Cookies Overview and HTTP Proxies

What is a Cookie?

- Small piece of data generated by a web server, stored on the client’s hard drive.
- Serves as an add-on to the HTTP specification (remember, HTTP by itself is stateless.)
- Still somewhat controversial, as it enables web sites to track web users and their habits…

Why use Cookies?

- Tracking unique visitors
- Creating personalized web sites
- Shopping Carts
- Tracking users across your site:
  - e.g. do users that visit your sports news page also visit your sports store?

Example Cookie Use

- Website wants to track the number of unique visitors who access its site.
- If the website checks the HTTP Server logs, it can determine the number of “hits”, but cannot determine the number of unique visitors.
- That’s because HTTP is stateless. It retains no memory regarding individual users.
- Cookies provide a mechanism to solve this problem.

Tracking Unique Visitors

- Step 1: Person A requests the website.
- Step 2: Web Server generates a new unique ID.
- Step 3: Server returns home page plus a cookie set to the unique ID.
- Step 4: Each time Person A returns to the website, the browser automatically sends the cookie along with the GET request.

Cookie Notes

- Created in 1994 for Netscape 1.1
- Cookies cannot be larger than 4K
- No domain (e.g. netscape.com, microsoft.com) can have more than 20 cookies.
- Cookies stay on your machine until:
  - they automatically expire
  - they are explicitly deleted
- Cookies work the same on all browsers.

Cookie Standards

- Version 0 (Netscape):
  - The original cookie specification
  - Implemented by all browsers and servers
  - We will focus on this Version
- Version 1
  - A proposed standard of the Internet Engineering Task Force (IETF)
  - Not very widely used (hence, we will stick to Version 0.)

Cookie Anatomy

- Version 0 specifies six cookie parts:
  - Name
  - Value
  - Domain
  - Path
  - Expires
  - Secure

Cookie Parts: Name/Value

- Name
  - Name of your cookie (Required)
  - Cannot contain white spaces, semicolons or commas.
- Value
  - Value of your cookie (Required)
  - Cannot contain white spaces, semicolons or commas.

Cookie Parts: Domain

- Only pages from the domain which created a cookie are allowed to read the cookie.
- For example, amazon.com cannot read yahoo.com’s cookies (imagine the security flaws if this were otherwise!)
- By default, the domain is set to the full domain of the web server that served the web page.
  - For example, myserver.mydomain.com would automatically set the domain to .myserver.mydomain.com

Cookie Parts: Domain

- Note that domains are always prepended with a dot.
  - This is a security precaution: all domains must have at least two periods.
- You can however, set a higher level domain
  - For example, myserver.mydomain.com can set the domain to .mydomain.com. This way hisserver.mydomain.com and herserver.mydomain.com can all access the same cookies.
- No matter what, you cannot set a domain other than your own.

Cookie Parts: Path

- Restricts cookie usage within the site.
- By default, the path is set to the path of the page that created the cookie.
- Example: user requests page from mymall.com/storea. By default, cookie will only be returned to pages for or under /storea.
- If you specify the path to / the cookie will be returned to all pages (a common practice.)

Cookie Parts: Expires

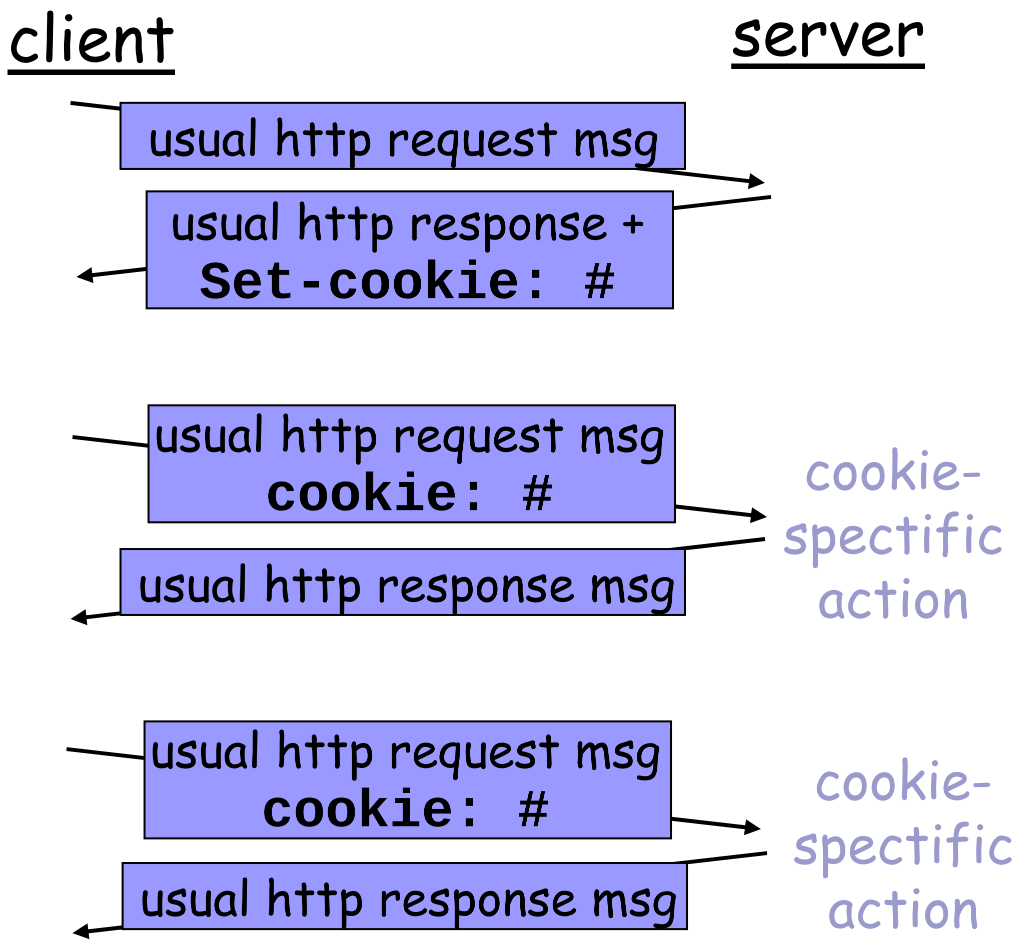
- Specifies when the cookie will expire.
- Specified in Greenwich Mean Time (GMT):
  - Wdy DD-Mon-YYYY HH:MM:SS GMT
- If you leave this value blank, browser will delete the cookie when the user exits the browser.
  - This is known as a *session cookies*, as opposed to a *persistent cookie*.

Cookie Parts: Secure

- The secure flag is designed to encrypt cookies while in transit.
- A secure cookie will only be sent over a secure connection (such as SSL.)
- In other words, if a cookie is set to secure, and you only connect via a non-secure connection, the cookie will <u>not</u> be sent.

User-server interaction: cookies

- server sends "cookie" to client in response msg
  `Set-cookie: 1678453`
- client stores & presents cookie in later requests
  `cookie: 1678453`
- server matches presented-cookie with server-stored info
  - authentication
  - remembering user preferences, previous choices



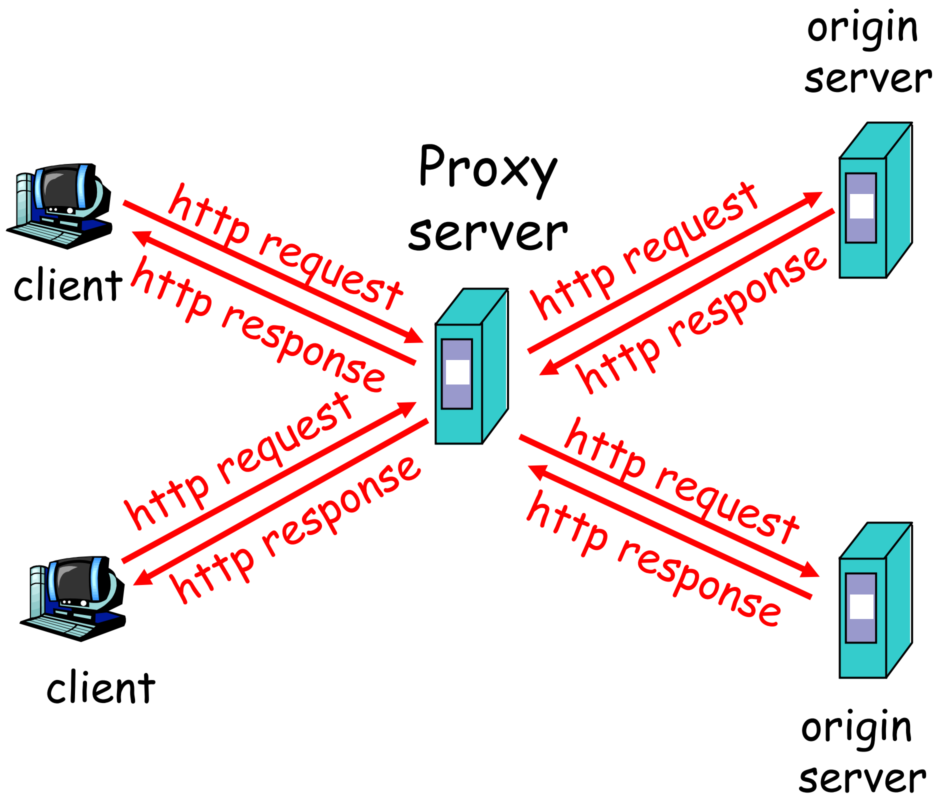
Cookie example

```
telnet www.google.com 80

Trying 216.239.33.99...
Connected to www.google.com.
Escape character is '^]'.

GET /index.html HTTP/1.0

HTTP/1.0 200 OK
Date: Wed, 10 Sep 2003 08:58:55 GMT
Set-Cookie:
    PREF=ID=43bd8b0f34818b58:TM=1063184203:LM=1063184203:S
    =DDqPgTb56Za88O2y; expires=Sun, 17-Jan-2038 19:14:07 GMT;
    path=/; domain=.google.com
.
.
```

Web Caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via web cache
- client sends all http requests to web cache
  - if object at web cache, web cache immediately returns object in http response
  - else requests object from origin server, then returns http response to client



More about Web caching

- Cache acts as both client and server
- Cache can do up-to-date check using `If-modified-since` HTTP header
    - Issue: should cache take risk and deliver cached object without checking?
    - Heuristics are used.
- Typically cache 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

Note: Meta tags and http-equiv

- HTTP servers use the property name specified by the http-equiv attribute to create an [RFC822]-style header in the HTTP response.

- The following sample META declaration:

```
<META http-equiv="Expires" content="Tue, 20 Aug 1996 14:25:27 GMT">
```

will result in the HTTP header:

```
Expires: Tue, 20 Aug 1996 14:25:27 GMT
```

This can be used by caches to determine when to fetch a fresh copy of the associated document.

References

- V22.0480-001, Sana` Odeh , Computer Science Department, New York University.
- Representation and Management of Data on the Internet (67633), Yehoshua Sagiv, The Hebrew University - Institute of Computer Science.
- Java Network Programming and Distributed Computing, Reilly & Reilly.
- *Computer Networking: A Top-Down Approach Featuring the Internet,* Kurose & Rose, Pearson Addison Wesley.