The Web (HTTP)

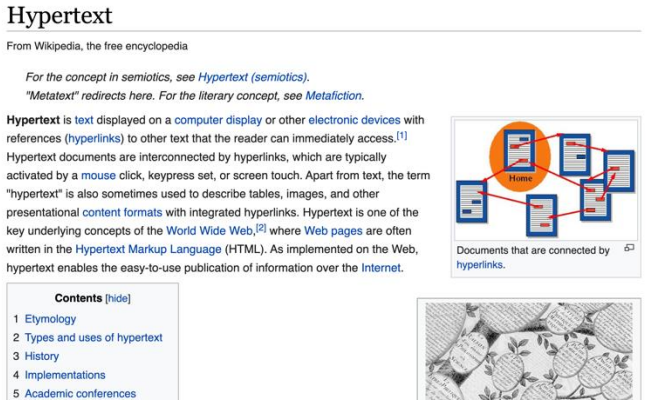
Lecture 6

http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Review of concepts



Hypertext

From Wikipedia, the free encyclopedia

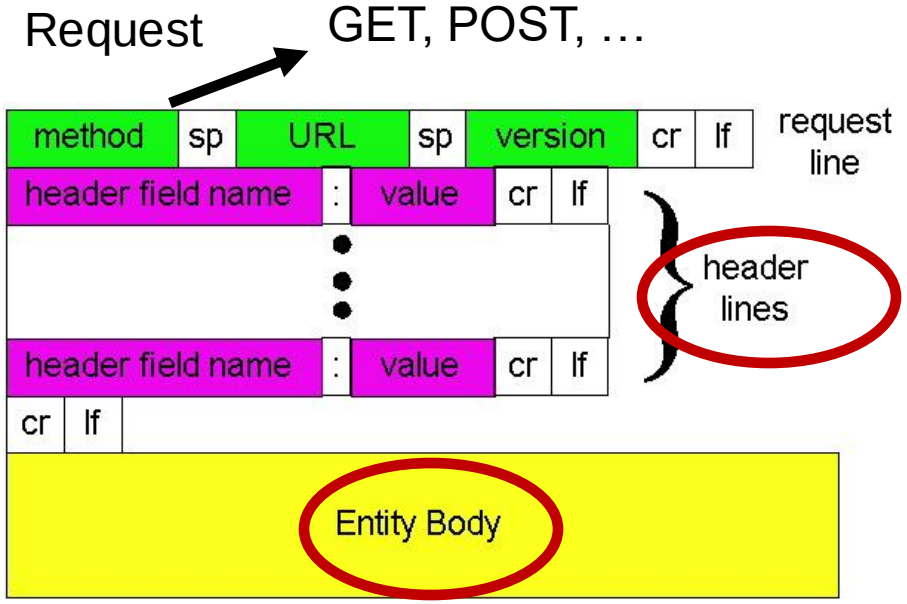
For the concept in semiotics, see Hypertext (semiotics).
"Metatext" redirects here. For the literary concept, see Metafiction.

Hypertext is text displayed on a computer display or other electronic devices with references (hyperlinks) to other text that the reader can immediately access.[1] Hypertext documents are interconnected by hyperlinks, which are typically activated by a mouse click, keypress set, or screen touch. Apart from text, the term "hypertext" is also sometimes used to describe tables, images, and other presentational content formats with integrated hyperlinks. Hypertext is one of the key underlying concepts of the World Wide Web,[2] where Web pages are often written in the Hypertext Markup Language (HTML). As implemented on the Web, hypertext enables the easy-to-use publication of information over the Internet.

Documents that are connected by hyperlinks.

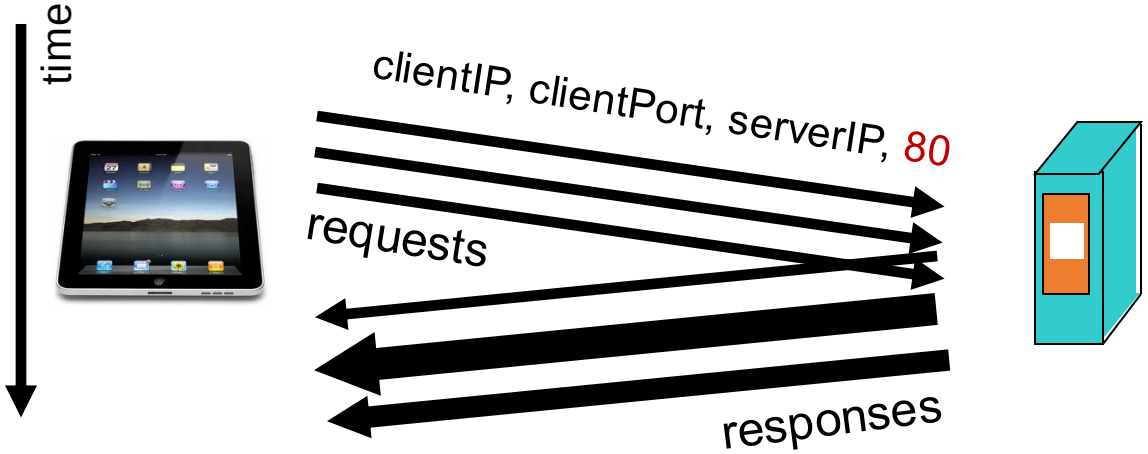
Contents [hide]

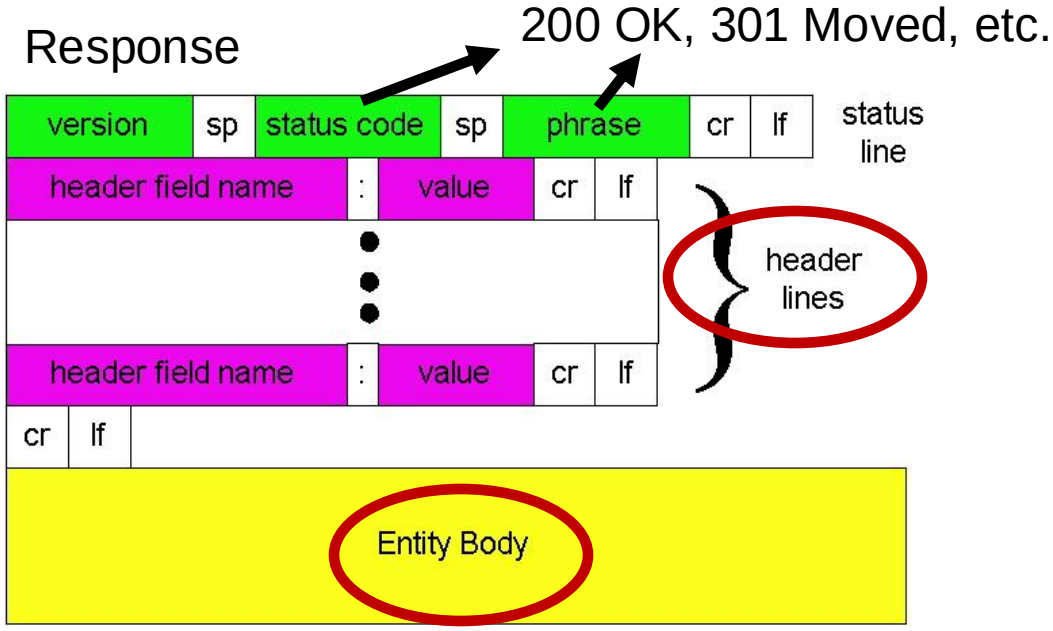
1 Etymology
2 Types and uses of hypertext
3 History
4 Implementations
5 Academic conferences



HyperText Transfer Protocol (HTTP)

Client-Server Protocol





HTTP Persistence

Two types of HTTP connectivity

Non-persistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses non-persistent connections

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

TCP is a reliable communication protocol provided by the transport layer. It requires setting up some resources (e.g., memory regions) for the connection to be set up at the endpoints before data communication.

Non-persistent HTTP (HTTP/1.0)

1a. HTTP client initiates TCP connection to HTTP server

1b. HTTP server at host “accepts” connection, notifying client

Suppose a user visits a page with text and 10 embedded images.

2. HTTP client sends HTTP request message

3. HTTP server receives request message, replies with response message containing requested object

time

Non-persistent HTTP (HTTP/1.0)

4. HTTP server closes TCP connection.

5. HTTP client receives response message containing HTML file, displays HTML. Parsing HTML file, finds 10 referenced image objects

time

6. Steps 1-5 repeated for each of 10 image objects

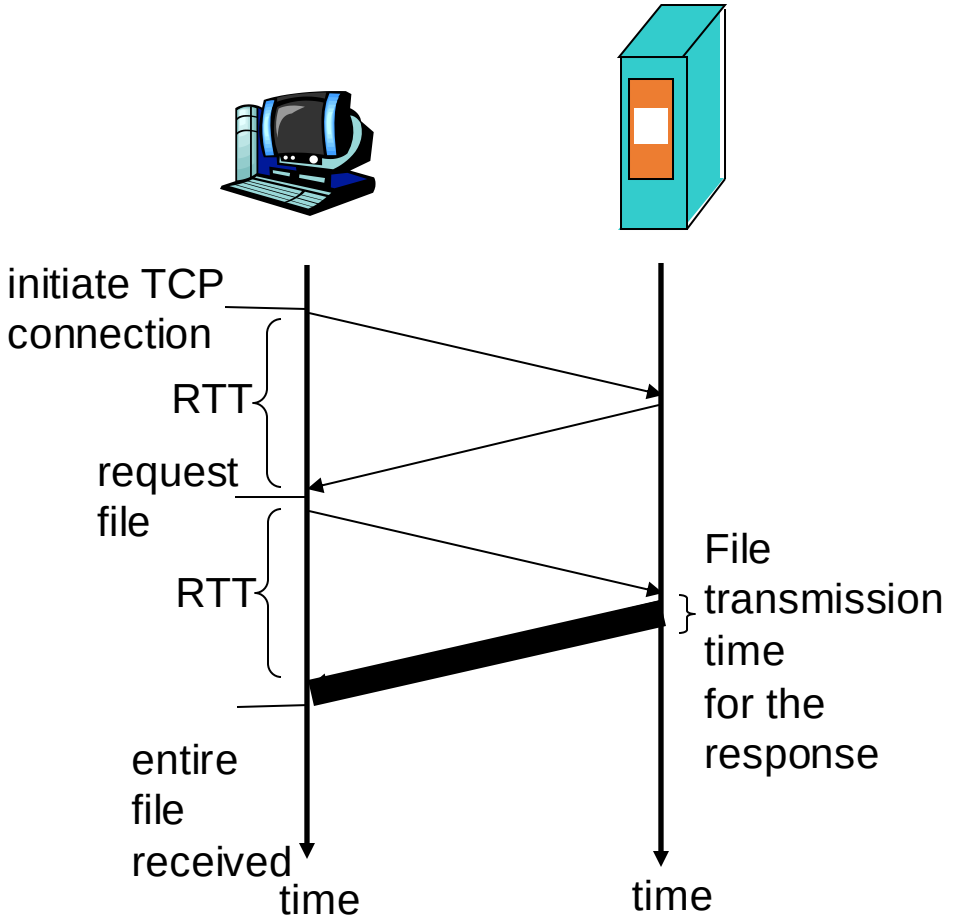
Single connection per object

Useful at a time when web pages contained 1 object: the base HTML file.

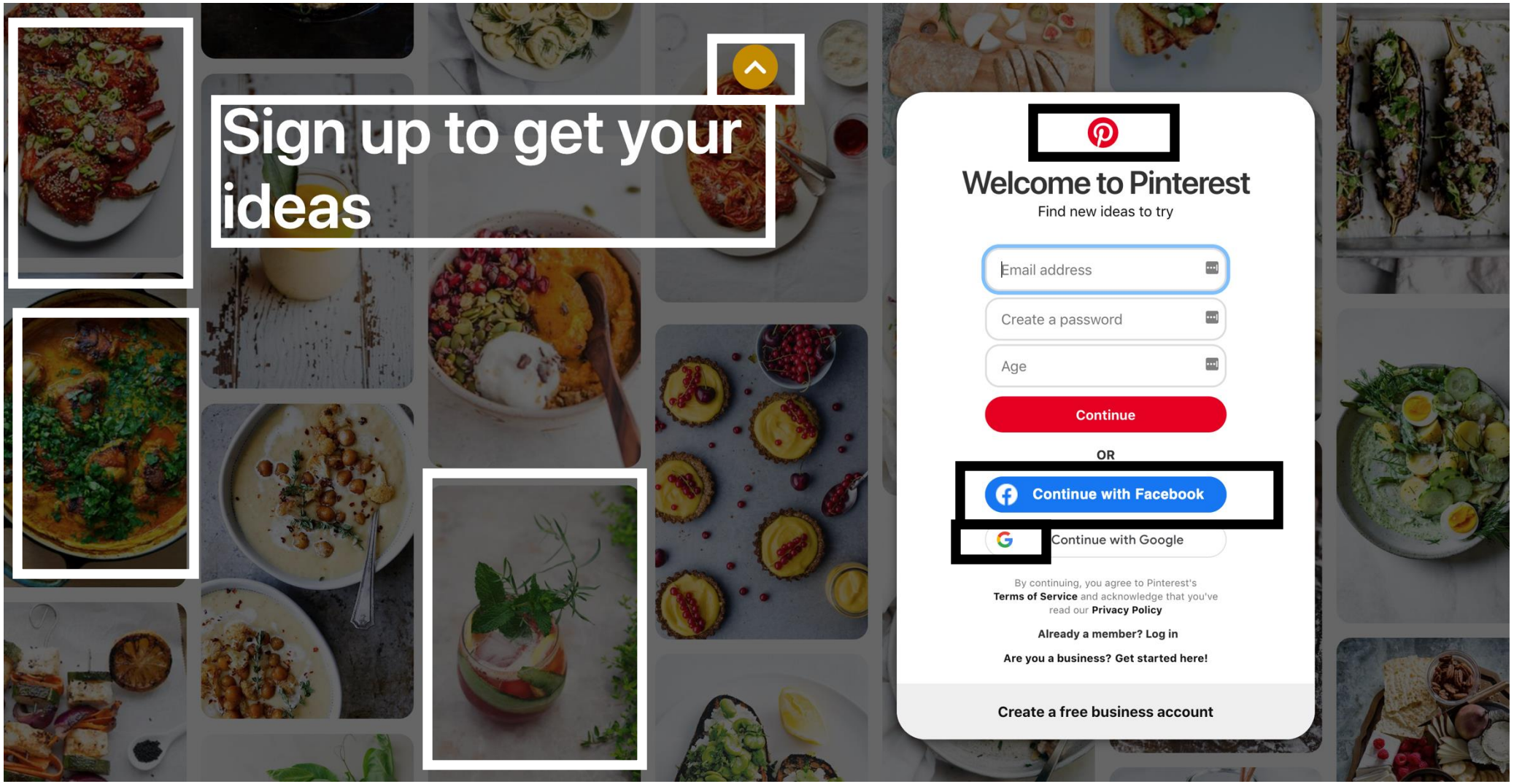
How long does it take to download an entire web page with non-persistent HTTP?
i.e.: before your browser can load the (entire) web page?

Non-persistent HTTP user response time

- Total delay = propagation + queueing + transmission
- Response time for the user
  - = sum of forward and backward total delays
- Round-Trip Time (RTT): total forward + backward delay for a “small” packet
  - Zero transmission delay
- Assumptions:
  - TCP initiation packet, response, HTTP requests are all “small” packets
  - No processing delays at the server
  - RTT is stable over time
- (2RTT + file transmission time) * #objects



Per-object overheads quickly add up



Modern web pages have 100s of objects in them.

Objects (e.g. images) may not be small.

Persistent HTTP (HTTP/1.1)

Suppose user visits a page with text and 10 images.

Web Server

1a. HTTP client initiates TCP connection to HTTP server

1b. HTTP server at host "accepts" connection, notifying client

2. HTTP client sends HTTP request message

3. HTTP server receives request message, replies with response message containing requested object

`Connection: keep-alive`

HTTP header introduced in HTTP/1.1

time

Persistent HTTP (HTTP/1.1)

4. HTTP server sends a response.

5. HTTP client receives response message containing HTML file, displays HTML. Parsing HTML file, finds 10 referenced image objects

Server keeps the TCP connection alive.

time

The 10 objects can be requested over the same TCP connection.

i.e., save an RTT per object (otherwise spent opening a new TCP connection in HTTP/1.0)

Persistent HTTP user response time

- Assume requests made one at a time (separate RTT per req)
- RTT + (RTT + file transmission time) * #objects
- Pipelining: send more than one HTTP request at a time
  - Extreme case: all requests in one (small) packet
  - RTT + (file transmission time) * #objects
  - In practice, dependencies between objects
- Compare with non-persistent:
  - (2RTT + file transmission time) * #objects
- Persistence (& pipelining) can save significant time, especially on high-RTT connections
- Other advantages of persistence: CPU savings, reduced network congestion, less memory (fewer connections)

Persistence vs. # of connections

- Persistence is distinct from the number of concurrent connections made by a client
- Your browser has the choice to open multiple connections to a server
  - HTTP spec suggests to limit this to a small number (2)
- Further, a single connection can have multiple HTTP requests in flight (pipelining) with persistent HTTP

> Clients that use persistent connections SHOULD limit the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than 2 connections with any server or proxy. A proxy SHOULD use up to 2*N connections to another server or proxy, where N is the number of simultaneously active users. These guidelines are intended to improve HTTP response times and avoid congestion.
>
> **8.2 Message Transmission Requirements**

Remembering Users On the Web

HTTP: Remembering users

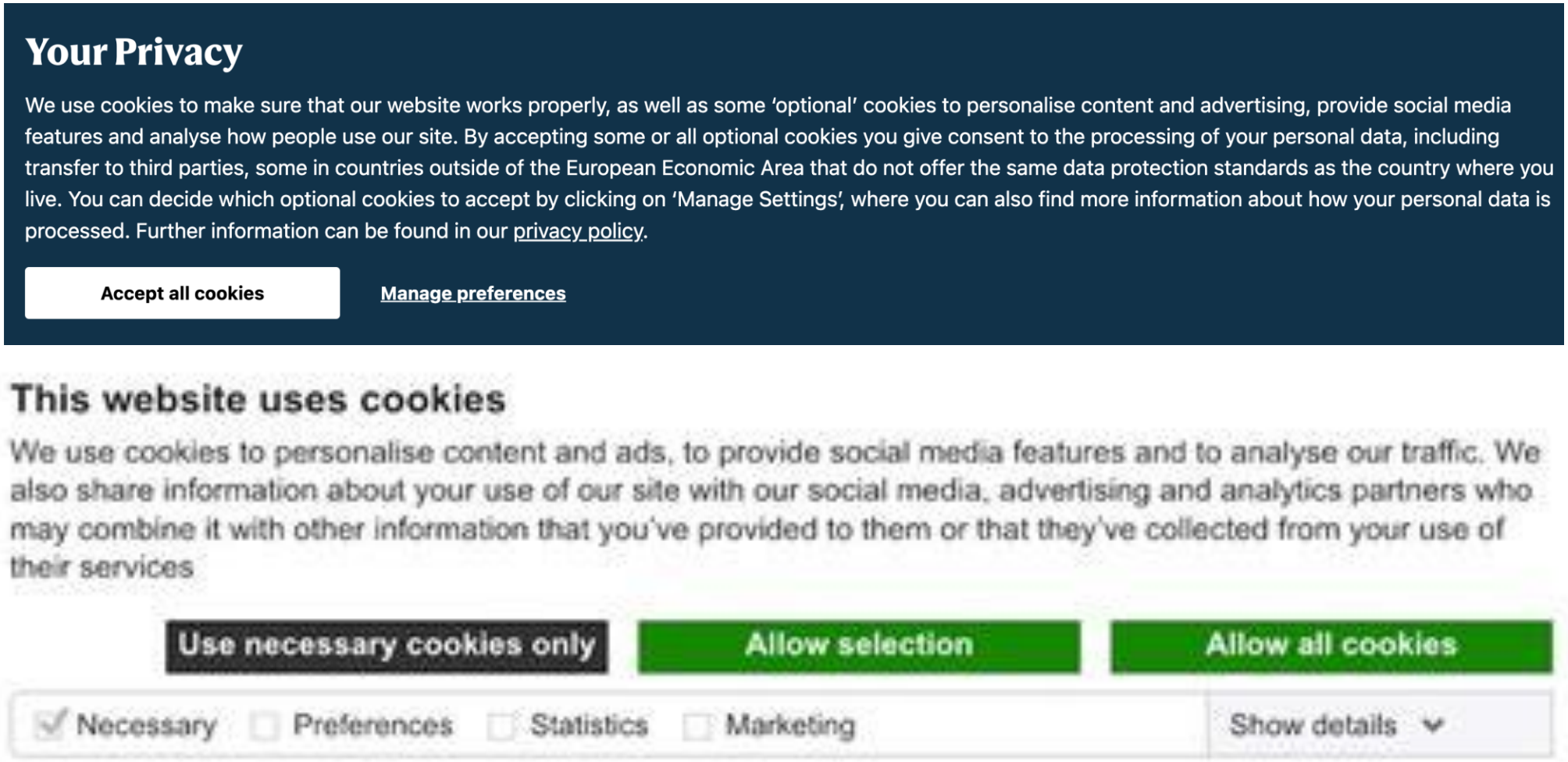
So far, HTTP mechanisms considered stateless

- Each request processed independently at the server
- The server maintains no memory about past client requests

However, state, i.e., memory, about the user at the server, is very useful!

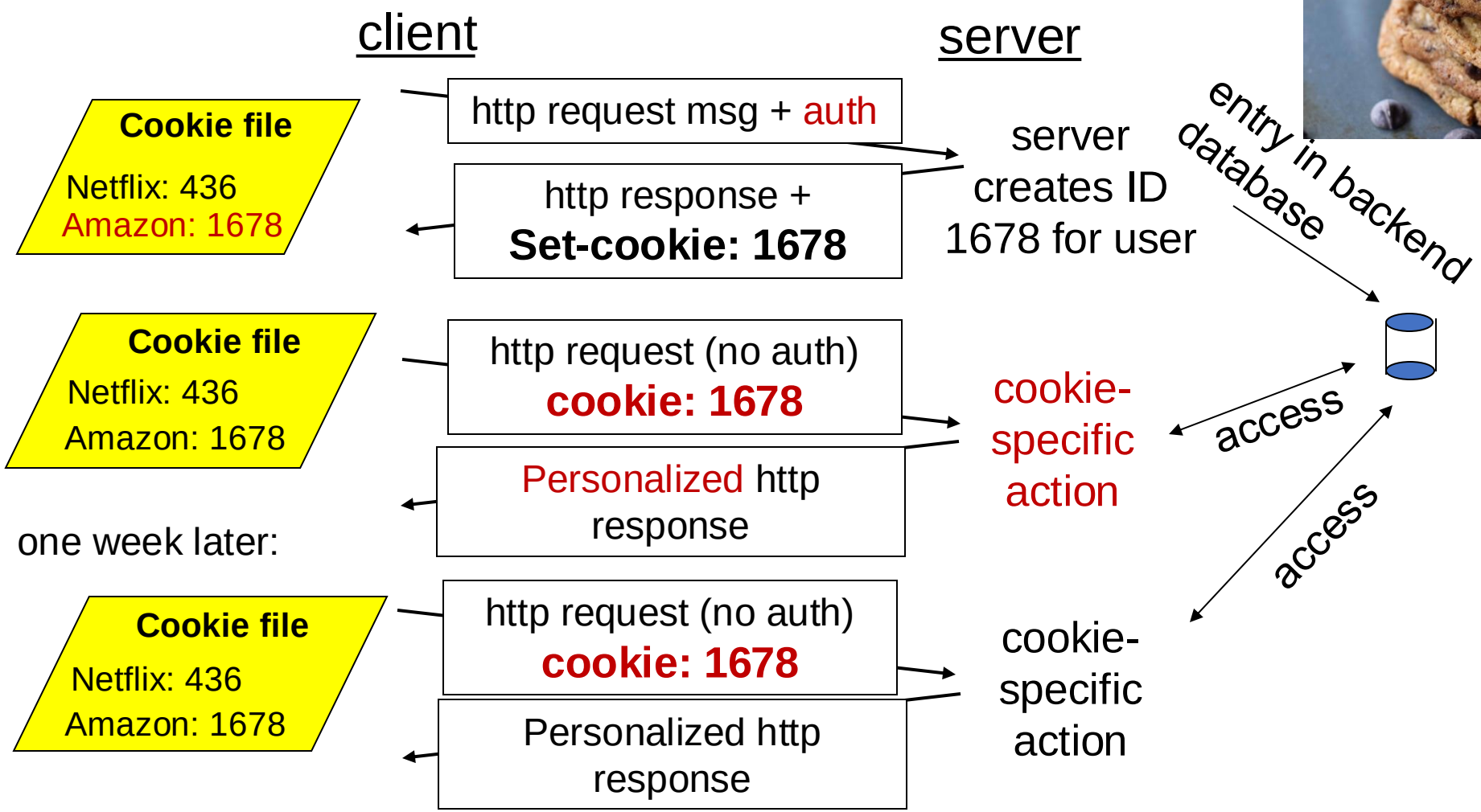
- User authentication (e.g., gmail)
- Shopping carts (e.g., Amazon)
- Video recommendations (e.g., Netflix)
- Any user session state in general

Familiar with these?



Cookies: Keeping user memory

Cookie is typically opaque to client.



How cookies work

Collaboration between client and server to track user state.

Four components:

1. cookie header line of HTTP response message
2. cookie header line in HTTP request message
3. cookie file kept on user endpoint, managed by user's browser
4. back-end database maps cookie to user data at Web endpoint

Cookies come with an expiration date (yet another HTTP header)

Cookies have many uses

- The good: Awesome user-facing functionality
  - Shopping carts, auth, … very challenging or impossible without it
- The bad: Unnecessary recording of your activities on the site
  - First-party cookies: performance statistics, user engagement, …
- The ugly: Tracking your activities across the Internet
  - Third-party cookies (played by ad and tracking networks) to track your activities across the Internet
  - personally identifiable information (PII)
  - Ad networks target users with ads; may sell this info
  - Scammers can target you too

PSA: Cookies and Privacy

- Disable and delete unnecessary cookies by default

- Suggested privacy-conscious browsers, websites, tools:

- DuckDuckGo (search)
- Brave (browser)
- AdBlock Plus (extension)
- ToR (distract targeting)
- … assuming it doesn't break the functions of the site

https://gdpr.eu/cookies/

Web Caching

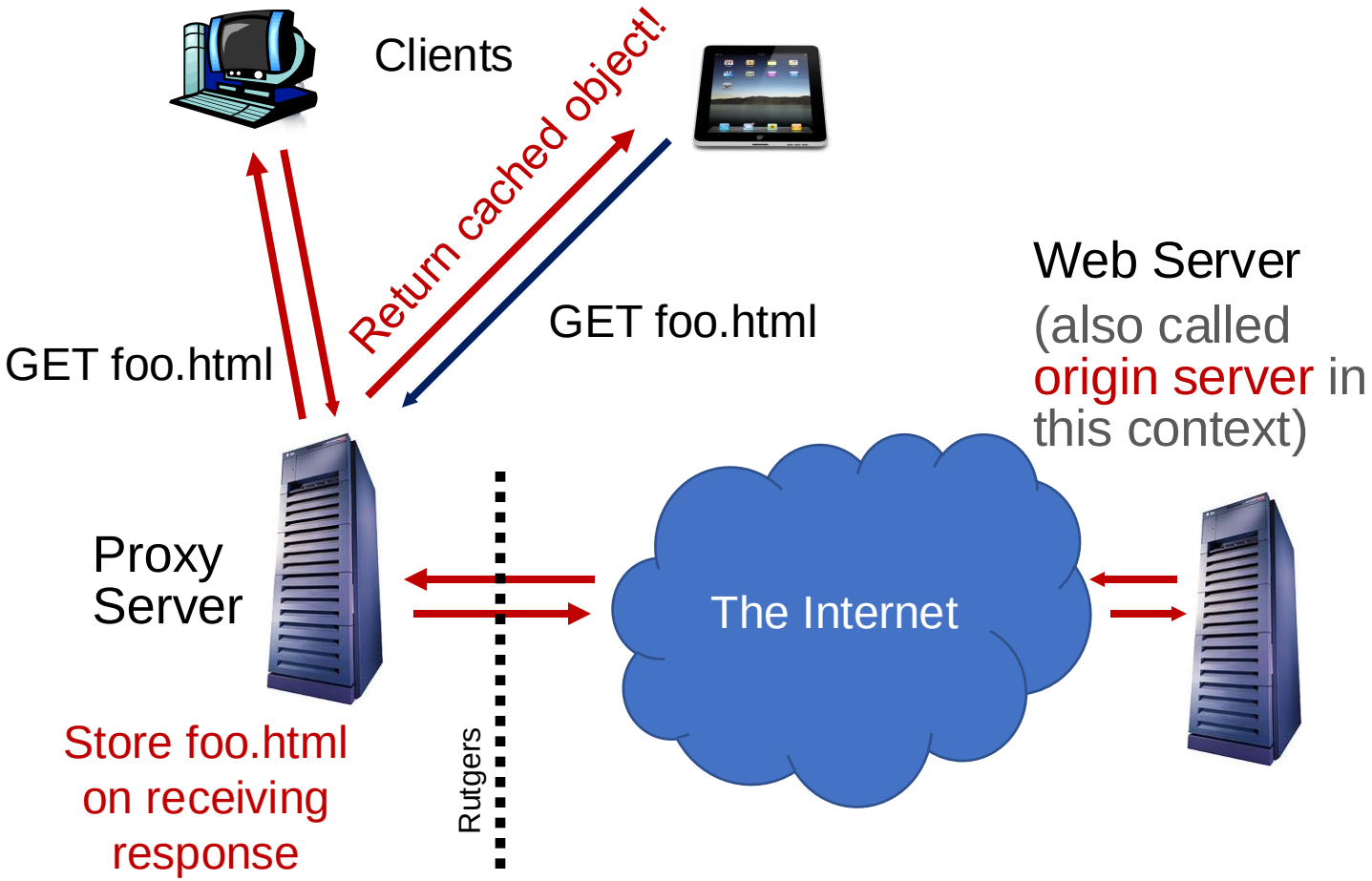
Web caches

Web caches: Machines that remember web responses for a network

Why cache web responses?

- Reduce response time for client requests
- Reduce traffic on an organization’s access link

Web caching using a proxy server

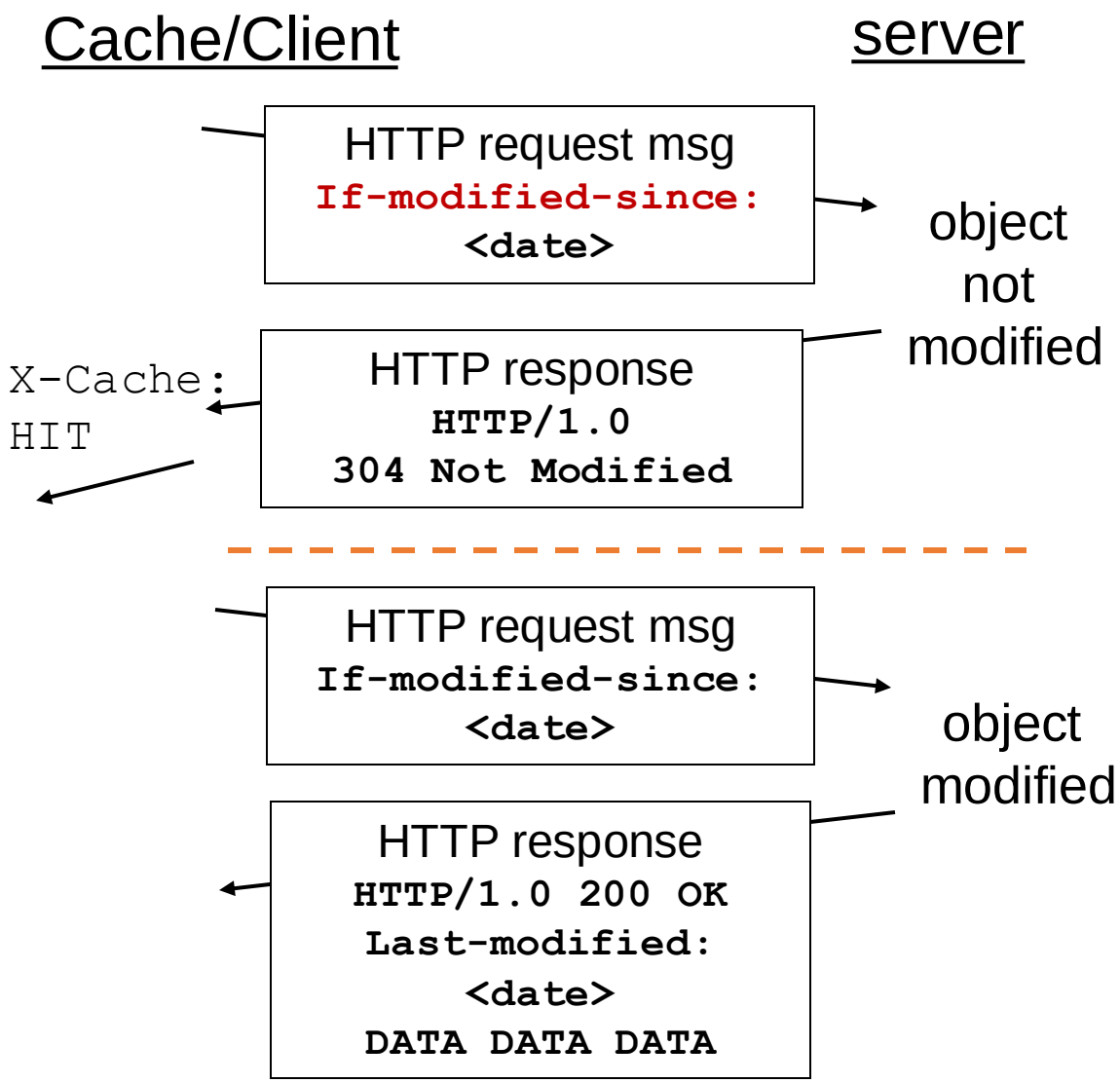


- You can configure a HTTP proxy on your laptop's network settings.
- If you do, your browser sends all HTTP requests to the proxy (cache).
- Hit: cache returns object
- Miss: obtain object from originating web server (origin server) and return to client
  - Also cache the object locally

Caching in the HTTP protocol

- Conditional GET
  guarantees cache content is up-to-date while still saves traffic and response time whenever possible

- Date in the cache's request is the last time the server provided in its response header Last-Modified



Content Distribution Networks (CDNs)

A global network of web caches

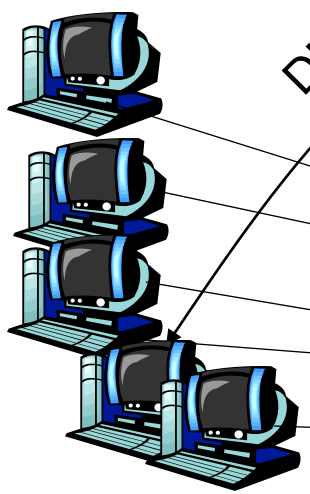
- Provisioned by ISPs and network operators
- Or content providers, like Netflix, Google, etc.

Uses (overlaps with uses of web caching in general)

- Reduce traffic on a network's Internet connection, e.g., Rutgers
- Improve response time for users: CDN nodes are closer to users than origin servers (servers holding original content)
- Reduce bandwidth requirements on the content provider
- Reduce cost to maintain origin servers

Without CDN

Clients distributed all over the world



DNS

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

Cluster of Rutgers CS origin servers (located in NJ, USA)

128.6.4.2

- Problems:
- Huge bandwidth requirements for Rutgers
- Large propagation delays to reach users

Where the CDN comes in

- Distribute content of the origin server over geographically distributed CDN servers
- But how will users get to these CDN servers?
- Use DNS!
  - DNS provides an additional layer of indirection
  - Instead of returning an IP address, return another DNS server (NS record)
    - Much like a response to any other iterative query
  - The second DNS server (run by the CDN) returns the IP address of the client
- The CDN runs its own DNS servers (CDN name servers)
  - Custom logic to send users to the “closest” CDN web server

With CDN

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	124.8.9.8 (NS record pointing to CDN name server)
www.google.com	74.125.225.243

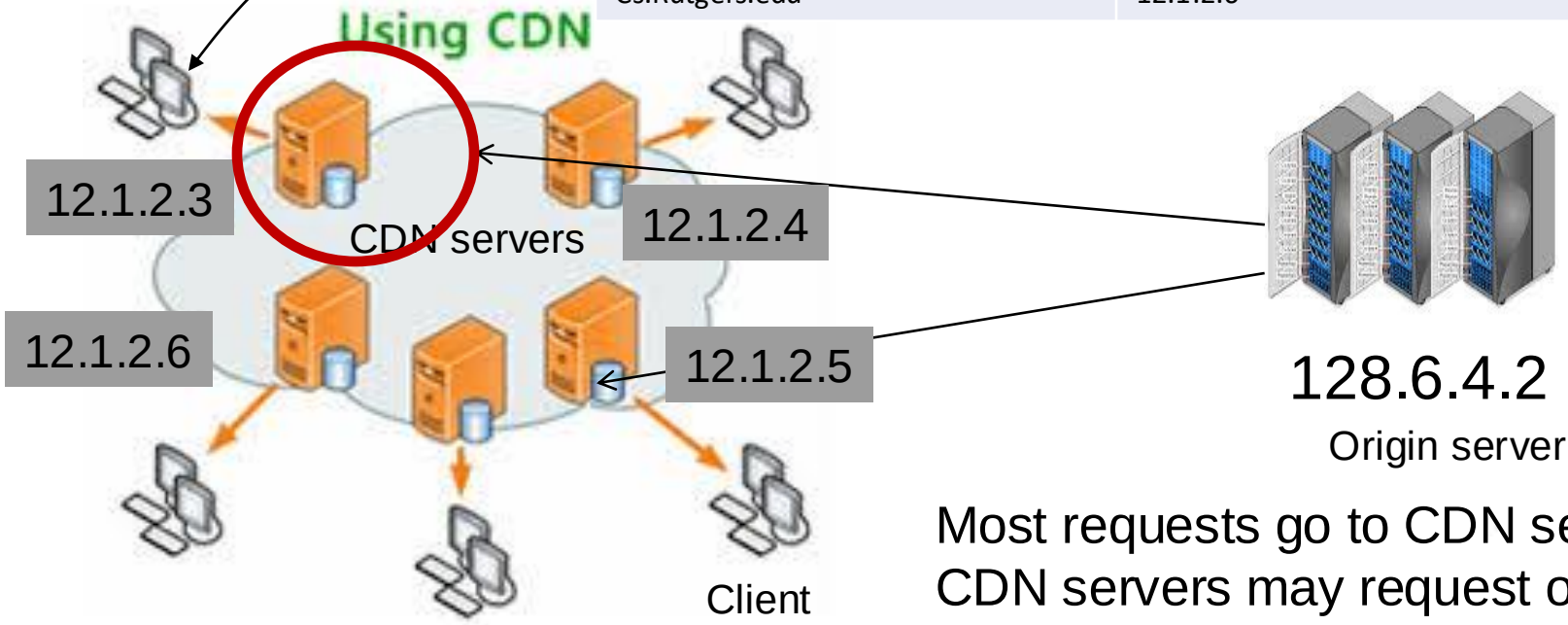
DNS reply

NS record delegates the choice of IP address to the CDN name server.

CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
Cs.Rutgers.edu	12.1.2.3
Cs.Rutgers.edu	12.1.2.4
Cs.Rutgers.edu	12.1.2.5
Cs.Rutgers.edu	12.1.2.6

Custom logic to map ONE domain name to one of many IP addresses!



Popular CDNs:
CloudFlare
Akamai
Level3
…

Most requests go to CDN servers (caches).
CDN servers may request object from origin
Few client requests go directly to origin server

Seeing a CDN in action

- `dig +trace freshtohome.com`

- `dig web.mit.edu` (or) `dig +trace web.mit.edu`

Summary of HTTP

- Request/response protocol
- ASCII-based human-readable message structures
- Enhanced stateful functionality using cookies
- Improve performance using caching and CDN
- Persistence and pipelining to improve performance
- Simple, highly-customizable protocol
  - Just add headers
- The protocol that is the basis of the web we enjoy today