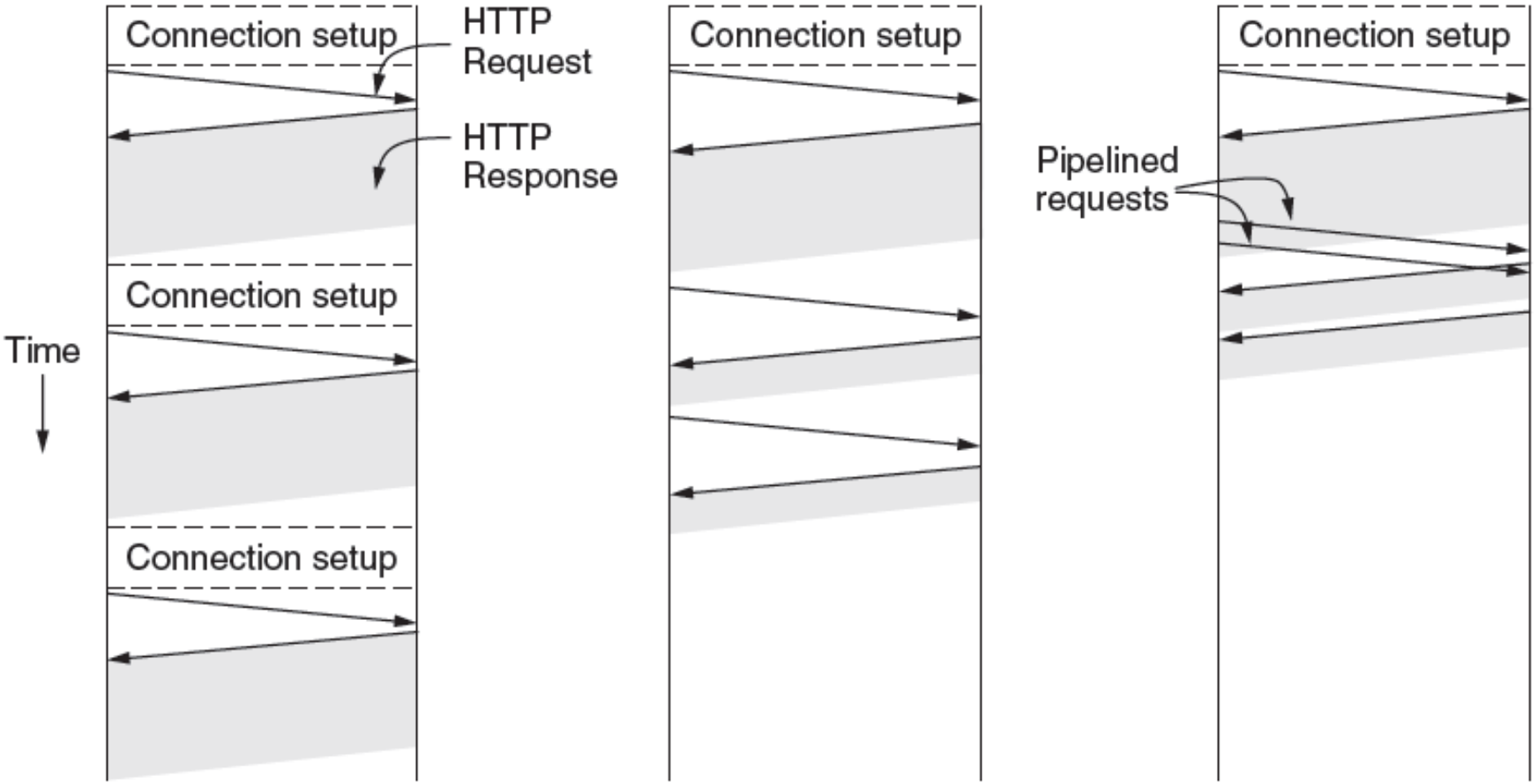
```
HTTP/1.1 200 OK
Date: Thu, 17 Nov 2011 15:54:10 GMT
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 14 Sep 2011 17:04:27 GMT
Content-Length: 285

<html> …
```

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

HTTP protocol

- HTTP/1.0
  - Early in the web
  - TCP connection established, page retrieved, connection tore down
  - Hard on TCP, can't ramp up from slow start
  - Not enough packets in flight for fast retransmission (which needs 3 dup ACKs).
- HTTP/1.1
  - Added support for persistent and pipelined requests

HTTP communication



Multiple connections and sequential requests.

Persistent connection and sequential requests.

Persistent connection and pipelined requests.

Persistent HTTP

- Non-persistent HTTP
  - OS must allocate resources for each TCP connection
  - Browsers often open parallel TCP connections
  - Requires 2 RTTs per object
- Persistent HTTP
  - Server leave connection open after sending response
  - Subsequent HTTP messages from same client/server sent over same connection
  - Client issues new request only when previous response has been received
  - 1 RTT per object

Persistent HTTP

- Persistent with pipelining
  - Clients sends multiple requests without waiting for previous
  - Requests need to be idempotent (e.g. GET)
  - As little as 1 RTT for whole page
  - Server must send responses in same order as requests
  - Default in HTTP/1.1 spec
  - Most browsers do not support/default to pipelining
  - Head-of-line blocking problem, use parallel connections instead

Multiple sites on one server

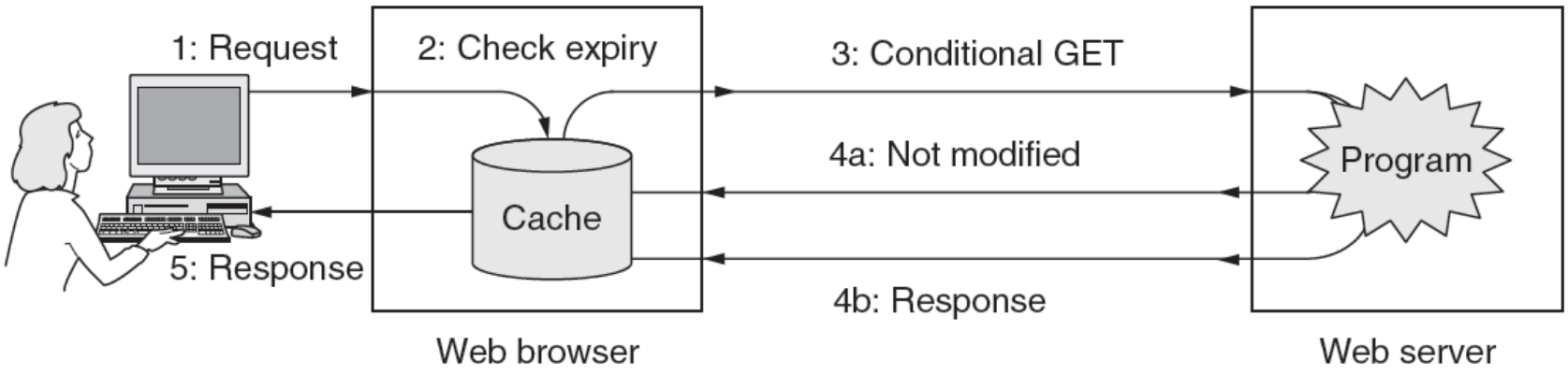
- Single server running host multiple web sites
  - Web hosting company with many companies on same physical server, e.g. www.widgets.com, www.junk.com
- How does it return correct response?
  - Solution 1: Each web site has a separate IP address
    - Server splits up based on IP address
    - Requires more IP addresses
  - Solution 2: Look in HTTP header host field
    - Mandatory in HTTP/1.1
    - Single server with a single IP address
    - Virtual hosting

Improving performance

- How do we make things faster?
- Minimize traffic between client/server
  - Conditional requests
  - Caching
  - Compression
- Speed up server's response
  - Multiple servers
  - Geographically distributed servers
  - Content delivery networks

HTTP Caching

- Clients often cache documents
  - How and when should they check for changes?
- HTTP has cache related headers
  - HTTP/1.0: "Expires: <date>"; "Pragma: no-cache"
  - HTTP/1.1:
    - Cache-Control: No-Cache, Private, Max-age: <seconds>
    - E-tag: <some value>



Conditional GETs

- Conditional GET
  - Fetch resource only if it has changed
  - Server avoids wasting resources to send again
  - Client sets "if-modified-since" header field
    - Server inspects "last modified" time of object
    - Returns "304 Not Modified" if unchanged, otherwise "200 OK" and new version.
  - Client sets "if-no-match" using previous received ETag for the desired object
    - Server compares with current "hash" of object

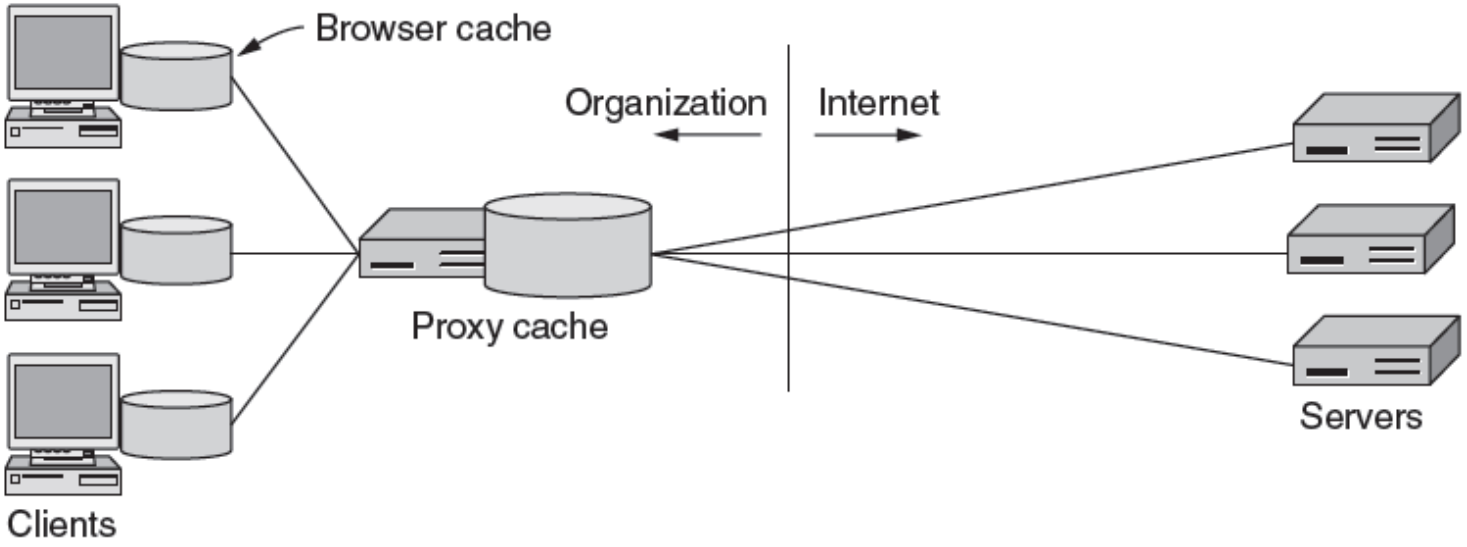
HTTP conditional request

```
GET / HTTP/1.1
Host: katie.mtech.edu
Connection: keep-alive
User-Agent: Mozilla/5.0
Accept: text/html, application/xhtml+xml
Accept-Encoding: gzip,deflate,sdch
If-None-Match: "c-221-4ace9c0b09cc0"
If-Modified-Since: Wed, 14 Sep 2011 17:04:27 GMT
```

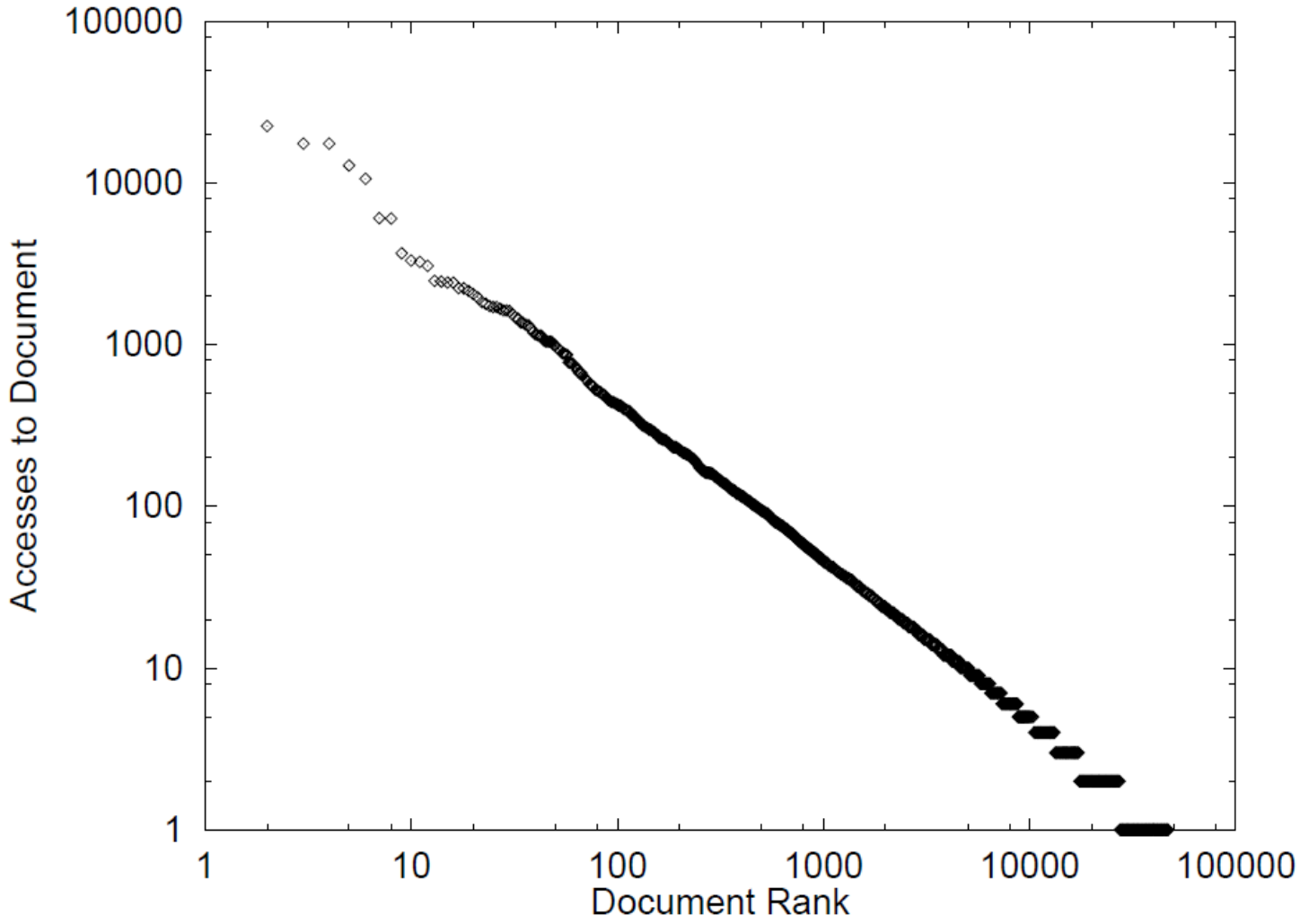
```
HTTP/1.1 304 Not Modified
Date: Thu, 17 Nov 2011 16:57:53 GMT
Server: Apache/2.2.16 (Debian)
Keep-Alive: timeout-15, max=100
Connection: Keep-Alive
ETag: " c-221-4ace9c0b09cc0"
```

Levels of caching

- Caching can occur at many levels:
  - In the client's browser
  - Client configs browser to use web proxy
  - Proxy at the ISP
  - "voluntary proxy" verus "intercepting/forced/transparent"



Long tail of unpopular pages



http://www.cs.bu.edu/techreports/pdf/1995-010-www-client-traces.pdf

Speeding up the server

- A single server doesn't scale
  - Even a really really powerful computer will not handle Netflix's traffic
- Server farms
  - Multiple servers each with replicated web site
- How do clients arrive at different servers?
  - Option 1: Client choose a "mirror" themselves
  - Option 2: DNS spreads request over a set of IP addresses pointing at different servers
  - Option 3: Front-end load balancing

dig youtube.com

;; ANSWER SECTION:

youtube.com.	3	IN	A	173.194.33.37
youtube.com.	3	IN	A	173.194.33.38
youtube.com.	3	IN	A	173.194.33.39
youtube.com.	3	IN	A	173.194.33.40
youtube.com.	3	IN	A	173.194.33.41
youtube.com.	3	IN	A	173.194.33.42

youtube.com.	300	IN	A	74.125.230.108
youtube.com.	300	IN	A	74.125.230.109
youtube.com.	300	IN	A	74.125.230.99
youtube.com.	300	IN	A	74.125.230.104
youtube.com.	300	IN	A	74.125.230.111
youtube.com.	300	IN	A	74.125.230.105
youtube.com.	300	IN	A	74.125.230.107
youtube.com.	300	IN	A	74.125.230.103
youtube.com.	300	IN	A	74.125.230.102
youtube.com.	300	IN	A	74.125.230.106
youtube.com.	300	IN	A	74.125.230.97
youtube.com.	300	IN	A	74.125.230.98
youtube.com.	300	IN	A	74.125.230.110
youtube.com.	300	IN	A	74.125.230.96
youtube.com.	300	IN	A	74.125.230.100

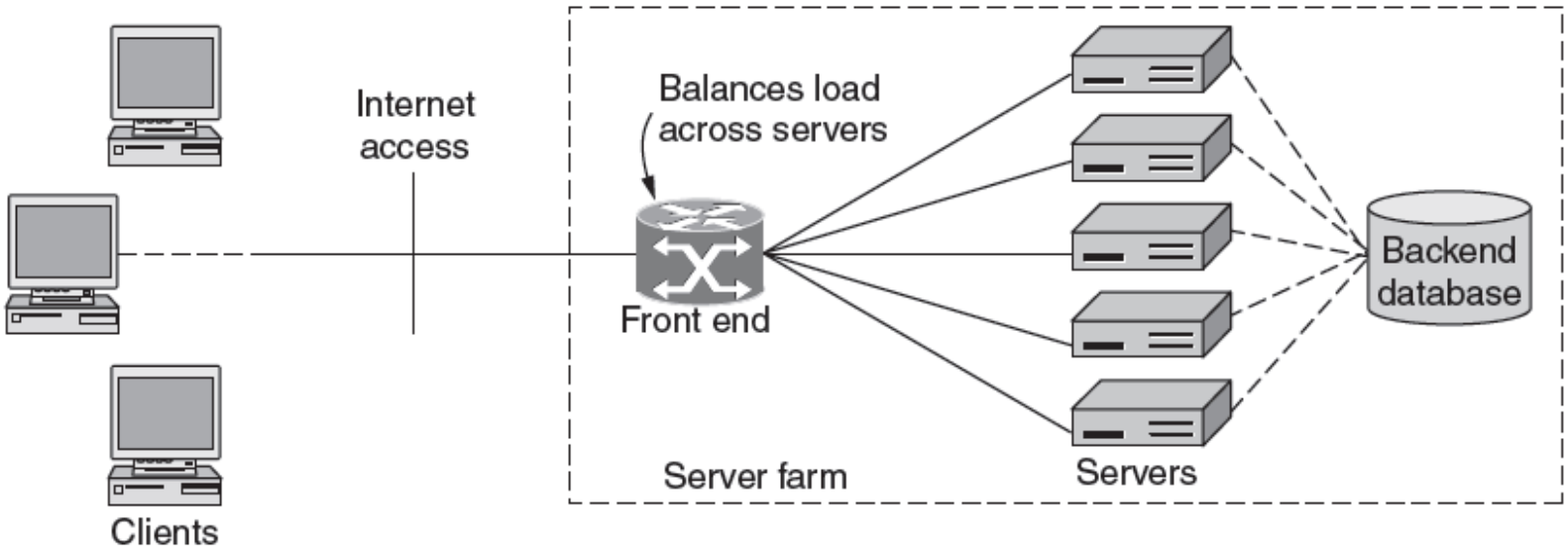
```
vertanen@katie:~$ ping 173.194.33.44
PING 173.194.33.44 (173.194.33.44) 56(84) bytes of data.
64 bytes from 173.194.33.44: icmp_req=1 ttl=54 time=16.7 ms
64 bytes from 173.194.33.44: icmp_req=2 ttl=54 time=16.2 ms
64 bytes from 173.194.33.44: icmp_req=3 ttl=54 time=16.6 ms
64 bytes from 173.194.33.44: icmp_req=4 ttl=54 time=15.8 ms
```

katie

```
kvertanen@li264-110:~$ ping 74.125.230.108
PING 74.125.230.108 (74.125.230.108) 56(84) bytes of data.
64 bytes from 74.125.230.108: icmp_req=1 ttl=57 time=1.13 ms
64 bytes from 74.125.230.108: icmp_req=2 ttl=57 time=1.14 ms
64 bytes from 74.125.230.108: icmp_req=3 ttl=57 time=1.11 ms
64 bytes from 74.125.230.108: icmp_req=4 ttl=57 time=1.08 ms
```

london

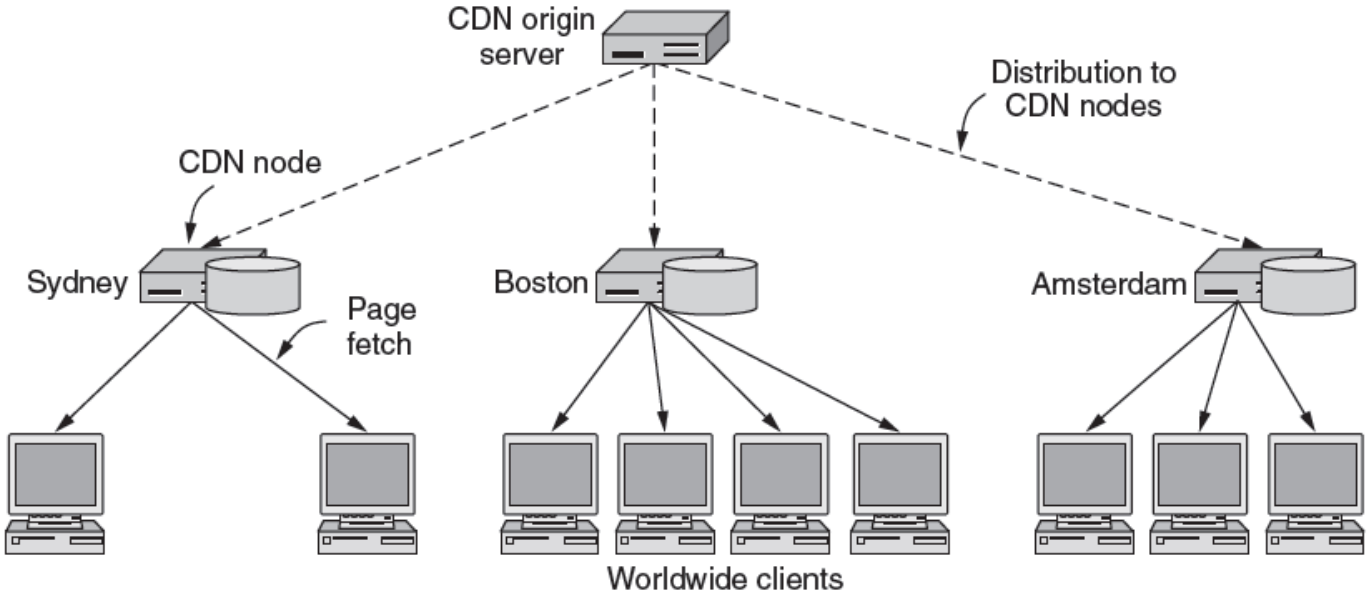
Front-end load balancing



- Single IP address for front-end
  - Front-end could broadcast all requests
    - Server responds by prior agreement (e.g. last 4-bits of source IP)
  - Front-end maps client to single server in a set
    - Load balancer inspects IP, TCP, and HTTP headers

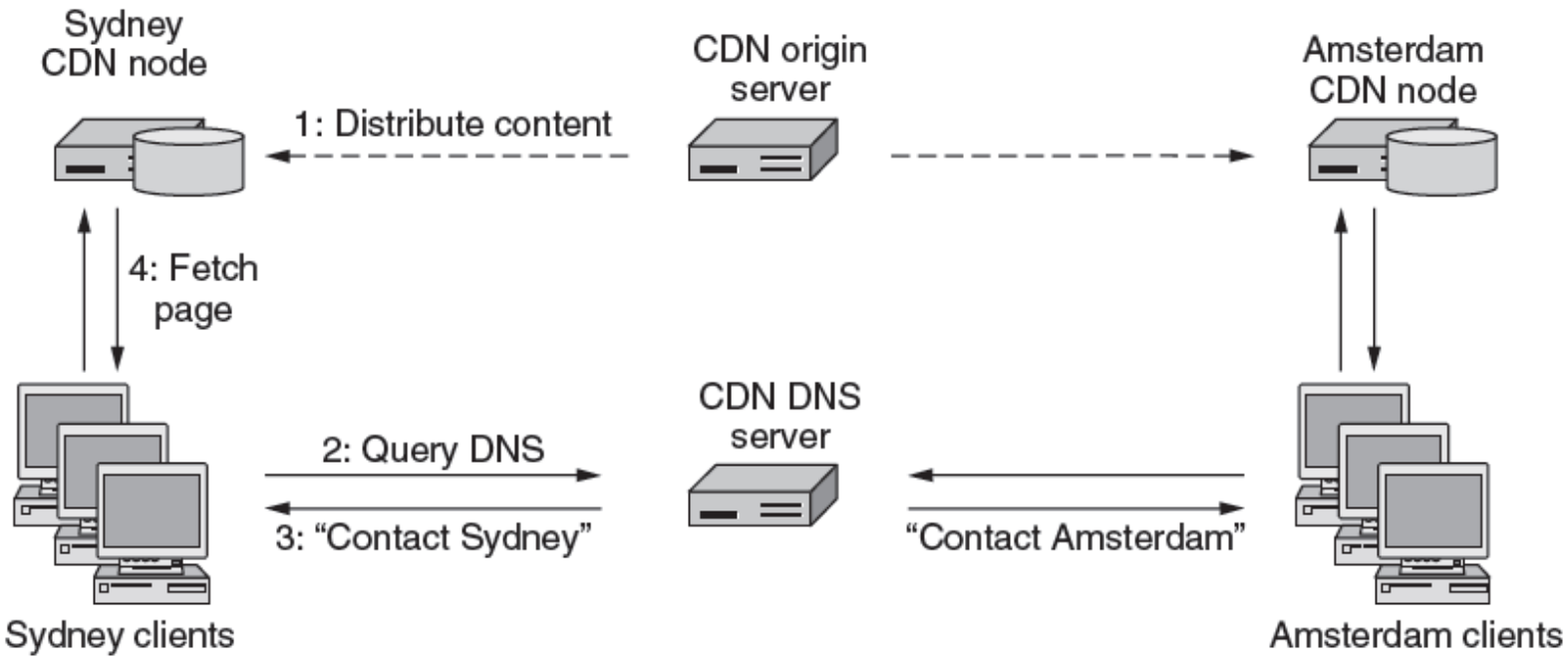
Caching for hire

- Content Distribution Networks (CDNs)
  - Pay someone to replicate your web site on many servers
  - CDN installs thousands of servers
    - In large datacenters with good connectivity
    - Close to users, lower latency → speeds up TCP slow-start
    - Amortize expense, customers sites can now handle "flash crowds"
  - Whenever you change content, update the CDN servers



DNS redirection

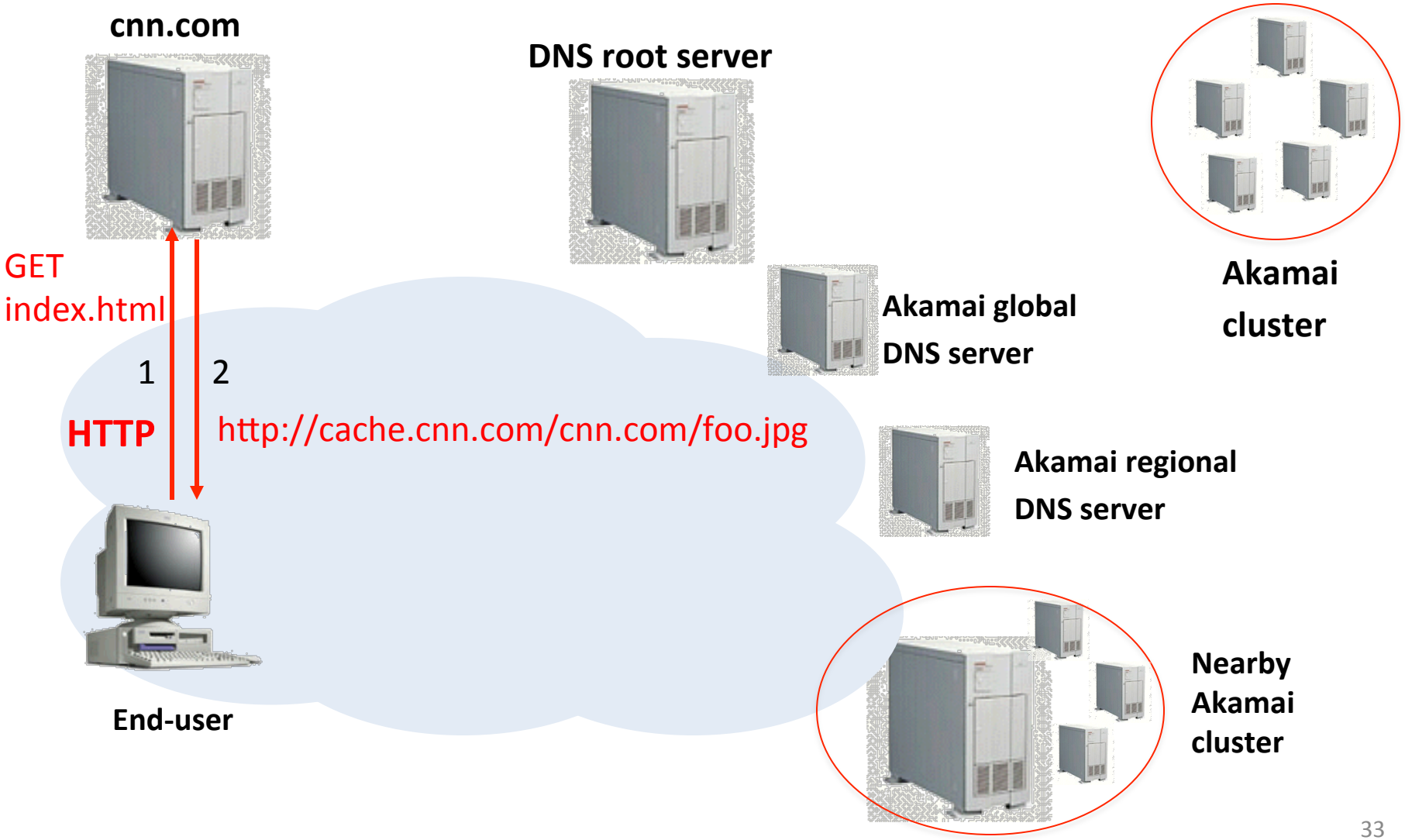
- How do clients use the CDN tree?
- DNS redirection
  - CDN runs the name server
  - Hands client the IP of the "best" CDN node



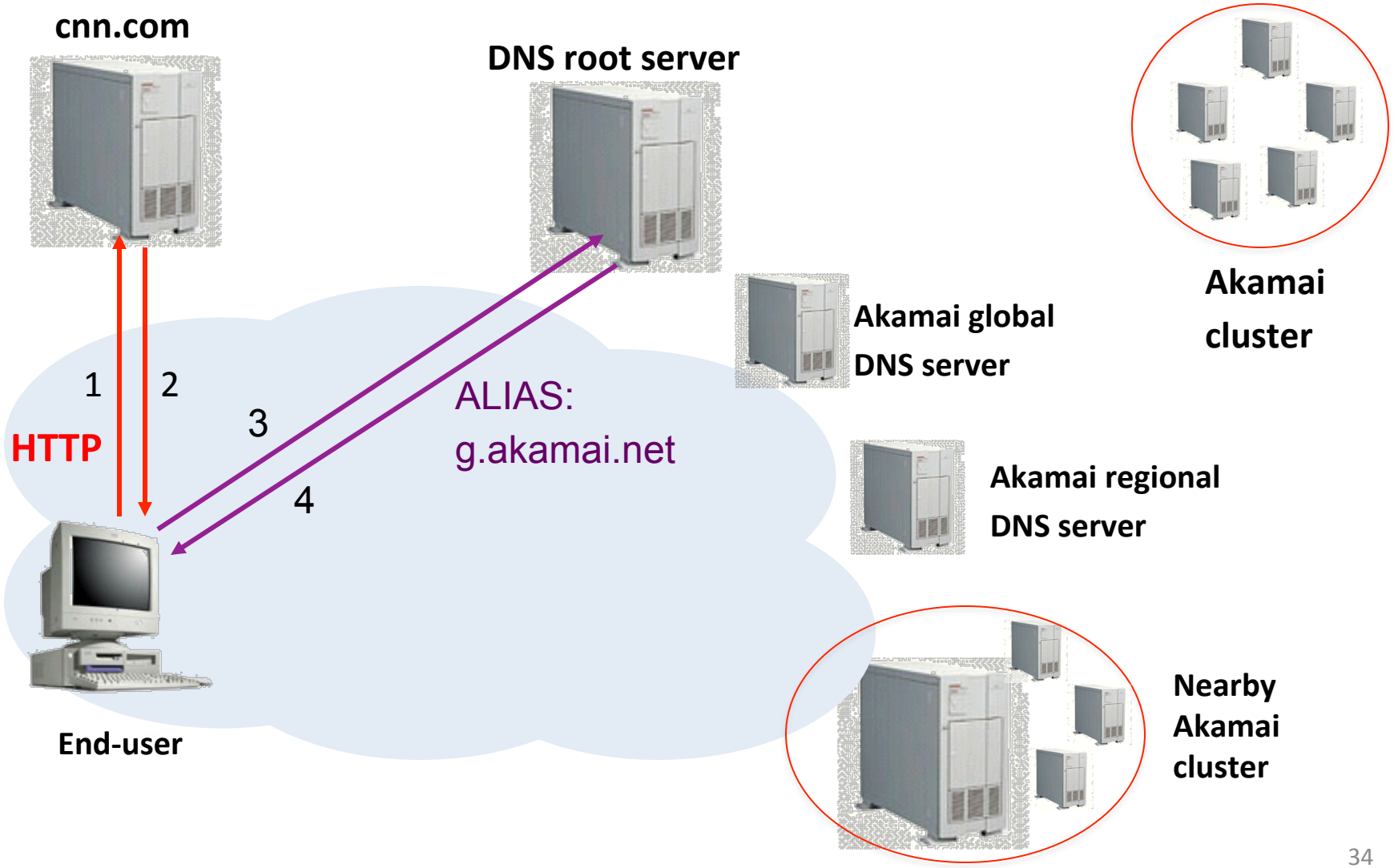
Akamai

- Akamai Technologies
  - First major CDN
  - Companies pay Akamai to deliver their content
  - Akamai wants nodes inside ISP networks
  - ISP networks want the nodes
    - Reduces their transit traffic
    - Make things faster for their customers

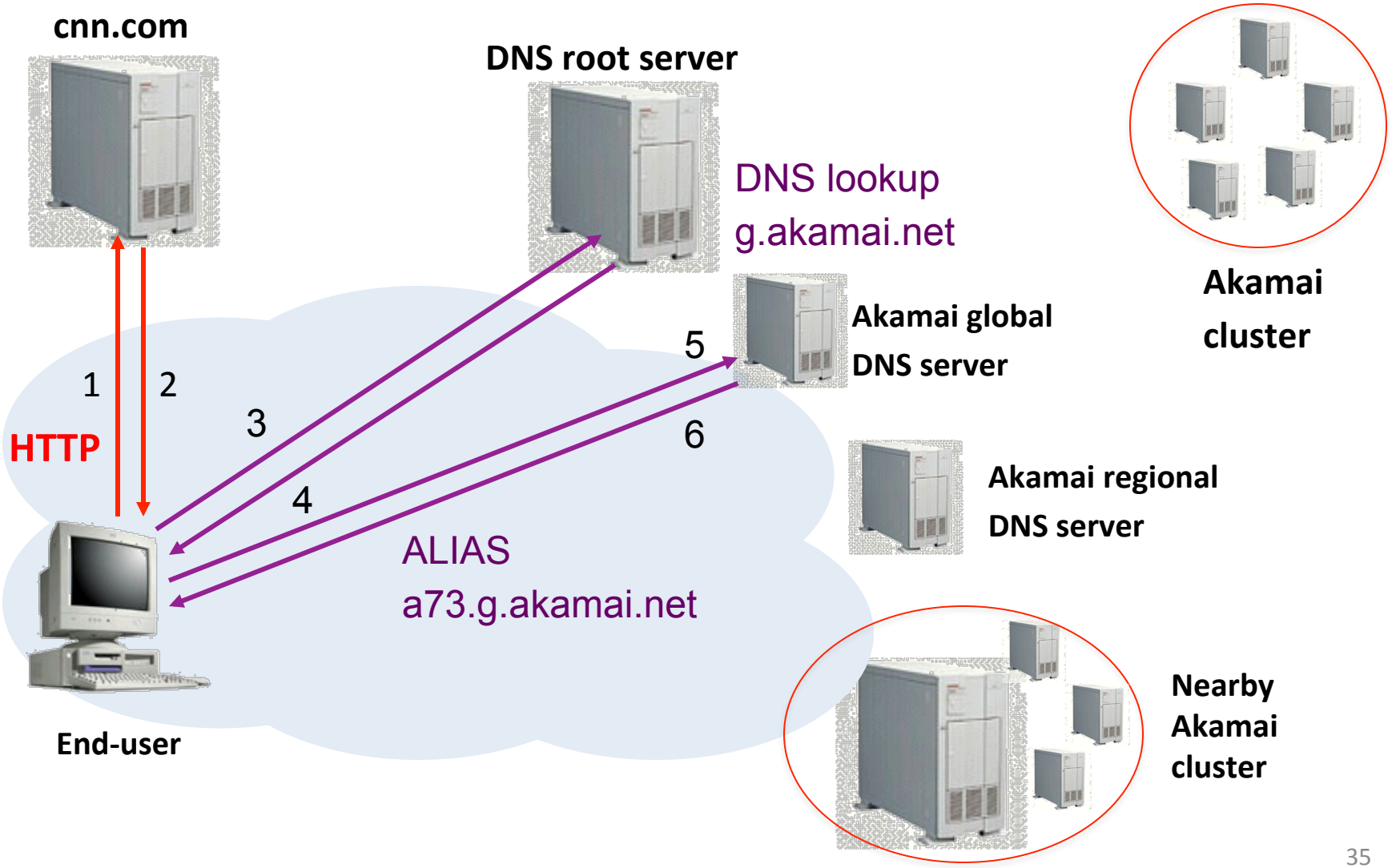
How Akamai Works



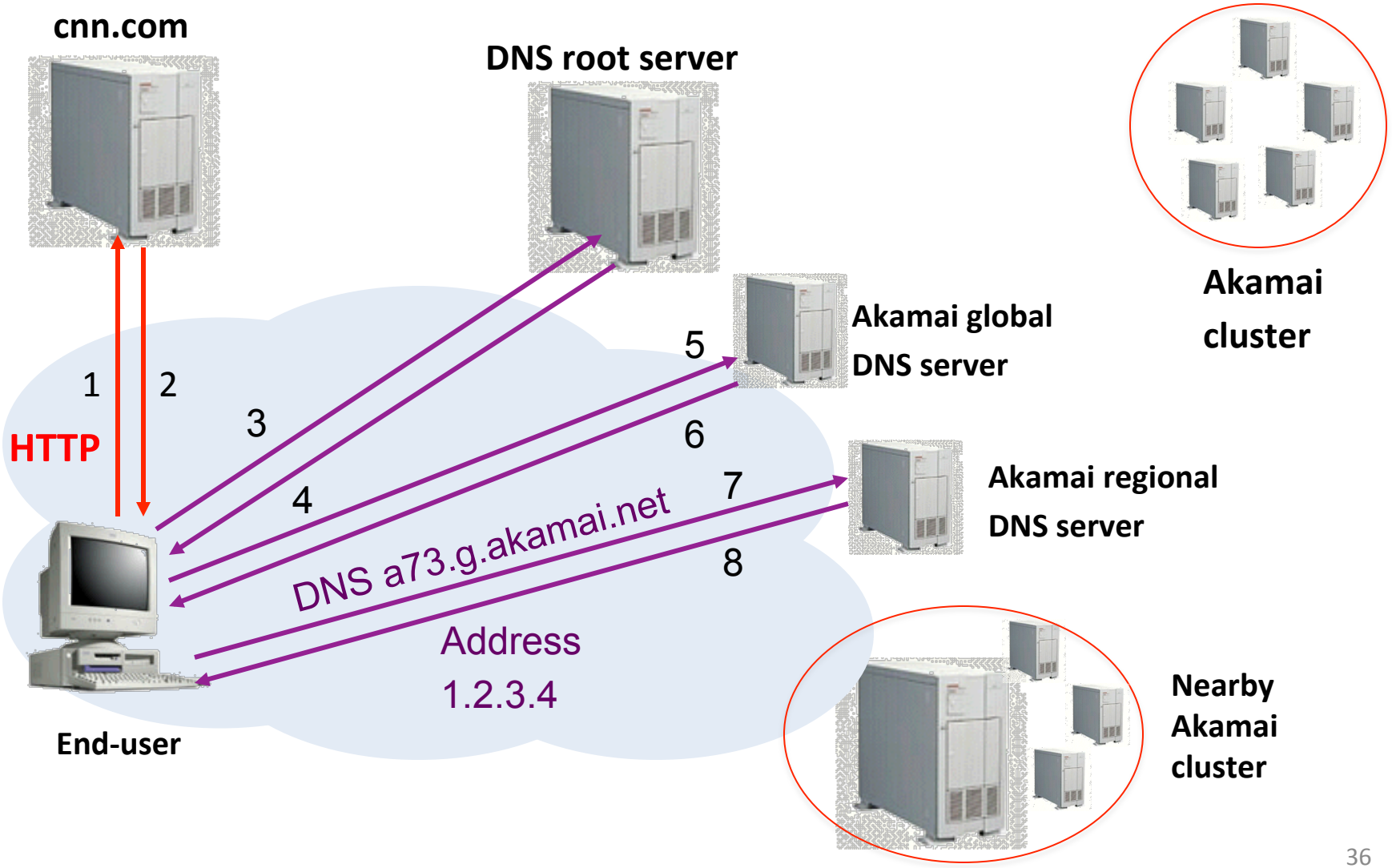
How Akamai Works



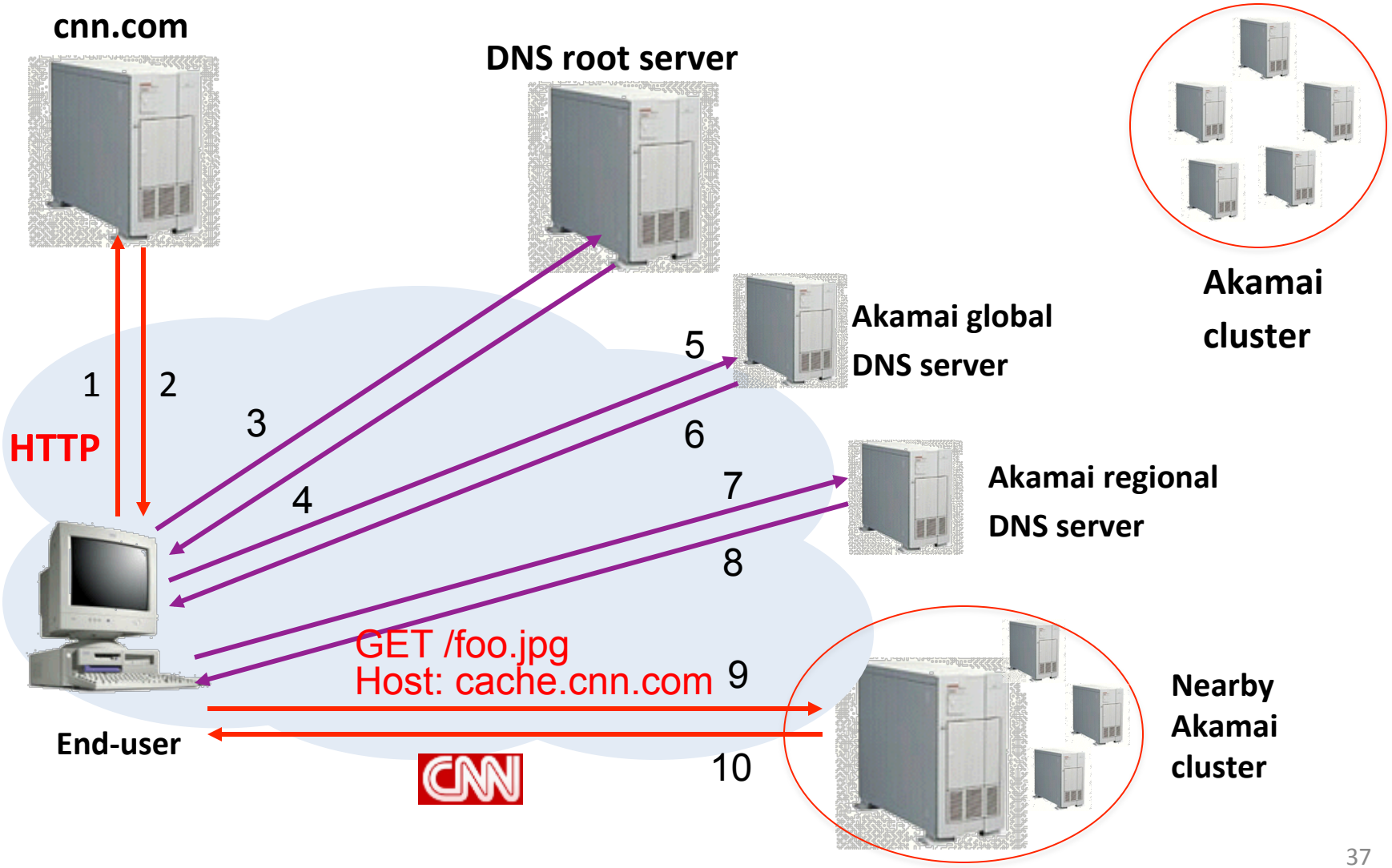
How Akamai Works



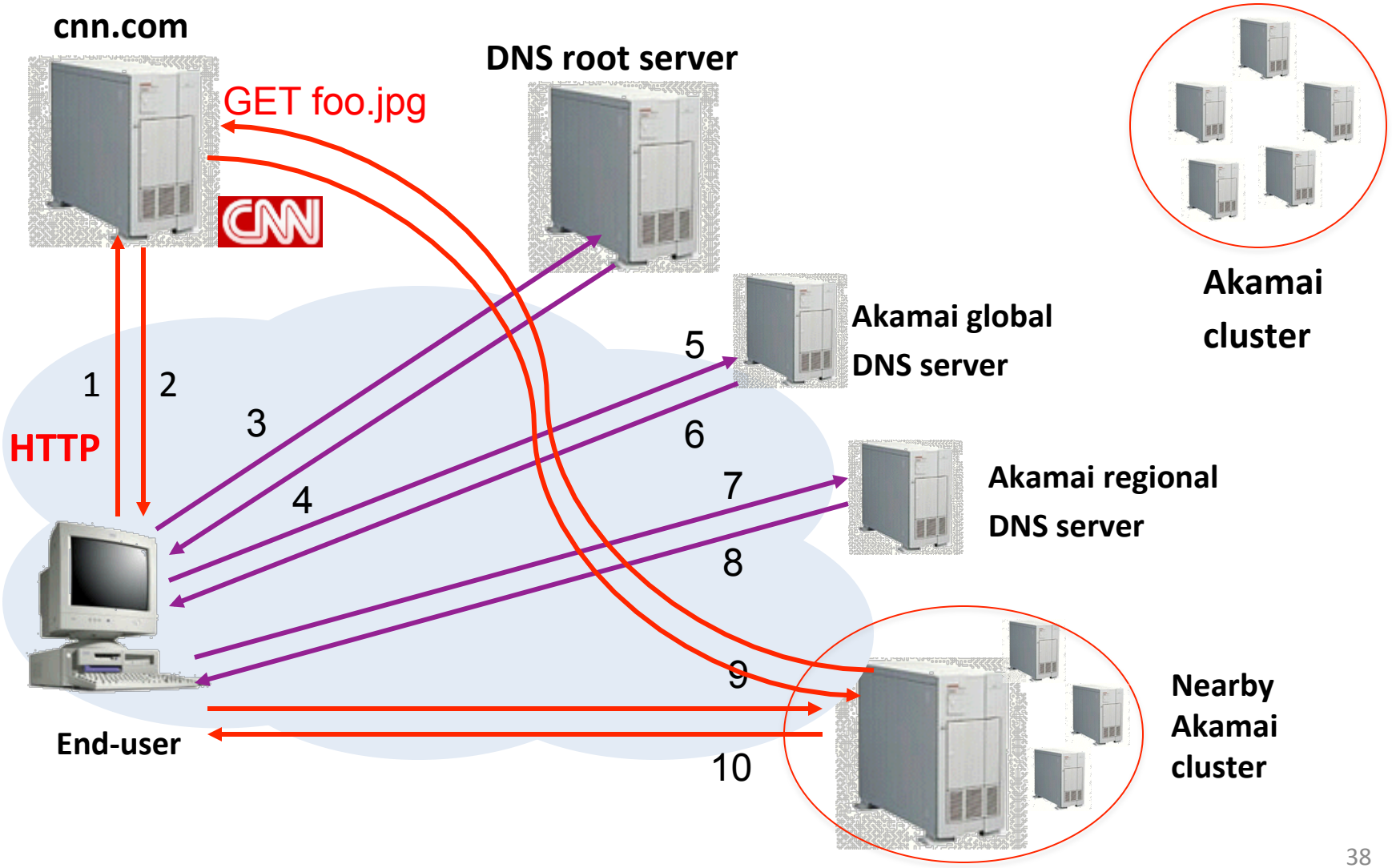
How Akamai Works



How Akamai Works



How Akamai Works



Summary

- How the web works
  - HTTP protocol over TCP
  - HTML standard for pages
  - URLs for locating things
- Making the web faster
  - Conditional requests
  - Caching on the host
  - Caching in the network
  - Content distribution networks