Cookies

HTTP – Stateless Protocol – **server forgets about each client as soon as it delivers response …**

- **stateless behavior is issue when:**
  - **server wants to have accurate count of site visitors**
  - **server wants to restrict user access, etc.**
  - **server wants to personalize pages for each client, or remember selections they made**

Cookie Technology – **allows sites to keep track of users**

(1) when a server identifies a new user, it adds Set-Cookie header to its response, containing an identifier for that user

(2) the client is expected to store the info from the Set-Cookie header on its disk, and send this info back to the server by means of Cookie header in all subsequent requests made to the same server / Web domain …

A cookie is a short piece of data, not code, which is sent from a web server to a web browser when that browser visits the server's site. The cookie is stored on the user's machine, but it is not an executable program and cannot directly harm the machine.

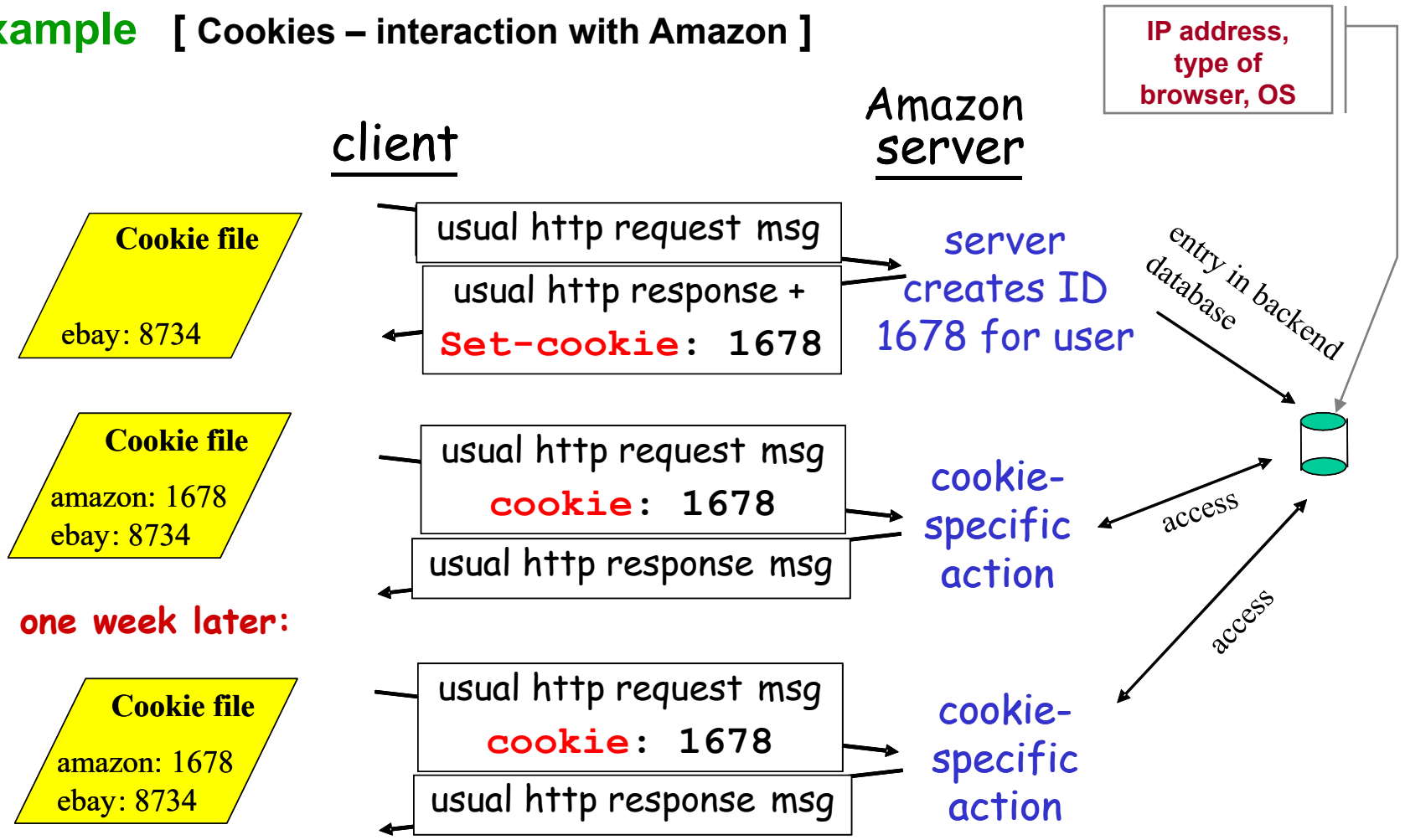
Cookies (cont.)

Set–Cookies & Cookies HTTP Headers – the two headers are used to set and exchange cookies between client and server machines

- **Set-Cookies** is a '<u>Response Header</u>' and is used by the server to set-up a new cookie on the client
- **Cookies** is a '<u>Request Header</u>' and is used by the client to return the cookie(s) to the server

Cookies (cont.)

Example **[Cookies – interaction with Amazon]**



Not all sites use cookies, but major portals (e.g. Yahoo), e-commerce (e.g. Amazon), and advertising sites make extensive use of cookies.

Cookies (cont.)

Example **[HTTP response and request containing cookies …]**

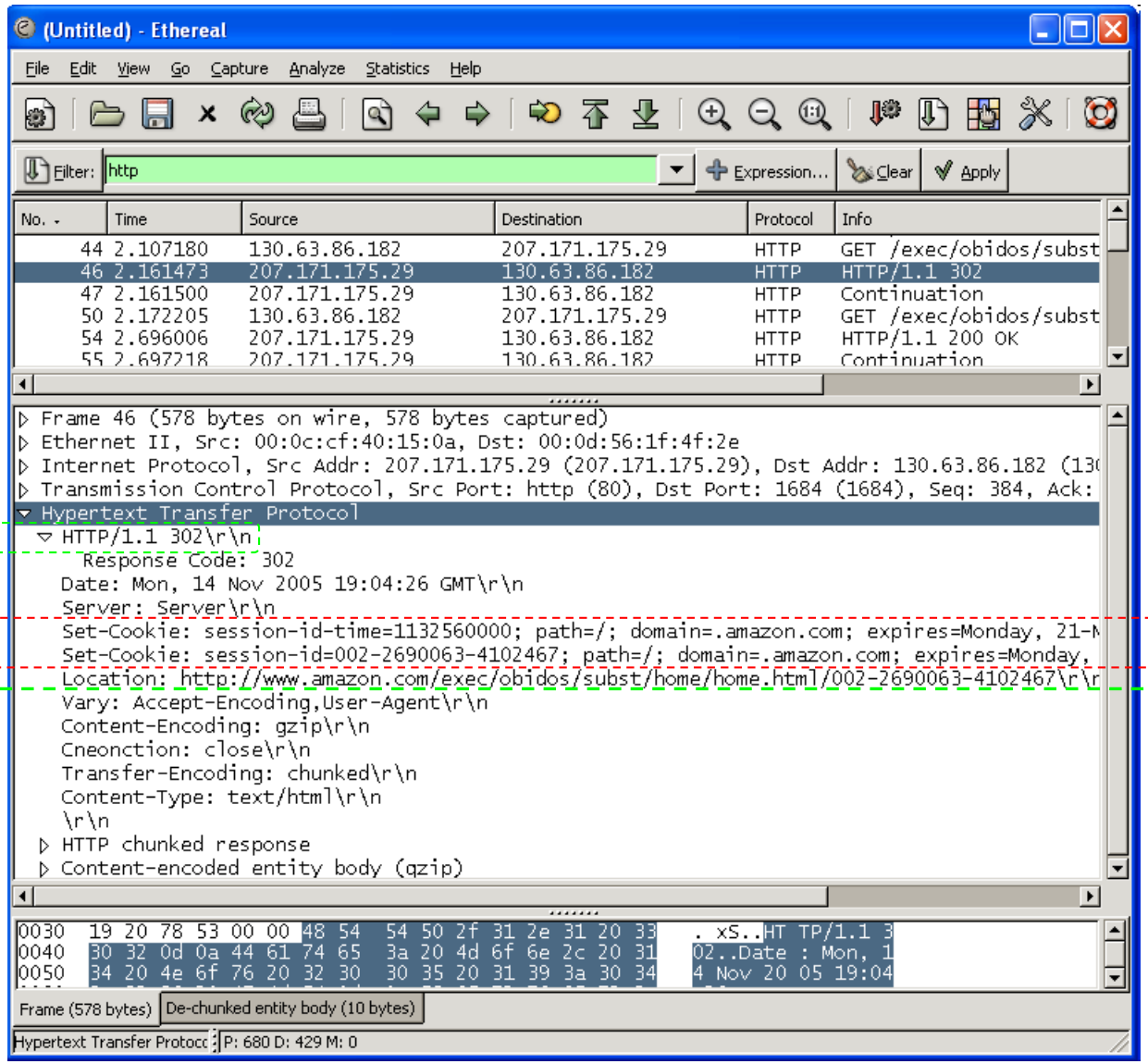
```
1  HTTP/2.0 200 OK
2  Content-type: text/html
3  Set-Cookie: yummy_cookie=choco
4  Set-Cookie: tasty_cookie=strawberry
5
6  [page content]
```

```
1  GET /sample_page.html HTTP/2.0
2  Host: www.example.org
3  Cookie: yummy_cookie=choco; tasty_cookie=strawberry
```

https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

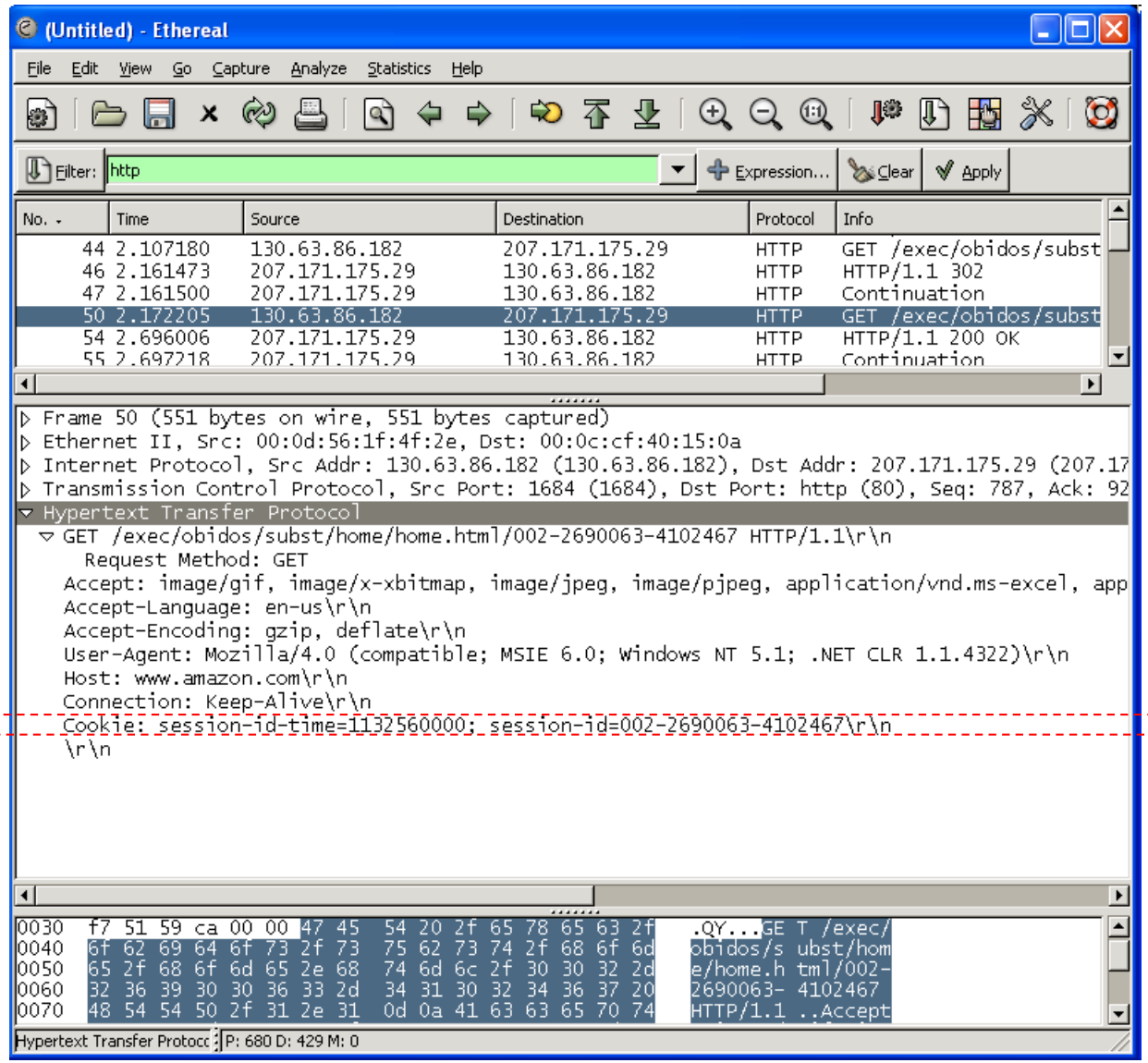
Cookies (cont.)

Example **[Cookies – interaction with www.amazon.com]**



Cookies (cont.)

Example **[Cookies – interaction with www.amazon.com (cont.)]**



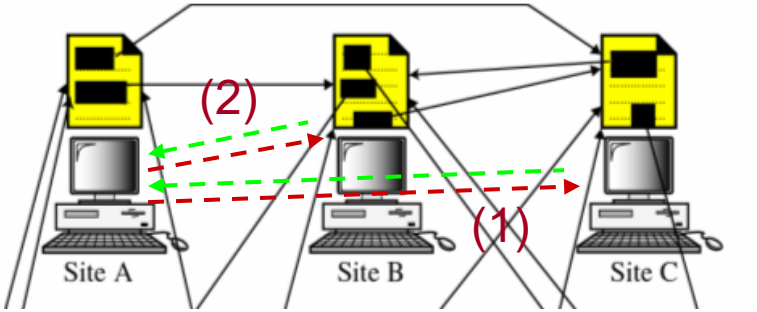
Cookies (cont.)

Issues with Cookies – although most of the time cookies are used for useful and benign purposes, there are ways to abuse them:

(1) **Undesirable Cookies**: any server can set a cookie for any reason – since some Web browsers do not inform the user when a cookie is being set, he may not even be aware that this is happening

(2) **Third-Party or Unintentional Cookies**: a cookie may be set by any server to which a request is sent, whether the user realizes it or not

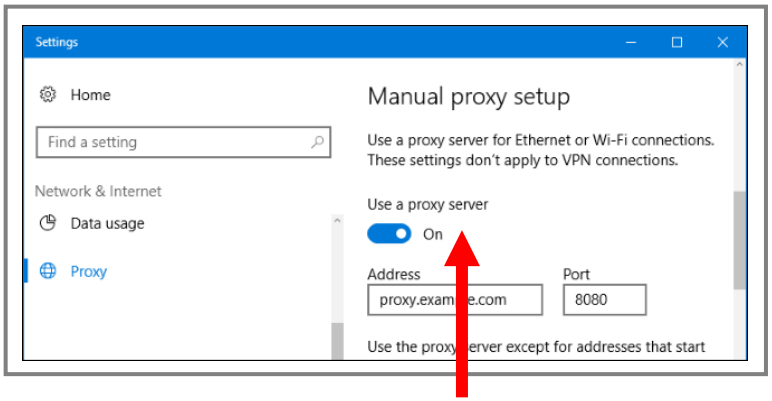
- by retrieving www.myfavoritesite.com/index.html that contains a reference to a tiny image at www.bigbrotherishere.com, the second site can set a cookie on your machine …



indirect HTML referencing

Web Caching

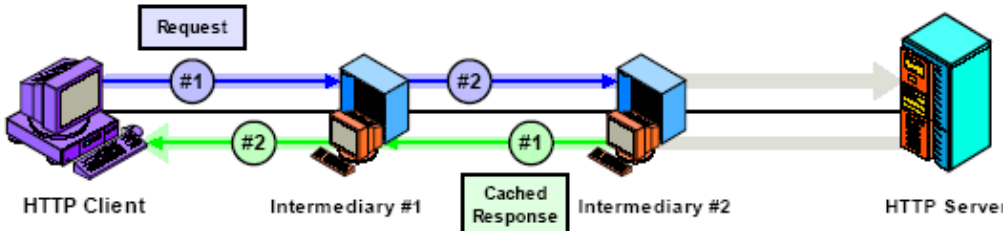
Web Cache / Proxy Server – **an intermediary entity that satisfies HTTP requests on the behalf of an origin Web server**



- **users browser must be configured so that all requests are first directed to its LAN's Web cache**
- **Web cache checks if it has a copy of a requested object stored locally**
  - **if it does, it sends cached object back to client**
  - **if it does NOT, it requests object from the origin server, and then returns it to the client**
- **advantages of caching:**

  (1) reduce response time for clients' requests
  (2) reduce traffic on an institution's access link
- **disadvantages of caching**

  slower performance if object is not cached – extra layer is added



time it takes for a response to arrive from the server to the first router

Web Caching (cont.)

Example [caching]

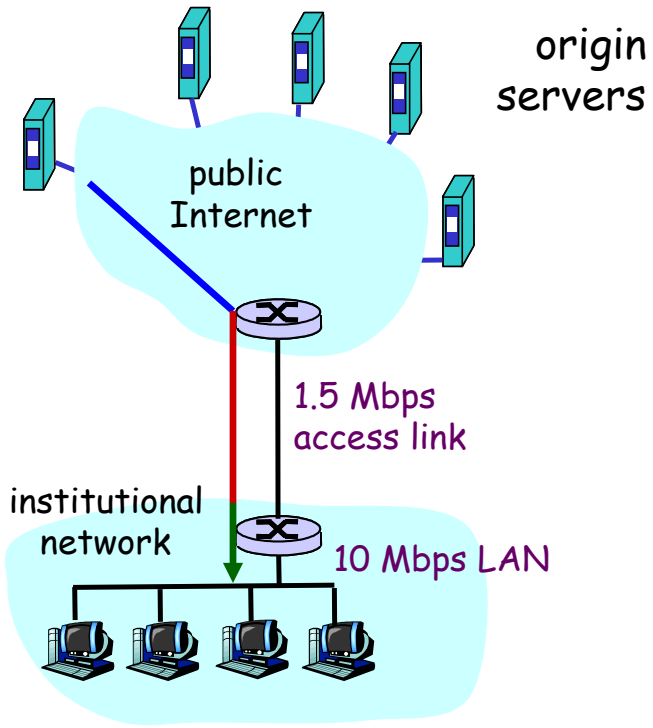
Assumptions

- average object size = 100,000 bits
- avg. request rate from institution's browsers to origin servers = 15/sec

Consequences

demanded bps available bps

- utilization on LAN =
  = (15 req/sec * 0.1 Mbit/req) / 10 Mbit/sec =
  = 0.15 = 15%
- utilization on access link =
  = (15 req/sec * 0.1 Mbit/req) / 1.5 Mbit/sec
  = 1 = 100%
- total delay =
  = Internet delay + access delay + LAN delay =
  = 2 sec + minutes + milliseconds = several min



Possible solution: increase access rate – costly!!!

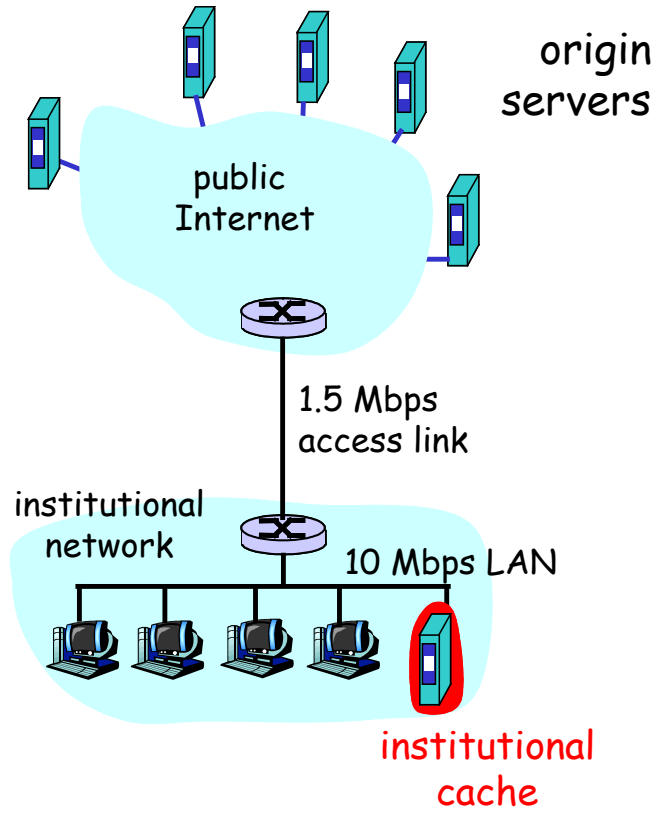
Web Caching (cont.)

Install cache

- suppose hit rate (fraction of requests satisfied by cache) is 0.4 (typically 0.2 – 0.7)

Consequence

- 40% requests will be satisfied almost immediately
- 60% requests satisfied by origin server
- utilization of access link reduced to 60%, resulting in negligible access delay (say 10 msec)
- total avegerage delay =
  0.4*LAN access delay + 0.6*WAN access delay <
  < 0.4*0.01 sec + 0.6*(2 + 0.01 + 0.01) sec ≈
  < several seconds

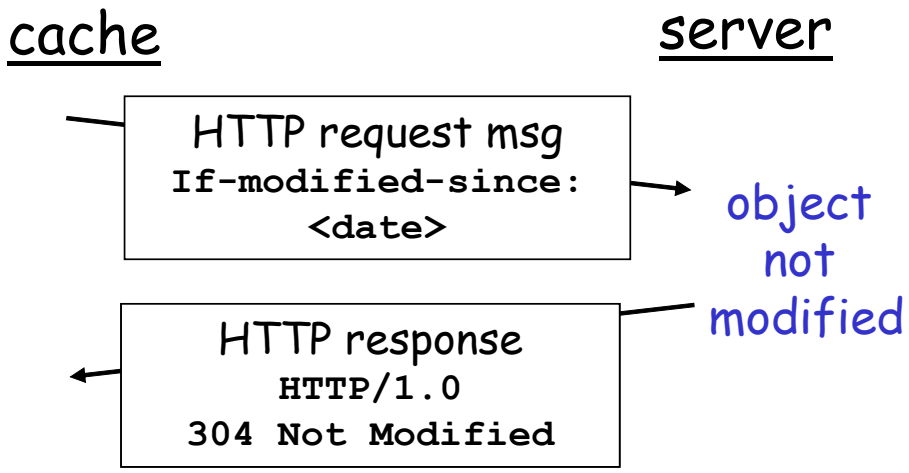


For this solution, the institution hast to (purchase and) install a Web cache. Many caches use public-domain software that runs on inexpensive PCs.

Web Caching (cont.)

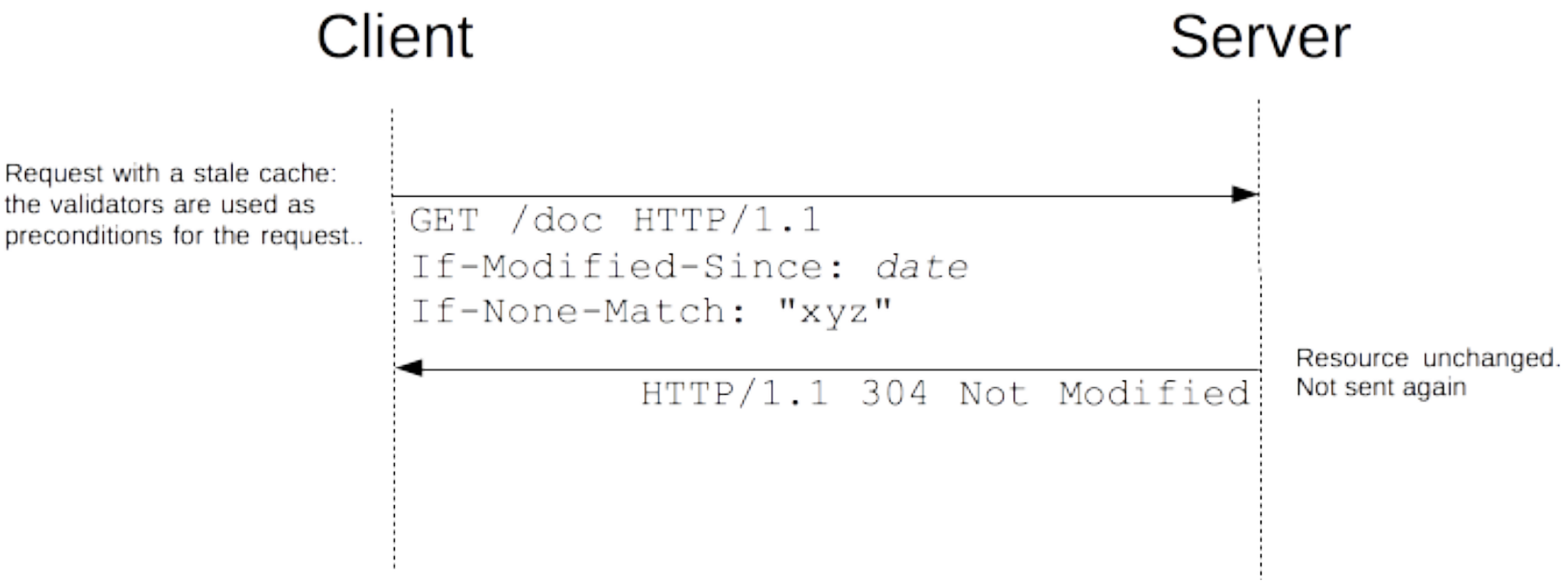
Web Cache Challenge – **an object residing in the cache might be stale, i.e. the object may have been modified since the copy was cached**

- **goal: want to keep cache up-to-date, but don't send object if cache latest version**
- **solution: conditional GET – requires that GET method contains If-Modified-Since header line**
  - **server will include requested object in response only if object has been modified since specified date**
  - **this saves bandwidth and reduces user-perceived response time, especially if object is large**



Web Caching (cont.)

Example **[If-Modified-Since HTTP Header]**



https://developer.mozilla.org/en-US/docs/Web/HTTP/Conditional_requests