CSCI-1680
HTTP II

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti

Administrivia

- Should have done your TCP Milestone II at this point

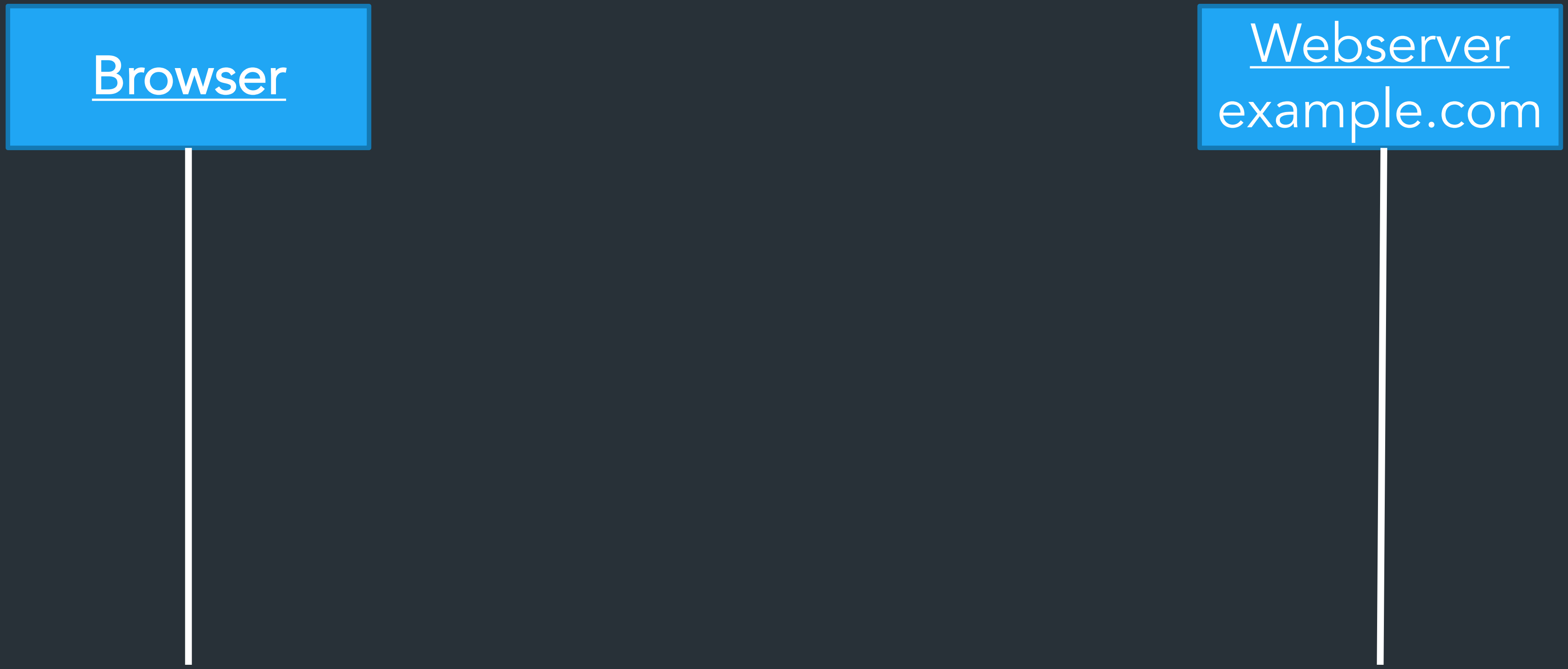
- TCP is due next Friday (11/22)
  => Look for some more testing/grading/SRC info soon

- Will announce some preliminary final project info next week

Warmup

Browser wants to fetch: http://example.com/page.html

Assuming no caching, what is the minimum number of packets the browser needs to wait for?



Browser

Webserver
example.com

```
> nc cs.brown.edu 80
GET / HTTP/1.0
Host: cs.brown.edu
Content-Type: text/html
Accept-Language: en


HTTP/1.1 200 OK
Date: Thu, 24 Mar 2011 12:58:46 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Thu, 24 Mar 2011 12:25:27 GMT
ETag: "840a88b-236c-49f3992853bc0"
Accept-Ranges: bytes
Content-Length: 9068
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en”>
...
```

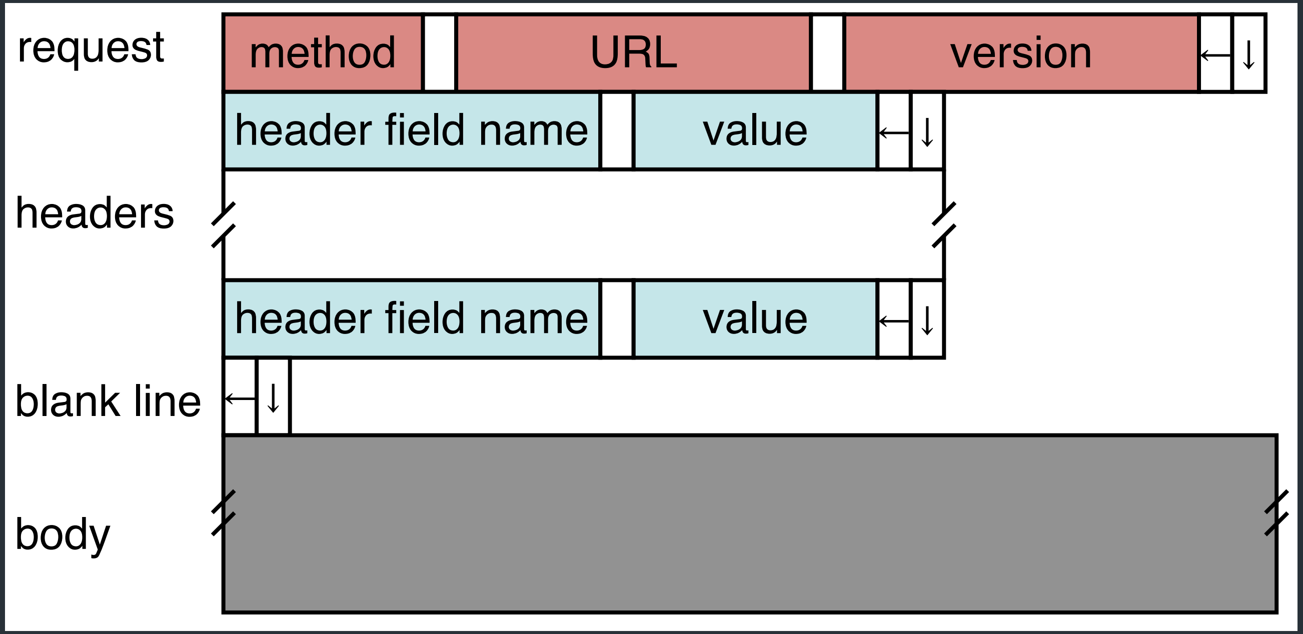
HTTP Request

Method:

- GET: current value of resource, run program
- POST: update a resource, provide input for a program. . .

Headers: useful info about request

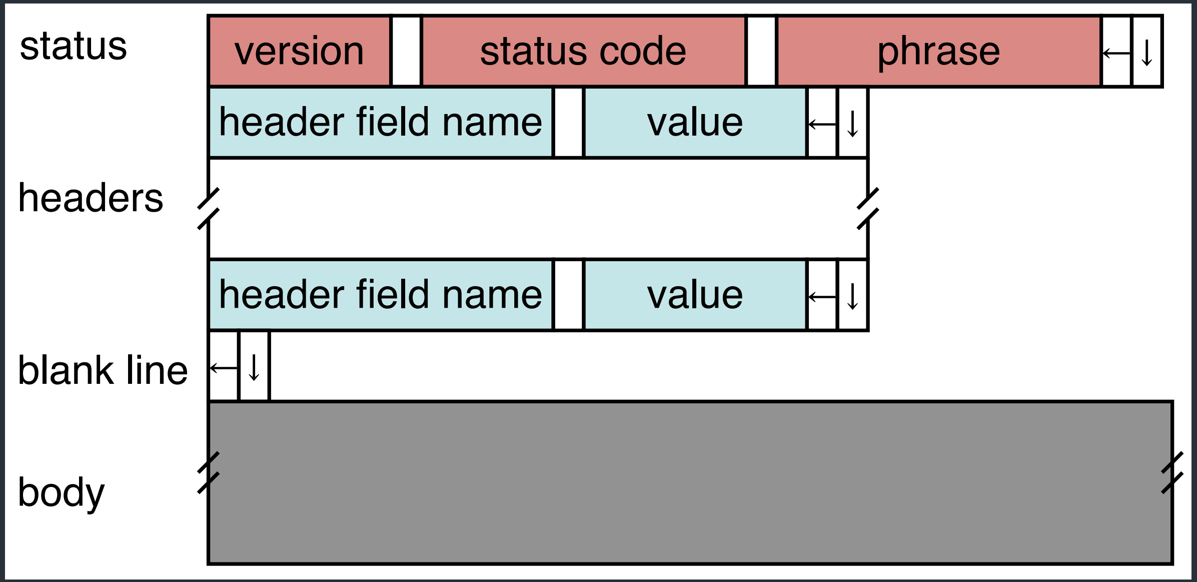
- E.g., desired language, text encoding



HTTP Responses

Status codes to indicate something about the result

- 1xx: Information e.g, 100 Continue
- 2xx: Success e.g., 200 OK
- 3xx: Redirection e.g., 302 Found (elsewhere),
- 4xx: Client Error e.g., 403 Forbidden, 404 Not Found
- 5xx: Server Error e.g, 503 Service Unavailable



status | version | status code | phrase ←↓

headers | header field name | value ←↓

header field name | value ←↓

blank line ←↓

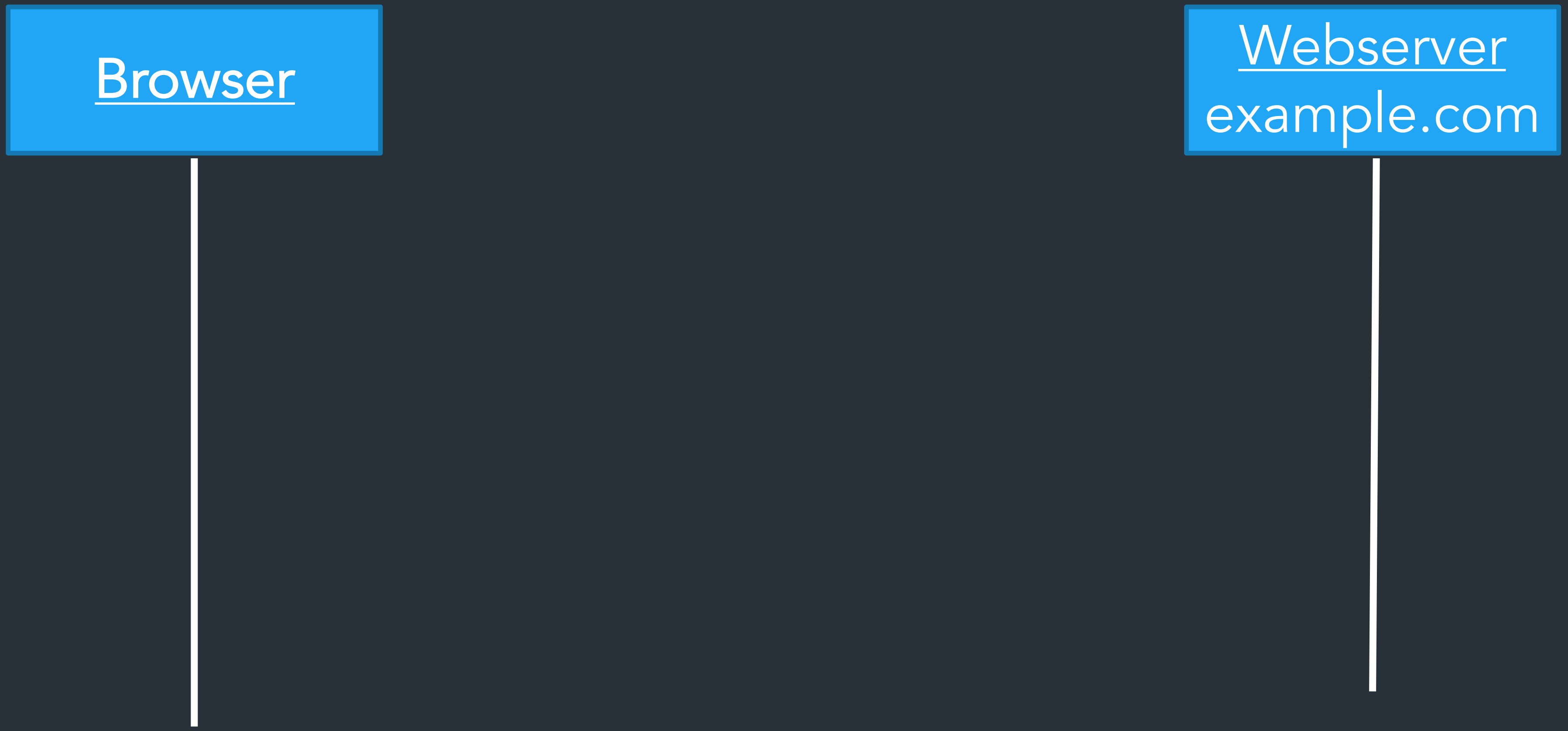
body

It gets worse

Modern web traffic almost always uses HTTPS: https://example.com/page.html

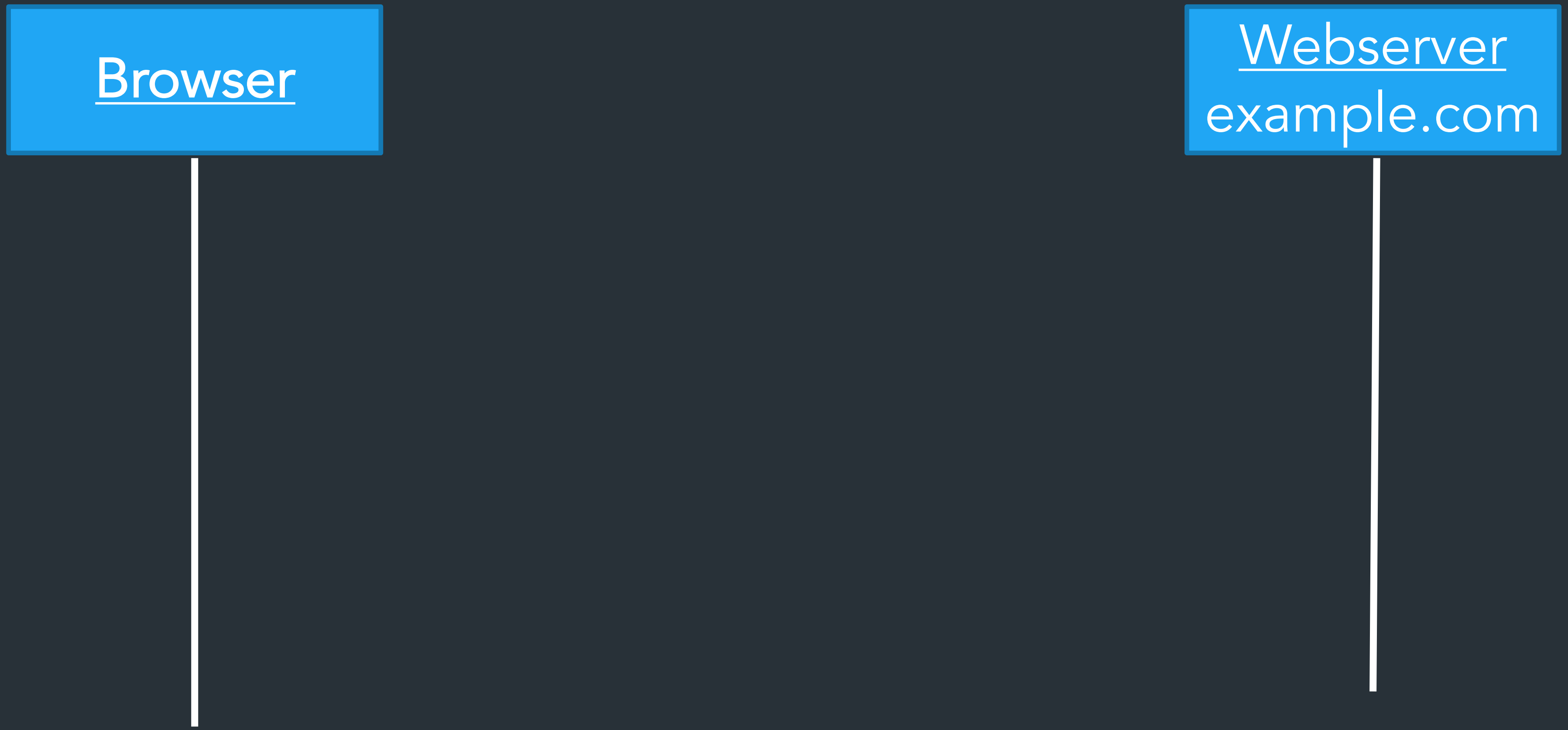
=> Creates a secure transport layer to prevent eavesdropping, etc

(more on this later)



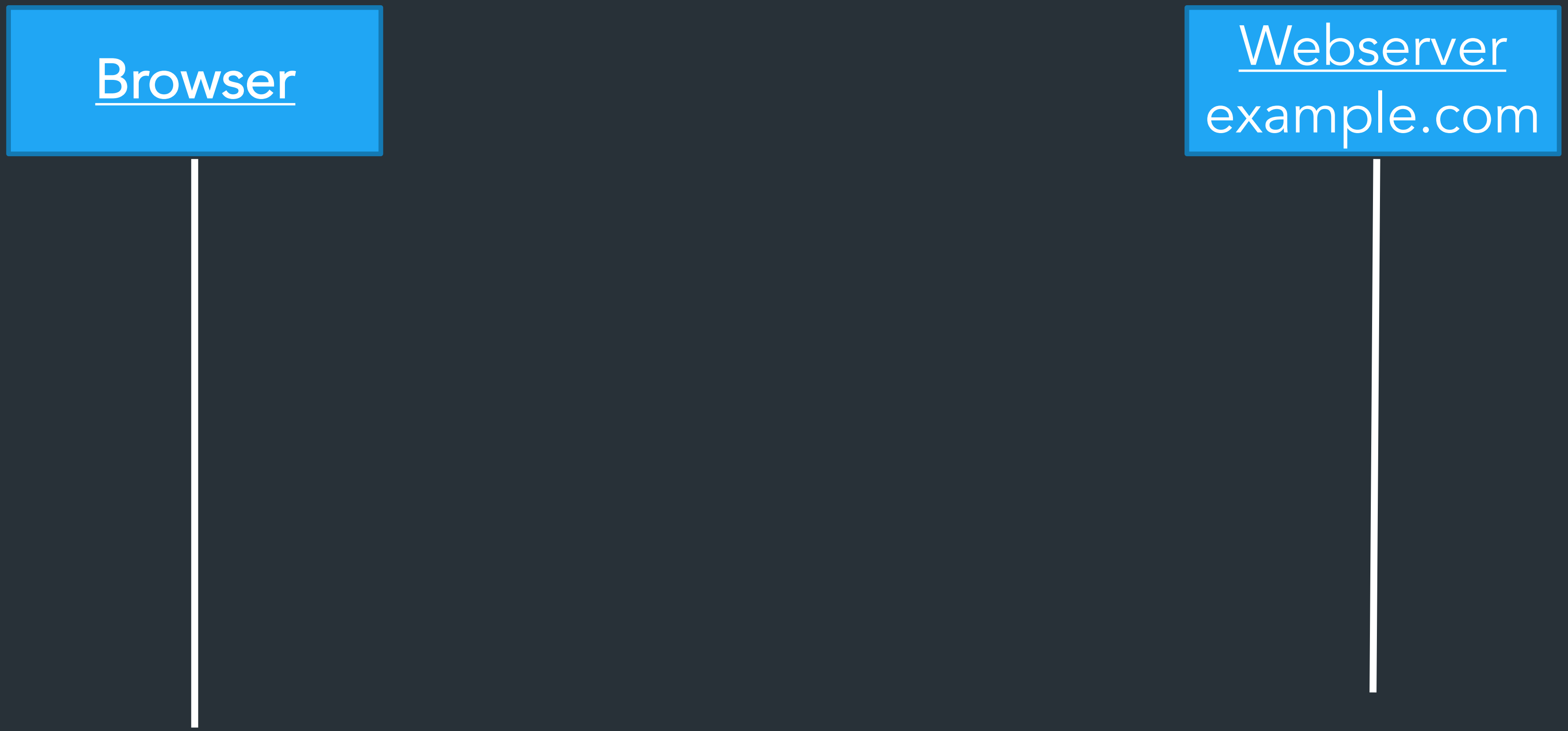
How does a browser load a page?

- Click a link, type in URL => browser fetches main page



How does a browser load a page?

- Click a link, type in URL => browser fetches main page
- Main page has links to more resources => need to fetch these too!
  - Images, CSS, Javascript, etc.



Browser

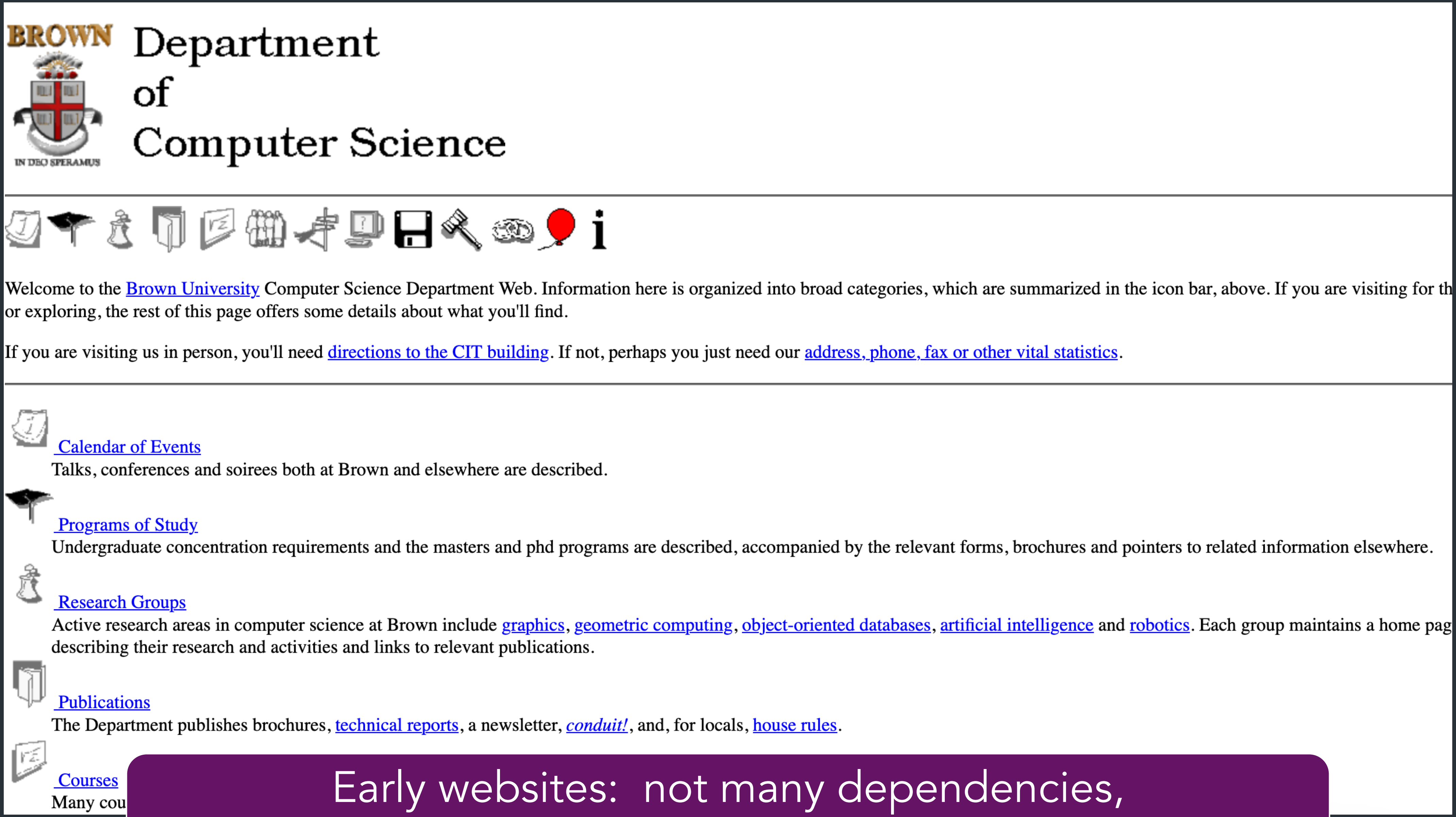
Webserver
example.com

How does a browser load a page?

- Click a link, type in URL => browser fetches main page
- Main page has links to more resources => need to fetch these too!
  - Images, CSS, Javascript, etc.

- New resources might load yet more resources…

Recursive process with many dependencies!



Department of Computer Science

Welcome to the Brown University Computer Science Department Web. Information here is organized into broad categories, which are summarized in the icon bar, above. If you are visiting for th or exploring, the rest of this page offers some details about what you'll find.

If you are visiting us in person, you'll need directions to the CIT building. If not, perhaps you just need our address, phone, fax or other vital statistics.

Calendar of Events
Talks, conferences and soirees both at Brown and elsewhere are described.

Programs of Study
Undergraduate concentration requirements and the masters and phd programs are described, accompanied by the relevant forms, brochures and pointers to related information elsewhere.

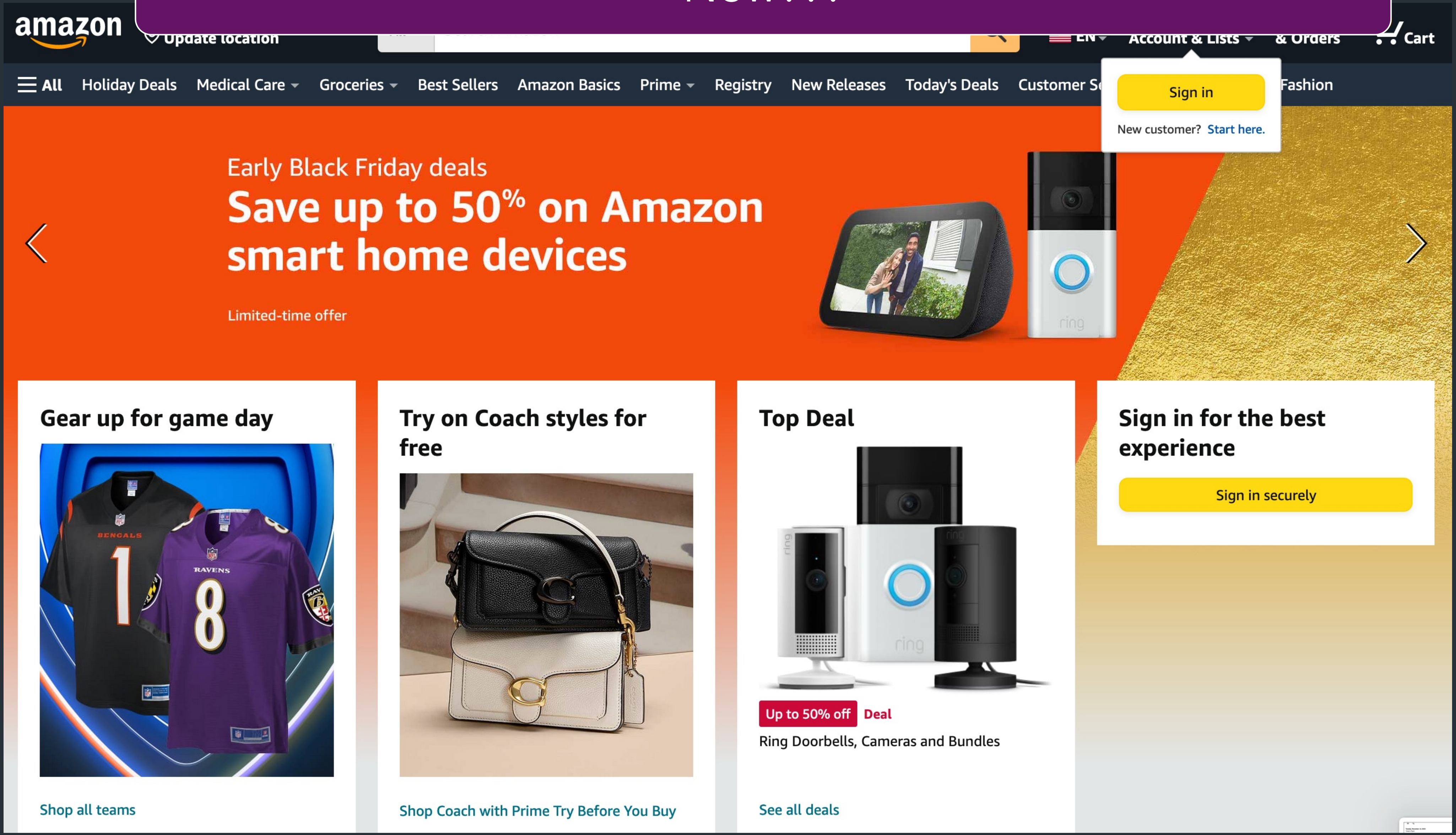
Research Groups
Active research areas in computer science at Brown include graphics, geometric computing, object-oriented databases, artificial intelligence and robotics. Each group maintains a home pag describing their research and activities and links to relevant publications.

Publications
The Department publishes brochures, technical reports, a newsletter, *conduit!*, and, for locals, house rules.

Courses
Many cou

Early websites: not many dependencies, usually served by one server

Now???


amazon
Update location
EN
Account & Lists
& Orders
Cart
All
Holiday Deals
Medical Care
Groceries
Best Sellers
Amazon Basics
Prime
Registry
New Releases
Today's Deals
Customer Se
Fashion
Sign in
New customer? Start here.
Early Black Friday deals
Save up to 50% on Amazon smart home devices
Limited-time offer
ring
Gear up for game day
Shop all teams
Try on Coach styles for free
Shop Coach with Prime Try Before You Buy
Top Deal
Up to 50% off
Deal
Ring Doorbells, Cameras and Bundles
See all deals
Sign in for the best experience
Sign in securely

On a modern webpage…

On a modern webpage…

- Huge number of dependencies, external resources
  - … from many different locations, not just one server!

- Lots of asynchronous operations => loading new resources as you are using the page

- Lots of dynamic content => generated by the server specifically for you (your feed, ad data, …)

How to make this fast?

How to make this fast?

What’s important for performance?

Observation: lots of small requests

Latency is a problem!   Need many RTTs just to fetch one resource!

Observation: lots of small requests

Latency is a problem! Need many RTTs just to fetch one resource!

HTTP/1.0: One TCP connection per request!

Can we do better?

HTTP/1.1 (1996): Persistent connections
=> Reuse TCP connection to for multiple requests

Problems?

Can we do better?

HTTP/1.1 (1996): Persistent connections

=> Reuse TCP connection to for multiple requests

Problems?
⇒One big request blocks others => head of line blocking
=> Same if connection has packet loss
=> Doesn't help when fetching from multiple locations

What can be done?

HTTP/1.1 Request

```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
```

What can be done?

Pipelining: have multiple “in-flight” requests at once

Two methods

- Multiple TCP connections in parallel

- Change the HTTP protocol: multiple requests per connection

What can be done?

Pipelining: have multiple “in-flight” requests at once

Two methods

- Multiple TCP connections in parallel

  => Browsers often do this (up to a limit)

- Change the HTTP protocol: multiple requests per connection

  => Newer HTTP versions: HTTP/2, HTTP/3

HTTP/2 (2015)

Adds support for multiplexed streams on one connection

What happens if a packet gets dropped?

https://www.twilio.com/blog/2017/10/http2-issues.html

HTTP/2 (2015)

Adds support for multiplexed streams on one connection

TCP provides a single, ordered byte stream
=> doesn’t know about multiple connections!

https://www.twilio.com/blog/2017/10/http2-issues.html

HTTP/2 (2015)

Adds support for multiplexed streams on one connection

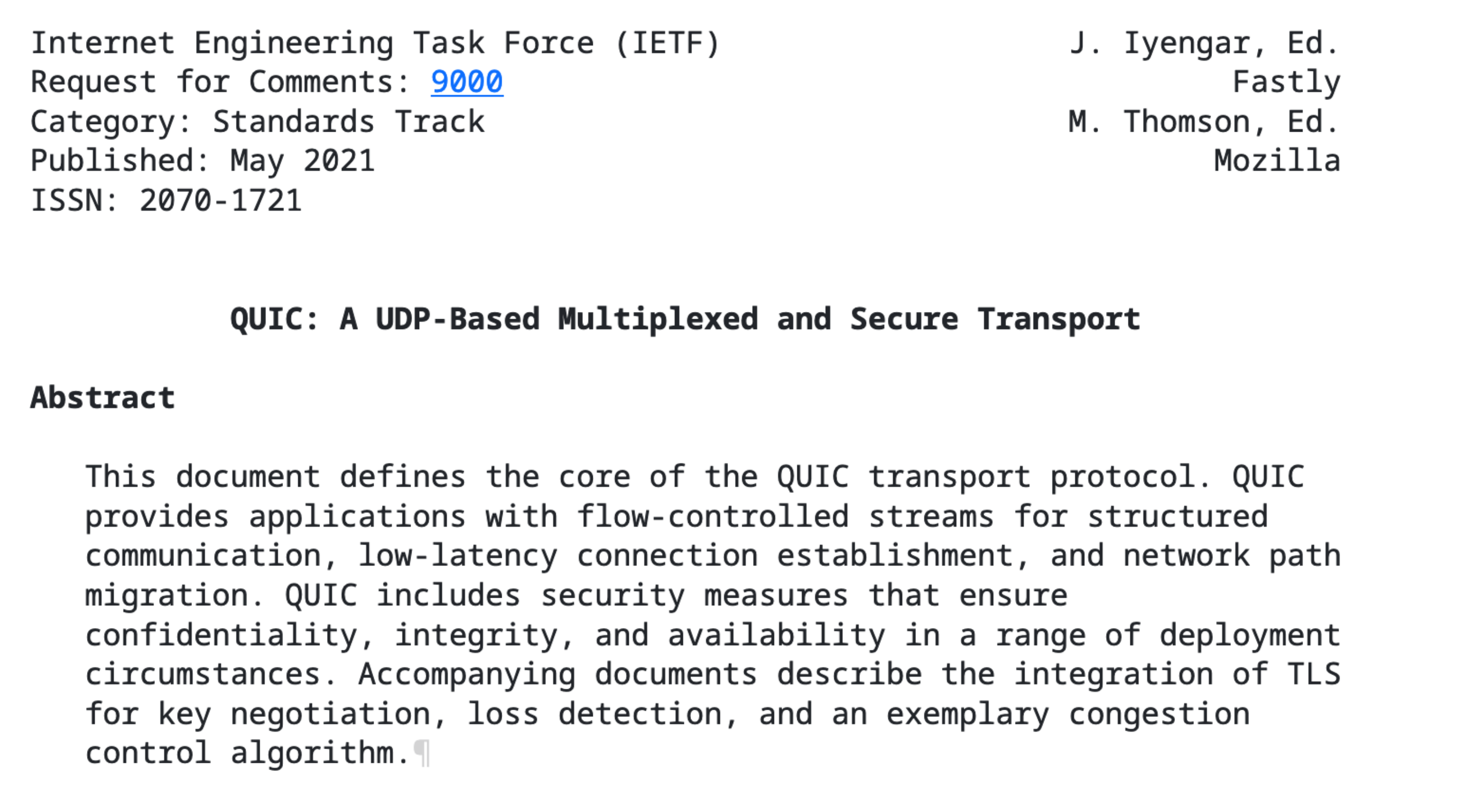
TCP provides a single, ordered byte stream
=> doesn't know about multiple connections!

Encumbered by TCP's semantics:
If a packet is lost, all streams suffer! 😭 😭 😭

=> Head of line blocking

https://www.twilio.com/blog/2017/10/http2-issues.html

HTTP/3 (2022): HTTP + QUIC



Internet Engineering Task Force (IETF) J. Iyengar, Ed.
Request for Comments: 9000 Fastly
Category: Standards Track M. Thomson, Ed.
Published: May 2021 Mozilla
ISSN: 2070-1721

QUIC: A UDP-Based Multiplexed and Secure Transport

Abstract

This document defines the core of the QUIC transport protocol. QUIC provides applications with flow-controlled streams for structured communication, low-latency connection establishment, and network path migration. QUIC includes security measures that ensure confidentiality, integrity, and availability in a range of deployment circumstances. Accompanying documents describe the integration of TLS for key negotiation, loss detection, and an exemplary congestion control algorithm.¶

HTTP/3 (2022): HTTP + QUIC

QUIC (RFC9000): Newer transport-layer protocol, same goals as TCP

HTTP/3 (2022): HTTP + QUIC

QUIC (RFC9000): Newer transport-layer protocol, same goals as TCP

- Supports multiple streams at once
- Various tricks to reduce message size and latency
- Integrates security by default (TLS)

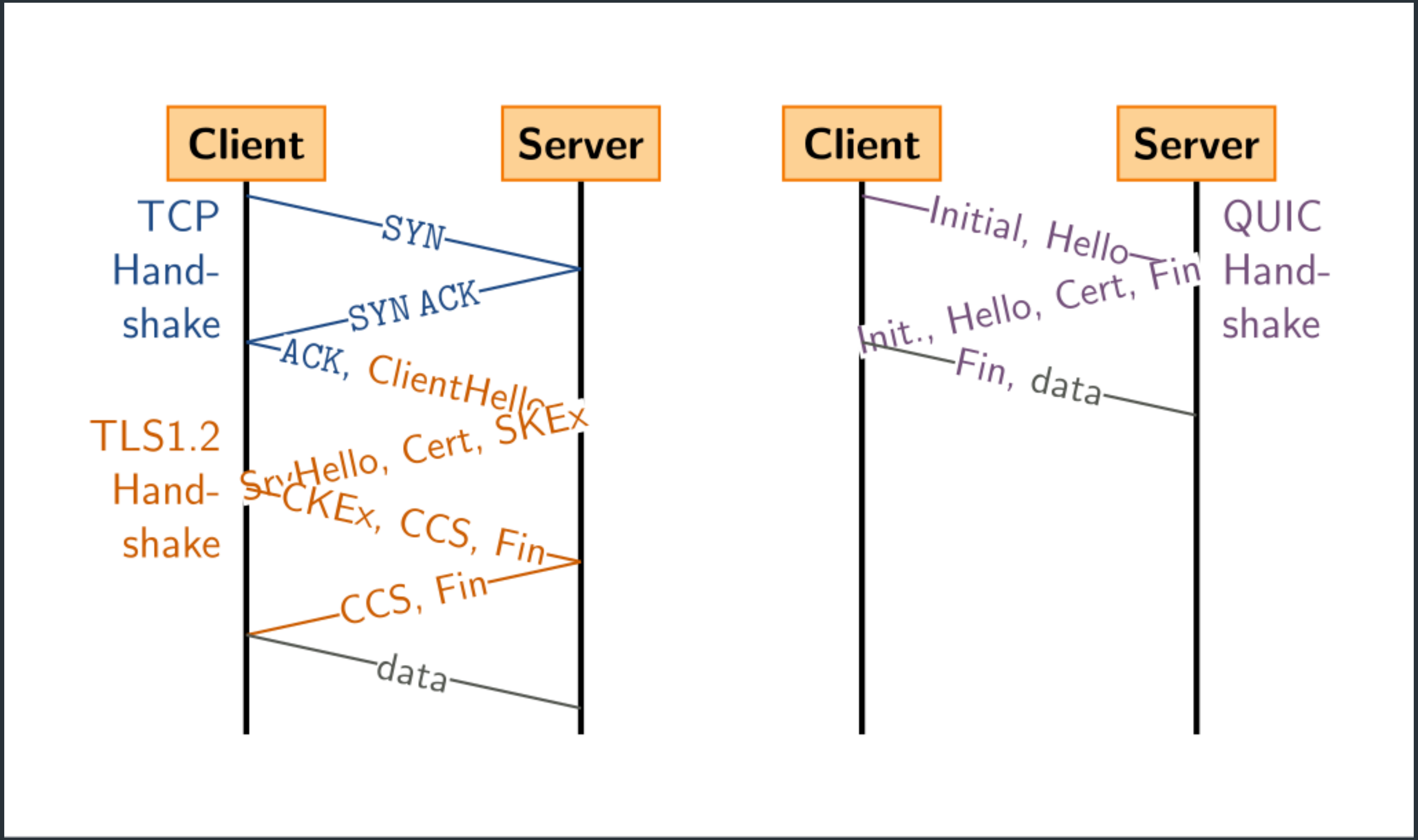
HTTP/3 (2022): HTTP + QUIC

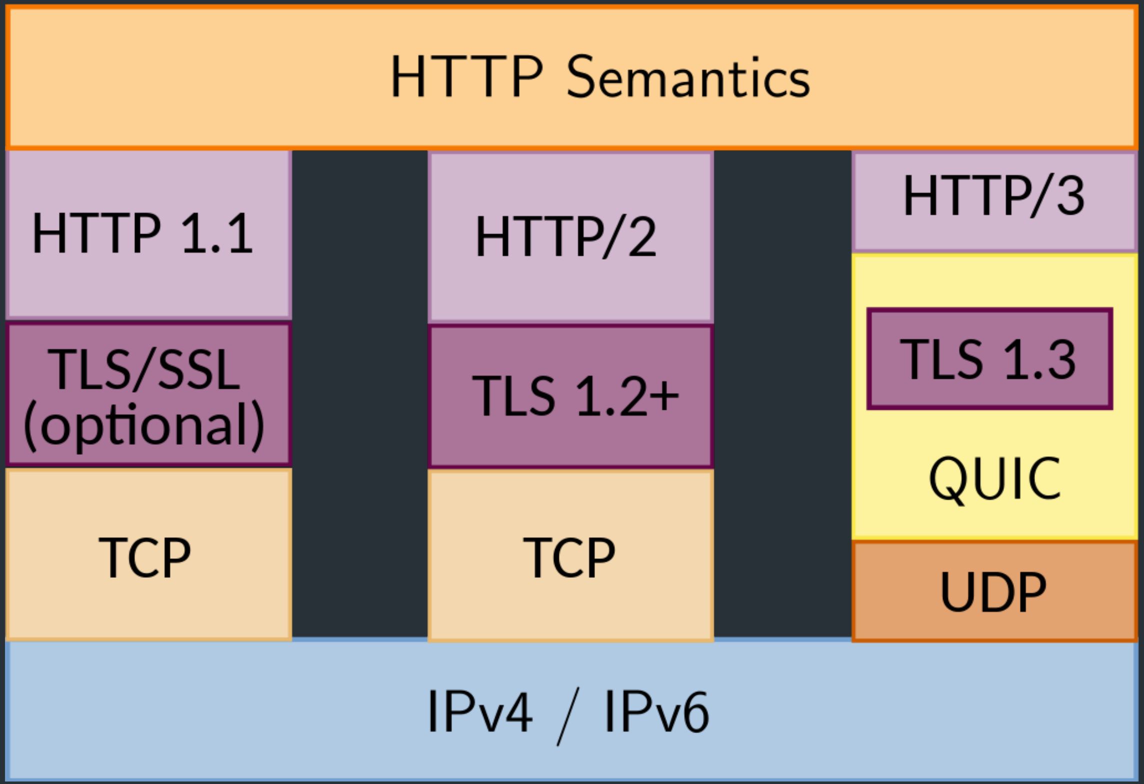
QUIC (RFC9000): Newer transport-layer protocol, same goals as TCP

- Supports multiple streams at once
- Various tricks to reduce message size and latency
- Integrates security by default (TLS)

- By moving multiplexing into the transport layer, can do so in a way that benefits HTTP (no head of line blocking!)

Comparison: QUIC's handshake





HTTP Semantics

HTTP 1.1

TLS/SSL (optional)

TCP

HTTP/2

TLS 1.2+

TCP

HTTP/3

TLS 1.3

QUIC

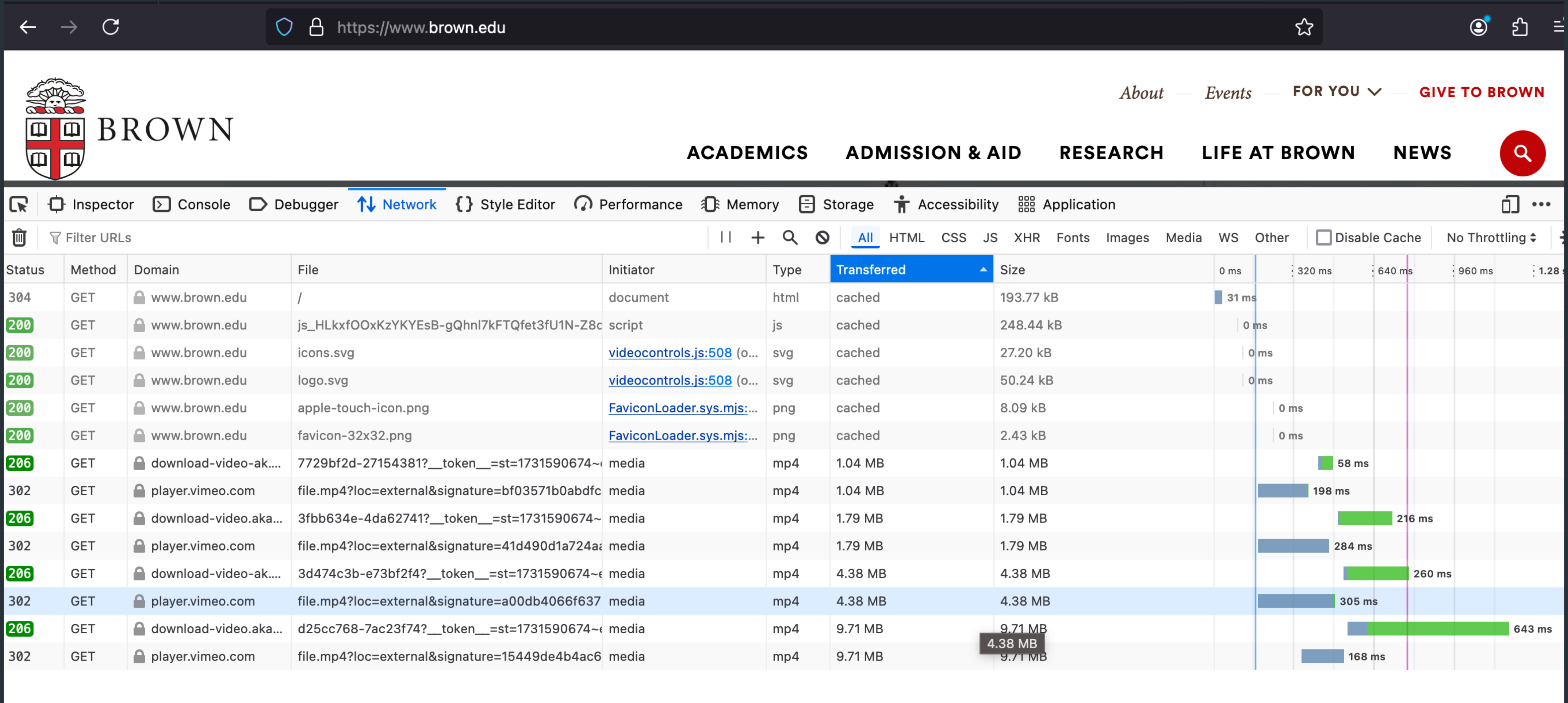
UDP

IPv4 / IPv6

http://httpwg.org/specs/rfc7540.html

What else can we improve performance?

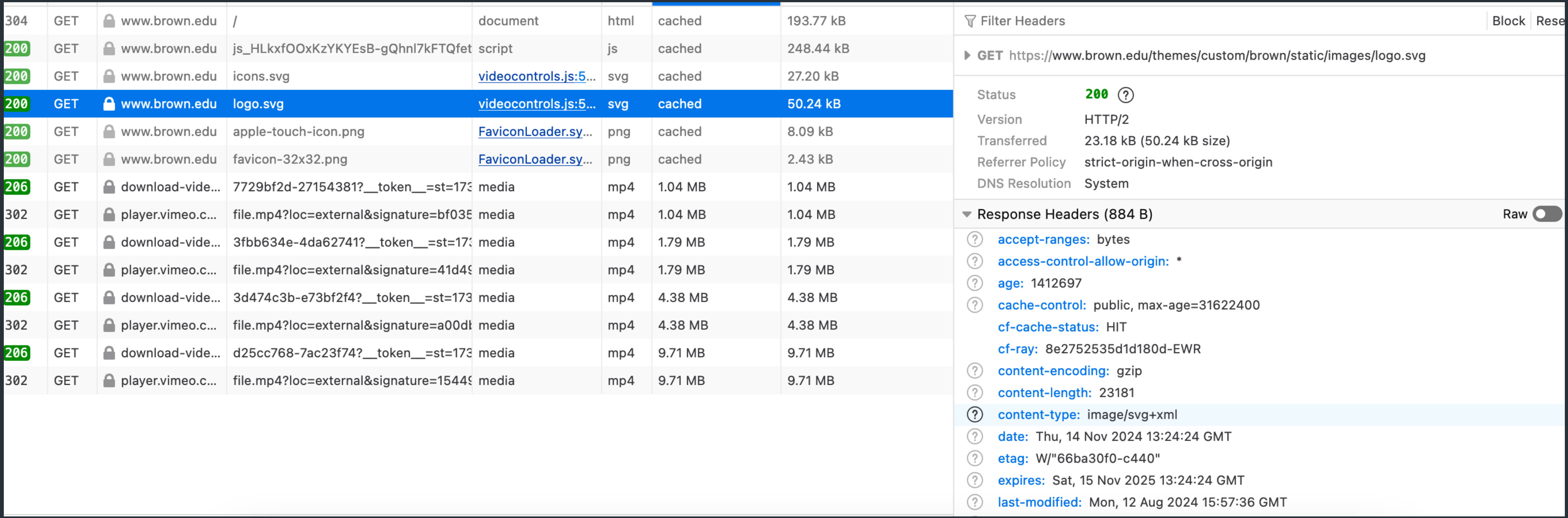
Caching: in the browser



Status	Method	Domain	File	Initiator	Type	Transferred	Size	0 ms 320 ms 640 ms 960 ms 1.28
304	GET	www.brown.edu	/	document	html	cached	193.77 kB	31 ms
200	GET	www.brown.edu	js_HLkxfOOxKzYKYEsB-gQhnl7kFTQfet3fU1N-Z8c	script	js	cached	248.44 kB	0 ms
200	GET	www.brown.edu	icons.svg	videocontrols.js:508 (o...	svg	cached	27.20 kB	0 ms
200	GET	www.brown.edu	logo.svg	videocontrols.js:508 (o...	svg	cached	50.24 kB	0 ms
200	GET	www.brown.edu	apple-touch-icon.png	FaviconLoader.sys.mjs:...	png	cached	8.09 kB	0 ms
200	GET	www.brown.edu	favicon-32x32.png	FaviconLoader.sys.mjs:...	png	cached	2.43 kB	0 ms
206	GET	download-video-ak....	7729bf2d-27154381?__token__=st=1731590674~	media	mp4	1.04 MB	1.04 MB	58 ms
302	GET	player.vimeo.com	file.mp4?loc=external&signature=bf03571b0abdfc	media	mp4	1.04 MB	1.04 MB	198 ms
206	GET	download-video.aka...	3fbb634e-4da62741?__token__=st=1731590674~	media	mp4	1.79 MB	1.79 MB	216 ms
302	GET	player.vimeo.com	file.mp4?loc=external&signature=41d490d1a724a	media	mp4	1.79 MB	1.79 MB	284 ms
206	GET	download-video-ak....	3d474c3b-e73bf2f4?__token__=st=1731590674~	media	mp4	4.38 MB	4.38 MB	260 ms
302	GET	player.vimeo.com	file.mp4?loc=external&signature=a00db4066f637	media	mp4	4.38 MB	4.38 MB	305 ms
206	GET	download-video.aka...	d25cc768-7ac23f74?__token__=st=1731590674~	media	mp4	9.71 MB	9.71 MB	643 ms
302	GET	player.vimeo.com	file.mp4?loc=external&signature=15449de4b4ac6	media	mp4	9.71 MB	9.71 MB	168 ms

What parts of this can be cached?

How do we know what to cache?



304	GET	www.brown.edu	/	document	html	cached	193.77 kB
200	GET	www.brown.edu	js_HLkxfOOxKzYKYEsB-gQhnl7kFTQfet	script	js	cached	248.44 kB
200	GET	www.brown.edu	icons.svg	videocontrols.js:5...	svg	cached	27.20 kB
200	GET	www.brown.edu	logo.svg	videocontrols.js:5...	svg	cached	50.24 kB
200	GET	www.brown.edu	apple-touch-icon.png	FaviconLoader.sy...	png	cached	8.09 kB
200	GET	www.brown.edu	favicon-32x32.png	FaviconLoader.sy...	png	cached	2.43 kB
206	GET	download-vide...	7729bf2d-27154381?__token__=st=173	media	mp4	1.04 MB	1.04 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=bf035	media	mp4	1.04 MB	1.04 MB
206	GET	download-vide...	3fbb634e-4da62741?__token__=st=17	media	mp4	1.79 MB	1.79 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=41d4	media	mp4	1.79 MB	1.79 MB
206	GET	download-vide...	3d474c3b-e73bf2f4?__token__=st=173	media	mp4	4.38 MB	4.38 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=a00d	media	mp4	4.38 MB	4.38 MB
206	GET	download-vide...	d25cc768-7ac23f74?__token__=st=173	media	mp4	9.71 MB	9.71 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=1544	media	mp4	9.71 MB	9.71 MB

GET https://www.brown.edu/themes/custom/brown/static/images/logo.svg

Status	200
Version	HTTP/2
Transferred	23.18 kB (50.24 kB size)
Referrer Policy	strict-origin-when-cross-origin
DNS Resolution	System

Response Headers (884 B)

```
accept-ranges: bytes
access-control-allow-origin: *
age: 1412697
cache-control: public, max-age=31622400
cf-cache-status: HIT
cf-ray: 8e2752535d1d180d-EWR
content-encoding: gzip
content-length: 23181
content-type: image/svg+xml
date: Thu, 14 Nov 2024 13:24:24 GMT
etag: W/"66ba30f0-c440"
expires: Sat, 15 Nov 2025 13:24:24 GMT
last-modified: Mon, 12 Aug 2024 15:57:36 GMT
```

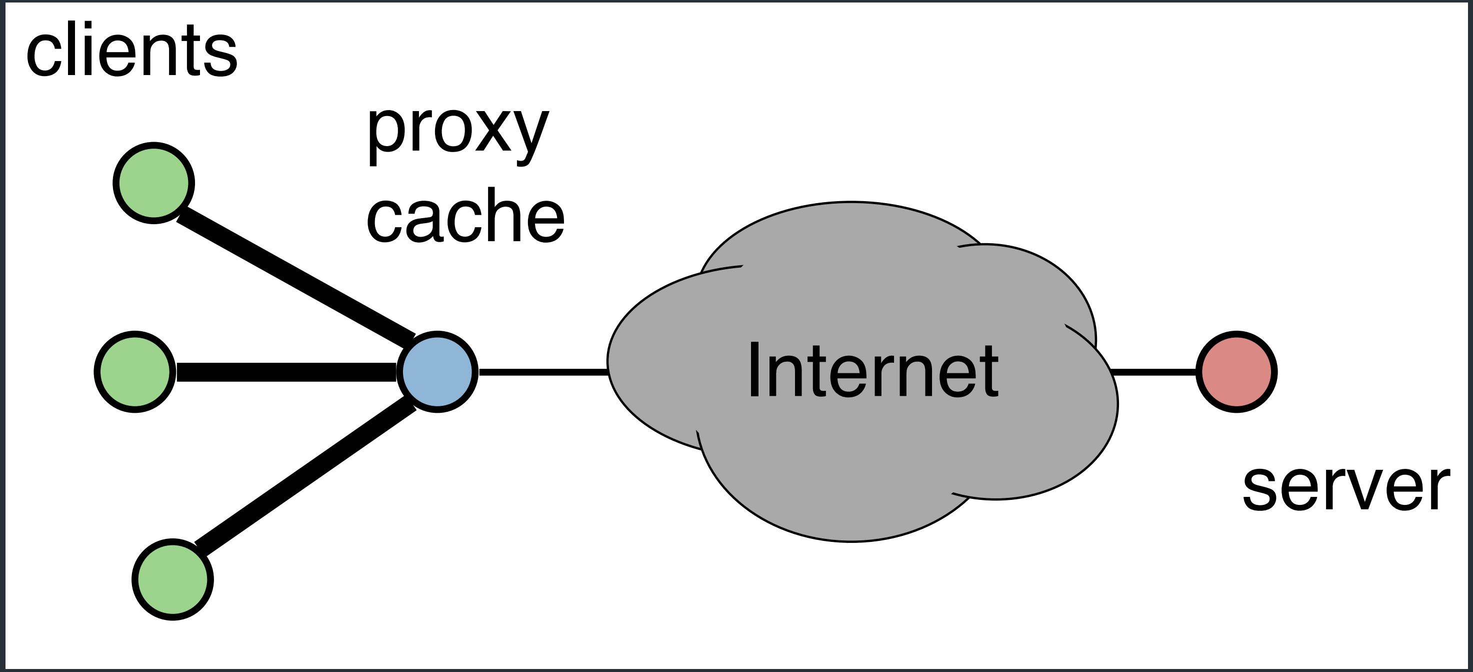
Headers returned with response
=> if caching is possible, how long to cache, etc.
=> Also possible to do conditional requests “If-Modified-Since” => server doesn’t send payload unless the resource has changed

How to Control Caching?

- Server sets options
  - Expires header
  - No-Cache header
- Client can do a conditional request:
  - Header option: if-modified-since
  - Server can reply with 304 NOT MODIFIED

Proxy caches

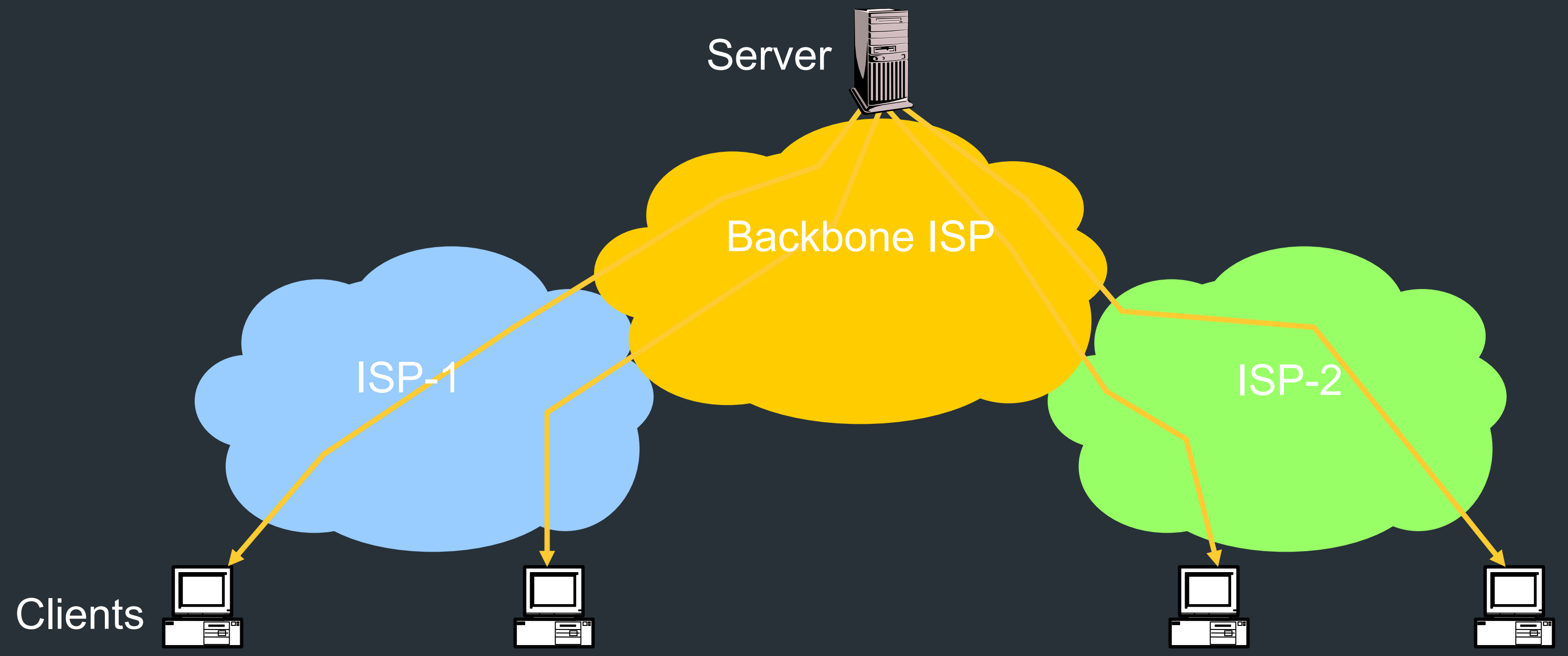
Classic way: proxy cache



⇒ Client first sends traffic to proxy server, which forwards to Internet
⇒ Proxy acts as cache

Implications
- Cache data close to clients (locality)
- Can also use to enforce security policies, or circumvent them (eg. open proxies)

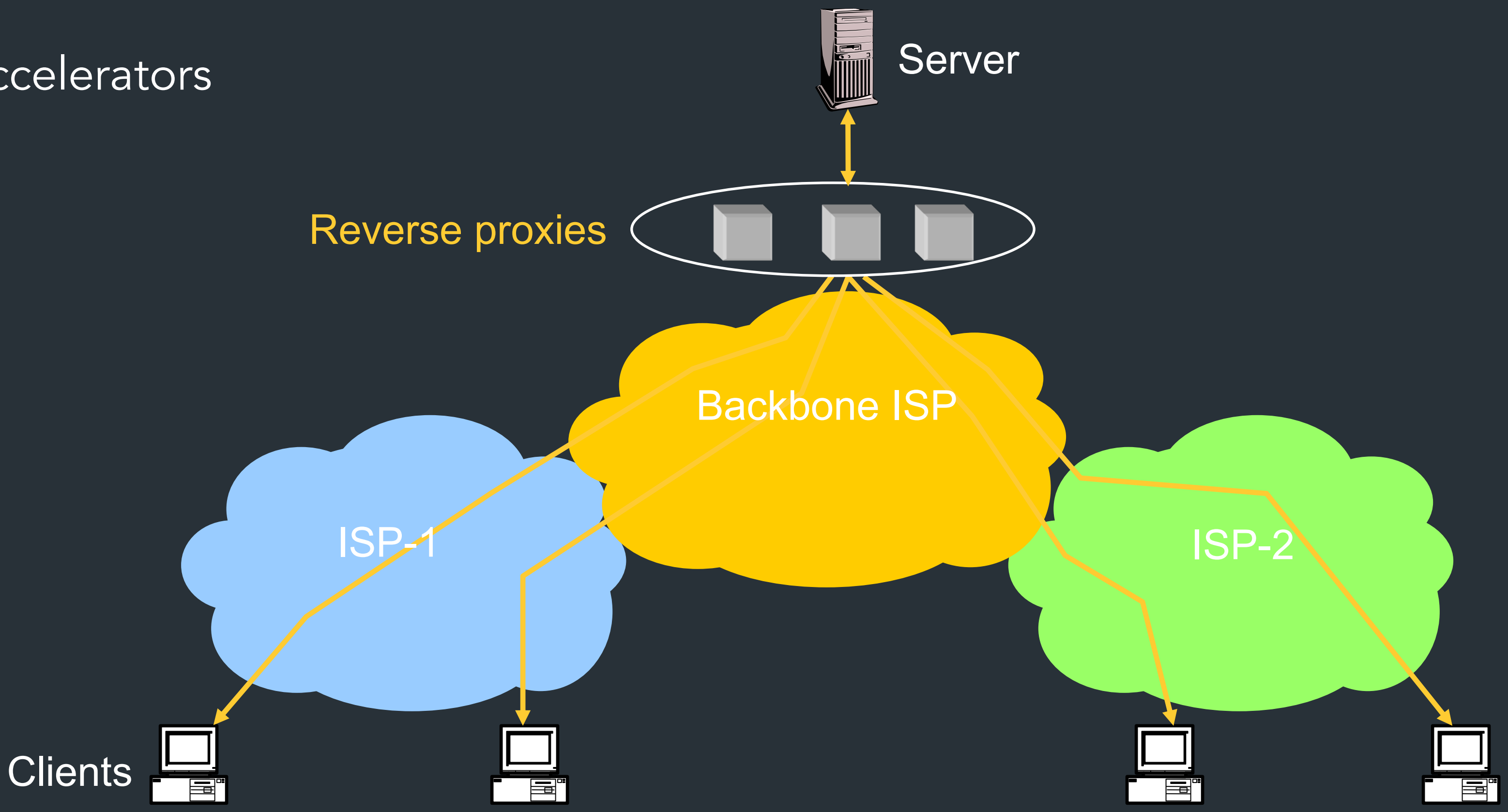
Caching *throughout* the network?



Reverse Proxies

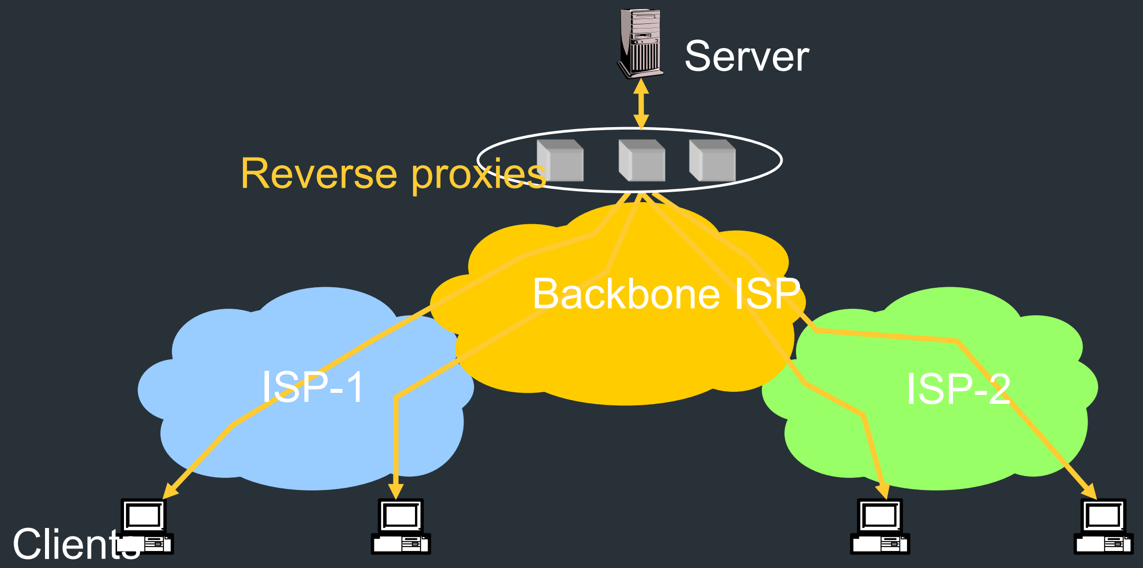
Idea: cache within network, appears like normal server
=> Reduce load on server, distribute load, do other tasks…

=> Also called accelerators



Reverse Proxies

Reasons for reverse proxying

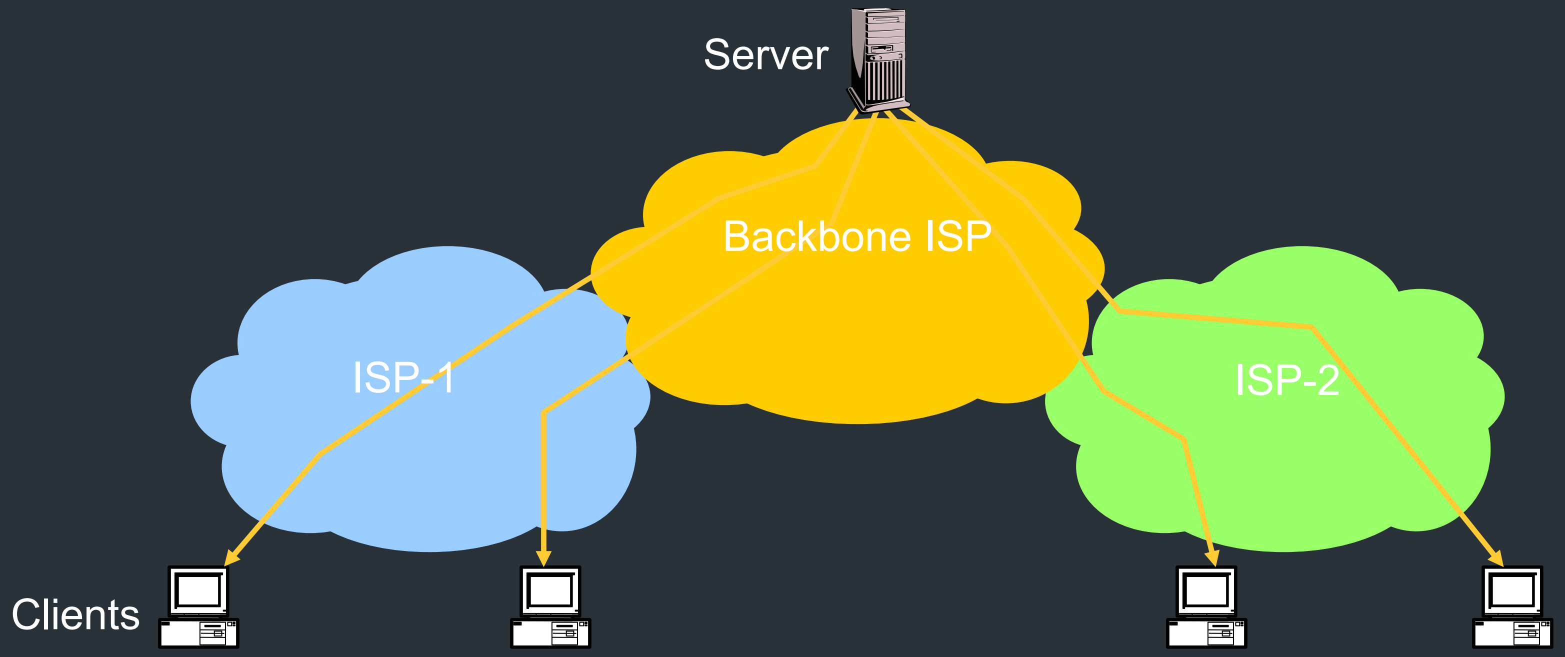


How well does caching work?

- Very well, up to a point
  - Large overlap in requested objects
  - Objects with one access place upper bound on hit ratio

- Example: Wikipedia
  - About 400 servers, 100 are HTTP Caches (Squid)
  - 85% Hit ratio for text, 98% for media

Where to cache content?



- Client (browser): avoid extra network transfers
- Server: reduce load on the server
- Service Provider: reduce external traffic

Content Distribution Networks (CDNs)

Companies that specialize in providing caching services (among other things)

=> Akamai, Cloudflare, …

Content Distribution Networks (CDNs)

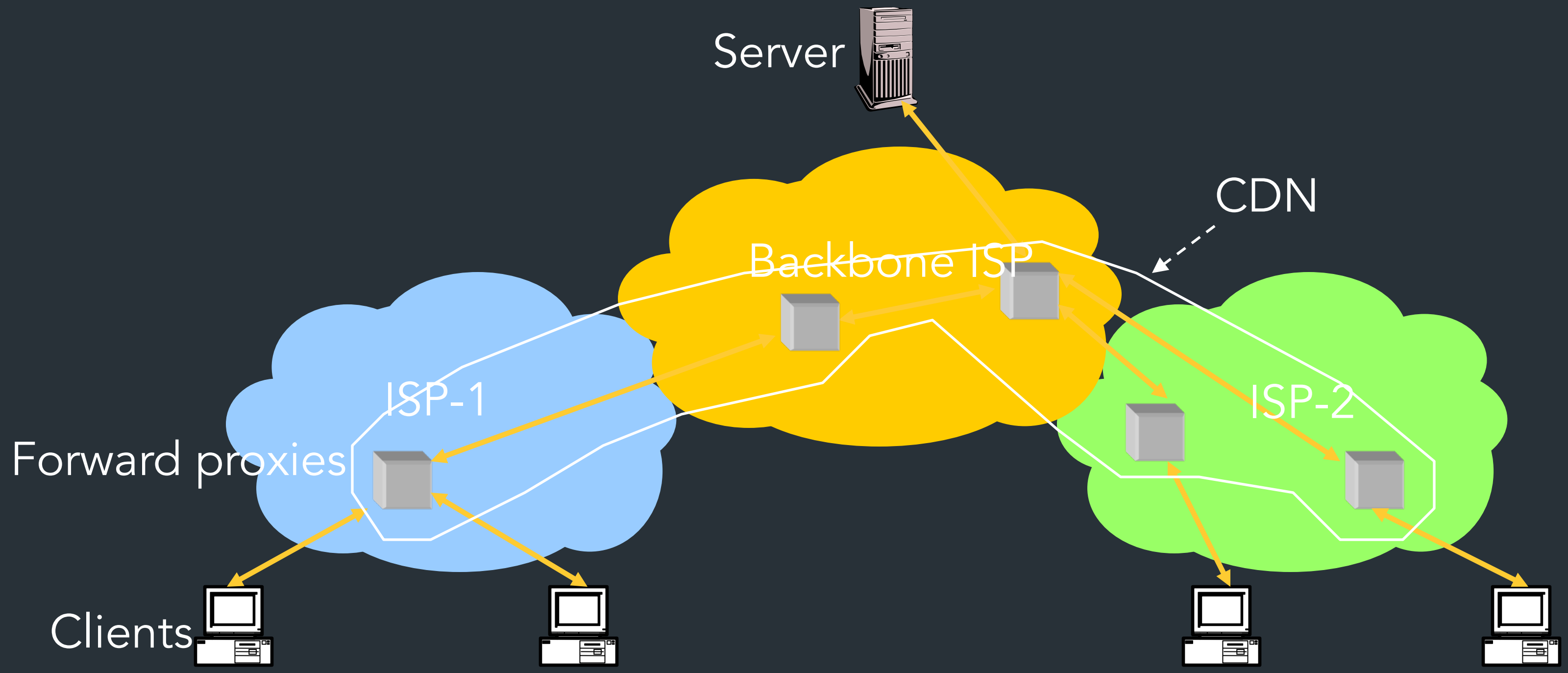
Companies that specialize in providing caching services (among other things)

=> Akamai, Cloudflare, …

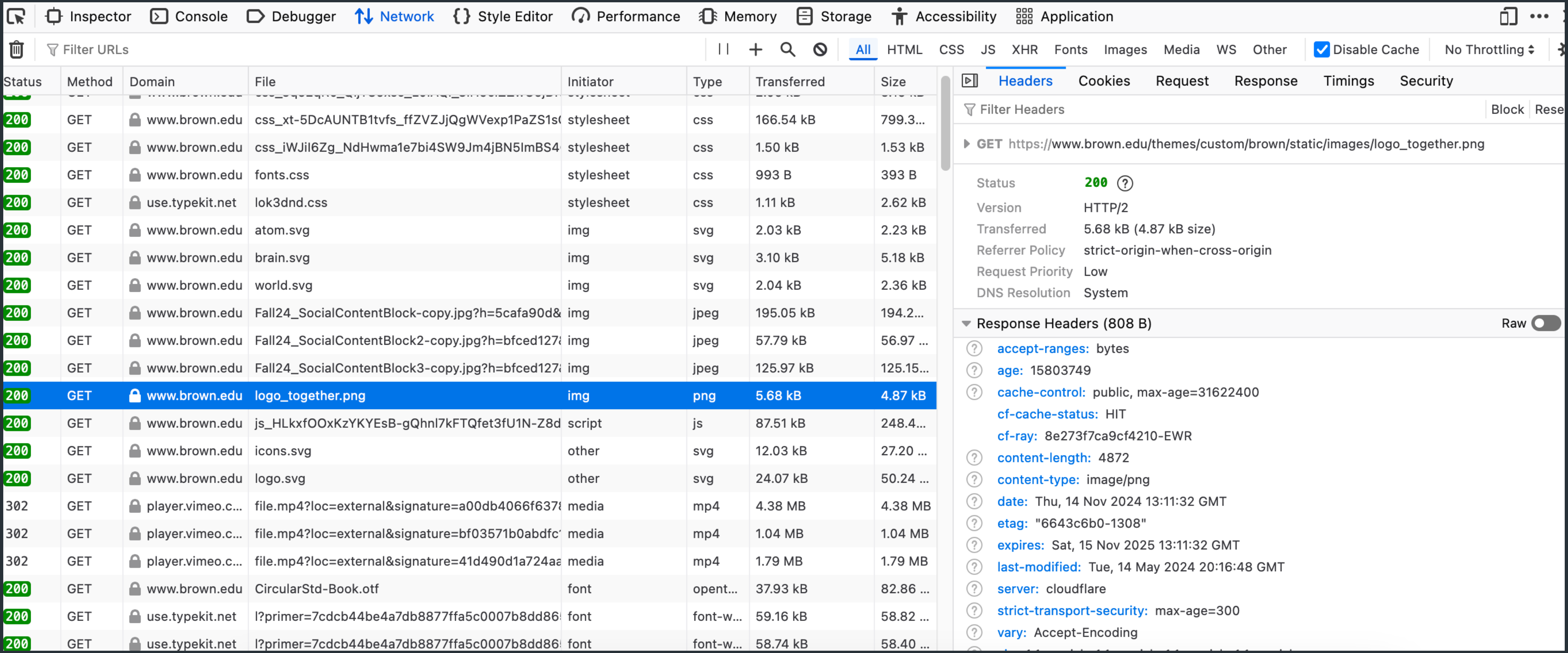
- Provides caching throughout network

- Can also do some processing

An Example CDN



```
dig www.brown.edu @10.1.1.10



;; ANSWER SECTION:
www.brown.edu. 3600 IN CNAME www.brown.edu.cdn.cloudflare.net.
www.brown.edu.cdn.cloudflare.net. 195 IN A 104.18.2.173
www.brown.edu.cdn.cloudflare.net. 195 IN A 104.18.3.173


;; Query time: 75 msec
;; SERVER: 10.1.1.10#53(10.1.1.10)
;; WHEN: Thu Nov 14 08:14:16 EST 2024
;; MSG SIZE  rcvd: 120
```



Status	Method	Domain	File	Initiator	Type	Transferred	Size
200	GET	www.brown.edu	css_xt-5DcAUNTB1tvfs_ffZVZJjQgWVexp1PaZS1s	stylesheet	css	166.54 kB	799.3...
200	GET	www.brown.edu	css_iWJil6Zg_NdHwma1e7bi4SW9Jm4jBN5lmBS4	stylesheet	css	1.50 kB	1.53 kB
200	GET	www.brown.edu	fonts.css	stylesheet	css	993 B	393 B
200	GET	use.typekit.net	lok3dnd.css	stylesheet	css	1.11 kB	2.62 kB
200	GET	www.brown.edu	atom.svg	img	svg	2.03 kB	2.23 kB
200	GET	www.brown.edu	brain.svg	img	svg	3.10 kB	5.18 kB
200	GET	www.brown.edu	world.svg	img	svg	2.04 kB	2.36 kB
200	GET	www.brown.edu	Fall24_SocialContentBlock-copy.jpg?h=5cafa90d&	img	jpeg	195.05 kB	194.2...
200	GET	www.brown.edu	Fall24_SocialContentBlock2-copy.jpg?h=bfced127	img	jpeg	57.79 kB	56.97 ...
200	GET	www.brown.edu	Fall24_SocialContentBlock3-copy.jpg?h=bfced127	img	jpeg	125.97 kB	125.15...
200	GET	www.brown.edu	logo_together.png	img	png	5.68 kB	4.87 kB
200	GET	www.brown.edu	js_HLkxfOOxKzYKYEsB-gQhnl7kFTQfet3fU1N-Z8d	script	js	87.51 kB	248.4...
200	GET	www.brown.edu	icons.svg	other	svg	12.03 kB	27.20 ...
200	GET	www.brown.edu	logo.svg	other	svg	24.07 kB	50.24 ...
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=a00db4066f637	media	mp4	4.38 MB	4.38 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=bf03571b0abdfc	media	mp4	1.04 MB	1.04 MB
302	GET	player.vimeo.c...	file.mp4?loc=external&signature=41d490d1a724a	media	mp4	1.79 MB	1.79 MB
200	GET	www.brown.edu	CircularStd-Book.otf	font	opent...	37.93 kB	82.86 ...
200	GET	use.typekit.net	l?primer=7cdcb44be4a7db8877ffa5c0007b8dd86	font	font-w...	59.16 kB	58.82 ...
200	GET	use.typekit.net	l?primer=7cdcb44be4a7db8877ffa5c0007b8dd86	font	font-w...	58.74 kB	58.40 ...

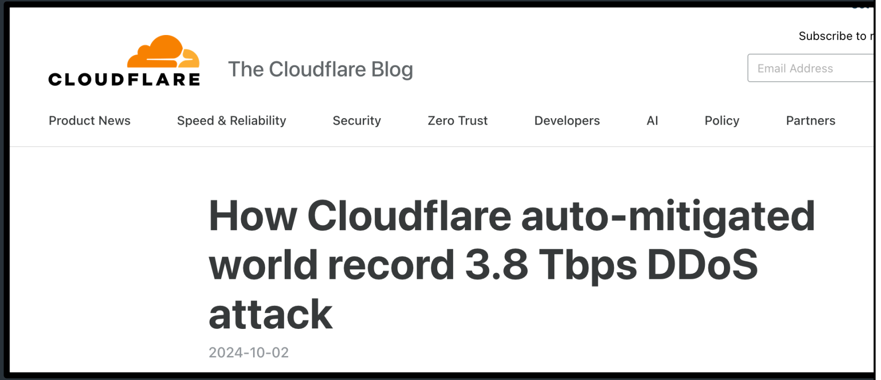
GET https://www.brown.edu/themes/custom/brown/static/images/logo_together.png

Status 200
Version HTTP/2
Transferred 5.68 kB (4.87 kB size)
Referrer Policy strict-origin-when-cross-origin
Request Priority Low
DNS Resolution System

Response Headers (808 B) Raw

```
accept-ranges: bytes
age: 15803749
cache-control: public, max-age=31622400
cf-cache-status: HIT
cf-ray: 8e273f7ca9cf4210-EWR
content-length: 4872
content-type: image/png
date: Thu, 14 Nov 2024 13:11:32 GMT
etag: "6643c6b0-1308"
expires: Sat, 15 Nov 2025 13:11:32 GMT
last-modified: Tue, 14 May 2024 20:16:48 GMT
server: cloudflare
strict-transport-security: max-age=300
vary: Accept-Encoding
```

DDoS mitigation via CDN



CLOUDFLARE The Cloudflare Blog

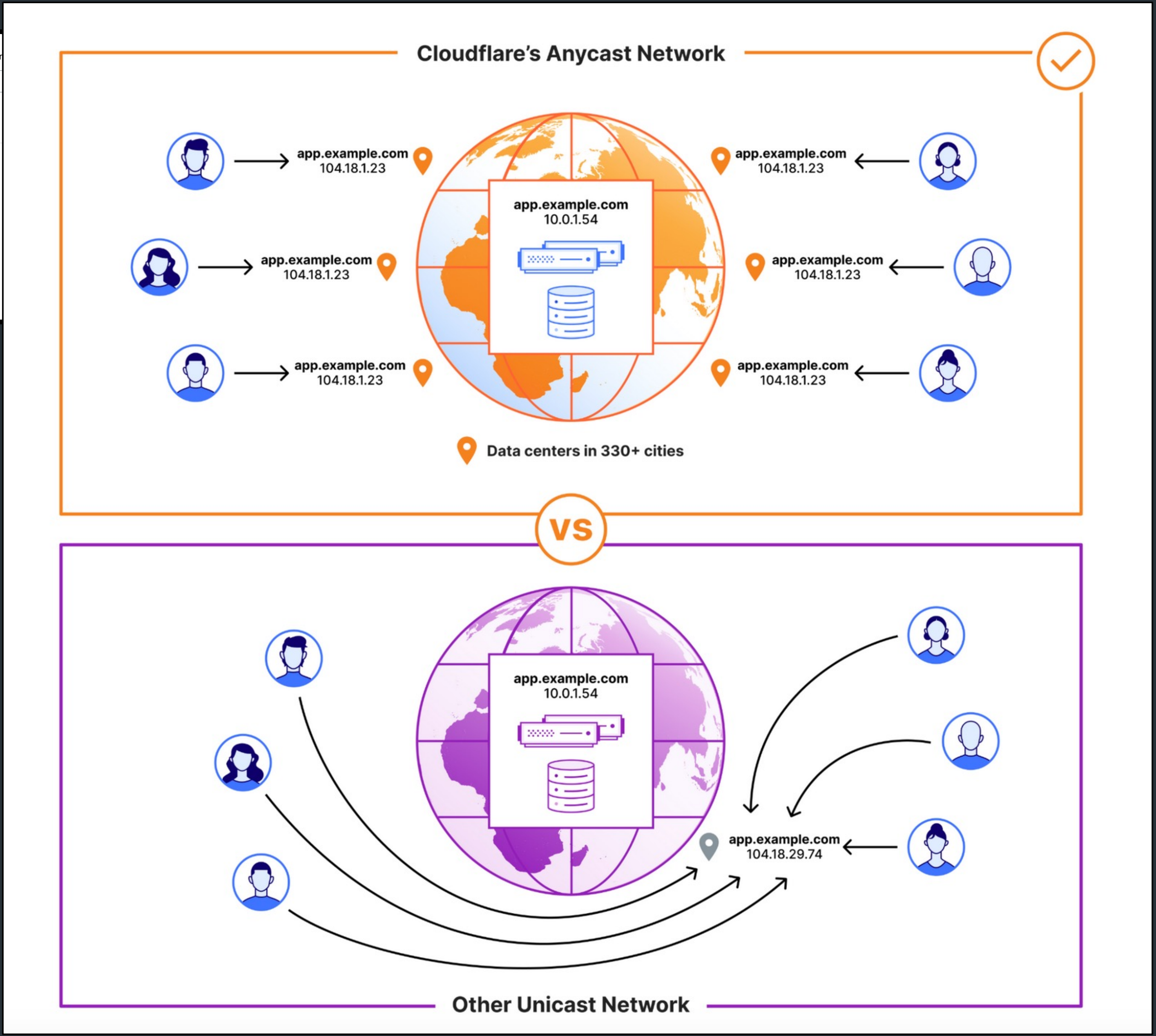
Subscribe to r

Email Address

Product News Speed & Reliability Security Zero Trust Developers AI Policy Partners

How Cloudflare auto-mitigated world record 3.8 Tbps DDoS attack

2024-10-02



Link

Content Distribution Networks (CDNs)

Companies that specialize in providing caching services (among other things)

=> Akamai, Cloudflare, …

- Provide both caching throughout network
  - Pull: result from client requests
  - Push: expectation of high access rates to some objects
- Can also do some processing
  - Deploy code to handle some dynamic requests
  - Can do other things, such as transcoding

How Akamai works

Akamai has cache servers deployed close to clients

- Co-located with many ISPs

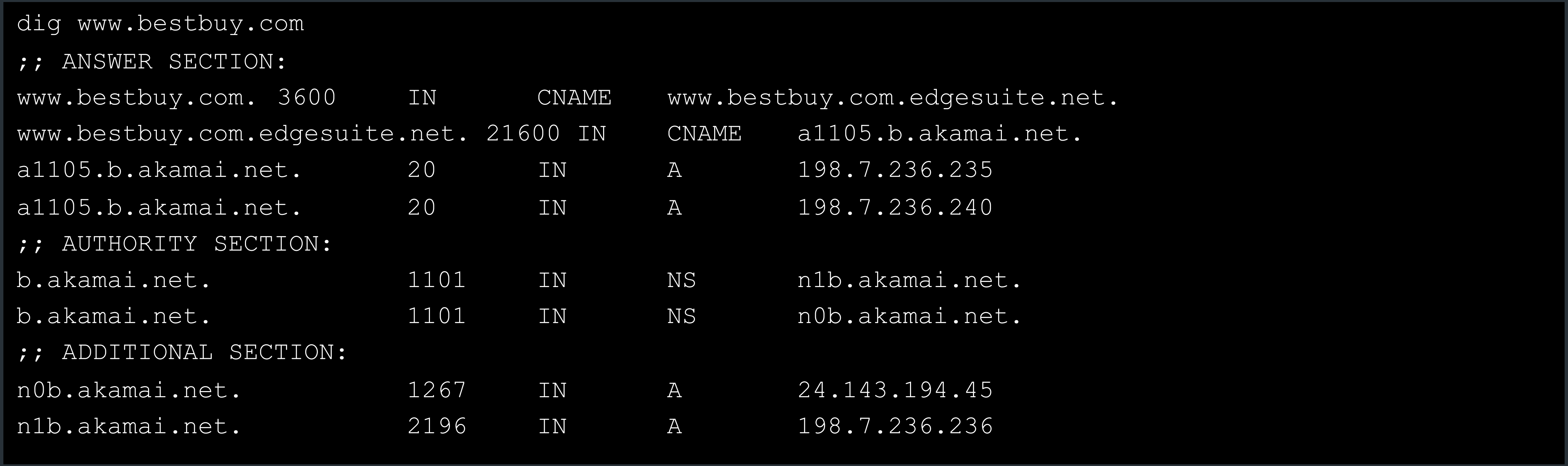
- Challenge: make same domain name resolve to a proxy close to the client

- Lots of DNS tricks. BestBuy is a customer
  - Delegate name resolution to Akamai (via a CNAME)

Other CDNs

- Akamai, Limelight, Cloudflare
- Amazon, Facebook, Google, Microsoft
- Netflix
- Where to place content?
- Which content to place? Pre-fetch or cache?

DNS Resolution



```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com. 3600     IN       CNAME    www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN     CNAME    a1105.b.akamai.net.
a1105.b.akamai.net.       20       IN       A        198.7.236.235
a1105.b.akamai.net.       20       IN       A        198.7.236.240
;; AUTHORITY SECTION:
b.akamai.net.             1101     IN       NS       n1b.akamai.net.
b.akamai.net.             1101     IN       NS       n0b.akamai.net.
;; ADDITIONAL SECTION:
n0b.akamai.net.           1267     IN       A        24.143.194.45
n1b.akamai.net.           2196     IN       A        198.7.236.236
```

- n1b.akamai.net finds an edge server close to the client’s local resolver
    - Uses knowledge of network: BGP feeds, traceroutes. *Their secret sauce…*

Example

From Brown

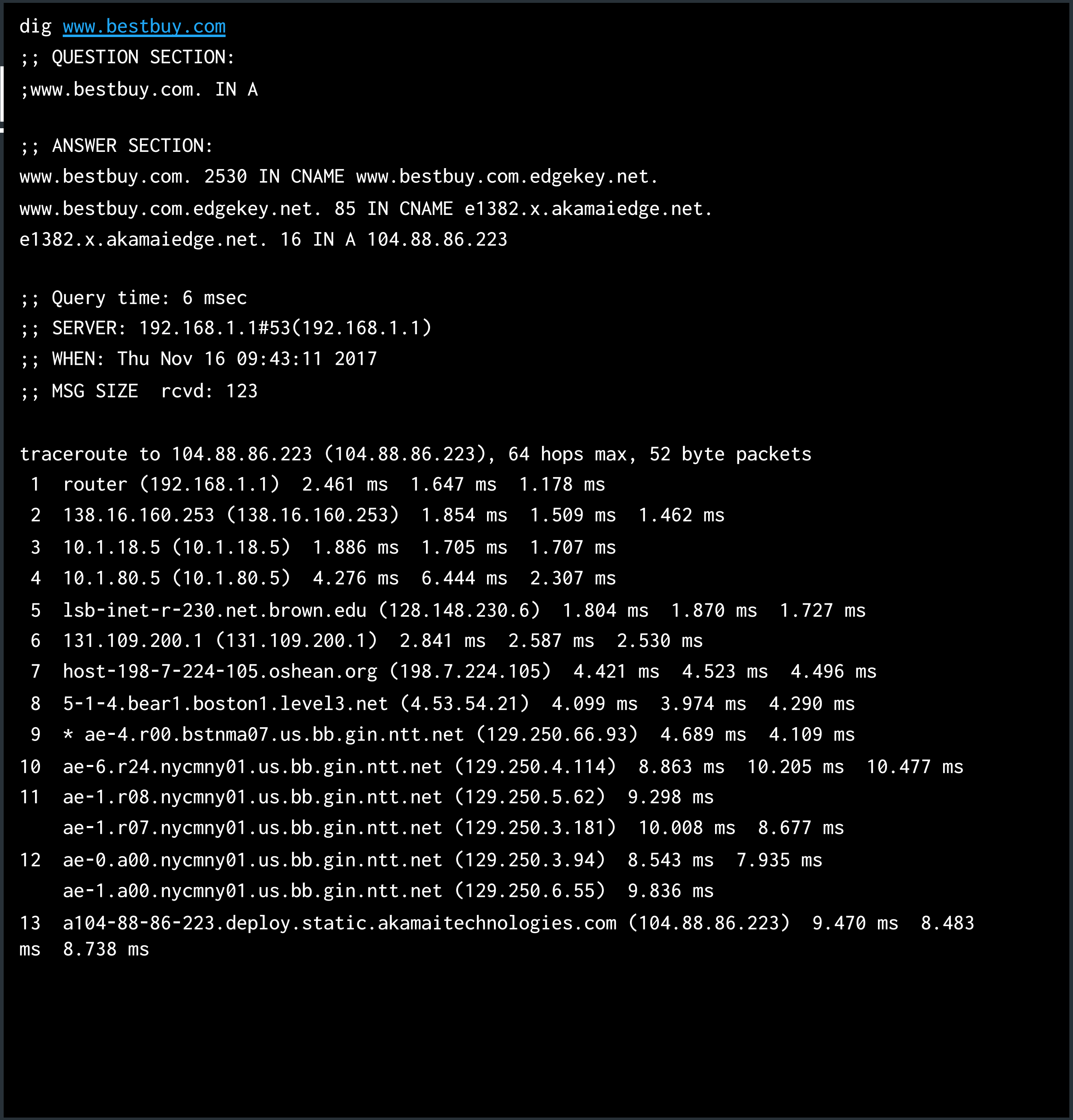
```
dig www.bestbuy.com
;; ANSWER SECTION:
www.bestbuy.com.  3600  IN    CNAME www.bestbuy.com.edgesuite.net.
www.bestbuy.com.edgesuite.net. 21600 IN  CNAME a1105.b.akamai.net.
a1105.b.akamai.net.     20    IN    A     198.7.236.235
a1105.b.akamai.net.     20    IN    A     198.7.236.240
```

- Ping time: 2.53ms

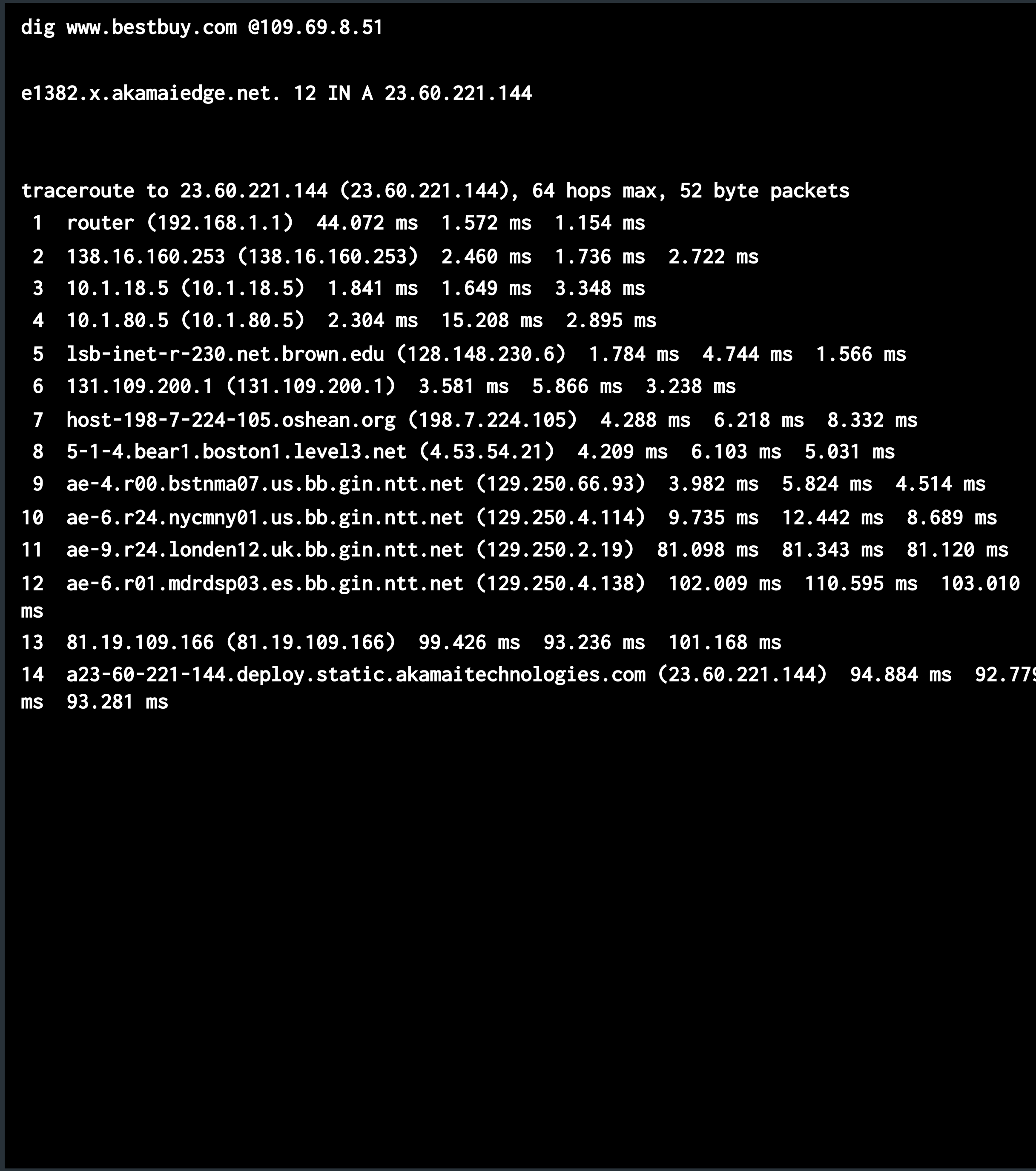
From Berkeley, CA

```
a1105.b.akamai.net.     20    IN    A     198.189.255.200
a1105.b.akamai.net.     20    IN    A     198.189.255.207
```

- Ping time: 3.20ms

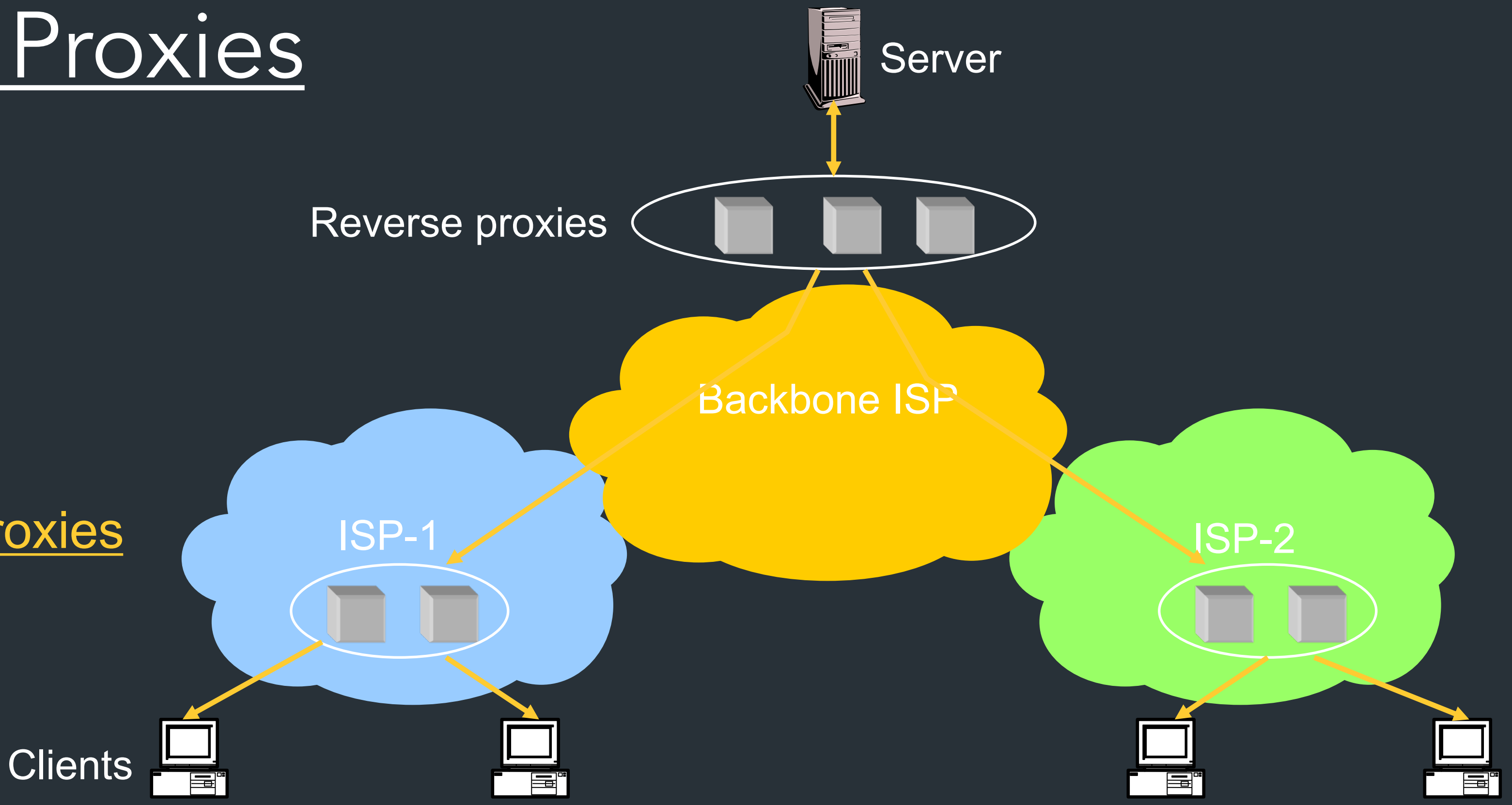


```
dig www.bestbuy.com
;; QUESTION SECTION:
;www.bestbuy.com. IN A

;; ANSWER SECTION:
www.bestbuy.com. 2530 IN CNAME www.bestbuy.com.edgekey.net.
www.bestbuy.com.edgekey.net. 85 IN CNAME e1382.x.akamaiedge.net.
e1382.x.akamaiedge.net. 16 IN A 104.88.86.223

;; Query time: 6 msec
;; SERVER: 192.168.1.1#53(192.168.1.1)
;; WHEN: Thu Nov 16 09:43:11 2017
;; MSG SIZE  rcvd: 123

traceroute to 104.88.86.223 (104.88.86.223), 64 hops max, 52 byte packets
 1  router (192.168.1.1)  2.461 ms  1.647 ms  1.178 ms
 2  138.16.160.253 (138.16.160.253)  1.854 ms  1.509 ms  1.462 ms
 3  10.1.18.5 (10.1.18.5)  1.886 ms  1.705 ms  1.707 ms
 4  10.1.80.5 (10.1.80.5)  4.276 ms  6.444 ms  2.307 ms
 5  lsb-inet-r-230.net.brown.edu (128.148.230.6)  1.804 ms  1.870 ms  1.727 ms
 6  131.109.200.1 (131.109.200.1)  2.841 ms  2.587 ms  2.530 ms
 7  host-198-7-224-105.oshean.org (198.7.224.105)  4.421 ms  4.523 ms  4.496 ms
 8  5-1-4.bear1.boston1.level3.net (4.53.54.21)  4.099 ms  3.974 ms  4.290 ms
 9  * ae-4.r00.bstnma07.us.bb.gin.ntt.net (129.250.66.93)  4.689 ms  4.109 ms
10  ae-6.r24.nycmny01.us.bb.gin.ntt.net (129.250.4.114)  8.863 ms  10.205 ms  10.477 ms
11  ae-1.r08.nycmny01.us.bb.gin.ntt.net (129.250.5.62)  9.298 ms
    ae-1.r07.nycmny01.us.bb.gin.ntt.net (129.250.3.181)  10.008 ms  8.677 ms
12  ae-0.a00.nycmny01.us.bb.gin.ntt.net (129.250.3.94)  8.543 ms  7.935 ms
    ae-1.a00.nycmny01.us.bb.gin.ntt.net (129.250.6.55)  9.836 ms
13  a104-88-86-223.deploy.static.akamaitechnologies.com (104.88.86.223)  9.470 ms  8.483
ms  8.738 ms
```



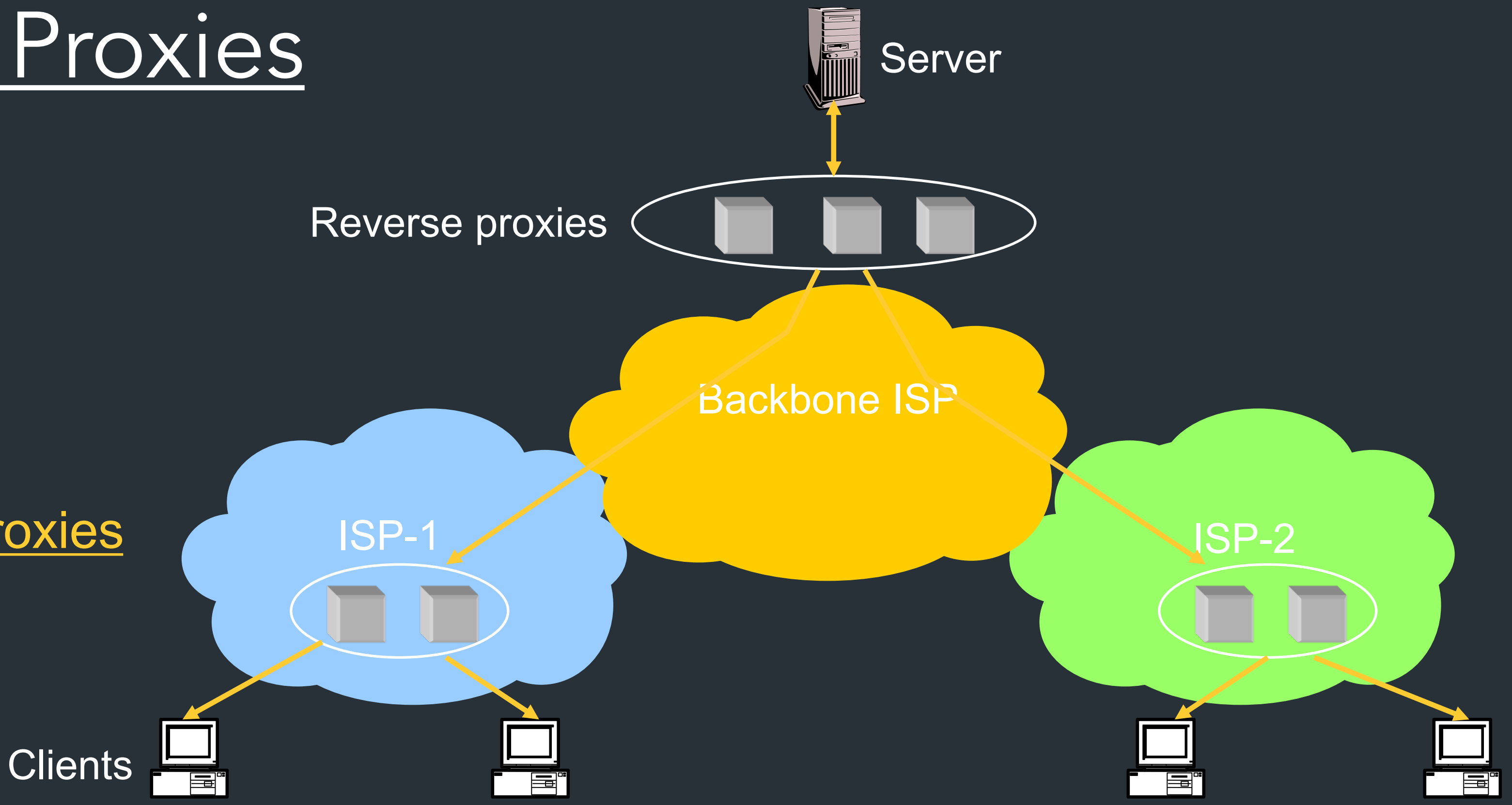
```
dig www.bestbuy.com @109.69.8.51

e1382.x.akamaiedge.net. 12 IN A 23.60.221.144


traceroute to 23.60.221.144 (23.60.221.144), 64 hops max, 52 byte packets
 1  router (192.168.1.1)  44.072 ms  1.572 ms  1.154 ms
 2  138.16.160.253 (138.16.160.253)  2.460 ms  1.736 ms  2.722 ms
 3  10.1.18.5 (10.1.18.5)  1.841 ms  1.649 ms  3.348 ms
 4  10.1.80.5 (10.1.80.5)  2.304 ms  15.208 ms  2.895 ms
 5  lsb-inet-r-230.net.brown.edu (128.148.230.6)  1.784 ms  4.744 ms  1.566 ms
 6  131.109.200.1 (131.109.200.1)  3.581 ms  5.866 ms  3.238 ms
 7  host-198-7-224-105.oshean.org (198.7.224.105)  4.288 ms  6.218 ms  8.332 ms
 8  5-1-4.bear1.boston1.level3.net (4.53.54.21)  4.209 ms  6.103 ms  5.031 ms
 9  ae-4.r00.bstnma07.us.bb.gin.ntt.net (129.250.66.93)  3.982 ms  5.824 ms  4.514 ms
10  ae-6.r24.nycmny01.us.bb.gin.ntt.net (129.250.4.114)  9.735 ms  12.442 ms  8.689 ms
11  ae-9.r24.londen12.uk.bb.gin.ntt.net (129.250.2.19)  81.098 ms  81.343 ms  81.120 ms
12  ae-6.r01.mdrdsp03.es.bb.gin.ntt.net (129.250.4.138)  102.009 ms  110.595 ms  103.010
ms
13  81.19.109.166 (81.19.109.166)  99.426 ms  93.236 ms  101.168 ms
14  a23-60-221-144.deploy.static.akamaitechnologies.com (23.60.221.144)  94.884 ms  92.779
ms  93.281 ms
```

Forward Proxies



Forward proxies

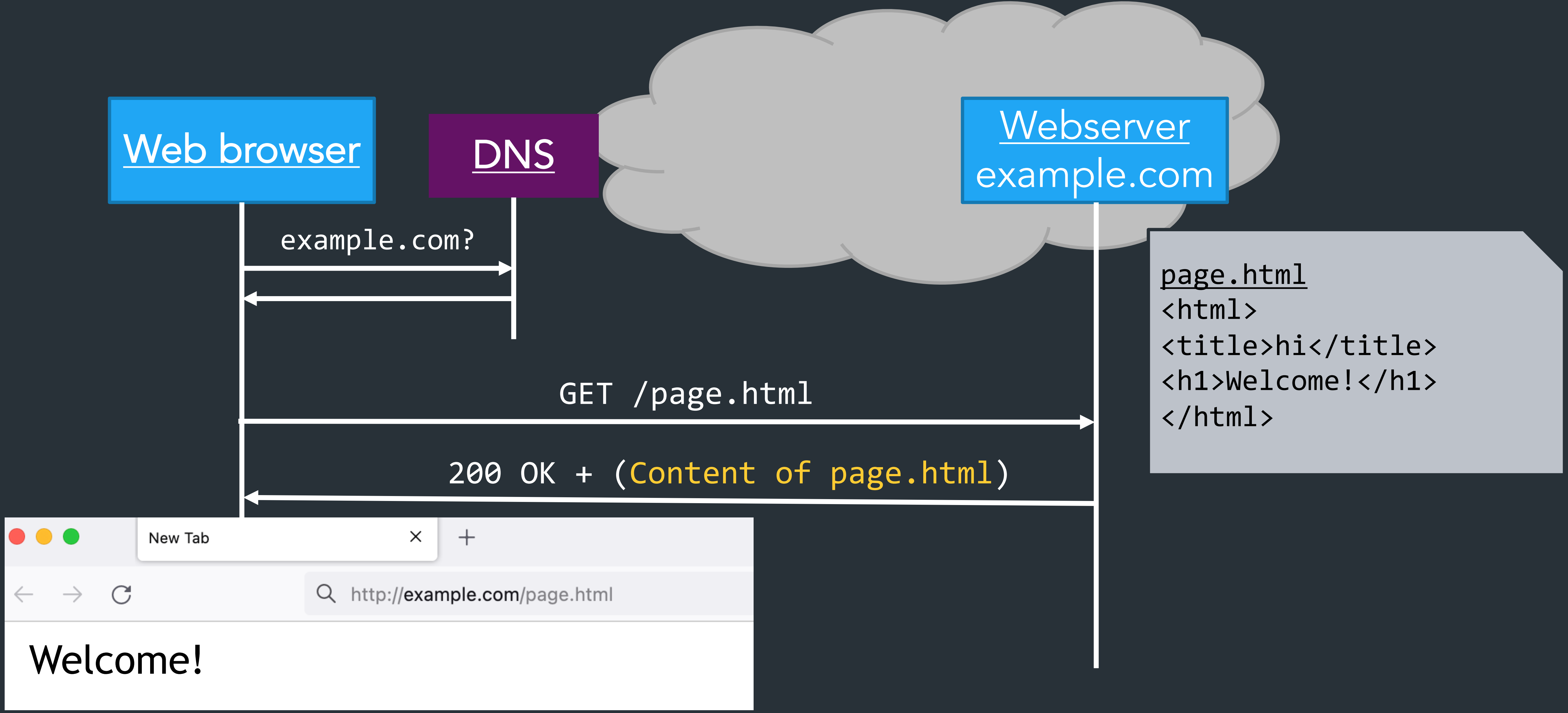
Forward Proxies



Forward proxies

Typically done by ISPs or Enterprises

- Reduce network traffic and decrease latency
- May be transparent or configured

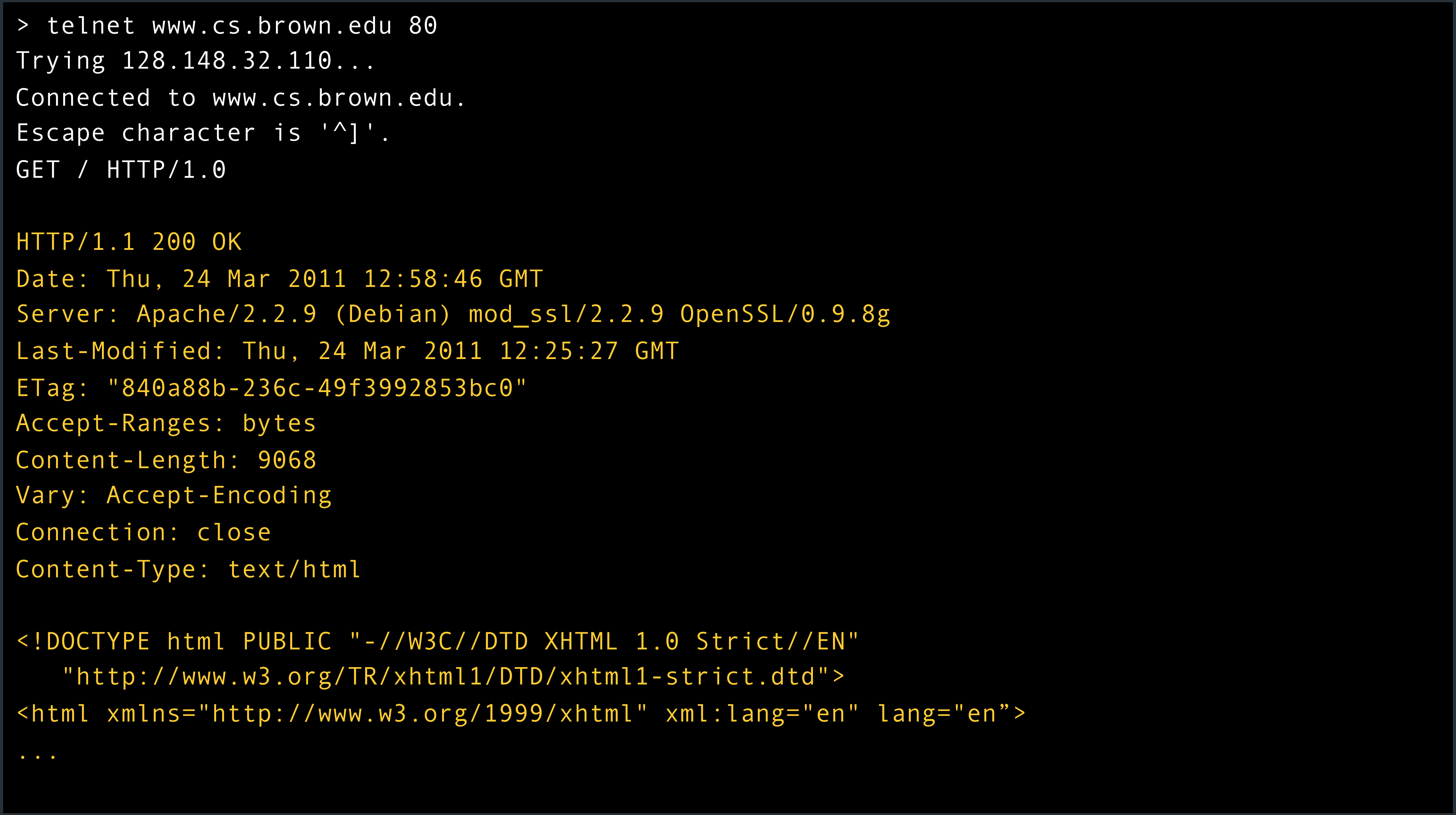


http://example.com/page.html

Welcome!

```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```

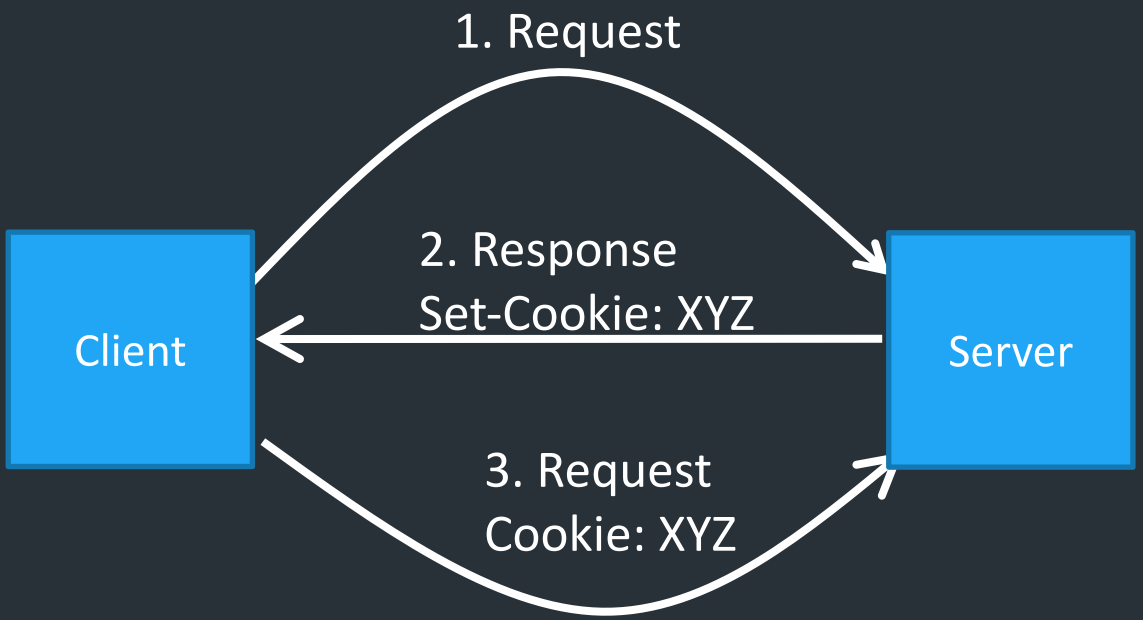
Server returns response (in this case, with HTML)



```
> telnet www.cs.brown.edu 80
Trying 128.148.32.110...
Connected to www.cs.brown.edu.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 24 Mar 2011 12:58:46 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Thu, 24 Mar 2011 12:25:27 GMT
ETag: "840a88b-236c-49f3992853bc0"
Accept-Ranges: bytes
Content-Length: 9068
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en”>
...
```

HTTP Cookies

- Client-side state maintenance
  - Client stores small state on behalf of server
  - Sends request in future requests to the server
  - Cookie value is meaningful to the server (e.g., session id)
- Can provide authentication



Modern web pages and HTTP

- Web APIs: HTTP response/requests are a standard way to ask for *anything*
- *Modern web pages: use Javascript to make lots of requests without reloading page*
  - *And can use APIs for all kinds of other stuff*

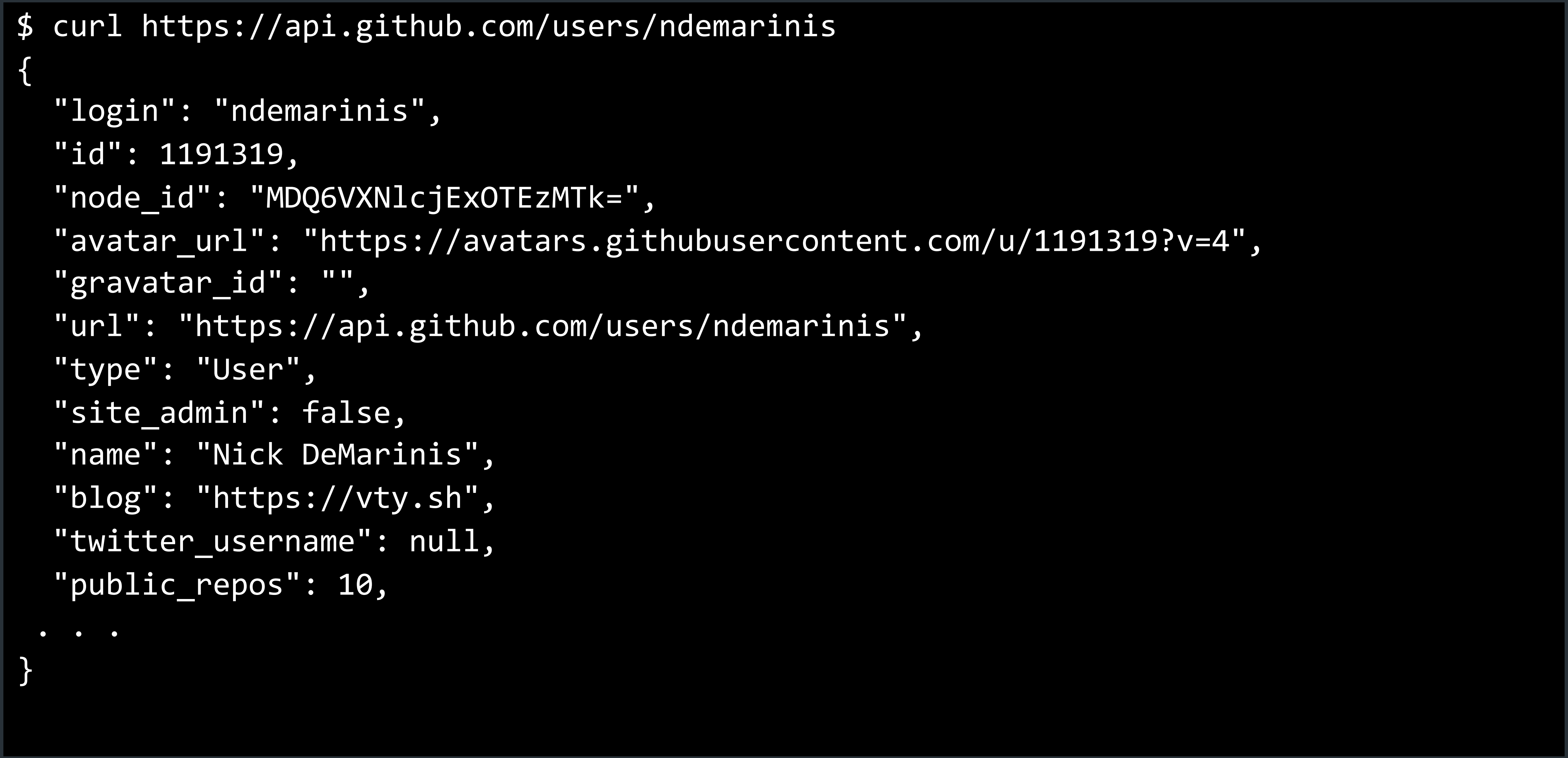
Example: Github public API

```
$ curl https://api.github.com/users/ndemarinis
{
  "login": "ndemarinis",
  "id": 1191319,
  "node_id": "MDQ6VXNlcjExOTEzMTk=",
  "avatar_url": "https://avatars.githubusercontent.com/u/1191319?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/ndemarinis",
  "type": "User",
  "site_admin": false,
  "name": "Nick DeMarinis",
  "blog": "https://vty.sh",
  "twitter_username": null,
  "public_repos": 10,
 . . .
}
```

Modern web pages and HTTP

- Web APIs: HTTP response/requests are a standard way to ask for *anything*
- *Modern web pages: use Javascript to make lots of requests without reloading page*
  - *And can use APIs for all kinds of other stuff*

Example: Github public API



```
$ curl https://api.github.com/users/ndemarinis
{
  "login": "ndemarinis",
  "id": 1191319,
  "node_id": "MDQ6VXNlcjExOTEzMTk=",
  "avatar_url": "https://avatars.githubusercontent.com/u/1191319?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/ndemarinis",
  "type": "User",
  "site_admin": false,
  "name": "Nick DeMarinis",
  "blog": "https://vty.sh",
  "twitter_username": null,
  "public_repos": 10,
 . . .
}
```

HTTP: What matters for performance?

Depends on type of request

- Lots of small requests (objects in a page)
- Some big requests (large download or video)

Small Requests

- Latency matters
- RTT dominates
- Major steps:
  - DNS lookup (if not cached)
  - Opening a TCP connection
  - Setting up TLS (optional, but now common)
  - Actually sending the request and receiving response

How can we reduce the number of connection setups?

- Keep the connection open and request all objects serially
  - Works for all objects coming from the same server
  - Which also means you don’t have to “open” the window each time

Persistent connections (HTTP/1.1)

Small Requests (cont)

- Second problem is that requests are serialized
  - Similar to stop-and-wait protocols!
- Two solutions
  - Pipelined requests (similar to sliding windows)
  - Parallel Connections
    - Browsers implement this differently—see “Inspect element”
  - How are these two approaches different?