

CN-Basic L15

HTTP Cacheing & Cookies

Dr. Ram P Rustagi
rprustagi@ksit.edu.in
<http://www.rprustagi.com>
<https://www.youtube.com/rprustagi>

Resources

- <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching>
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Caching>
-

HTTP Redirect

- **Content type**
 - accessing `hello.html` vs `hello.txt`
 - `Content-Type: text/html`
 - `Content-Type: text/plain`
- **Location (i.e. URL Redirect)**
 - `workshops.rprustagi.com` →
 `rprustagi.com/workshops/workshops.html`
- **Access `google.com`**
 - user browser or (`wget -d <URL>`)

Web Cache and Proxy

- A Good resource on Web Cache
 - https://www.mnot.net/cache_docs/
 - <https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching?hl=en>
- Resources and acknowledgements
 - http://wps.pearsoned.com/ecs_kurose_compnetw_6/216/55463/14198700.cw/index.html

Conditional GET

- *Goal*: don't send object if cache has up-to-date cached version
 - No object transmission delay
 - Lower link utilization
- *cache*: specify date of cached copy in HTTP request

If-modified-since: <date>

- *Server*: response contains no object if cached copy is up-to-date:

HTTP/1.0 304 Not Modified

Conditional GET

- **Goal:** don't send object if cache has up-to-date cached version

- no transmission delay
- lower link utilization

- **cache:** specify date of cached copy in HTTP request

- If-modified-since:
<date>

- **server:** response contains no object if cached copy is up-to-date:

- HTTP/1.0 304 Not Modified

client



server



HTTP request msg
If-modified-since:
<date>

object
not
modified
before
<date>

HTTP response
HTTP/1.0 304 Not Modified

HTTP request msg
If-modified-since:
<date>

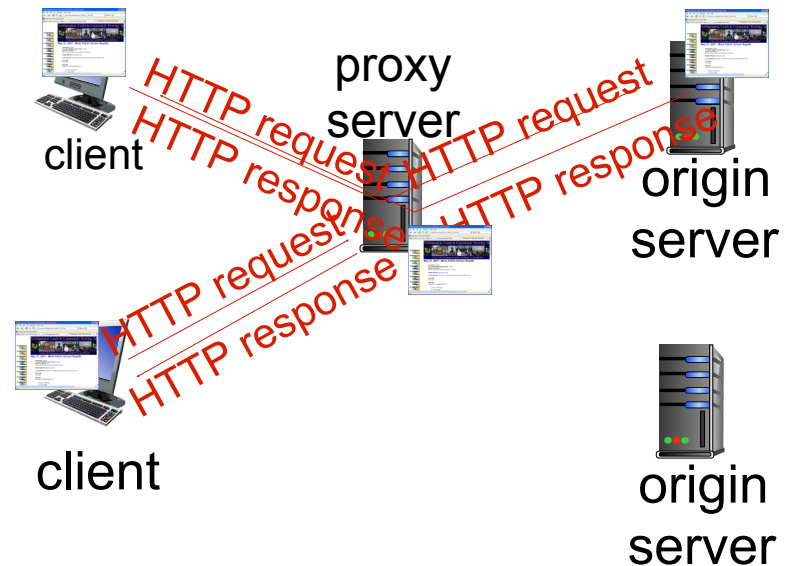
object
modified
after
<date>

HTTP response
HTTP/1.0 200 OK
<data>

Web caches (proxy server)

- user sets browser: Web accesses via proxy server
- browser sends all HTTP requests to cache
- cache requests object from origin server, then returns object to client
 - object in cache: cache returns object

goal: satisfy client request without involving origin server

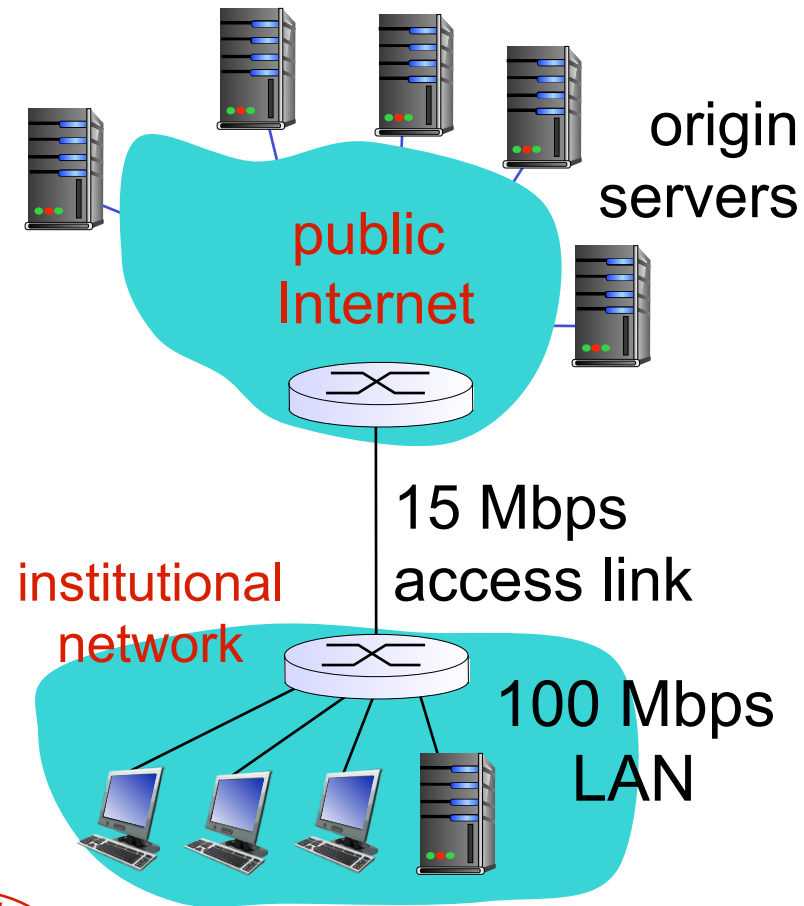


More about Web caching

- Proxy server acts as both client and server
 - server for original requesting client
 - client to origin server
- typically proxy server is installed by ISP (university, company, residential ISP)
- *why Web caching?*
 - reduce response time for client request
 - reduce traffic on an institution's access link
 - Internet dense with caches: enables “poor” content providers to effectively deliver content (so too does P2P file sharing)

Caching example:

- **assumptions:**
- avg object size: 1M bits
- avg request rate from browsers to origin servers: 15/sec
- RTT from institutional router to any origin server: 2 sec (**internet delay**)
- access link rate: 15 Mbps
- **consequences:**
- LAN Traffic intensity =
 - $15 * 1\text{Mb} / 100\text{Mbps} = 15\%$
- Access link traffic intensity
 - $15 * 1\text{Mb} / 15\text{Mbps} = 100\%$
- total delay = Internet delay + access delay + LAN delay
- = 2s + minutes + μs **problem!**



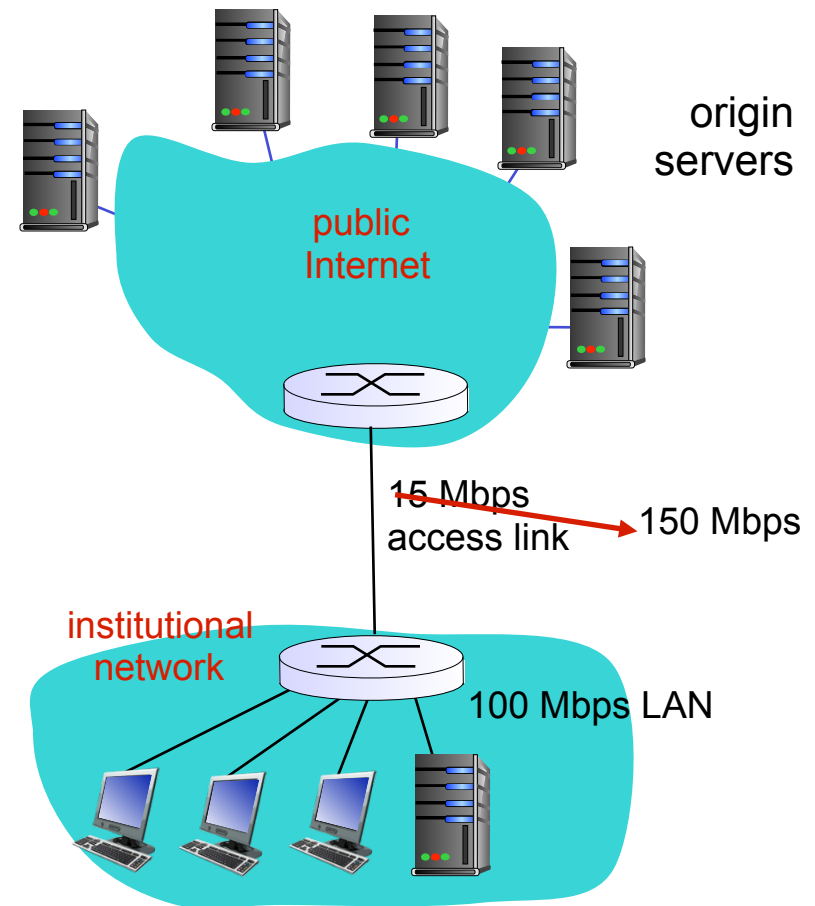
Caching example: fatter access link

assumptions:

- avg object size: 1M bits
- avg request rate from browsers to origin servers: 15/sec
- RTT from institutional router to any origin server: 2 sec
- access link rate: 15 Mbps

consequences:

- LAN utilization: 15% → 10%
- access link utilization = 100% → msecs
- total delay = Internet delay + access delay + LAN delay
- = 2 sec + minutes + μ s



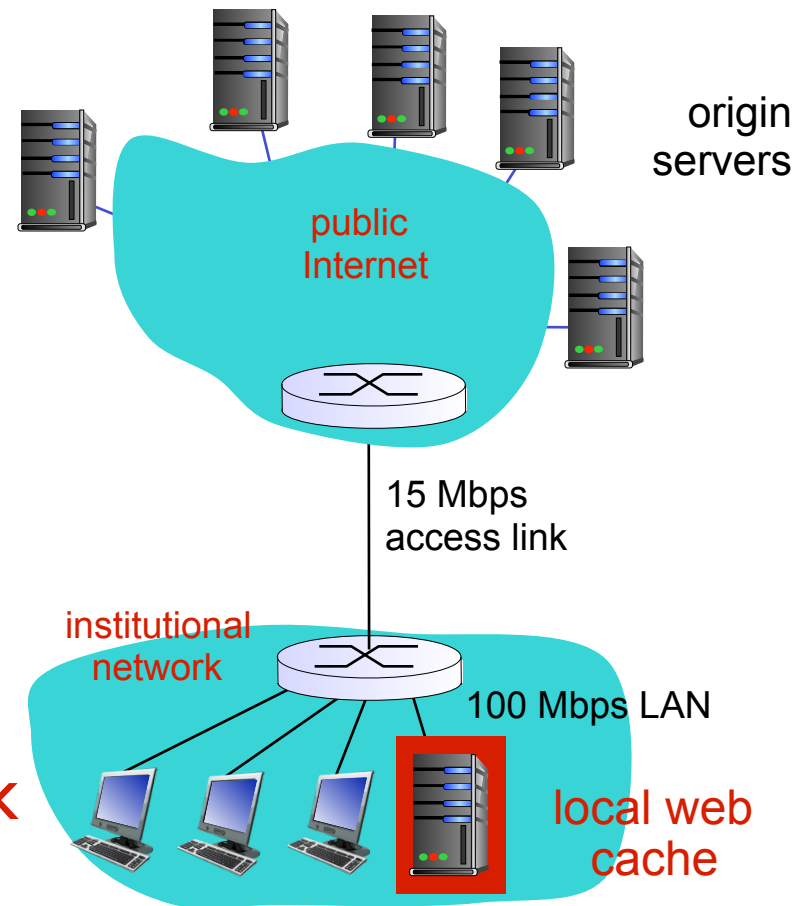
Cost: increased access link speed (not cheap!)

Caching example: install local cache

- *consequences:*
- LAN utilization: 15%
- access link utilization = 100%
- total delay =
Internet delay ?
+ access delay ?
+ LAN delay
= 2 sec + minutes + μ s

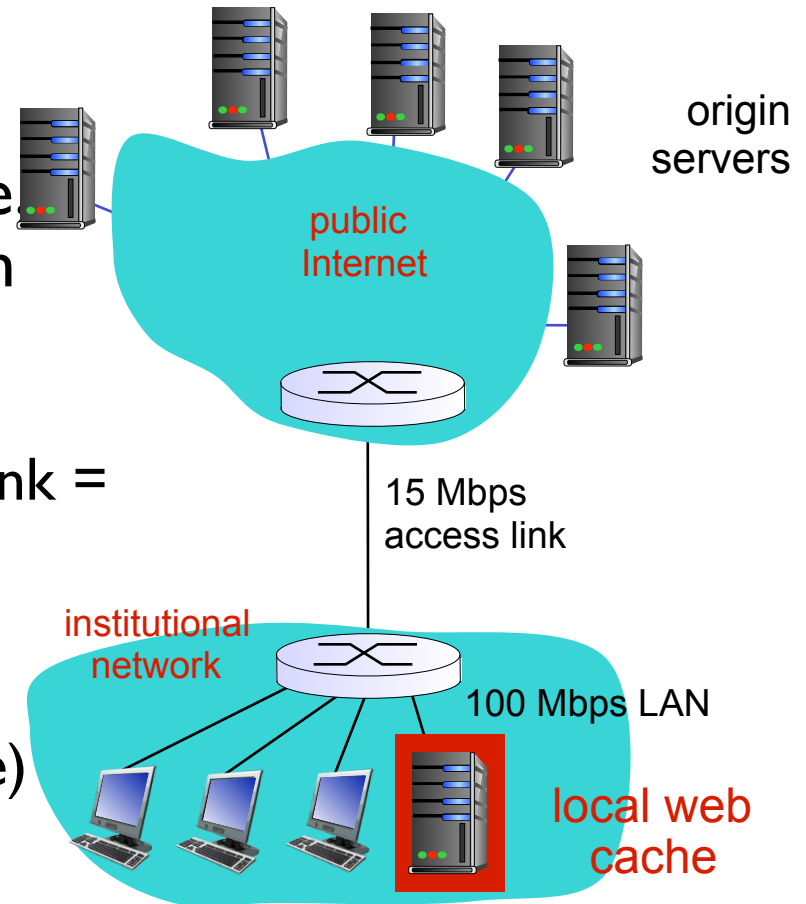
How to compute link utilization, delay?

Cost: web cache (cheap!)



Caching example: install local cache

- *Calculating access link utilization, delay with cache:*
- suppose cache hit rate is 0.4
 - 40% requests satisfied at cache
 - 60% requests satisfied at origin
- access link utilization:
- 60% of requests use access link
- data rate to browsers over access link = $0.6 * 15 \text{ Mbps} = 9 \text{ Mbps}$
- utilization = $9 / 15 = .6$
- total delay
 - = $0.6 * (\text{delay from origin servers})$
 - + $0.4 * (\text{delay when satisfied at cache})$
 - = $0.6 (2.01) + 0.4 (\sim \text{msecs})$
 - = $\sim 1.2 \text{ secs}$
- less than with 150 Mbps link (and cheaper too!)



Types of Cache

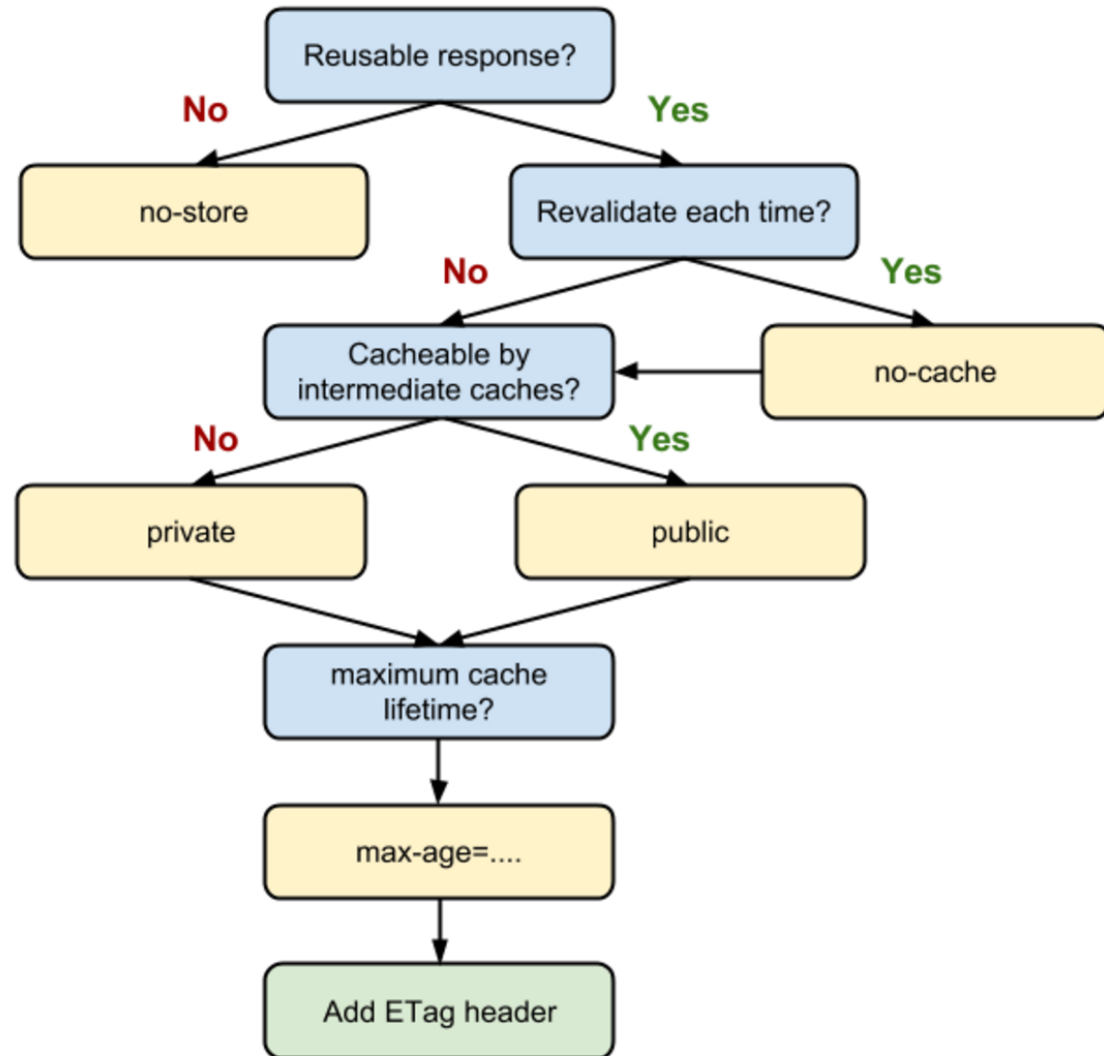
- private cache: Exclusive browser cache
- public cache
 - proxy cache
 - reduces bandwidth requirements
 - reduces delays
 - gateway cache : (aka reverse proxy cache)
 - deployed by web masters for scalability
 - examples: memcached, varnish

Cache Headers

- Last-modified
- If-modified-since / If-unmodified-since
- Etag
- If-none-match
- Vary
- Age
- Pragma directive
- Date
- Expires
- Cache-Control

Cache-control

src: [https://
developers.google.com/web/
fundamentals/performance/
optimizing-content-efficiency/
http-caching?hl=en](https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching?hl=en)



User-server state: cookies

many Web sites use cookies

four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in next HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

- *example:*

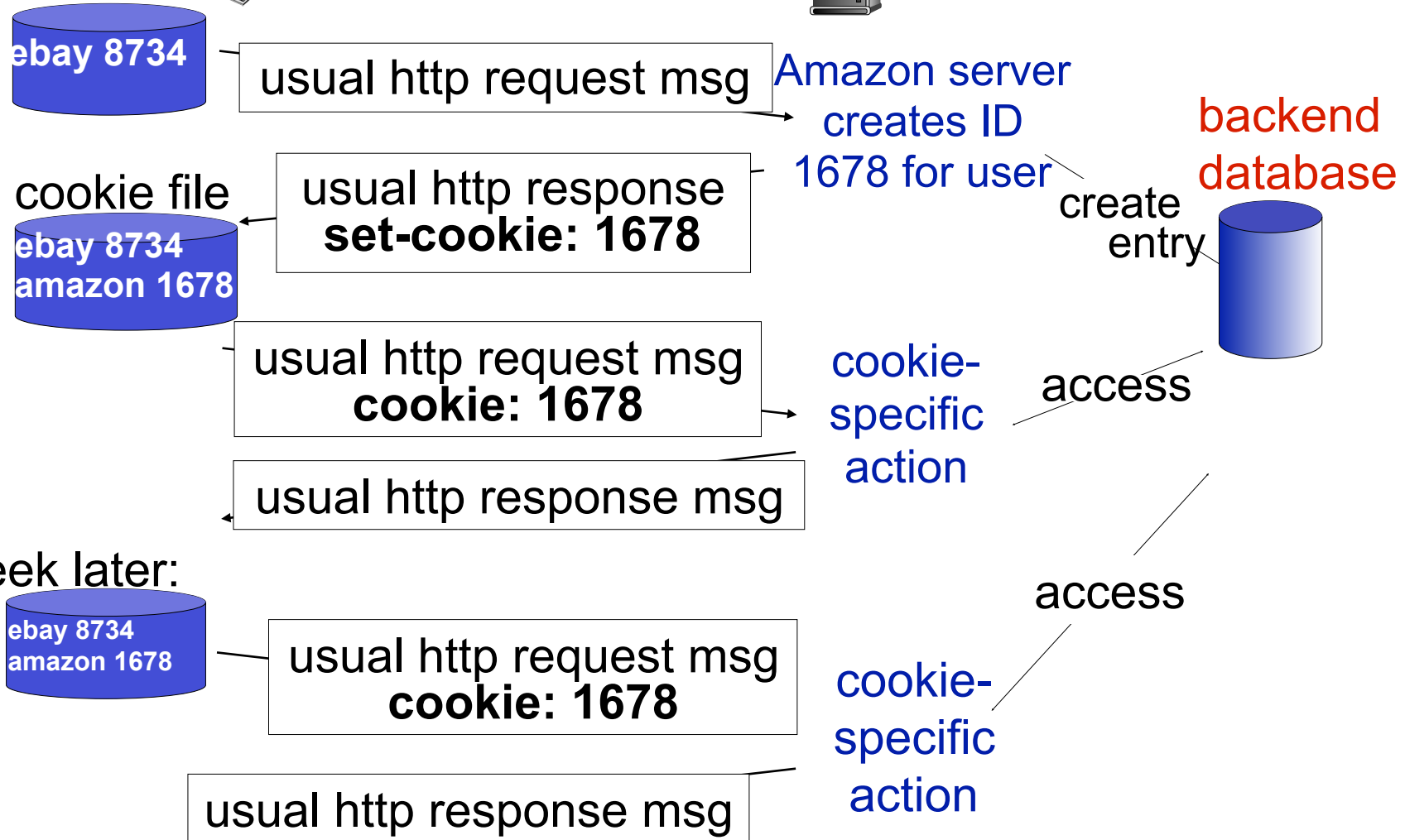
- Susan always access Internet from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - unique ID
 - entry in backend database for ID

Cookies: keeping “state” (cont.)

client



server



Cookies (continued)

- *what cookies can be used for:*

- authorization
- shopping carts
- recommendations
- user session state
 - (Web e-mail)

aside

- *cookies and privacy:*
- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

- *how to keep “state”:*

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

Cookie Settings

- <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Set-Cookie>
- Domain
 - domain and sub domain
- Path
 - covers sub paths
- Secure
 - only with HTTPS
- HttpOnly
 - Only web page access, not with javascript
- Max-Age, Expires
- SameSite

Cookies (continued)

- **Examples**

rprustagi.com/workshops/web/cookie-expiry.php

rprustagi.com/workshops/web/cookie-path.php

rprustagi.com/workshops/web/cookie-secure.php

- **How browser stores it**

- **Firefox:** preferences → privacy → Cookies

- **Remove the cookie and re-access the page**

- **Chrome:** Settings → Advanced → privacy & Security → Site Settings → Cookies → See all cookies and site data

Summary

- HTTP Redirect
- HTTP Cookies
- HTTP Cacheing