Google Adsense

- business model: sells adspace (on its and your webpages) to advertisers
- launched ~ 2003
- collects revenue from advertisers
  - $9.7B annual revenue (2011), from advertisers
  - 68% out to website publisher (you), 32% to Google
- contextual advertising: knowing content of webpage, demographic (geo location, past history) of client view webpage, can narrowly target ad to webpage viewer
  - higher chance of viewer buying

HTTP: example

Jim accesses http://gaia.cs.umass.edu/cs290nw from home (9/22/2013)

HTTP GET message
(getting the homepage base file)

```
GET /cs290nw HTTP/1.1
Host: gaia.cs.umass.edu
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:22.0) Gecko/20100101 Firefox/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

gaia.cs.umass.edu/cs290nw home page
HTTP: example

Jim accesses http://gaia.c.umass.edu/cs290nw from home (9/22/2013)

Client browser

HTTP reply message

HTTP/1.1 200 OK
Date: Mon, 23 Sep 2013 00:15:02 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Wed, 04 Sep 2013 19:39:41 GMT
ETag: "daddb-1d22-7f6e5940"
Accept-Ranges: bytes
Content-Length: 7458
Keep-Alive: timeout=10, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

Line-based text data: text/html
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
...
</head>
<body>
<img src="http://gaia.c.umass.edu/cs290nw/shutterstock_90734176.jpg" width="161" height="128" alt="networked world logo" />
</body>
</html>

HTTP: example

Jim accesses http://gaia.c.umass.edu/cs290nw from home (9/22/2013)

Client browser

HTTP GET message

GET /cs290nw/shutterstock_90734176.jpg HTTP/1.1
Host: www-net.cs.umass.edu
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:22.0) Gecko/20100101 Firefox/22.0
Accept: image/png,image/*;q=0.8,image/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www-net.cs.umass.edu/cs290nw/
Connection: keep-alive

Name of web containing the reference to the image being requested
**HTTP: example**

Jim accesses http://gaia.cs.umass.edu/cs290nw from home (9/22/2013)

---

**HTTP reply message**

(contains JPEG image)

---

**HTTP reply message header**

---

**This is a JPEG image below**

---

**JPEG image file**

---

---

---

**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  

---

Application Layer 2-6
Cookies: keeping “state” (cont.)

Cookies (continued)

what cookies can be used for:
- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

how to keep “state”:
- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

aside
cookies and privacy:
- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites
Cookies: HTTP response from nytimes.com

HTTP/1.1 200 OK
Date: Mon, 23 Sep 2013 00:41:32 GMT
Server: Apache
expires: Thu, 01 Dec 1994 16:00:00 GMT
cache-control: no-cache
pragma: no-cache
Set-Cookie: RMID=007f0100191a523f8e3c0064; Expires=Tue, 23 Sep 2014 00:41:32 GMT;
Path=/; Domain=.nytimes.com
Set-cookie: adxcs=-; path=/; domain=.nytimes.com
Vary: Host
Content-Length: 202615
Connection: close
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Transfer-Encoding: chunked

HTTP chunked response
Content-encoded entity body (gzip): 46739 bytes -> 202615 bytes
Line-based text data: text/html
   <!DOCTYPE html>
   ......

Cookies returned from nytimes.com

Cookies: next GET to nytimes.com

Hypertext Transfer Protocol
GET / HTTP/1.1
Host: www.nytimes.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:22.0) Gecko/20100101 Firefox/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://www.nytimes.com
Cookie: RMID=007f0100191a523f8e3c0064; adxcs=; ... 
Connection: keep-alive

Cookies that were returned to me previously by nytimes.com
HTTP: improving performance

Concepts to cover:

- IF-MODIFIED-SINCE header: no need to download a page that has been cached (saved in browser) and not changed
- web-cache: download from a closer-by server that may have a cached copy

Web browser caching

- browser cache: web browser stores copy of all downloaded content (typically, on your disk)
- motivation: before downloading content, as server if content has changed
  - HTTP message/reply to check if content has changed is small, compared to content itself
  - don’t download cached content that hasn’t changed

How Do I Clear My Browser's Cache? - PC Support - About.com
 pcsupport.about.com › Internet & Network

by Tim Fisher - in 1,457 Google+ circles
Tutorials on how to clear the browser cache in Firefox, IE, Chrome, and Safari.
To clear the cache means to remove copies of web pages stored by the browser.
**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

**Web caches (proxy server)**

**goal:** satisfy client request without involving origin server

- **user sets browser:** Web accesses via cache
- **browser sends all HTTP requests to cache**
  - object in cache: cache returns object
  - else cache requests object from origin server, then returns object to client
More about Web caching

- cache acts as both client and server
  - server for original requesting client
  - client to origin server
- 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

HTTP: cookies and advertising

Concepts to cover:

- web page content (including ads) from multiple site: composition at client
- cookies
- third-party cookies: ad network server tracking user web page accesses across multiple sites
A short history of the best

music ever.

Disco is a genre of

music from the

1970's ...

Disco hits!

HTML file contains text, and two <IMG> tags. Both images are stored on ilovedisco.com

1. HTTP GET request to ilovedisco.com for homepage
2. ilovedisco.com server send homepage HTML file to browser via HTTP reply
3. Browser reads homepage HTML file, sees <img> tag (John Travolta), requests first image via HTTP GET
4. ilovedisco.com server sends first image to browser via HTTP reply
5. Browser sees second <img> tag (the ad) and requests ad image via HTTP GET
6. ilovedisco.com server sends ad image to browser via HTTP reply; web page displayed at client

HTTP: homepage, image, ad (v1):

- all web page content at ilovedisco.com
- HTML file, Travolta image, ad are separate files on server - composed into webpage at client
- same content would be served to all browsers
- ilovedisco.com would sell ad space directly to Disco Hits
A short history of the best music ever. Disco is a genre of music from the 1970's...

1-4. As before, home page, and first image (J. Travolta) downloaded from ilovedisco.com
5. Browser reads second <img> tag (ad) requests ad image from MegaAd.com via HTTP GET, with referer field: ilovedisco.com
6. MegaAd.com server sends ad image to browser via HTTP reply, knowing image is to be embedded in page from ilovedisco.com. Page displayed at client

HTTP: homepage, image, ad (v2): observations

- ad content not served by ilovedisco.com
- ilovedisco.com could sell ad space directly to Disco Hits who provides content
- ilovedisco.com could sell ad space to ad network, who serves content
  - ad network serves as aggregator for many products/companies,
  - knows “referer”
  - ilovedisco wouldn’t even know what ad is going to be displayed in its page!
A short history of the best music ever.
Disco is a genre of music from the 1970's...

1. HTTP GET request to ilovedisco.com for homepage
2. ilovedisco.com server sends homepage HTML file to browser via HTTP reply, with ilovedisco cookie: 82. ilovedisco cookie: 82 Cookie stored at client.
3. HTTP GET for image, GET message contains with ilovedisco cookie: 82
4. ilovedisco.com server sends first image to browser via HTTP reply
5. Browser sees requests ad image via HTTP GET message with ilovedisco cookie: 82
6. ilovedisco.com server sends ad image to browser via HTTP reply; page displayed at client.

7. HTTP request to ilovedisco.com for homepage
8. ilovedisco.com server sees cookie in GET msg, sends homepage HTML file to browser via HTTP reply containing DIFFERENT AD IMAGE from last time
9. HTTP GET for Travolta image, GET contains with ilovedisco cookie: 82
10. ilovedisco.com server sends Travolta image
11. Browser requests new ad image via HTTP GET with ilovedisco cookie: 82
12. ilovedisco.com server sends new ad image to browser via HTTP reply; page displayed.
HTTP: homepage, image, ad (v3): observations

- cookies can be used to personalize (target) content (e.g., ads) to client based on past interaction with this server
- web server can dynamically generate content depending on what client has done/seen in past

Let’s watch WSJ video about 3rd part cookies!

HTTP: Third party cookies

1-5. As before, home page, and first image (J. Travolta) downloaded from ilovedisco.com, request made for ad image from MegaAd.com via HTTP GET, with referer field: ilovedisco.com

6. MegaAd.com server sends ad image to browser via HTTP reply, knowing image is to be embedded in page from ilovediscocom, adds its own cookie MegaAd: 814. Remembers that cookie #814 owner had visited ilovedisco.com

Third party cookie: when you visit a web page, a third website is able to put a cookie on your browser (as shown here).
HTTP: Targeted advertising (v4)

1-6 client visits ilovedisco.com, disco ad served by MegaAd.com

7-10 client visits iloveNY.com, HTML text and image served by iloveNY.com

11 client contacts MegaAd.com to get ad to display, includes MegaAd cookie # 814

12 MegaAd.com sees referred request from iloveNY.com, sees cookie 814, knows client visited disco site earlier, serves targeted content ad: disco + NY

HTTP: Targeted advertising - observations

- *third party cookies* allow third party (e.g., MegaAds.com) to track user access over multiple web sites (any site with MegaAd link)

- MegaAd uses past user activity to micro-target specific ads to specific users
  - MegaAd can charge ad creators more to place their ads in micro-targeted manner (since user is more likely to be interested in ad)

- users not aware of third party cookies and tracking
  - invasion of privacy ????
Using Cookies to track users

- Wall St. Journal video: [Video Link]

- Mozilla (maker of Thunderbird browser) recently (Feb. 2013) announced it will block 3rd party cookie use in next release.

- Interactive Advertising Bureau (IAB) is concerned

- “Do Not Track” legislation being discussed/introduced
  - Sept. 2013 in CA
  - US Senate: Do Not Track Me Online Act of 2011
  - EU: Right to be Forgotten

Ad networks case study: Google’s Adsense

The players:

- **Web server (publisher)**: HTML file contains text end embedded reference to Google Adsense. You want to show content and make $ from ad display.

- **Google Adsense**: chooses ad to serve, shows up in your webpage, displayed at client. Wants to make $.

- **Client**: who will visit your website. Advertisers want to sell to them. You want to show them content (AND make $ from selling ad space).

- **Advertisements (advertisers)**, who will pay to put ads on your webpage.
AdSense: bidding

- **ahead of time**: website publisher uses AdSense: layout ad space on web page, provide context (e.g., keywords about website content) for website
- client HTTP request to fetch publisher webpage eventually results in HTTP request to AdSense to provide ad
  - ad composed with publisher webpage, displayed at client
- AdSense runs *auction*, in real time (milliseconds) among all advertisers who *bid* for price they will pay to place their ad on webpage

Reflections on AdSense

- 
- 
- 
- 
- 

Introduction to the Ad Auction, Hal Varian
http://www.youtube.com/watch?v=gbbYrFJah3c